Online Optimization with Lookahead

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)
von der Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von
Dipl.-Wi.-Ing. Fabian Dunke

Tag der mündlichen Prüfung:	24. Juli 2014
Referent:	Prof. Dr. Stefan Nickel
Korreferent:	Prof. Dr.-Ing. Kai Furmans

Karlsruhe, 2014

Acknowledgements

The work you are looking at is my thesis, but it would definitely not have been possible without the help, support and guidance of others. I first would like to thank those people.

To Prof. Dr. Stefan Nickel:
For your farsighted mentoring, for your valuable ideas throughout supervising this thesis, and for giving me the opportunity to work in OR.

To Prof. Dr.-Ing. Kai Furmans:
For your kind interest in this work and for co-supervising it, and not to forget, for the first steps that I could take in logistics when I was a student.

To Prof. Dr. Oliver Stein:
For your long-lasting and careful attention, for arousing my interest in optimization, and for teaching me how to work precisely.

To Prof. Dr. J. Philipp Reiss:
For your sincere and friendly conduct as the chairman of the examination committee.

To Prof. Dr. Gerhard Woeginger:
For your lucid recommendations with regard to the realization of mathematical concepts and for your gracious hospitality.

To Alex, Anne, Brita, Eric, Ines, Jörg and Melanie:
For the amicable working atmosphere which makes work a pleasant duty and for the comments on this thesis which doubtlessly helped me improve the quality of this work.

To Marcel and Lars:
For calling me over the last five years to have an always delicious meal at mensa.

To My Family:
For the support and encouragement that you have given to me all my life.

Fabian Dunke

Abstract

Online optimization with lookahead deals with sequential decision making under incomplete information where each decision must be made on the basis of a limited, but certain preview (lookahead) of future input data. In many applications, this optimization paradigm provides a better description of a decision maker's informational state than the well-established disciplines of offline and online optimization since not all may be known about the future, but also not nothing.

Despite the growing importance of the resource information as a result of technological advances, lookahead is often still only deemed an add-on to online optimization in problem-specific contexts. We argue that in order to understand how algorithm performance can be enhanced by additional information, it requires a common understanding of lookahead and its implications on instance processing by algorithms. The main contributions of this thesis consist of the development of a systematic groundwork for comprehensive performance evaluation of algorithms in online optimization with lookahead and the subsequent validation of the presented approaches in theoretical analysis and computational experiments.

In the first part, we embed the paradigm of online optimization with lookahead into the theory of optimization and develop a precise definition of the term *lookahead*. We find that the lookahead effect on the objective value can be subdivided into an informational and a processual component: The former yields the improvement attainable by forwarded information release, while the latter expresses the improvement attainable by the change in a problem's "rules" immanent to lookahead. Since it is widely acknowledged that competitive analysis – still the standard gauge for performance measurement of online algorithms – fails to display the typical behavior of algorithms, we lay out a holistic distributional approach of performance analysis which takes into account both the absolute behavior of an algorithm as well as its behavior relative to some reference algorithm. This approach facilitates an explicit consideration of different information regimes. Further, we establish the link to discrete event systems which finally leads to the formulation of a generic modeling framework for online optimization with lookahead.

The second part applies the proposed method of distributional performance analysis to online algorithms endowed with various degrees of information preview and provides structural insights with regard to observable lookahead effects in the respective problem settings: We first perform an exact analysis in basic settings of the ski rental, bin packing and traveling salesman problem. From the proofs, we obtain explanations for the fact that lookahead leads to different magnitudes of improvement depending on the respective problem types. Subsequently, we expand our analysis to more general settings of the above problems and additionally to the paging and scheduling problem: Extensive sample-based numerical experiments are conducted to examine the algorithms' reactions to different levels of supplied information. Obtained results are gathered in an information pool concerning the impact of additional information in several standard problems of online optimization. Results on the lookahead effect from these problems can conditionally be transferred to more complex settings as seen in simulation studies on the real world applications of an order picking system in a warehouse and a pickup and delivery service in a road network. We conclude that our approach to performance analysis of algorithms in online optimization with lookahead can be employed in problem settings of various complexities to assess the value of information and to determine the most suitable algorithm from a set of potential algorithm candidates for different lookahead levels.

Contents

1 Introduction

Although there is undisputed agreement on the importance of coping with unexpected events in today's systems for production and logistics ([78], [148], [154]), recent implementations of planning and scheduling systems such as Advanced Planning Systems (APS) still suffer from their deficiency in dealing with uncertainty over time: In a rolling time horizon, future plans are determined on the basis of forecasted data by offline optimization methods ([154]). However, since only decisions of the next period are implemented before the problem gets resolved with updated forecasts, this approach exhibits a high degree of redundancy. Additionally, predictions are destined to be wrong, and it is only a matter of time before deviations between plan and reality will occur.

On the other hand, the number of problem settings where input data can be collected and processed in real time is continuously increasing due to technological advances ([78]). Since planning systems built for these environments are subject to steady information disclosure, they are said to be online. Optimization problems arising in this context are called online optimization problems and algorithms for them have to operate dynamically. This paradigm is completely opposite to that of classical offline optimization where all input data is assumed to be known in advance. Between these two extremes, there is an intermediate setting which we will call online optimization with lookahead. Here, the amount of accessible information is governed by some lookahead mechanism. Online optimization with lookahead represents an alternative approach for dealing with unpredictable events: Instead of having to rely on forecasted data, uncertainty is tackled by sequential decision making where each decision is made based only on the small, but certain part of the future known at that time.

In an organizational context, the task of solving online optimization problems (with lookahead) is a recurring pattern needed to operate and control industrial applications. The functional logic of a dynamic system repeatedly requires decision making in order to continue ([135]). For each of these decisions, an online algorithm is called as a subroutine. It has to determine partial solutions based on the currently available input data such that the overall solution which will be composed of all partial solutions will be as good as possible.

Figure 1.1 sums up the hierarchical relation between the logic in a dynamic system and the online optimization module needed therein (see also [119]).

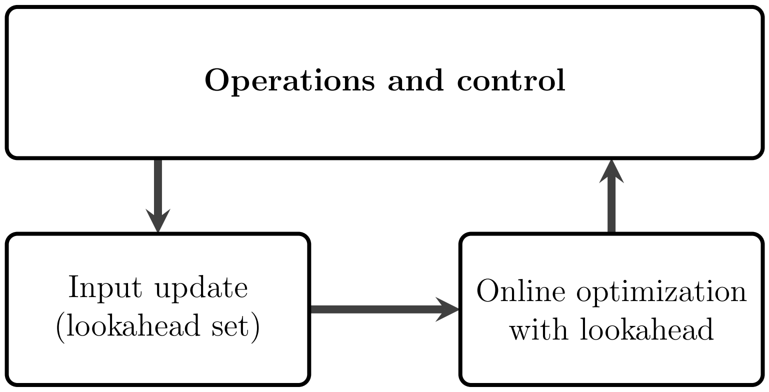
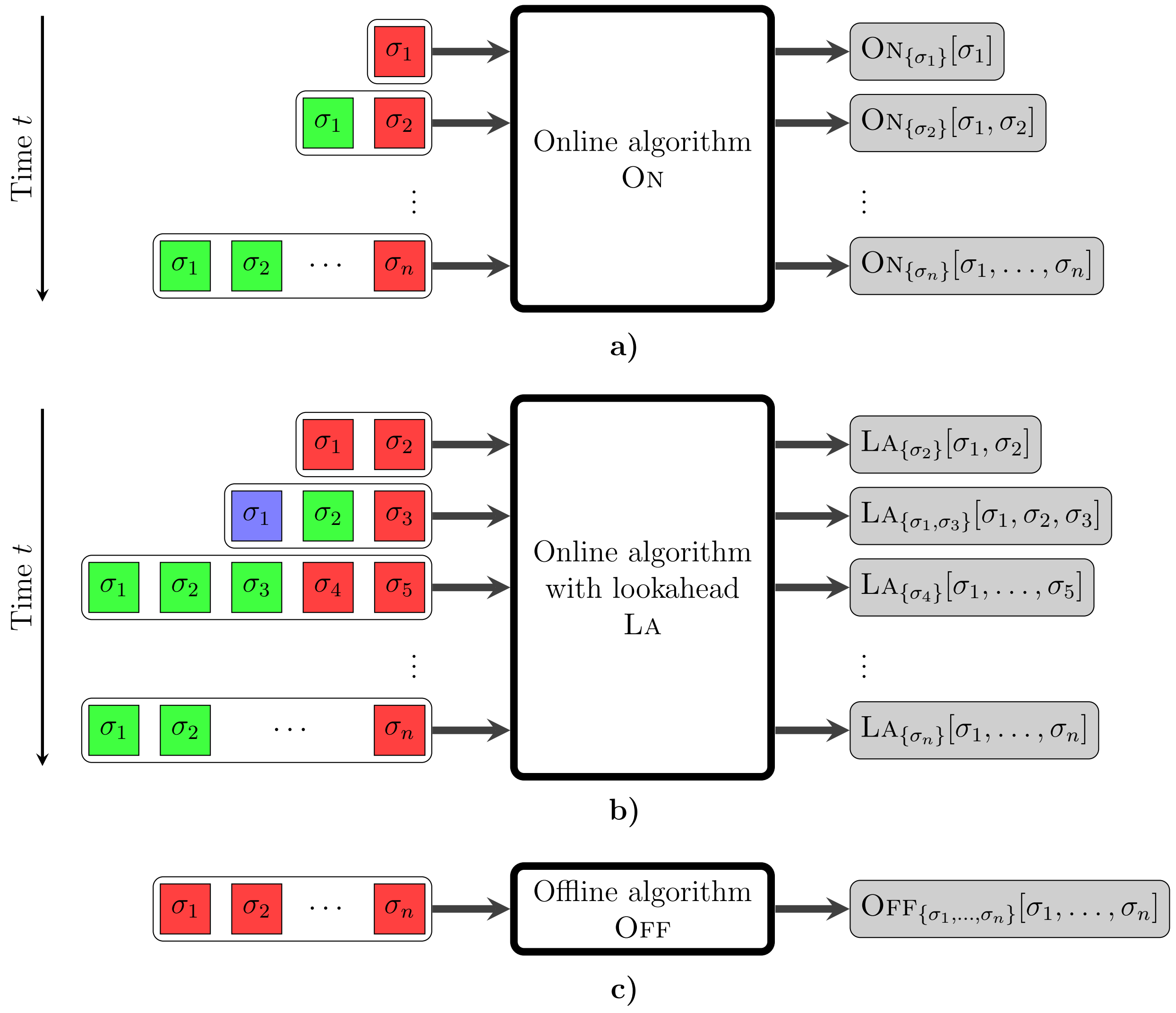

Operations and control
Input update (lookahead set)
Online optimization with lookahead

Figure 1.1: Hierarchical relation between operations and control of a dynamic system and online optimization with lookahead.

Whether the assumption of complete or incomplete information applies, depends on the application: On the strategic level, almost all problems are offline, e.g., facility location, supplier selection or distribution channel selection. On the operational level, problems are often intrinsically online, e.g., order picking, scheduling or transportation planning. Problems on the tactical planning level, such as capacity planning or distribution planning, appear to be of either kind. A variety of problems is solved by concatenating offline and online optimization methods: In the first stage, offline optimization is carried out with all data available at the start of the planning horizon; in the second stage, input data is collected and processed repeatedly in an online manner where fixed decisions from the first stage are respected (cf. also [88]). We conclude that online optimization problems with lookahead are encountered primarily on the operational and occasionally on the tactical level of control.

Algorithms for solving online optimization problems – both with and without lookahead – have to obey regimes of incomplete information while making their choices. Contrarily, offline optimization algorithms are privileged to resort to complete information while computing a solution. Solution methodologies for tackling the different types of these problems strongly differ from each other as illustrated exemplarily in Figure 1.2 for the processing of an input sequence consisting of input elements $\sigma_1, \sigma_2, \ldots, \sigma_n$ with $n \in \mathbb{N}$. Arriving input elements are represented by red rectangles, already processed input elements appear in green rectangles, and already known but still unprocessed input elements are printed in blue rectangles. For algorithm Alg and $i, j, m \in \mathbb{N}$ with $i \leq j \leq m$, we denote by $\text{Alg}_{\{\sigma_i,\ldots,\sigma_j\}}[\sigma_1, \ldots, \sigma_m]$ the costs incurred by Alg for processing the input elements in $\{\sigma_i, \ldots, \sigma_j\}$ based on information $\sigma_1, \ldots, \sigma_m$.



Time t
σ_1
σ_2
σ_n
Online algorithm ON
$\text{ON}_{\{\sigma_1\}}[\sigma_1]$
$\text{ON}_{\{\sigma_2\}}[\sigma_1, \sigma_2]$
$\text{ON}_{\{\sigma_n\}}[\sigma_1, \dots, \sigma_n]$
a)
Time t
σ_3
σ_4
σ_5
Online algorithm with lookahead LA
$\text{LA}_{\{\sigma_2\}}[\sigma_1, \sigma_2]$
$\text{LA}_{\{\sigma_1,\sigma_3\}}[\sigma_1, \sigma_2, \sigma_3]$
$\text{LA}_{\{\sigma_4\}}[\sigma_1, \dots, \sigma_5]$
$\text{LA}_{\{\sigma_n\}}[\sigma_1, \dots, \sigma_n]$
b)
Offline algorithm OFF
$\text{OFF}_{\{\sigma_1,\dots,\sigma_n\}}[\sigma_1, \dots, \sigma_n]$
c)

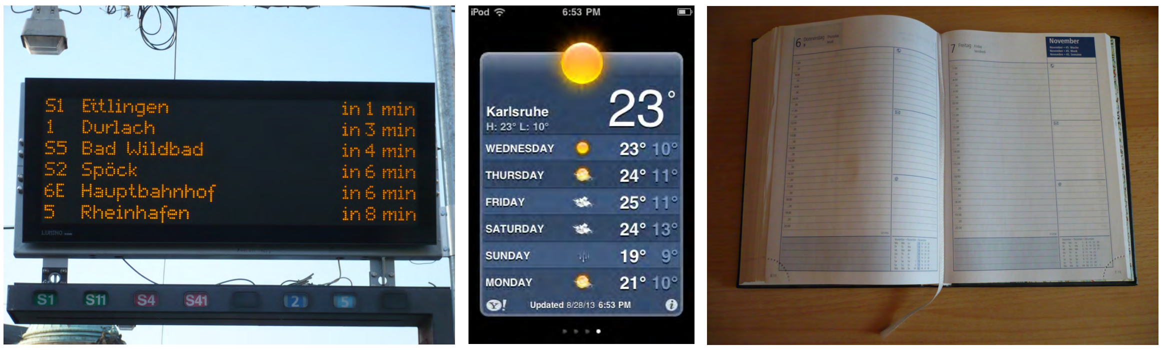
Figure 1.2: Comparison of optimization paradigms. **a)** Online optimization without lookahead. **b)** Online optimization with lookahead. **c)** Offline optimization.

Significant research efforts have been put into tackling continuous online optimization problems arising in process industries (e.g., chemical production, raw materials processing) and in control problems (e.g., regulatory control of power plants, robotics, aerospace). Related problems are coined by continuous nonlinear dynamical systems, and the task consists of monitoring and controlling the processes by adjusting parameters in order to keep the system in a steady state (see, e.g., [84], [125]). To solve these problems, methods from control theory are applied where mainly continuous decision variables appear within differential equations.

In this thesis, we deal with discrete online optimization problems which means that decisions can be traced back to a discrete structure ([85]). Most problems emerge from combinatorial optimization where one searches for a best element in a discrete set of feasible solutions or from integer programming where one aims at solving mathematical programs with decision variables constrained to take on integer values[1]. Online problems of this type occur in a multitude of domains (see, e.g., [24], [32], [84], [74], [85]) including

- production and logistics, e.g., routing, packing, scheduling, load balancing,
- telecommunications, e.g., call admission, circuit routing,
- memory management, e.g., caching, paging, file migration,
- self-organizing data structures, e.g., list accessing, binary search trees,
- financial engineering, e.g., rent-or-buy decisions, portfolio selection, trading, and
- theoretical problems, e.g., graph coloring, graph matching, online linear programming.

Lookahead information is also encountered in plenty of situations in our everyday lives as shown in Figure 1.3 and it has a major influence on our decision making: Dynamic passenger information boards provide predictions about expected vehicle arrivals within the next minutes; the information can be used in order to update travel routes based on the current traffic scene. The weather forecast influences decisions concerning weather-dependent outdoor activities and prevents us from booking them when the weather is announced to be bad. A calendar can be seen as the ultimate embodiment of lookahead as it allows to record all known pieces of future information which seem relevant to organize our personal or professional schedules.



a) b) c)

Figure 1.3: Everyday life situations where decisions can be improved due to lookahead. **a)** Dynamic passenger information board. **b)** Weather forecast. **c)** Organizer and calendar.

[1] Combinatorial optimization and integer programming are closely related to each other due to the fact that many combinatorial optimization problems can be formulated as integer programs and, vice versa, many integer programs can be understood in terms of a combinatorial optimization problem.

1.1 Problem Statement and Scope of the Thesis

Basic variants of online problems have been studied in the mathematical framework known as competitive analysis (see, e.g., [32], [74]): Algorithms for an online optimization problem have to compete with an optimal offline algorithm which knows the whole input in advance and quality guarantees have to hold for arbitrary input sequences. Hence, competitive analysis is a worst-case consideration of a worst-case analysis; results are overly pessimistic and do not reflect an algorithm's practical abilities to suitably deal with a given problem.

Most theoretical results were derived based on the taxonomy prevalent in a specific problem and not based on a general notation valid for problems of all kinds. Likewise, the intermediate setting of online optimization with lookahead has been addressed by the online optimization community every now and then only in specific problems arising in routing and transportation ([7], [8], [11], [34], [96], [95], [155]), scheduling ([51], [123], [129], [130], [134], [161], [168], [169]), organization of data structures ([2], [3], [41], [113], [156], [163], [164]), data transfer ([64], [93]), packing ([83], [86]), lot sizing ([1]), metrical task systems ([20], [115]) or graph theory ([45], [87], [94]). To the best of our knowledge, there have been no attempts to formalize different degrees of available information in a general framework.

A reason for the lack of general concepts lies in the unsettled role of the factor time. In some problems it is just used to establish an order of events (sequential model); others intrinsically rely upon time as a part of the instance specification (time stamp model). This issue also accounts for various perceptions of the term lookahead: Does it mean that a certain number of future input elements is known? Does it mean that all future input elements occurring in a particular time window are foreseen?

Endowing an algorithm with lookahead should lead to better results due to improved planning opportunities. Therefore, lookahead is deemed a mechanism for increasing the power of an algorithm ([96]) and we may ask for the value of a preview on future information within the class of online optimization problems.

Obviously, lookahead without an algorithm which can make use of it renders itself worthless. Therefore, determining the value of lookahead and performance analysis of algorithms are closely intertwined. By the nature of sequential decision making under incomplete information, possible "errors" of an algorithm cannot necessarily be corrected later when one realizes that another decision would have been better ([151]). Due to the inevitability of failure, it is impossible to find an algorithm which solves an online optimization problem to optimality and all we can do is to find algorithms for a certain problem which are as good as possible.

Asking for the value of additional information gives rise to the idea of comparing algorithms under lookahead to algorithms with small information resources rather than to an optimal offline algorithm which is in sharp contrast to competitive analysis. Since we consider it unfair to use an optimal offline algorithm as the performance yardstick in online optimization, we seek for other, more practically oriented methods for the analysis of algorithms.

In summary, we recognize that a number of significant questions in online optimization with lookahead have not yet been addressed satisfactorily. Motivated by the above shortcomings in the current state of the theory, we formulate four major research questions (RQ):

RQ1 What do we understand by lookahead?

RQ2 Which formalism can be used to model the solution process in an online optimization problem with lookahead in a generally applicable framework?

RQ3 Which performance measurement approach is best suitable to analyze the performance of algorithms in online optimization problems with lookahead and to relate the quality of algorithms to each other?

RQ4 What is the value of different degrees of lookahead in specific online optimization problems with lookahead?

In industrial settings, algorithms often have to terminate in a couple of seconds ([85]). Traditionally, information is the only scarce resource in online optimization and no attention is paid to computing time. In awareness of this gap between theory and practice, we will have a look at real-time requirements whenever they may become crucial in distinguishing between the quality of algorithms, e.g., when $\mathcal{NP}$-hard (sub-) problems are encountered.

Finally, we point out that the approach of online optimization with lookahead taken in this thesis can be distinguished from other approaches for optimization under incomplete information (see, e.g., [5], [29]): In stochastic programming, probability distributions for scenarios that take into account all uncertain factors are known and solution quality is typically evaluated by average case measures such as to immunize the solution probabilistically to incomplete information. In addition, stochastic programming is rather concerned with sporadic than with frequent decision making. Robust optimization, in contrast, does not rely on probability distributions but on a given range of possible values for uncertain factors. The goal is to construct a solution which is feasible for all possible realizations and exhibits optimality in some robustness-related sense. A conceptual framework called online stochastic optimization which assumes given distributions for future requests has been devised in [22] and [24]. Generic algorithms that express different goals and exploit given stochastic information are proposed, e.g., optimizing expectation, consensus or regret.

Online optimization with lookahead as treated in this thesis differs from the previous approaches by its perception of uncertainty: Rather than presuming a particular probability model or possible value ranges, it strives for a more holistic analysis of uncertainty as justified by increased volatilities in today's markets. We opt for a method of performance analysis which incorporates typical and worst-case behavior of an algorithm as well as its overall performance range in an equal measure. Although the traditional focus of online optimization is on hedging against worst-case scenarios, recent application-driven developments show that in a more comprehensive view on the topic also aspects like sensitivity to additional lookahead or integration into simulation environments need to be addressed.

1.2 Applications of Online Optimization with Lookahead

Online optimization problems with lookahead arise in applications of different domains. The following examples suggest that lookahead is polymorphic depending on the context.

Online Routing with Lookahead

A recurring task in transportation and logistics is vehicle routing ([118]). As a result of increased usage of geographic information systems (GIS) and global positioning systems (GPS), the research focus has shifted from the static to the dynamic version of the problem ([139]); these variants refer to the offline and online version, respectively. Applications can be found in emergency, taxi and repair services as well as in order picking.

A number of requests has to be served by a set of vehicles each starting and ending in a given depot O with the aim of optimizing some costs such as the total travel distance. Every request has a release time representing the earliest time for service. Providing lookahead makes both locations and release times of the requests known earlier. In Figure 1.4, the offline situation is compared to the online situation. In the latter case, dots in gray correspond to unknown requests at snapshot time and numbers indicate the request revelation order.

Providing additional lookahead is expected to lead to enhanced performance by incorporating more requests into an algorithm's strategy. However, based on customer requirements it needs to be clarified first whether earlier notification due to lookahead also facilitates earlier customer service, or whether earliest service times from the online problem without lookahead are retained. This requirement strongly impacts the optimization potential induced by lookahead. We point out three notions of lookahead known from literature:

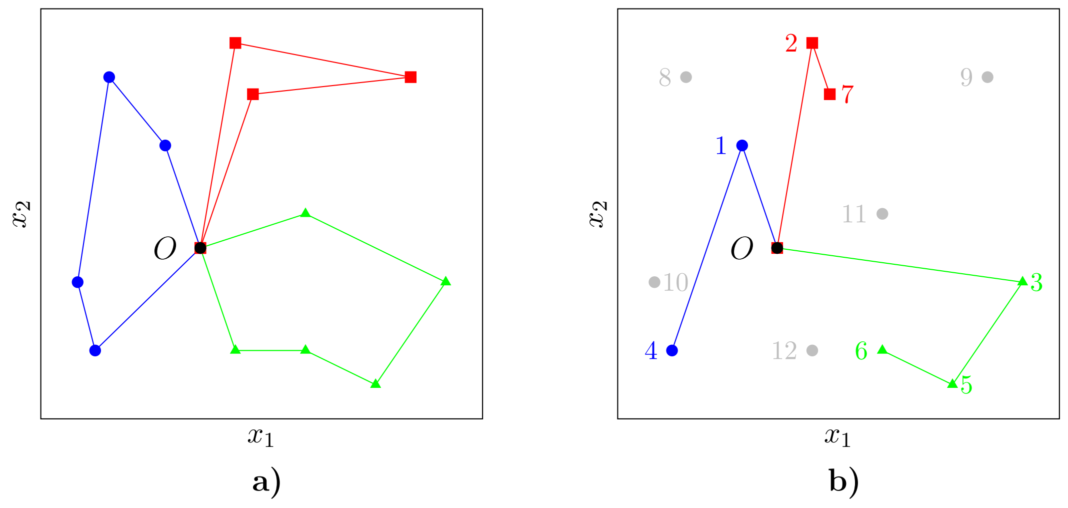


Figure 1.4: Vehicle routing. **a)** Offline (static) version. **b)** Online (dynamic) version.

- Request lookahead allows an algorithm to foresee a fixed number of upcoming requests ([7], [8]).
- Time lookahead as discussed by Allulli et al. ([7], [8]) and Ausiello et al. ([11]) allows an algorithm to foresee all requests having a release time within a fixed time window starting at the current time.
- Disclosure times of requests introduced by Jaillet and Wagner ([96]) explicitly specify the notification times of requests and differ from their earliest service times.

Request lookahead is probably the most unrealistic among these concepts ([11]), whereas disclosure times allow for a customer-specific model of lookahead ([96]). Time lookahead sets the same temporal offset between notification and release of a request for all customers. Competitive analysis in [7] and [96] yields that time lookahead and disclosure dates may lead to (slight) improvements depending on the objective function and metric space.

We mention that there are numerous refinements and generalizations of the vehicle routing problem with industrial relevance ([158]), e.g., pickup and delivery problems or inventory routing problems, which all lend themselves to an integration of additional lookahead information. The design of real-time compliant algorithms has to take into account the computational complexity of vehicle routing problems[2], e.g., by devising decomposition methods such as cluster-first route-second strategies ([101]). Moreover, one has to be aware of counterintuitive problem features such as the fact that waiting for requests located in spatial proximity may be beneficial although there are still unserved requests.

[2] The vehicle routing problem is $\mathcal{NP}$-hard because its decision version contains `Hamiltonian Circuit` which is known to be an $\mathcal{NP}$-complete problem.

Online Bin Packing with Lookahead

Packing comprises the task of combining objects from a set of small items in order to pack them into elements of large objects such that some objective function is optimized ([69]). The practical scope of packing is twofold: First, it includes the combinatorial task of grouping small items into subsets and assigning each of them to a large object; second, it includes the geometric task of ensuring that within each large object the small items are laid out such that they are entirely contained in the large object without overlapping. Applications include packaging logistics, assembly line balancing, memory allocation and layout design.

A fundamental packing problem is the (one-dimensional) bin packing problem ([77]) where a number of items $\sigma_1, \sigma_2, \ldots, \sigma_n$ for $n \in \mathbb{N}$ with sizes $s_i \in (0,1]$ for $i = 1, \ldots n$ is given and the task is to pack them into a minimum number of unit-capacitated bins. We seek to find a partition of $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ into a minimum number m of subsets $B_1, B_2, \ldots, B_m$ such that

$$\sum_{\sigma_i \in B_j} s_i \leq 1$$

for all $j = 1, \ldots, m$. The problem is computationally complex[3].

The problem instance in Figure 1.5 shows why it is important to have a look at different modes of information disclosure. In the pure online setting, items arrive and have to be packed one after another without knowledge of any other future item. In an exemplary lookahead setting, two items to be packed are known at each time except when only one item remains to be packed. While an optimal offline algorithm needs only six bins, all online algorithms without lookahead which do not open a new bin when the item to be packed fits in an already open bin end up with eight bins. Seven bins are needed by all online algorithms with lookahead of two items which try to generate bins occupied as much as possible.

The input sequence in Figure 1.5 is somewhat pathologic with respect to the item sizes and the input sequence length. If the input sequence was much longer, the unoccupied space of the depicted bins would probably be filled. Thus, the performance degradation due to incomplete information is expected to be small for sufficiently long item sequences.

In contrast to the assumptions of the basic bin packing problem, there will be bounds on the number of open bins as a result of space restrictions in practice: In packaging logistics one would have to obey the number of packaging stations or loading docks; in memory allocation one would have to respect storage capacities ([83]). This problem is called the

[3] Bin packing is $\mathcal{NP}$-hard because its decision version can be reduced from `Partition` which is known to be an $\mathcal{NP}$-complete problem.

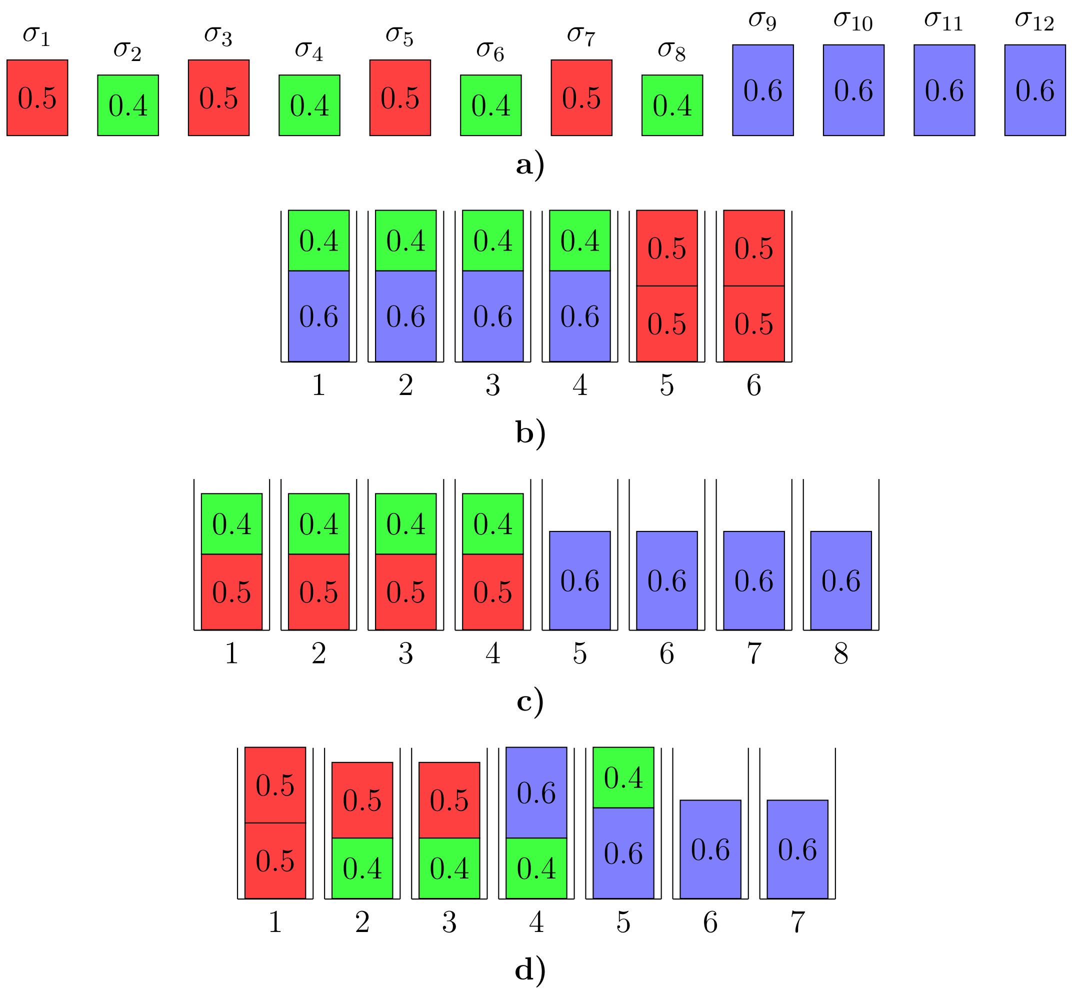

σ_1 σ_2 σ_3 σ_4 σ_5 σ_6 σ_7 σ_8 σ_9 σ_{10} σ_{11} σ_{12}
0.5 0.4 0.5 0.4 0.5 0.4 0.5 0.4 0.6 0.6 0.6 0.6
a)
1 2 3 4 5 6
b)
1 2 3 4 5 6 7 8
c)
1 2 3 4 5 6 7
d)

Figure 1.5: Bin packing. **a)** Item sequence. **b)** Optimal offline solution. **c)** Solution of an online algorithm without lookahead. **d)** Solution of an online algorithm with lookahead.

(one-dimensional) bounded-space bin packing problem: Each time a new bin needs to be opened, one of the K open bins has to be closed first. In packaging logistics this means to send a bin or truck away; in memory allocation this amounts to deleting the contents of some memory module. Because bins cannot remain open arbitrarily long and may be sent away although not fully laden, the improvement to be expected by lookahead in the bounded-space problem should be bigger than in the unbounded case.

Apart from informational benefits, lookahead in packing may serve as a buffer for input elements. In a warehouse, items can be consolidated before their assignment to a destination container ([83], [101]). Thus, lookahead equips the decision maker with more alternatives through the accumulation of items. Clearly, this only holds if the processing order of known items is arbitrary. We require to define lookahead always in conjunction with a specification of a corresponding processing mode which tells us whether permuting input elements is allowed (e.g., by sorting physically small items in a buffer) or not (e.g., by enqueuing physically large items in a job sequence).

We mention two types of lookahead from literature applicable to packing problems:

- In (conventional) request lookahead, a fixed number of future objects to be packed is seen at any time ([83]).
- In property lookahead, the lookahead consists of those items which jointly fulfill a given property ([83], [155]).

An instantiation of property lookahead has been laid out by Grove ([83]) for bin packing: The lookahead consists of those items which jointly do not exceed a threshold cost value when being processed by some algorithm. Another instantiation, due to Tinkl ([155]), is to collect those elements in the lookahead which do not exceed a given threshold weight or size. With respect to practical considerations, we need to guarantee that no item stays in the buffer and no bin stays in the warehouse for too long in order to prevent starvation.

The aforementioned problems can be generalized to two or three spatial dimensions and there are numerous additional problem variants (see, e.g., [55]) of which we mention batch bin packing where items become available in blocks, dynamic bin packing where possible events include departures and variable-sized bin packing where bin sizes may vary.

Online Paging with Lookahead

Memory management and data organization intrinsically feature an online character due to data communication over time. Algorithms try to organize memory or data structures such that the total costs for access are lowest possible. The paging problem is a fundamental problem in computer science ([32]) and gave rise to competitive analysis in the 1980s ([149]). It is concerned with efficiently managing a two-level store of memory consisting of a small fast cache memory of size k and a large slow memory of unbounded size. The input sequence corresponds to a sequence of requested pages and a requested page can only be accessed when it is in the cache. Thus, whenever the request is on a page already in the cache, no cost is incurred (cache hit), but whenever an algorithm has to bring the requested page to the cache first, a unit cost is charged (page fault, cache miss). The problem is to decide which cache page to evict upon a page fault. As opposed to the previous applications, there is a polynomial-time optimal offline algorithm: Algorithm LONGESTFORWARDDISTANCE (LFD, Belady's optimal replacement algorithm in [17]) serves every request sequence with the minimum number of faults by evicting the page in the cache which will be requested farthest in the future when a page fault occurs. An online algorithm knows nothing about future requests. Unfavorably, this may escalate to every request producing a cache miss.

Consider a cache of size 3 and a slow memory containing all 26 letters of the standard alphabet. Initially, the cache is filled with $\{a, b, c\}$ as displayed in Figure 1.6, and the sequence of requested pages is $\sigma = (f, a, b, i, a, n)$. Offline algorithm Lfd incurs three page faults. Online algorithm LeastRecentlyUsed (Lru) evicts a page whose last request was earliest, i.e., least recently, among the cache pages. Lru incurs five page faults on σ.

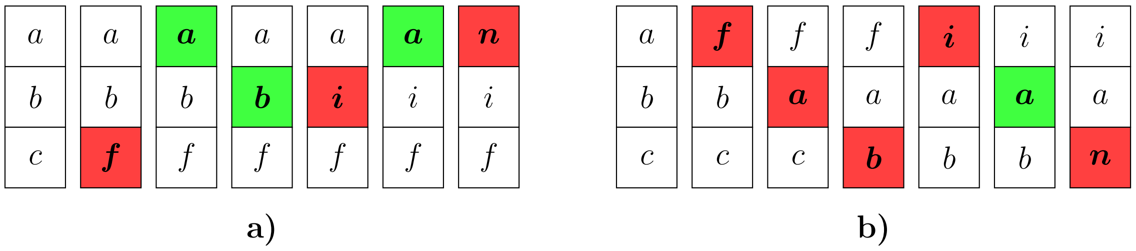


Figure 1.6: Paging. Offline algorithm Lfd in **a)** and online algorithm Lru in **b)** lead to a different number of page faults on input sequence $\sigma = (f, a, b, i, a, n)$.

Requests usually arrive in fixed-size blocks in data communications, thereby giving a natural preview of requests. The model of conventional request lookahead where a fixed number of pages is seen at each time has been repelled because of its ineffectiveness in competitive analysis ([164]): Denote by σ_i^k a request on page σ_i for k times in a row. Then the ratio of the costs incurred by online algorithm $\textsc{Alg}_1$ to the costs incurred by Lfd on page sequence $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is the same as the ratio incurred by an online algorithm $\textsc{Alg}_k$ with lookahead k to the costs incurred by Lfd on page sequence $(\sigma_1^k, \sigma_2^k, \ldots, \sigma_n^k)$ when $\textsc{Alg}_k$ mimics $\textsc{Alg}_1$ on the first of each σ_i^k with $i = 1, \ldots, n$. Lookahead becomes useless in this case since it hides new future requests. To eliminate this shortcoming, we give three alternatives of lookahead that have been devised in literature:

- Strong lookahead of size k as introduced by Albers ([2]) consists of the current request and k additional pairwise different pages which also have to differ from the current request.

- Resource-bounded lookahead of size k as suggested by Young ([164]) consists of those upcoming pages that fulfill the property that no more than $k+1$ page faults will occur when processed by the online algorithm under consideration.

- By natural lookahead of size k as devised by Breslauer ([41]), we understand the knowledge of $k+1$ distinct pages which in contrast to strong lookahead collectively are not in the cache.

Note that strong lookahead is independent of an algorithm, resource-bounded lookahead depends both on an algorithm's past and future behavior which makes it admittedly unrealistic for use, and natural lookahead relies on an algorithm's past behavior. It is shown for each of these lookahead types that mild improvements in competitive analysis are achieved because pathological cases as described above are bypassed. Paging algorithms turn out to strongly benefit already from conventional request lookahead in empirical studies ([41]).

1.3 Overview of the Thesis

The overall structure of this thesis is divided into seven chapters as shown in Figure 1.7.

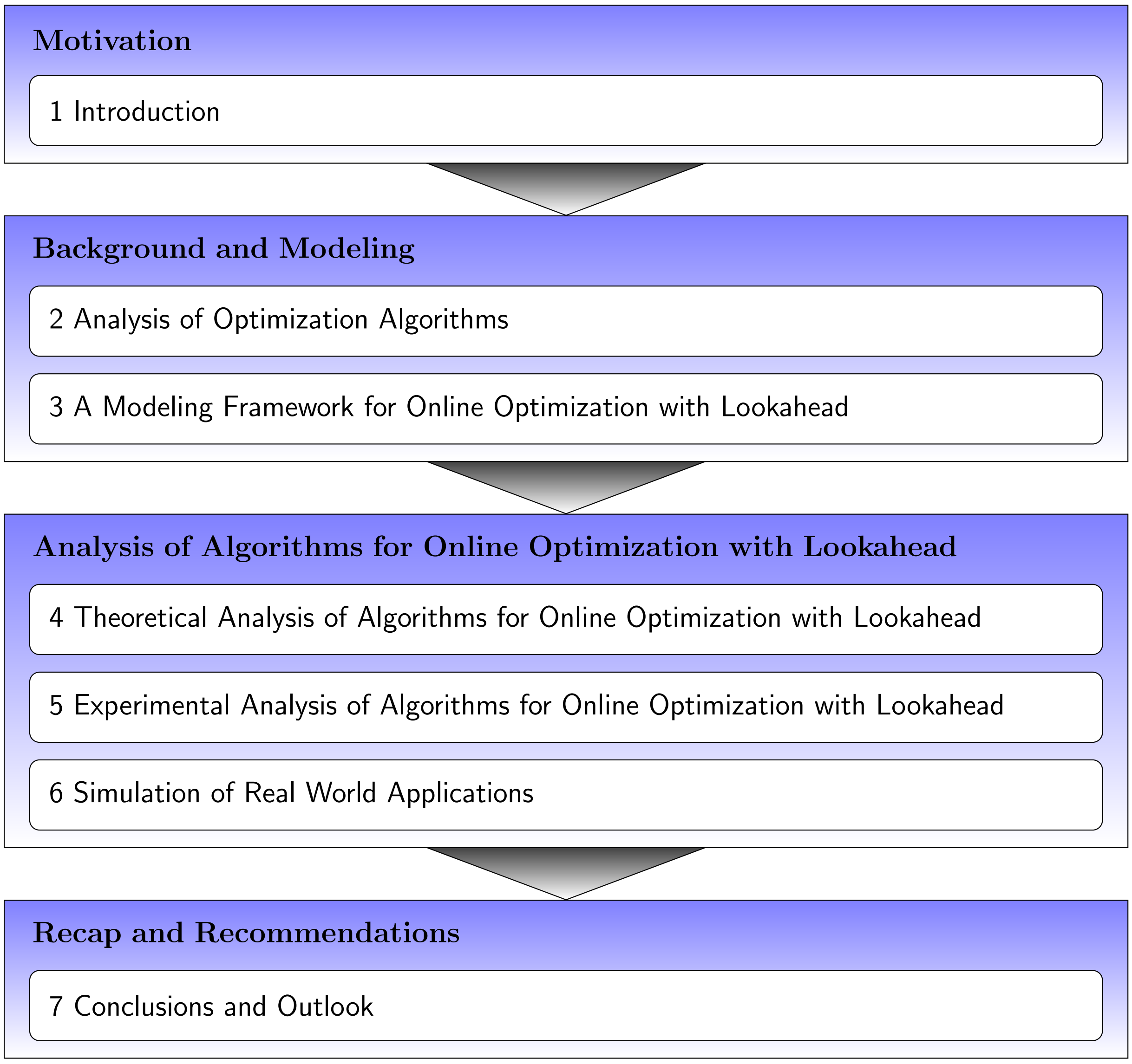

Motivation
1 Introduction
Background and Modeling
2 Analysis of Optimization Algorithms
3 A Modeling Framework for Online Optimization with Lookahead
Analysis of Algorithms for Online Optimization with Lookahead
4 Theoretical Analysis of Algorithms for Online Optimization with Lookahead
5 Experimental Analysis of Algorithms for Online Optimization with Lookahead
6 Simulation of Real World Applications
Recap and Recommendations
7 Conclusions and Outlook

Figure 1.7: Structure of the thesis.

Chapter 1 introduced the subject of online optimization with lookahead, motivated desired research outcomes and laid out some application examples.

The following two chapters are devoted to the definition of a general modeling framework for online optimization problems with lookahead: In Chapter 2, we take a closer look at different optimization paradigms with respect to the amount of information provided, and we discuss different concepts for performance analysis of optimization algorithms, especially in the context of online optimization under varying degree of informational preview. Finally, we interrelate different modeling techniques for discrete event systems and find that methods for solving online optimization problems with lookahead adhere to these techniques as well due to their sequential decision making character. Chapter 3 focuses on the modeling of online optimization problems with lookahead: We collect basic definitions for lookahead and look at their peculiarities. Since one of our main goals is to analyze the performance of algorithms in different applications using unified concepts and a common taxonomy, we propose a generic process model which all online algorithms using lookahead have to obey. In the remainder of this thesis, we continue by instantiating this framework for a variety of important applications and by applying the proposed methods of algorithm analysis to assess the value of lookahead.

Chapters 4 to 6 delve into the analysis of online optimization algorithms endowed with various degrees of lookahead in particular problem settings: In Chapter 4, a theoretical analysis is conducted for basic academic problem settings. First hints are found concerning the role of lookahead as a promoter of improved algorithm performance. In Chapter 5, we explore the effects of lookahead in classic online optimization problems. Detailed numerical experiments are conducted from a sampling-based point of view. The results indicate that the behavior of online algorithms in practice depends on the amount of lookahead, but also that the extent of the lookahead value strongly relies on the problem itself. Online optimization algorithms exploiting lookahead information are used within simulation models of two real world applications in Chapter 6. We learn that due to the higher number of restrictions in practical settings, the lookahead effect is mitigated to a certain extent.

Chapter 7 subsumes the findings of this thesis and recurs to the four central research questions that were specified previously in this introduction. Likewise, we point out limitations of our approach and provide starting points for possible future research directions.

2 Analysis of Optimization Algorithms

Algorithms are computational methods to solve any kind of computational problem, i.e., to provide the correct output for any input ([54]). Optimization algorithms face the task of determining a best possible element out of a set of solution candidates. Unfortunately, in online optimization – both with and without lookahead – the input is revealed only gradually, and due to the inevitability of failure in decision making under incomplete information, it is impossible for any algorithm operating in an online manner to halt with the correct, i.e., best possible, output on any input. This chapter clarifies and resolves the relationships between the different modes of information disclosure and discusses the role of algorithms in this context. In particular, we provide a clear definition of the optimization paradigm *online optimization with lookahead* and decompose the effect of lookahead into an informational and a processual component. To facilitate an analysis of the lookahead impact on solution quality, we develop a holistic approach to performance measurement of algorithms. Finally, general analogies between the solution process in an online optimization problem and discrete event systems are deduced.

2.1 Optimization Paradigms

In online optimization, input data is revealed sequentially. Optimization problems arising in practice often exhibit this type of information disclosure as opposed to standard offline optimization where all data is known in advance[4]. Essentially, offline and online optimization differ in the amount of accessible informational content: Offline optimization assumes complete information, online optimization assumes incomplete information. The definition of *complete* is unique in terms of representing 100 %, but the definition of *incomplete* admits an infinite number of levels representing t % with $t \in [0, 100)$. This leads to the definition of

[4] According to [74], the terms *online* and *offline* are likely to origin from cryptographic systems where decryption was either done continuously during data transfer (*on* the communication line) or after all data were transferred (*off* the communication line).

a more profound notion which we will refer to as online optimization with lookahead. Here, we can quantify the amount of information which is available to any algorithm operating in this information regime. We start with a series of definitions in order to ensure a common taxonomical and notational basis. The first definitions are based on Garey and Johnson ([77]) as well as on Ausiello et al. ([12]); subsequent definitions for online optimization problems – both with and without lookahead – are introduced first in this thesis.

Definition 2.1 (Problem ([77]))**.**
A problem is a general question containing a set of parameters. △

Definition 2.2 (Instance of a problem ([77]))**.**
An instance of a problem is a set of parameter values describing a concrete version of the problem. △

We remark that the term *input* is often used as a synonym for the term *instance*.

Definition 2.3 (Solution ([77]))**.**
A solution to a given instance of a problem is an adequate answer to the concrete version of the problem obtained by replacing the parameters in the general question with the provided parameter values of the instance. △

There are several types of problems requiring different kinds of answers: Decision problems and search problems require yes-/no-answers, counting problems require integer answers, and optimization problems require answers encoding the best solution. In order to give a concise definition of the class of optimization problems, we first need an underlying concept of optimality which allows us to judge on the quality of solutions.

Definition 2.4 (Optimality concept)**.**
An optimality concept is a correspondence returning for each set of solutions to a given instance of a problem a subset of this set to be considered best. △

A natural form of an optimality concept is based on a scalar-valued objective function f which assigns each solution s a number $f(s) \in \mathbb{R}$, the $\leq$-relation on $\mathbb{R}$ and an optimization goal $\text{opt} \in \{\min, \max\}$. There are other concepts of optimality, e.g., Pareto optimality in multicriteria optimization, but for us the above optimality concept induced by $(f, \leq, \text{opt})$ will do. We are in a position to give a formal definition of an optimization problem which implicitly subsumes the previous concepts.

Definition 2.5 (Optimization problem ([12])).
An optimization problem Π is a quadruple (I, S, f, opt) where I is a set of instances, S is a function returning the set of solutions $S(i)$ for any $i \in I$, f is a function returning the objective value for any pair $(i, s) \in I \times S(i)$, and $\text{opt} \in \{\min, \max\}$ is the optimization goal. △

We note that the set $S(i)$ is also called the feasible set of $i \in I$.

In a given instance of an optimization problem, we can account for a best possible solution among the set of all solution candidates in terms of the problem's optimality concept.

Definition 2.6 ((Exact) solution to an instance of an optimization problem ([12])).
Let $\Pi = (I, S, f, \text{opt})$ be an optimization problem.

a) A solution to $i \in I$ is a pair $(s, f(i, s))$ where $s \in S(i)$.

b) An exact solution to $i \in I$ is a pair $(s^*, f(i, s^*))$ where $s^* \in S(i)$ such that $f(i, s^*) \leq f(i, s)$ for all $s \in S(i)$ if $\text{opt} = \min$ and $f(i, s^*) \geq f(i, s)$ for all $s \in S(i)$ if $\text{opt} = \max$.

△

The term *optimal solution* is used as a synonym for the term *exact solution*.

Because we consider optimization problems where instances are not known at the outset but disclosed over time in an instance revelation process, Definition 2.5 exhibits two major shortcomings:

- It does not account for the sequentiality in the instance revelation process that any solution method has to obey.
- It disregards that the instance revelation process may depend on previous (partial) answers given by the solution method.

We introduce the instance revelation rule as a mechanism to account for dynamic aspects in the revelation process of an instance.

Definition 2.7 (Instance revelation rule).
An instance revelation rule is a rule that governs the temporal course of events in the release of information on the problem instance. △

The dynamic disclosure of an instance of an optimization problem is respected in the following definition by associating a sequence of input elements and the instance revelation rule itself to the instance.

Definition 2.8 (Instance of an optimization problem)**.**
An instance of an optimization problem consists of a set of parameter values including a sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ and an instance revelation rule r. $\triangle$

The sequence σ in an instance of an optimization problem is called input sequence; its elements $\sigma_1, \sigma_2, \ldots$ are called input elements. We say that the elements of σ await processing by some solution method. Once an input element has been processed, it is considered finished. We give three examples of general nature for an instance revelation rule:

- σ_{i+1} with $i = 1, 2, \ldots$ is revealed when σ_i is considered finished.
- $\sigma_1, \sigma_2, \ldots$ are revealed at prescribed release times $\tau_1, \tau_2, \ldots$
- σ is known completely at time 0.

Observe that choosing different instance revelation rules r and r' on the same input sequence σ establishes two different problem instances. Since different instance revelation rules may be used for the same problem, we need a possibility to settle all sources of unclarity with respect to the dynamic processing of the input elements which may inherently arise by the introduction of lookahead. To this end, we associate a set of rules with a problem Π.

Definition 2.9 (Rule set)**.**
A rule set of a problem is a set of restrictions on the solution to an instance of the problem. $\triangle$

Observe that choosing two different rule sets P and P' establishes two different problems. We list three examples which may appear as elements of a rule set. Note that the first rule cannot be used in conjunction with the second or third rule, respectively.

- σ_i with $i = 1, 2, \ldots$ has to be finished before σ_j with $j > i$ can be finished.
- The finishing order of the input elements in σ is arbitrary.
- At most $m \in \mathbb{N}$ input elements with $m > 1$ can be finished at the same time.

The instance revelation rule and the rule set allow us to make a clear distinction between the informational implications caused by lookahead and the consequences on processing of the input elements inherent to lookahead. Regrettably, except for [155], none of the existing literature on online optimization in conjunction with lookahead is concerned with this kind of split of lookahead into forwarded information and implied processing requirements. As a result, it is not clear which of the mechanisms is responsible for improvements when additional lookahead is granted.

Whenever we want to make the rule set P of an optimization problem Π explicit, we may write Π_P instead of Π and provide a specification of P; whenever we want to make the instance revelation process of an instance explicit, we specify the input sequence σ and the revelation rule r in a pair (σ, r) along with the parameter values of the problem instance.

Example 2.10 (Paging)**.**
Consider the paging problem introduced in Chapter 1.2 with page alphabet $\mathcal{A}$, cache size k, an initially filled cache, and denote by $C = \{c_1, c_2, \ldots, c_k\}$ a cache configuration, i.e., a set containing the k cache pages. We identify the elements of $\Pi_P = (I, S, f, \text{opt})$.

Rule set P may contain the following restrictions:

- Elements from $\sigma = (\sigma_1, \sigma_2, \ldots)$ are requested in ascending order of their index.
- Only one non-cache page can be brought into the cache at a time.
- A requested non-cache page has to be brought into the cache as soon as possible.

I is the set of all instances $i = (\sigma, r)$ where $\sigma = (\sigma_1, \sigma_2, \ldots)$ with $\sigma_i \in \mathcal{A}$ for $i = 1, 2, \ldots$ is the sequence of requested pages and r is an instance revelation rule such as:

- All elements of σ are known at the outset (offline optimization problem).
- Elements of σ become available one after another in equidistant intervals (online optimization problem with independent release).
- One input element is known at a time; the next input element is revealed when the previous one has been brought into the cache (online optimization problem with processing-dependent release).
- Exactly $k \in \mathbb{N}$ input elements are known at a time; a new input element is revealed when a known one has been brought into the cache (online optimization problem with lookahead and processing-dependent release).

The set of solution candidates S contains all sequences of cache configurations which comply with r and P, i.e.,

$$S = \Big\{(C_1, C_2, \ldots) \,\big|\, C_i \text{ for } i = 1, 2, \ldots \text{ respects } r \text{ and } P\Big\}.$$

The objective function

$$f = \sum_{j>1} |C_j \backslash C_{j-1}|$$

maps a sequence of cache configurations to the total number of page evictions. opt = min demands to minimize the total number of page evictions. $\Diamond$

2.1.1 Offline Optimization

The key characteristic of an offline optimization problem is that there is no uncertainty about any of its instances.

Definition 2.11 (Offline optimization problem)**.**
An offline optimization problem is an optimization problem where in each instance the input sequence is known at time 0. △

Example 2.12 (Offline paging)**.**
Paging is a sequential problem by nature, i.e., although all page requests are known at the outset, it is forbidden to permute them when bringing them into the cache. This is reflected by the specification of rule set $P = \{p_1, p_2, p_3\}$ with

- $p_1 :=$ σ_i has to be brought into the cache before σ_j if $i < j$,
- $p_2 :=$ Two successive cache configurations have to differ in exactly one page, and
- $p_3 :=$ A page has to be brought into the cache as soon as it is available.

◊

2.1.2 Online Optimization

The key characteristic of an online optimization problem is that there is uncertainty with respect to the input sequence in at least one instance of that problem.

Definition 2.13 (Online optimization problem)**.**
An online optimization problem is an optimization problem where at least one instance exists for which the input sequence is not known completely at time 0. △

We next specify two refinements of online optimization problems frequently addressed in literature ([85], [116], [157]). In the sequential model of online optimization, new input elements are revealed in a processing-dependent fashion.

Definition 2.14 (Online optimization problem in the sequential model)**.**
An online optimization problem in the sequential model is an optimization problem where any instance of the problem has an instance revelation rule which says that a new input element only becomes known when another one has finished processing. △

In the time stamp model of online optimization, input elements are released by an independent external input element generator irrespective of any processing. The release times correspond to time stamps. In contrast to the sequential model, input elements can be accumulated in the time stamp model.

Definition 2.15 (Online optimization problem in the time stamp model)**.**
An online optimization problem in the time stamp model is an optimization problem where any instance of the problem has an instance revelation rule which assigns each input element an independent release time. △

Example 2.16 (Online paging)**.**
Online paging is classically understood in the sequential model. Hence, we have for each instance an instance revelation rule in the form

$$\begin{aligned} r := {} & \text{At time 0, only } \sigma_1 \text{ is seen; the next page of } \sigma \text{ is revealed when the} \\ & \text{currently known unprocessed page is in the cache.} \end{aligned}$$

Concerning rule set P, we can drop rules p_1 and p_2 from the offline problem in Example 2.12 because they are implied automatically as a consequence of r and we obtain

$$P := \{\text{A page has to be brought into the cache as soon as it is available}\}.$$

In the (barely known) time stamp model of paging, page requests pop up independently at their release times and the restriction of sequential processing is dropped. Hence, we have another instance revelation rule, namely

$$r := \text{A page becomes available when its release time is reached.}$$

We demand page evictions after time intervals of a given length have elapsed by rule set

$$P := \{\text{The cache is updated after a given amount of time has elapsed}\}.$$

In implementations, we have to think of resolution strategies for the case of too many page arrivals in a short period of time (buffer overflow). ◊

2.1.3 Online Optimization with Lookahead

Online optimization problems with lookahead *are* online optimization problems. Their separate discussion results from the significant research question of how much the outcome in an online optimization problem can be enhanced through the provision of input elements at an earlier point in time. Since lookahead makes information available earlier, an equivalent approach would be to speak of preponed or forwarded information release.

Introducing lookahead in an online optimization problem amounts to exchanging the instance revelation rule of each instance with another one that has enhanced lookahead capabilities. Hence, we have to view any instance with lookahead in the light of a reference instance without lookahead. We further recognize that by making parts of the instance known earlier, new possibilities with regard to input element processing may arise as stated in the rule set.

We establish the notion of an instance revelation rule substitution in order to manifest reference to another online optimization problem.

Definition 2.17 (Instance revelation rule substitution).
An instance revelation rule substitution $r \to r'$ of instance revelation rule r with instance revelation rule r' transforms any instance of an optimization problem containing r into the same instance with r replaced by r'. $\triangle$

If there is a functional relation between the input element release times in the original and those in the transformed instance, we can refine the abstract concept of an instance revelation rule substitution by the specification of a function which gives for each input element σ_i the amount of time by which σ_i becomes known earlier under lookahead than in the reference instance without lookahead.

Definition 2.18 (Lookahead operator).
A lookahead operator is a function $L : \mathbb{N} \to \mathbb{R}^{\geq 0}$ where $L(i)$ gives the amount of time that input element σ_i of an instance containing input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ becomes known earlier through lookahead. $\triangle$

Lookahead is worthless for $L \equiv 0$ and almighty (with respect to the resource information) for $L \equiv \tau$ where τ represents the vector of release times of the input elements in the reference instance.

Apart from transforming problem instances by changing their instance revelation rule, lookahead may impose different restrictions on how to treat the elements of σ.

Definition 2.19 (Rule set substitution).
A rule set substitution $P \to P'$ of rule set P with rule set P' transforms optimization problem $\Pi_P = (I, S, f, \text{opt})$ into optimization problem $\Pi_{P'} = (I, S', f, \text{opt})$ where for $i \in I$ any $s \in S(i)$ satisfies P and any $s' \in S'(i)$ satisfies P'. $\triangle$

Essentially, as a result of a rule set substitution, the feasible sets of Π_P and $\Pi_{P'}$ no longer coincide because P' imposes different requirements on a solution than P.

We are in a position to give a definition of lookahead in online optimization:

Definition 2.20 (Lookahead).
A lookahead is a pair $(r \to r', P \to P')$ consisting of an instance revelation rule substitution $r \to r'$ and a rule set substitution $P \to P'$. $\triangle$

The following definition accounts for the transformation of a given reference online optimization problem induced by lookahead: On the one hand, transformation affects problem instances due to the instance revelation rule substitution; on the other hand, problem restrictions may be updated as a result of the rule set substitution.

Definition 2.21 (Online optimization problem with lookahead).
An online optimization problem with lookahead is an online optimization problem which is obtained from a reference online optimization problem by applying the instance revelation rule substitution and the rule set substitution of a given lookahead to the instances and to the feasible set of the reference online optimization problem, respectively. $\triangle$

Figure 2.1 illustrates the decomposition of lookahead into an informational and a processual component.

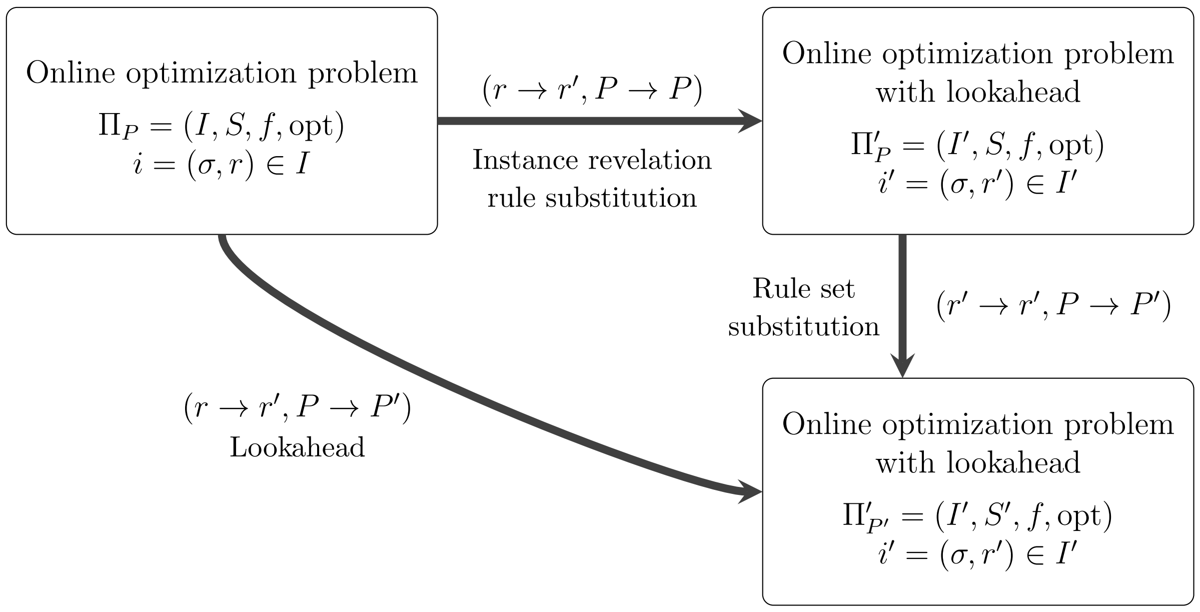

Online optimization problem
$\Pi_P = (I, S, f, \text{opt})$
$i = (\sigma, r) \in I$
$(r \to r', P \to P)$
Instance revelation rule substitution
Online optimization problem with lookahead
$\Pi'_P = (I', S, f, \text{opt})$
$i' = (\sigma, r') \in I'$
Rule set substitution
$(r' \to r', P \to P')$
$(r \to r', P \to P')$
Lookahead
Online optimization problem with lookahead
$\Pi'_{P'} = (I', S', f, \text{opt})$
$i' = (\sigma, r') \in I'$

Figure 2.1: Change of problem instances and problem through application of lookahead.

Example 2.22 (Online paging with lookahead).
First, recall the instance revelation rule in the reference online optimization problem from Example 2.16 as

$r :=$ At time 0, only σ_1 is seen; the next page of σ is revealed when the currently known unprocessed page is in the cache.

Consider now lookahead where essentially a fixed number k of pages is known at each time, but pages are still required to be processed in their order of release (request lookahead without permutation); we have instance revelation rule

$r' :=$ At time 0, $\sigma_1, \dots, \sigma_k$ are seen; the next page of σ is revealed when the currently known unprocessed page with smallest index is in the cache.

Rule set P of the reference online optimization problem remains valid after introduction of lookahead, i.e., the lookahead is of the form $(r \to r', P \to P)$.

We also give the instance revelation rule for the strong type of lookahead as introduced by Albers ([2]) where one can foresee additionally the next k different pages.

$r' :=$ At time 0, $\sigma_1, \dots, \sigma_{k'}$ with $|\{\sigma_1, \dots, \sigma_{k'}\}| = k + 1$ are seen; whenever a page from the lookahead is served for the last time among the lookahead requests, the remaining pages of σ are revealed such that there are $k + 1$ pairwise different pages in the lookahead again.

$\Diamond$

Because of the absence of a neutral performance benchmark in any optimization problem under incomplete information (in contrast to the existence of exact solutions in offline optimization), we recognize that we cannot derive any statement about the value of lookahead in online optimization without specifying the solution method on which any of these statements would depend.

2.2 Algorithm Analysis

Since we cannot expect a given instance of a problem to be solved by mere intuition or clairvoyance, we are in need of a structured computational method for solution retrieval.

Definition 2.23 (Algorithm ([54], [77])).
An algorithm is a list of computational statements designed to find an adequate answer for any instance of a given problem. △

Algorithms take a set of values as initial input, optionally collect values as additional input during execution of the statements and produce another set of values as output. In optimization problems, algorithms return an element from the set of solutions.

Definition 2.24 ((Exact) algorithm for an optimization problem ([12], [54])).
Let $\Pi = (I, S, f, \text{opt})$ be an optimization problem.

a) An algorithm is a function $\text{ALG} : I \to S$ which assigns every $i \in I$ a solution $s_{\text{ALG}}(i) \in S(i)$.

b) An exact algorithm is a function $\text{ALG} : I \to S$ which assigns every $i \in I$ an exact solution $s_{\text{ALG}}(i) \in S(i)$.

△

The term *optimal algorithm* is used as a synonym for the term *exact algorithm*.

For problems under incomplete information, exact algorithms do not exist by nature; for hard offline optimization problems, they may exceed available computing resources on some instances. We conclude that in either situation, we have to be satisfied with algorithms which produce possibly good solutions in favor of the decision maker. Moreover, data in applications is often fuzzy or inaccurate, rendering the usage of exact algorithms inexpedient.

The two main pillars in the field of algorithms are design and analysis. We are concerned with the latter since we intend to decide for a given set of algorithm candidates which one is most promising. Concerning algorithms for online optimization problems under lookahead, the resource information represents the biggest asset of an algorithm, and we seek to analyze the influence of this resource on obtainable solution quality.

2.2.1 Complexity of Problems and Algorithms

Motivated by the resource boundedness of computers, complexity analysis of *algorithms* opts at investigating resource utilization such as run time, memory or bandwidth consumption of algorithms. The primary yardstick for algorithm complexity is run time which stems from the desire to measure efficiency in terms of speed. Transferring the tractability of problems through (known) algorithms, *problems* are also categorized in complexity classes.

The underlying computational model of the following discussion is the unit-cost random access machine where only the number of elementary steps is counted ([54]): Each elementary step takes $O(1)$ time independent of the length of involved operands. Elementary steps comprise arithmetic, data and control operations. Memory hierarchies and multi-core processors are neglected in this model of computation as well. However, it gives suitable run time predictions of algorithms on computers. We recall that a polynomial-time algorithm for a problem is an algorithm which terminates for any instance of size $n \in \mathbb{N}$ in $O(n^k)$ time with $k \in \mathbb{N}_0$.

In order to organize optimization problems in complexity classes, we have to make one step back and consider a class of problems related to optimization problems.

Definition 2.25 (Decision problem ([77])).
A decision problem is a problem where the answer to each instance is either *yes* or *no*. $\triangle$

Any optimization problem $\Pi = (I, S, f, \mathrm{opt})$ can be associated to a corresponding decision problem by equipping it with a bound $b \in \mathbb{R}$ on the objective value and asking for instance $i \in I$ whether $s \in S(i)$ with $f(s) \leq b$ if $\mathrm{opt} = \mathrm{min}$ or $f(s) \geq b$ if $\mathrm{opt} = \mathrm{max}$ exists. It follows that an optimization problem is at least as hard as the related decision version. Fortunately, decision problems can be organized in complexity classes.

Definition 2.26 (Complexity class $\mathcal{P}$ ([77])).
The complexity class $\mathcal{P}$ consists of all decision problems for which an algorithm exists which determines the correct solution to any instance in polynomial time. $\triangle$

Unfortunately, for plenty of decision problems no polynomial-time algorithm has been found until today. In order to admit a classification of these problems with respect to their tractability as well, we merely demand that verification of an answer to a problem instance is computationally tractable when a certificate for the answer is provided. Verification is accomplished by a verification algorithm which outputs *yes* upon receival of an instance and a certificate if and only if the instance is indeed a *yes*-instance. Because it is unclear how many of these certificates need to be supplied until a *yes*-instance is encountered, the $\mathcal{N}$ in the following definition stands for non-deterministic.

Definition 2.27 (Complexity class $\mathcal{NP}$ ([77])).
The complexity class $\mathcal{NP}$ consists of all decision problems for which any *yes*-answer along with a certificate can be verified as a *yes*-instance in polynomial time. $\triangle$

The majority of the decision versions of real world computational problems belongs to the hardest problems in $\mathcal{NP}$. To characterize them, we need polynomial-time reducibility.

Definition 2.28 (Polynomial-time reducibility ([77])).
A decision problem Π' is called polynomial-time reducible to decision problem Π if there is a polynomial-time algorithm with respect to the size of $i' \in I_{\Pi'}$ such that

1. $i' \in I_{\Pi'}$ is transformed into $i \in I_{\Pi}$,
2. i' is a *yes*-instance if and only if i is a *yes*-instance.

△

Clearly, any algorithm for Π can be used as a subroutine for Π'. If there is no polynomial-time algorithm for Π', then there cannot be a polynomial-time algorithm for Π since otherwise the polynomial-time algorithm for Π could be used for Π' along with polynomial-time reducibility.

Definition 2.29 (Complexity class $\mathcal{NP}$-complete ([77])).
The complexity class $\mathcal{NP}$-complete consists of all decision problems Π which fulfill the following two properties:

1. $\Pi \in \mathcal{NP}$,
2. Π' is polynomial-time reducible to Π for any $\Pi' \in \mathcal{NP}$.

△

It follows that if one $\mathcal{NP}$-complete problem can be solved in polynomial time, then all problems in $\mathcal{NP}$ can be solved in polynomial time, i.e., $\mathcal{P} = \mathcal{NP}$; also, if for any $\mathcal{NP}$-complete problem no polynomial-time algorithm exists, then there is no such algorithm for all $\mathcal{NP}$-complete problems. This is why $\mathcal{NP}$-complete problems are considered the hardest problems in $\mathcal{NP}$. Starting with the satisfiability problem ([52]), the existence of numerous $\mathcal{NP}$-complete problems could be established. Recalling that $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{NP}$-complete are defined only for decision problems, we extend the classification to general problems next. For polynomial-time reductions between different problem classes, we refer the reader to [12].

Definition 2.30 (Complexity class $\mathcal{NP}$-hard ([77])).
The complexity class $\mathcal{NP}$-hard consists of all problems Π for which it holds that any $\Pi' \in \mathcal{NP}$ is polynomial-time reducible to Π. △

If already the decision problem version of an optimization problem is $\mathcal{NP}$-complete, then the optimization problem itself has to be at least as hard as all problems in $\mathcal{NP}$. Figure 2.2 summarizes the relations between the complexity classes.

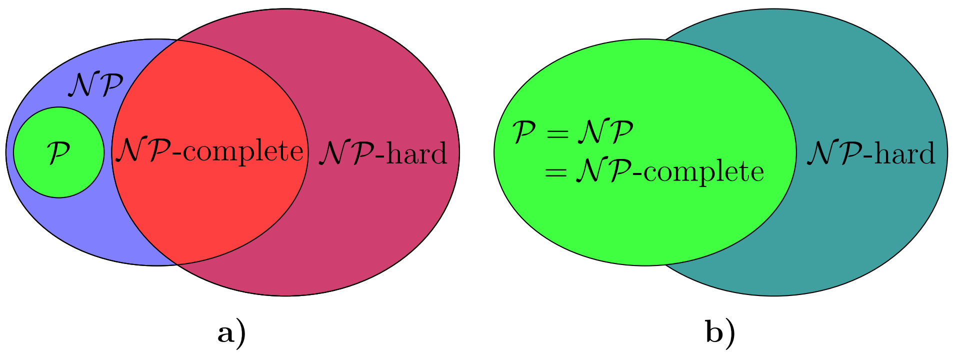


Figure 2.2: Relations between complexity classes. **a)** $\mathcal{P} \neq \mathcal{NP}$. **b)** $\mathcal{P} = \mathcal{NP}$.

The extensive list of $\mathcal{NP}$-complete problems in [77] including settings from graph theory, network design, partitioning, storage and retrieval, and mathematical programming contains numerous elementary problems which can be identified as subproblems in thousands of practical applications, making them intrinsically difficult. Although the vast majority of known algorithms for $\mathcal{NP}$-hard problems run in exponential time and we cannot fare better than super-polynomial (given $\mathcal{P} \neq \mathcal{NP}$), for some problems faster exact algorithms are known than for others. Over the last years, researchers have fostered interest in determining the exact complexity of $\mathcal{NP}$-hard problems to distinguish between different exponential run-time behaviors of algorithms. We refer the interested reader to the survey of Woeginger ([160]).

Complexity classes for optimization problems are established in relation to $\mathcal{P}$ and $\mathcal{NP}$. $\mathcal{NPO}$ comprises all optimization problems whose decision version is in $\mathcal{NP}$; $\mathcal{PO}$ consists of all $\mathcal{NPO}$ problems which can be solved exactly in polynomial time.

Definition 2.31 (Complexity class $\mathcal{NPO}$ ([12]))**.**
The complexity class $\mathcal{NPO}$ consists of all optimization problems $\Pi = (I, S, f, \mathrm{opt})$ which fulfill the following properties:

1. $i \in I$ is recognizable as an instance of the problem in polynomial time.
2. For any $i \in I$, it is decidable in polynomial time whether a given $s \in S(i)$, and the size of any $s \in S(i)$ is bounded by $O(|s|^k)$ for some $k \in \mathbb{N}_0$.
3. f is computable in polynomial time.

$\triangle$

Definition 2.32 (Complexity class $\mathcal{PO}$ ([12]))**.**
The complexity class $\mathcal{PO}$ consists of all optimization problems $\Pi = (I, S, f, \mathrm{opt}) \in \mathcal{NPO}$ such that there exists an exact polynomial-time algorithm for Π. $\triangle$

In the sequel, we discuss how complexity arises in online optimization problems. Due to successive instance revelation, no optimization algorithm can be exact and not even an exponential-time algorithm with infinite capacities is able to determine an exact solution. Thus, any online optimization problem is $\mathcal{NP}$-hard. Yet, because solving an instance of an online optimization problem comprises solving a series of subproblems, we have to be more concerned about the complexity of the subproblems.

Whenever lookahead implies that the informational preview in terms of future input elements is bounded, subproblems occurring over time can be solved in fixed polynomial time. Whenever lookahead admits an arbitrary number of input elements to be foreseen, the sizes of the subproblems may become arbitrarily large such that their complexity equals that of the offline variant of the subproblem. Irrespective of a polynomial-time upper bound on run time, solving medium to large scale instances of $\mathcal{NP}$-hard optimization problems represents a computational burden for any online procedure in case of real-time requirements.

Since future information is uncertain anyway, finding exact solutions to the subproblems does not automatically imply better solutions for the overall problem. We will see in Chapters 5 and 6 that there are counterexample instances in some problems where exact reoptimization turns out disadvantageous due to myopic decision making based on current data because future (currently unknown) data is likely to necessitate deviating from once computed plans.

We remark that it is not clear (i.e., it is not exogenously given) how optimization goal and constraints have to look like in the subproblems. One expects them to coincide with those of the overall problem, but perhaps it may be more advantageous to include additional or different criteria and restrictions in the subproblems (see also Chapters 5.3, 5.4 and 7.2).

2.2.2 Classification of Optimization Algorithms

Besides classifying algorithms according to their run time, we have to account for solution quality when dealing with optimization algorithms. Among the set of algorithms which do not necessarily provide an exact solution to an optimization problem, we distinguish between different aspiration levels ([12], [54]). To this end, let $s_{\text{ALG}}(i) \in S(i)$ be the solution computed by algorithm ALG for instance $i \in I$ in an optimization problem $\Pi = (I, S, f, \text{opt})$.

Approximation algorithms find a solution to an *offline* optimization problem with provable solution quality. If opt = min, ALG is called c-approximative if for all $i \in I$ it holds that

$$f(i, s_{\text{ALG}}(i)) \leq c \cdot \min_{s \in S(i)} f(i, s).$$

If opt = max, Alg is called a c-approximation algorithm if for all $i \in I$ it holds that

$$c \cdot f(i, s_{\text{Alg}}(i)) \geq \max_{s \in S(i)} f(i, s).$$

Approximation algorithms[5] are used when the limiting resource is computing time.

Competitive algorithms find a solution to an *online* optimization problem with provable solution quality with respect to a hypothesized offline algorithm. If opt = min, Alg is called c-competitive if for all $i \in I$ it holds that

$$f(i, s_{\text{Alg}}(i)) \leq c \cdot \min_{s \in S(i)} f(i, s).$$

If opt = max, Alg is called c-competitive if for all $i \in I$ it holds that

$$c \cdot f(i, s_{\text{Alg}}(i)) \geq \max_{s \in S(i)} f(i, s).$$

The limiting resource is scarce information ([32]) and not computing time.

Heuristics find a good, but not necessarily optimal solution to an (offline or online) optimization problem without assuring any guarantee of solution quality. Nonetheless, computational experiments show that a large number of heuristic algorithms perform considerably well for instances of practical problems.

Example 2.33 (Algorithms in a practical problem)**.**
Consider a repair service which has to send a technician to customers upon their request. To save costs, the optimization problem consists of selecting a customer order that causes a minimum length of the service technician's tour starting and ending at the company building. How can the problem be tackled? Realizing that at each day the company faces an online version of the $\mathcal{NP}$-hard traveling salesman problem (TSP), there are several possibilities:

- The company ignores that customer requests arrive online and puts their requests off to be served at the next day. This day-ahead scheduling approach amounts to solving instances of the offline TSP over night such that customers will be served according to the obtained plan the next day. If the company possesses two computers, it can try to solve a mathematical programming formulation ([19], [42]) of the instance with an exact algorithm on one computer, whilst on the other it uses Christofides' $\frac{3}{2}$-approximation

[5] The definitions for opt $\in \{\min, \max\}$ can be collapsed into one by $\max\left\{\frac{f(i,s_{\text{Alg}}(i))}{f(i,s^*)}, \frac{f(i,s^*)}{f(i,s_{\text{Alg}}(i))}\right\} \leq c$ with exact solution s^* on any instance i; the maximum is attained by the first term in a minimization problem and by the second term in a maximization problem.

algorithm ([44]) for the TSP or the k-opt-heuristic with $k \in \{2, 3, 4\}$ ([124]) which is known to produce sufficiently good results for practical needs.

- Arriving customer requests are served in an online manner. Suitable objectives are to minimize the makespan, i.e., the time to finish all requests, or to minimize the latency, i.e., the sum of all completion times of the customers. For minimizing the makespan a 2-competitive algorithm PlanAtHome is known ([13]), while for minimizing the latency only a 6-competitive algorithm Interval is available ([117]). We mention two generic algorithms for solving online optimization problems ([116]): Replan computes an optimal tour for all known requests whenever a new request arrives and follows this tour until either it returns to the origin or a new request arrives; Ignore computes an optimal tour for all known requests and follows this tour until it returns to the origin where it computes a new optimal tour for those requests that have arrived in the meantime. Both algorithms are $\frac{5}{2}$-competitive for minimizing the makespan.
- Customer requests are assumed to pop up some time ahead of earliest possible visit (e.g., due to contractually prescribed durations between service request and delivery), giving an algorithm a form of time lookahead. It is shown in [7] that lookahead leaves the competitive factor at 2 which means that no better algorithm with respect to the optimal offline solution is found in the competitive analysis framework.

$\diamondsuit$

2.2.3 Algorithms and Lookahead

With algorithms as solution procedures for online optimization problems, we are led to the question of what can be achieved by them upon provision of additional lookahead. For most problem instances, lookahead is expected to change the instance and the underlying rules in favor of the decision maker. Yet, there are situations where decisions of an algorithm based on lookahead are worse than those that would have been taken without lookahead. To quantify the impact of lookahead in terms of a change in the objective value, we introduce the lookahead value which we further decompose into two components according to the instance revelation rule and rule set substitutions of a lookahead.

Definition 2.34 (Lookahead value).
Let $\Pi_P = (I, S, f, \text{opt})$ and $\Pi_{P'} = (I', S', f, \text{opt})$ be online optimization problems where the instances $i' \in I'$ result from applying lookahead $(r \to r', P \to P')$ to the instances $i \in I$, and let $s_{\text{ALG}} \in S(i)$ and $s_{\text{ALG}'} \in S'(i')$ be the solution candidates obtained by Alg and Alg$'$, respectively.

- If opt = max, then
$$\Delta f^{r,r',P,P'}_{\text{ALG},\text{ALG}'}(i) := f(i', s_{\text{ALG}'}) - f(i, s_{\text{ALG}})$$
is called the lookahead value of $(r \to r', P \to P')$ on input sequence i with respect to algorithm pair $(\text{ALG}, \text{ALG}')$.

- If opt = min, then
$$\Delta f^{r,r',P,P'}_{\text{ALG},\text{ALG}'}(i) := f(i, s_{\text{ALG}}) - f(i', s_{\text{ALG}'})$$
is called the lookahead value of $(r \to r', P \to P')$ on i with respect to algorithm pair $(\text{ALG}, \text{ALG}')$.

△

We artificially decompose the lookahead value to account for the partial improvements due to the instance revelation rule substitution and due to the rule set substitution, respectively. The decomposition is artificial because it relies on the members of the set $\mathcal{ALG}$ of admissible algorithms for problems that have to operate under rule set P.

Definition 2.35 (Partial lookahead value due to instance revelation rule substitution).
Let $\Pi_P = (I, S, f, \text{opt})$ be an online optimization problem, let ALG be an algorithm for Π_P, and let $\mathcal{ALG}$ be a set of admissible algorithms for Π_P. Further, let i'' be the instance which results from applying lookahead $(r \to r', P \to P')$ to instance $i \in I$, and let $s_{\text{ALG}} \in S(i)$ and $s_{\text{ALG}''} \in S(i'')$ be the solutions obtained by ALG and $\text{ALG}'' \in \mathcal{ALG}$, respectively.

- If opt = max, then
$$\Delta f^{r,r'}_{\text{ALG}}(i) := \max_{\text{ALG}'' \in \mathcal{ALG}} \big\{ f(i'', s_{\text{ALG}''}) \big\} - f(i, s_{\text{ALG}})$$
is called the partial lookahead value of $(r \to r', P \to P')$ due to instance revelation rule substitution $r \to r'$ on i with respect to ALG and $\mathcal{ALG}$.

- If opt = min, then
$$\Delta f^{r,r'}_{\text{ALG}}(i) := f(i, s_{\text{ALG}}) - \min_{\text{ALG}'' \in \mathcal{ALG}} \big\{ f(i'', s_{\text{ALG}''}) \big\}$$
is called the partial lookahead value of $(r \to r', P \to P')$ due to instance revelation rule substitution $r \to r'$ on i with respect to ALG and $\mathcal{ALG}$.

△

Note that in the previous definition $\textsc{Alg}''$ has to operate under P although the lookahead $(r \to r', P \to P')$ also comprises a rule set substitution to P'. However, to determine the partial lookahead value attributable to the instance revelation rule substitution, we have to maintain algorithm processing under rule set P. The value of $\Delta f^{r,r'}_{\textsc{Alg}}(i)$ specifies the absolute improvement attainable for a given set $\mathcal{ALG}$ of algorithm candidates operating under P.

Definition 2.36 (Partial lookahead value due to rule set substitution).
Let $\Pi_P = (I, S, f, \text{opt})$ and $\Pi_{P'} = (I', S', f, \text{opt})$ be online optimization problems, let $\textsc{Alg}$ and $\textsc{Alg}'$ be algorithms for Π_P and $\Pi_{P'}$, respectively. Further, let i' be the instance which results from applying instance revelation rule substitution $(r \to r')$ to instance $i \in I$, and let $s_{\textsc{Alg}'} \in S(i')$ be the solution obtained by $\textsc{Alg}'$. The value

$$\Delta f^{P,P'}_{\textsc{Alg},\textsc{Alg}'}(i) := \Delta f^{r,r',P,P'}_{\textsc{Alg},\textsc{Alg}'}(i) - \Delta f^{r,r'}_{\textsc{Alg}}(i)$$

is called the partial lookahead value of $(r \to r', P \to P')$ due to rule set substitution $P \to P'$ on i with respect to algorithm pair $(\textsc{Alg}, \textsc{Alg}')$. △

By definition, the lookahead value is decomposed into the two components, i.e., for lookahead $(r \to r', P \to P')$ it holds that

$$\Delta f^{r,r',P,P'}_{\textsc{Alg},\textsc{Alg}'}(i) = \Delta f^{r,r'}_{\textsc{Alg}}(i) + \Delta f^{P,P'}_{\textsc{Alg},\textsc{Alg}'}(i).$$

It deserves mentioning that the partial lookahead value due to rule set substitution implicitly presumes that additional information is made known earlier. Hence, it can be seen as the part of the lookahead effect that could not be elicited just by making use of the additional information. In a similar form, [155] considers lookahead to be either a means of information preview or a means of selection.

The following two examples show that – depending on the application – both partial lookahead values can be prevalent.

Example 2.37 (Online bin packing with lookahead).
Consider bin packing with two item sizes $\{0.4, 0.6\}$ and item sequence $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (0.4, 0.4, 0.6, 0.6)$. For the pure online version, we have instance revelation rule r as

$$r := \sigma_1 \text{ is known at the beginning; } \sigma_{i+1} \text{ is revealed after } \sigma_i \text{ has been assigned}$$

and rule set P as

$$P := \{\sigma_i \text{ has to be assigned before } \sigma_{i+1}\}.$$

For the lookahead version, we assume request lookahead of size 2, i.e., we have instance revelation rule r' as

$$r' := \sigma_1, \sigma_2 \text{ are known at time 0; an item is revealed when another one has been assigned}$$

and rule set P' as

$$P' := \{\text{Any previously revealed item can be assigned when it has not yet been assigned}\}.$$

Let instances i and i' correspond to σ as revealed under r and r', respectively. We use algorithm BestFit (Bf) in the pure online setting and algorithm BestFitDecreasing (Bfd) in the setting with lookahead. Bf puts the next item into the fullest bin available, whereas Bfd first sorts the known unassigned items by non-increasing sizes and assigns the largest of them to the fullest bin available. Both algorithms comply with rule sets P and P', respectively. Moreover, we let intermediary algorithm BestFitModified (Bfm) operate under rule set P, i.e., it puts the items into the bins in their order of appearance. However, Bfm bases its decision on all available information: When Bfm is supplied with information according to r' it operates identically to Bf except for the case where an open bin at level 0.4 exists and two unassigned items with sizes 0.4 and 0.6 have to be assigned in this order (as required by P). In this case, the item of size 0.4 is put in a new bin and the item of size 0.6 is packed afterwards in one of the two bins at level 0.4.

Applying Bf yields one bin at level 0.8 and two bins at level 0.6, i.e., $s_{\text{Bf}}(i) = \big((0.4, 0.4), (0.6), (0.6)\big)$ and $f(i, s_{\text{Bf}}(i)) = 3$. Applying Bfd yields two fully laden bins, i.e., $s_{\text{Bfd}}(i') = \big((0.4, 0.6), (0.6, 0.4)\big)$ and $f(i', s_{\text{Bfd}}(i')) = 2$. We therefore observe a lookahead value of $\Delta f^{r,r',P,P'}_{\text{Bf},\text{Bfd}}(i) = 3 - 2 = 1$. Applying Bfm yields $s_{\text{Bfm}}(i') = \big((0.4, 0.6), (0.4, 0.6)\big)$ in compliance with P and $f(i', s_{\text{Bfm}(i')}) = 2$. Moreover, $f(i', s) \geq 2$ for any solution $s \in S(i')$. Thus, we have a lookahead value due to instance revelation rule substitution of $\Delta f^{r,r'}_{\text{Bf}}(i) = 3-2 = 1$ on i. There is no lookahead value due to rule set substitution because $\Delta f^{P,P'}_{\text{Bf},\text{Bfd}}(i) = 3-2-1 = 0$.

In this example, the improvement is a result of provision of *information* at an earlier point in time; allowing to permute the items is of no value. We can think of this as an algorithm fictively processing lookahead information and reserving spaces according to the lookahead case, whilst factually still adhering to the original release order during item packing. $\diamondsuit$

Example 2.38 (Online traveling salesman with lookahead).
Consider a traveling salesman problem with two locations $\{0, 1\}$ and request sequence $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (0, 1, 0, 1, 0)$. The server starts in 0. For the pure online version, we have

instance revelation rule r as

$$r = \sigma_1 \text{ is known at the beginning; } \sigma_{i+1} \text{ is revealed after } \sigma_i \text{ has been visited}$$

and rule set P as

$$P = \{\sigma_i \text{ has to be visited before } \sigma_{i+1}\}.$$

For the lookahead version, we assume request lookahead of size 2, i.e., we have instance revelation rule r' as

$$r' = \sigma_1, \sigma_2 \text{ are known initially; a request is revealed when another one has been visited}$$

and rule set P' as

$$P' := \{\text{Any previously revealed request can be visited when it has not yet been visited}\}.$$

Let instances i and i' correspond to σ as revealed under r and r', respectively. Note that rule set P' makes some locations visitable at an earlier time, e.g., at time 0, both σ_1 and σ_2 can be visited, whilst under P only σ_1 can be visited. Due to P, we must use algorithm FirstComeFirstServed (Fcfs) in the pure online setting; in the setting with lookahead, we use algorithm NearestNeighbor (Nn). Fcfs visits the requests in their order of release, whereas Nn stays in the current location when this location is contained in the two unvisited known requests. Both algorithms comply with rule sets P and P', respectively. Moreover, there is no algorithm other than Fcfs to operate under P, i.e., we must choose Fcfs as an intermediary algorithm that is supplied with information according to r', but obviously cannot capitalize from it.

Applying Fcfs on i yields the visiting order $s_{\text{Fcfs}}(i) = (0,1,0,1,0)$ and $f(i, s_{\text{Fcfs}}(i)) = 4$. Applying Nn on i' yields the visiting order $s_{\text{Nn}}(i') = (0,0,1,1,0)$ and $f(i', s_{\text{Nn}}) = 2$. We therefore observe a lookahead value of $\Delta f^{r,r',P,P'}_{\text{Fcfs},\text{Nn}}(i) = 4 - 2 = 2$. Applying "intermediary" algorithm Fcfs on i' yields $s_{\text{Fcfs}}(i') = (0,1,0,1,0)$ in compliance with P and $f(i', s_{\text{Fcfs}}) = 4$. Moreover $f(i', s) = 4$ for any solution $s \in S(i')$. Thus, we have a lookahead value due to instance revelation rule substitution of $\Delta f^{r,r'}_{\text{Fcfs}}(i) = 4 - 4 = 0$ on i. The complete lookahead value is made up of the lookahead value due to rule set substitution because $\Delta f^{P,P'}_{\text{Fcfs},\text{Nn}}(i) = 4 - 2 - 0 = 2$.

In this example, the improvement is a result of the *change of circumstances* under which requests have to be visited; allowing to permute the release order of the known requests in the visiting order is responsible for the complete lookahead value and it enables us to get rid of Fcfs which is ultimately susceptible to detours. Without this allowance we could not make use of any additional information no matter at which time it is given. $\Diamond$

2.3 Performance of Optimization Algorithms

There are many different perspectives on assessing algorithm performance, and to our mind, a holistic judgement can only be obtained by intermixing them: Worst-case analysis gives strong worst-case guarantees of algorithm quality, but lacks in displaying the overall behavior. Average-case analysis addresses an algorithm's overall behavior, but it assumes distributions on the instances to be given and provides no guarantee in form of a maximum deviation from an (offline) optimum. Distributional analysis intends to illustrate the spectrum of an algorithm's behavior over all input instances, thereby eliminating weaknesses of worst-case and average-case analysis; yet, distributional results are hard to obtain. We desire the following properties for an ideal performance measure:

Generality The performance measure can be formulated regardless of a specific problem.

Applicability The performance measure can be applied to all types of problem settings.

Representativeness The performance measure allows for a statement about the overall behavior of an algorithm over all instances.

Comparability Algorithms can be compared directly using the performance measure.

Analyzability and computability The performance measure can be expressed analytically or at least calculated in a sample-based method.

It is hardly possible for a performance measure to fulfill all criteria at once. Subsequently, we first study performance measures for online algorithms suggested in literature along with their strengths and weaknesses (Chapter 2.3.1), before we present our approach to performance assessment of algorithms in online optimization that allows for the consideration of different information regimes (Chapter 2.3.2).

2.3.1 Performance Measures for Online Optimization Algorithms

Analyzing the relation between an online algorithm (with absent or present lookahead) and an optimal offline algorithm from a worst-case perspective has become the standard tool, called competitive analysis, for analyzing online algorithms. However, this approach comes along with some notable disadvantages and limitations ([65], [73]):

- Results are overly pessimistic because single worst-case instances, often pathologically construed, decide upon the quality of an algorithm.
- Competitive analysis is oblivious to overall algorithm performance.
- Algorithm performance is reduced to a single, worst-case-related key figure.
- Discriminating between algorithms with equivalent worst-case, but differing average-case behavior is impossible.
- Competing with an omniscient offline algorithm may be irrelevant in practice because only short previews of future information are eligible.
- Direct comparison between two candidate algorithms is impossible.
- Although suggested by computational experiments, competitive analysis often fails to reproduce the beneficial impact of lookahead.
- If the generation of input elements depends on algorithm processing, it is impossible to formulate an offline counterpart of an online optimization problem.
- Competitive analysis is only amenable for academic problems, but not for real world applications.

In order to mitigate these effects, several enhancements to competitive analysis were proposed (see, e.g., the surveys in [38], [65], [91]) such as increasing the power of the online algorithm, reducing the power of the offline algorithm, restricting the instance space, applying alternative objective functions, or randomizing the online processing. Other approaches intend to eliminate the weaknesses of competitive analysis, e.g., by comparing algorithms directly, assuming some distribution for the input sequences, or expanding performance assessment from a single number to distribution functions. Table 2.1 gives a systematical overview of the performance measures found in literature used for the analysis of online optimization algorithms.

To harmonize notation with literature, we will denote the objective value $f(i, s_{\text{ALG}}(i))$ of online algorithm ALG on instance i containing input sequence σ as $\text{ALG}[\sigma] := f(i, s_{\text{ALG}}(i))$; instead of instance set I, we use the set of all input sequences Σ. Moreover, we assume minimization as optimization goal and denote an optimal (hypothesized) offline algorithm by OPT. Except for the competitive ratio, none of the developed alternatives is widely accepted and recognized as a performance measure. The reason for this is that the measures were mainly developed in a problem-specific context in order to unfold some special property of an algorithm that has been observed empirically.

Deterministic worst-case analysis

Competitive analysis and refinements

- Competitive ratio ([80], [32], [99], [100], [104], [149], [150])
- Competitive ratio under resource augmentation (On ↑) ([56], [71], [102], [149])
- Competitive ratio under fair adversaries (Off ↓) ([11], [31])
- Competitive ratio under locality of reference (Off ↓) ([4], [33])
- Loose competitive ratio (Off ↓) ([165], [167])
- Accommodating function (Off ↓) ([37], [39])
- Cooperative ratio ([66])

Other approaches

- Comparative ratio ([115])
- Max/Max ratio ([20])
- Relative worst-order ratio ([35])

Probabilistic worst-case analysis

Competitive analysis and refinements

- Smoothed competitive ratio (Off ↓) ([16], [153])
- (Randomized algorithms (Off ↓) ([21], [32])

Other approaches

- Random order ratio ([108])

Average-case analysis

Competitive analysis and refinements

- Expected competitive ratio ([151])
- Expected performance ratio ([136], [145], [151])
- Expected competitive ratio under diffuse adversaries ([115], [15], [166])

Distributional analysis

Other approaches

- Relative interval analysis ([67])
- Stochastic dominance ([90])
- Bijective analysis ([9], [10])
- Average analysis ([9], [10])
- Counting distribution of objective value ([68])
- Counting distribution of performance ratio ([68])

Table 2.1: Performance measures for algorithms in online optimization problems. Decreased (increased) power of an offline (online) algorithm is denoted by Off ↓ (On ↑).

2.3.1.1 Deterministic Worst-Case Performance Measures

Competitive analysis has become the standard for measuring performance of online algorithms since its advent in the 1970s and 1980s[6]. The idea of competitive analysis is to directly compare the performance of $\textsc{Alg}$ to that of $\textsc{Opt}$. $\textsc{Alg}$ is called c-competitive if there is a constant a such that

$$\textsc{Alg}[\sigma] \leq c \cdot \textsc{Opt}[\sigma] + a, \qquad \sigma \in \Sigma. \tag{2.1}$$

The role of the additive constant a is to facilitate an asymptotic analysis and to make results independent of initial conditions for finite input sequences. In the case $a = 0$, $\textsc{Alg}$ is called strictly c-competitive if

$$\frac{\textsc{Alg}[\sigma]}{\textsc{Opt}[\sigma]} \leq c, \qquad \sigma \in \Sigma. \tag{2.2}$$

A c-competitive algorithm is a c-approximation algorithm with the additional restriction that it has to compute online ([32]). The competitive ratio c_r of $\textsc{Alg}$ is the greatest lower bound over all c such that $\textsc{Alg}$ is c-competitive, i.e.,

$$\begin{aligned} c_r &= \inf\{c \geq 1 \,|\, \textsc{Alg}[\sigma] \leq c \cdot \textsc{Opt}[\sigma] + a, \sigma \in \Sigma\} && (2.3) \\ &= \inf\{c \geq 1 \,|\, \textsc{Alg} \text{ is } c\text{-competitive}\}. && (2.4) \end{aligned}$$

The competitive ratio states how much the performance of $\textsc{Alg}$ degrades with respect to $\textsc{Opt}$ due to the lack of information in the worst-case. If $\textsc{Alg}$ can be shown to be c-competitive, but c cannot be shown to be the competitive ratio because we only have a lower bound $\underline{c}$ for the competitive ratio, there is a competitiveness gap of $c - \underline{c}$ for $\textsc{Alg}$. The competitive ratio's worst-case nature leads to the critique mentioned in the introduction of this section. The first competitive analysis is probably due to Graham ([80]) who analyzed the $\textsc{List}$ algorithm for minimum makespan scheduling on m parallel machines showing that $c_r = 2 - \frac{1}{m}$.

The weakness of $\textsc{Alg}$ compared to $\textsc{Opt}$ can partially be overcome by tuning instance parameters in favor of $\textsc{Alg}$ and leaving them unchanged for $\textsc{Opt}$. Since informational nescience is compensated with additional resources, this is called *resource augmentation*. It was introduced for scheduling ([102]) where machines are allowed to process tasks at a higher speed in the online problem. Generalizations to other problems are apparent: In paging, the cache size is increased ([149]); in bin packing, larger bins are used ([56], [71]). The performance

[6] Graham's guarantee for the $\textsc{List}$ scheduling algorithm ([80]) from 1966 and Johnson's work on approximation algorithms for bin packing ([99], [100]) in the 1970s fostered research interest in worst-case performance measurement of online algorithms. In the mid 1980s, the papers by Sleator and Tarjan ([149], [150]) continued this flourishing interest which led to the term *competitive analysis* in 1988 by Karlin ([104]).

measure depends on the amount ϵ of extra resource that is given to $\textsc{Alg}$ which now operates with resource amount $(1+\epsilon)\cdot 100\%$ (denoted by $\textsc{Alg}_{1+\epsilon}$). $\textsc{Alg}$ is c-competitive[7] under resource augmentation ϵ if it holds that

$$\frac{\textsc{Alg}_{1+\epsilon}[\sigma]}{\textsc{Opt}[\sigma]} \le c, \qquad \sigma \in \Sigma.$$

Similarly to augmenting resources of $\textsc{Alg}$, we can diminish those of $\textsc{Opt}$. A realization of this idea is to restrict the behavior of $\textsc{Opt}$ to acting as a *fair adversary* ([11], [31]) in a sense to be specified in the problem: In the traveling salesman problem, the server has to move within the convex hull of the requests known to $\textsc{Alg}$ so far to prevent cheap capitalizing by moving to a future request which $\textsc{Alg}$ cannot see; a fair adversary in bin packing has to pack items in the same order as $\textsc{Alg}$. Let $\textsc{Opt}'$ be an optimal algorithm for the offline problem under fairness constraints, then $\textsc{Alg}$ is called c-competitive under fair adversaries if Condition 2.1 holds with $\textsc{Opt}$ substituted by $\textsc{Opt}'$.

Restricting instances to typical input patterns encountered in practice narrows the gap between $\textsc{Alg}$'s and $\textsc{Opt}$'s power. An application prototype is paging where it is observed that over a short time only pages from a local neighborhood of the current cache content are referenced (*locality of reference*). Numerous models of locality were proposed ([4], [33]): In the access graph model, a graph G whose vertices correspond to page requests is used; edges are established between two vertices corresponding to two requests when these can be requested consecutively. Any feasible input sequence corresponds to a walk in G. The concave function model ([4]) is based on the idea that the number of distinct pages in a subsequence has to grow slower than linear in the size of the subsequence. This concept coincides with the working set model in memory management ([59], [60]) where a function $f : \mathbb{N} \to \mathbb{R}^{>0}$ is concave if and only if $f(1) = 1$ and $f(n+1) - f(n) \ge f(n+2) - f(n+1)$ for all $n \in \mathbb{N}$. An input sequence in the max-model of concave analysis is admissible if for all subsequences of length n at most $f(n)$ distinct pages are in the subsequence; an input sequence in the average-model of concave analysis is admissible if the average number of distinct pages in all subsequences of length n is at most $f(n)$. Denote the set of input sequences consistent with the chosen model of locality by Σ^{loc}, then $\textsc{Alg}$ is called c-competitive under locality of reference if Condition 2.1 holds with Σ replaced by Σ^{loc}.

Loose competitiveness has been introduced in [165] for two reasons: First, worst-case instances in competitive analysis are often tailored to specific problem parameters. Second, it is arguable whether input sequences with small cost shall be considered in the same way as

[7] Most publications do not differentiate between *competitive* and *strictly competitive*; the case $a = 0$ in Condition 2.1 is often tacitly assumed.

costly ones because of setup and overhead. We pay attention to one problem parameter k, and let $\mathcal{K}$ be the set of possible values for k. $\textsc{Alg}$ is $(\epsilon, \delta, \mathcal{K})$-loosely c-competitive if for any $\sigma \in \Sigma$ at least $(1-\delta) \cdot 100\,\%$ of the $|\mathcal{K}|$ parameter values for k satisfy

$$\textsc{Alg}_k[\sigma] \leq \max\{c \cdot \textsc{Opt}[\sigma], \epsilon \cdot |\sigma|\}$$

where $\textsc{Alg}_k[\sigma]$ is the objective value of $\textsc{Alg}$ on σ when the parameter is k. Hence, input sequences with costs no larger than $\epsilon \cdot |\sigma|$ are ignored for appropriate constant ϵ. Parameter k corresponds to the cache size in paging and to the bin capacity in bin packing. Using loose competitiveness in paging, deviations between empirical performance ratios and the lower bound of the competitive ratio can be explained ([167]).

The *accommodating function* ([37], [39]) is defined for online optimization problems with a limited resource which comes in a default amount, say k, but may be varied. Similar to locality of reference, we restrict input sequences: Let $\textsc{Opt}_k[\sigma]$ denote the costs of $\textsc{Opt}$ on σ when $\textsc{Opt}$ has a resource level of k, then an input sequence σ is called an α-sequence if $\textsc{Opt}_{\alpha k}[\sigma] = \textsc{Opt}_{k'}[\sigma]$ for all $k' \geq \alpha k$. Clearly, for 1-sequences the default amount of the resource is enough for $\textsc{Opt}$ to find the optimal solution. Let Σ^α be the set of all α-sequences, then $\textsc{Alg}$ is c-competitive under α-sequences if Condition 2.1 holds with Σ replaced by Σ^α; $\textsc{Alg}$ has competitive ratio $c_r(\alpha)$ under α-sequences if Condition 2.3 holds with Σ replaced by Σ^α. Plotting $c_r(\alpha)$ over α yields the accommodating function $A(\alpha) = c_r(\alpha)$. $A(1)$ is called accommodating ratio of $\textsc{Alg}$ and it expresses the worst-case ratio on those input sequences which are easy enough for $\textsc{Opt}$ to produce an optimal solution given the default amount of resources. The accommodating function has been applied to resource-constrained problems such as seat reservation, fair bin packing, unrestricted bin packing and paging. The authors claim that only small values $\alpha \geq 1$ should be considered in order to get rid of those input sequences leading to the conventional competitive ratio.

The *cooperative ratio* ([66]) replaces $\textsc{Opt}[\sigma]$ with some modified cost value $\textsc{Opt}'[\sigma]$ that implicitly accounts for the difficulty of σ: On a difficult input sequence, $\textsc{Alg}$ is explicitly allowed to incur a higher cost. Let $\textsc{Opt}' : \Sigma \to \mathbb{R}$ be a function with $\textsc{Opt}'[\sigma] \geq \textsc{Opt}[\sigma]$ for all $\sigma \in \Sigma$, then $\textsc{Alg}$ has cooperative ratio c_r if Condition 2.3 holds with $\textsc{Opt}[\sigma]$ replaced by $\textsc{Opt}'[\sigma]$. The idea is to reduce the impact of badly behaving input sequences on the performance ratio. In applications, difficult instances are expected to be isolated in the input sequence space, e.g., requests in paging are expected to be generated according to some locality property. Hence, the adversary is trusted to cooperate with and not to sabotage $\textsc{Alg}$. It has been shown that the cooperative ratio leads to a better separation between performances of different paging and list update algorithms compared to competitive analysis.

Comparative analysis ([115]) differs from competitive analysis in that the latter relates $\textsc{Alg}$'s objective value to that of $\textsc{Opt}$, whereas the former relates the best objective value of a class of algorithm candidates to that of another class of algorithm candidates which are weaker than $\textsc{Opt}$, but stronger than those in the first class. Let $\mathcal{A}, \mathcal{B}$ be two algorithm classes where $\mathcal{B}$ is more powerful than $\mathcal{A}$, e.g., due to more computational resources or due to more available information, then the comparative ratio c_r^{comp} is defined as

$$c_r^{comp} := \max_{\mathrm{B} \in \mathcal{B}} \min_{\mathrm{A} \in \mathcal{A}} \max_{\sigma \in \Sigma} \frac{\mathrm{A}[\sigma]}{\mathrm{B}[\sigma]}.$$

In order to maximize c_r^{comp}, $\mathcal{B}$ chooses candidate B, whereupon $\mathcal{A}$ answers with candidate A, whereupon B chooses the worst instance $\sigma \in \Sigma$ for A. The idea of comparative analysis is also the foundation of our approach in Chapter 2.3.2: Compare algorithm classes with varying lookahead levels to each other. Its advantages become apparent in the possibility to compare algorithms without reference to $\textsc{Opt}$. Whereas competitive analysis fails to capture the benefit of (request) lookahead of size l in paging with cache size k, it has been shown in [115] that lookahead leads to a comparative ratio of $c_r^{comp} = \min\{k, l+1\}$ as opposed to $c_r = k$ for algorithms both with and without lookahead in competitive analysis.

Rather than comparing $\textsc{Alg}$ with $\textsc{Opt}$ on the same worst-case instance, the *max/max ratio* ([20]) compares the (amortized) behavior of both algorithms on their respective worst-case instances. Amortization refers to the costs per input element: The amortized costs $A_{\textsc{Alg},n}$ of $\textsc{Alg}$ over all input sequences $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ of length n are the worst-case costs per input element over all input sequences of length n, i.e.,

$$A_{\textsc{Alg},n} = \max_{\sigma \in \Sigma, |\sigma| = n} \frac{\textsc{Alg}[\sigma]}{n}$$

The max/max ratio $c_r^{\max}$ of $\textsc{Alg}$ is defined as

$$c_r^{\max} := \limsup_{n \to \infty} \frac{A_{\textsc{Alg},n}}{A_{\textsc{Opt},n}} = \limsup_{n \to \infty} \frac{\max_{\sigma \in \Sigma, |\sigma| = n} \textsc{Alg}[\sigma]}{\max_{\sigma \in \Sigma, |\sigma| = n} \textsc{Opt}[\sigma]}.$$

Direct comparison of algorithms $\textsc{Alg}_1$ and $\textsc{Alg}_2$ is possible by replacing $\textsc{Alg}$ with $\textsc{Alg}_1$ and $\textsc{Opt}$ with $\textsc{Alg}_2$. Using the max/max ratio, non-competitive algorithms in competitive analysis are deemed competitive in the framework of the max/max ratio: Consider an insurance against (repeated) theft where the only option is to buy or not to buy ([20]). Any algorithm that wants to be competitive has to buy; hence, the malicious adversary in competitive analysis never presents a theft. Likewise, when the algorithm never buys, the adversary presents thefts only. In total, the algorithm is non-competitive. On finite input sequences $\sigma = (\sigma_1, \dots, \sigma_n)$ where $\sigma_i = 1$ means theft and $\sigma_i = 0$ means no theft in period i, $c_r^{\max}$ attains a finite value, and buying always induces a lower max/max ratio than not buying because the worst-case for buying is the fixed price, whereas for not buying it is the number of periods multiplied with the item value. Moreover, it is shown that the max/max ratio is able to discriminate between algorithms with different levels of lookahead for the k-server problem which is impossible in competitive analysis. Note that paging with cache size k coincides with the k-server problem where all request points are equidistant.

The *relative worst-order ratio* ([35]) combines the ideas of considering the behavior on input sequence permutations (cf. random-order ratio on page 44) and selecting the worst-case sequence for either algorithm (cf. max/max ratio). Let Π be the set of all permutations of $\{1, 2, \dots, n\}$ and denote by σ_π the permutation of $\sigma \in \Sigma$ according to $\pi \in \Pi$. By $\textsc{Alg}'[\sigma]$ we denote the costs of the worst-case permutation of σ for $\textsc{Alg}$, i.e.,

$$\textsc{Alg}'[\sigma] = \max_{\pi \in \Pi} \{\textsc{Alg}[\sigma_\pi]\}.$$

For $c \in \mathbb{R}$, define statements $S_1(c)$ and $S_2(c)$ as

$$\begin{aligned} S_1(c) &: \text{There exists a constant } b \text{ such that } \textsc{Alg}_1'[\sigma] \leq c \cdot \textsc{Alg}_2'[\sigma] + b, \\ S_2(c) &: \text{There exists a constant } b \text{ such that } \textsc{Alg}_1'[\sigma] \geq c \cdot \textsc{Alg}_2'[\sigma] - b. \end{aligned}$$

The relative worst-order ratio of $\textsc{Alg}_1$ and $\textsc{Alg}_2$ is defined if $S_1(1)$ or $S_2(1)$ holds; in this case, $\textsc{Alg}_1$ and $\textsc{Alg}_2$ are called comparable. Let $\textsc{Alg}_1$ and $\textsc{Alg}_2$ be two comparable algorithms, then the relative worst-order ratio $c_{\textsc{Alg}_1,\textsc{Alg}_2}$ is

$$c_{\textsc{Alg}_1,\textsc{Alg}_2} := \begin{cases} \sup\{r \mid S_2(r)\}, & \text{if } S_1(1) \text{ holds,} \\ \inf\{r \mid S_1(r)\}, & \text{if } S_2(1) \text{ holds.} \end{cases}$$

For $\textsc{Alg}_2 = \textsc{Opt}$, we speak of the worst-order ratio. Observe that for problems where all permutations of an input sequence induce the same (optimal) costs, the worst-order ratio reduces to the competitive ratio. For comparable algorithms, a relative worst-order ratio $c_{\textsc{Alg}_1,\textsc{Alg}_2} = 1$ means that $\textsc{Alg}_1$ and $\textsc{Alg}_2$ are considered equal, whereas $c_{\textsc{Alg}_1,\textsc{Alg}_2} < 1$ ($c_{\textsc{Alg}_1,\textsc{Alg}_2} > 1$) indicates that $\textsc{Alg}_1$ is better (worse) than $\textsc{Alg}_2$. The relative worst-order ratio allows an algorithm to perform bad on some specific input sequence which may be totally different from the worst-case sequence of the other algorithm. The authors have evaluated the measure for bin packing, seat reservation, paging and 2-server problems on the line to compare several online algorithms in these problems ([36], [35], [40], [38]).

2.3.1.2 Probabilistic Worst-Case Performance Measures

Probabilistic worst-case analysis is based on the use of some randomization mechanism. In contrast to average-case analysis, randomness is not used to impute probabilities on the occurrence of input sequences, but to blur worst-case instances.

Smoothed competitive analysis ([16]) origins from smoothed (complexity) analysis ([153]) where it was introduced to explain why the observed complexity of the simplex algorithm is polynomial-time, whereas its worst-case complexity is exponential-time. The basic idea to confirm these observations is to show that worst-case inputs are fragile in the sense that the worst-case character vanishes under small (smoothing) perturbations. The idea has been transferred to the analysis of online optimization algorithms by slightly perturbing instances according to some probability distribution and analyzing the expected competitive ratio on the perturbed sequences. Let p be some probability distribution and let $\Sigma^p(\sigma) \subset \Sigma$ be the set of input instances that are obtained by smoothening $\sigma \in \Sigma$ according to p, then the smoothed competitive ratio c_r^s of $\textsc{Alg}$ under p is given by

$$c_r^s := \sup_{\sigma \in \Sigma} \mathbb{E}_p \Big(\frac{\textsc{Alg}[\sigma^p]}{\textsc{Opt}[\sigma^p]} \Big)$$

where $\mathbb{E}_p$ is the expectation taken over all input sequences $\sigma^p \in \Sigma^p(\sigma)$[8]. For processor scheduling in a time-sharing multitasking operating system, some algorithm is shown to behave better under smoothed instances than its competitive ratio would suggest ([16]).

The *random-order ratio* ([108]) works similar to the smoothed competitive ratio. It considers the neighborhood $\Sigma^\Pi(\sigma)$ of an input sequence $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ as the set of all permutations of σ, i.e.,

$$\Sigma^\Pi(\sigma) = \{(\sigma_{\pi^{-1}(1)}, \sigma_{\pi^{-1}(2)}, \dots, \sigma_{\pi^{-1}(n)}) \,|\, \pi \text{ is a permutation of } \{1, 2, \dots, n\}\}.$$

The random-order ratio c_r^{ro} of $\textsc{Alg}$ is given by

$$c_r^{ro} := \limsup_{\textsc{Opt}[\sigma] \to \infty} \frac{\mathbb{E}_{\Sigma_\Pi(\sigma)}(\textsc{Alg}[\sigma])}{\mathbb{E}_{\Sigma_\Pi(\sigma)}(\textsc{Opt}[\sigma])}$$

where $\textsc{Opt}[\sigma] \to \infty$ (as given in [108]) characterizes the set of all input instances with length tending to infinity and all permutations of an input sequence are considered equally likely, i.e., uniformly distributed. For some applications, such as bin packing, this assumption is

[8] Instead of the expectation of the ratio, also the ratio of expectations could be used; however, the instance-wise perspective is preferred in [16] because of its stronger notion of competitiveness.

appropriate, whereas for others, such as paging with locality of reference, it is not. In [108], it is shown that BestFit outperforms other online bin packing algorithms: The algorithm is shown to have $c_r^{ro} \in [1.08, 1.5]$, whereas the competitive ratio is 1.7.

We only mention the use of *randomized algorithms* in order to improve worst-case results: A randomized online algorithm RandAlg is a probability distribution over a set of deterministic online algorithms $\{\textsc{Alg}_1, \textsc{Alg}_2, \ldots, \textsc{Alg}_m\}$ which will be drawn at random. The objective value of $\textsc{RandAlg}[\sigma]$ on input sequence σ then becomes a random variable. The strength of a randomized algorithm arises as a result of its uncertain realization that the malicious adversary has to cope with. We will not go into detail as we seek for an assessment of the behavior of a single deterministic algorithm and refer the reader to [21] and [32].

2.3.1.3 Average-Case Performance Measures

In average-case analysis, stochasticity refers to probabilities for input sequence occurrences. The notion of competitiveness has been transferred to stochastic settings ([136], [151]). Let D be a probability distribution over all input sequences, then Alg is called c-competitive under D if there is a constant a such that

$$\mathbb{E}_D(\textsc{Alg}[\sigma]) \leq c \cdot \mathbb{E}(\textsc{Opt}[\sigma]) + a.$$

In the case $a = 0$, Alg is called strictly c-competitive under D if

$$\frac{\mathbb{E}_D(\textsc{Alg}[\sigma])}{\mathbb{E}_D(\textsc{Opt}[\sigma])} \leq c.$$

The *expected competitive ratio* c_r of Alg under D is

$$\begin{aligned} c_r^D &= \inf\{c \geq 1 \mid \mathbb{E}_D(\textsc{Alg}[\sigma]) \leq c \cdot \mathbb{E}_D(\textsc{Opt}[\sigma]) + a\} \\ &= \inf\{c \geq 1 \mid \textsc{Alg} \text{ is } c\text{-competitive under } D\}. \end{aligned}$$

In the same way, the expectation can be taken instance-wise over all ratios between Alg's and Opt's costs: *The expected performance ratio* $c_r'^D$ of Alg under D is

$$c_r'^D = \inf\Big\{c \geq 1 \mid \mathbb{E}_D\Big(\frac{\textsc{Alg}[\sigma]}{\textsc{Opt}[\sigma]}\Big) \leq c\Big\}.$$

The author in [151] concludes that the expected performance ratio should be favored over the expected competitive ratio because sequences with small (large) objective value would be underrepresented (overrepresented) in the isolated expectations. Moreover, the expected performance ratio indicates which algorithm performs better on most of the sequences, and using Markov's inequality the probability for a sequence whose ratio is far from the expectation can be bounded. Several publications consider an asymptotic version of the expected performance ratio where one restricts attention to input sequences of infinite length (see, e.g., [26], [50], [106], [121]): Let Σ_n comprise the set of all input sequences of length n, $\sigma_n \in \Sigma_n$ and

$$R^D(n) := \mathbb{E}_D\Big(\frac{\textsc{Alg}[\sigma_n]}{\textsc{Opt}[\sigma_n]}\Big),$$

then the *asymptotic expected performance ratio* $R(\infty)$ of algorithm Alg is

$$R^D(\infty) = \lim_{n\to\infty} R^D(n).$$

Expected competitiveness has been extensively studied in the context of bin packing with a focus on the asymptotic expected performance ratio: $R^D(\infty)$ is derived for NextFit under uniformly distributed item sizes ([50], [106], [142]); Harmonic has $R^D(\infty) = \frac{\pi^2}{3} - 2 \approx 1.289$ ([121]). In [26], FirstFit is shown to be asymptotically optimal, i.e., $R^D(\infty) = 1$. [147] introduced the expected waste as a subsitute performance measure and succeeded in separating BestFit and FirstFit: In the limit both perform equally good, whereas BestFit converges faster. Results for discrete item size distributions are given in [49]. [6] and [109] adopt similar techniques to determine whether BestFit, FirstFit and RandomFit are stable, i.e., whether the expected waste remains asymptotically bounded. Results for discrete item sizes in the bounded-space bin packing problem are due to [136]. Surveys including average-case results in bin packing can be found in [46], [47] and [55]. In paging with locality of reference, [105] extends the access graph model with edge probabilities such that input sequences are generated by a Markov chain. The asymptotic expected fault rate is used to characterize an algorithm that achieves an optimal asymptotic expected fault rate on any Markov chain. Moreover, a procedure to determine an online algorithm whose asymptotic expected fault rate is bounded with respect to Opt is devised. However, in [75] it has been shown previously that none of the famous algorithms such as Lru or Fifo can have an asymptotic expected fault rate that is bounded with respect to Opt. In [76], $c_r'^D$ is analyzed for the ski rental problem where skiing ends after any period with probability $\lambda \in [0, 1]$. Upper bounds on $c_r'^D$ for List in makespan scheduling were derived in [48] under several distributions; for completion time scheduling, $c_r'^D \in O(1)$ was proven in [145] for ShortestExpectedProcessingTime under a class of distributions with known expected processing time. Bounds on $c_r'^D$ are found in [144], [151], [152] for minimum spanning trees, paging and completion time scheduling.

The *diffuse adversary model* introduced in [115] decreases adversary power by restricting instances to a class of admissible distributions Δ whereof a worst distribution $D \in \Delta$ is selected to determine the value of the performance measure. $\textsc{Alg}$ is said to have expected competitive ratio $c_r^d(\Delta)$ under diffuse adversaries with distribution class Δ if

$$c_r^d(\Delta) = \inf\Big\{c \geq 1 \,|\, \frac{\mathbb{E}_D(\textsc{Alg}[\sigma])}{\mathbb{E}_D(\textsc{Opt}[\sigma])} \leq c, D \in \Delta\Big\}.$$

$\textsc{Alg}$ is said to have expected performance ratio $c_r'^d(\Delta)$ under diffuse adversaries with distribution class Δ if

$$c_r'^d(\Delta) = \inf\Big\{c \geq 1 \,|\, \mathbb{E}_D\Big(\frac{\textsc{Alg}[\sigma]}{\textsc{Opt}[\sigma]}\Big) \leq c, D \in \Delta\Big\}.$$

The diffuse adversary model was first proposed in [115] to improve competitive analysis results on paging where a class of distributions Δ_ϵ is given to model the opposite of locality of reference: For any given history of previously requested pages and any page p, the probability that p will be requested next is not larger than ϵ. It is shown that $\textsc{Lru}$ attains the optimal expected performance ratio under diffuse adversaries with distribution class Δ_ϵ. The computation of $c_r'^d(\Delta_\epsilon)$ has been achieved in [166] showing $\textsc{Lru}$'s superiority over $\textsc{Fwf}$ and $\textsc{Fifo}$. A diffuse adversary to model locality of reference is proposed in [15]: Given some history of previous pages, the probability of requesting page p is a non-increasing function of the last point in time when p was requested. The observed superiority of $\textsc{Lru}$ is confirmed because $c_r'^d(\Delta) \in O(1)$ when distributions exhibit a sufficiently large degree of locality.

2.3.1.4 Distributional Performance Measures

The main advantage of distributional performance analysis is that an algorithm is judged by a distribution instead of a single key figure. Although no probabilistic assumptions are made in online optimization, we recognize that we obtain deterministic counting results when we impute a uniform distribution over all instances ([90]).

Relative interval analysis ([67]) is a preliminary stage of distributional analysis since it only considers the extreme values of a distribution for two algorithms. Define

$$\mathrm{Min}_{\textsc{Alg}_1,\textsc{Alg}_2}(n) := \min_{|\sigma|=n}\{\textsc{Alg}_1[\sigma] - \textsc{Alg}_2[\sigma]\},$$
$$\mathrm{Max}_{\textsc{Alg}_1,\textsc{Alg}_2}(n) := \max_{|\sigma|=n}\{\textsc{Alg}_1[\sigma] - \textsc{Alg}_2[\sigma]\},$$

then the relative interval of $\textsc{Alg}_1$ and $\textsc{Alg}_2$ is

$$I_{\textsc{Alg}_1,\textsc{Alg}_2} = \left[\liminf_{n\to\infty} \frac{\textsc{Min}_{\textsc{Alg}_1,\textsc{Alg}_2}(n)}{n}, \limsup_{n\to\infty} \frac{\textsc{Max}_{\textsc{Alg}_1,\textsc{Alg}_2}(n)}{n}\right].$$

The relative interval of an algorithm pair corresponds to its asymptotic range of amortized costs. For $I_{\textsc{Alg}_1,\textsc{Alg}_2} = [0, c]$ with $c \geq 0$, $\textsc{Alg}_2$ dominates $\textsc{Alg}_1$ in the sense that $\textsc{Alg}_2$ never incurs larger (asymptotic) amortized costs than $\textsc{Alg}_1$. Analogously, for $I_{\textsc{Alg}_1,\textsc{Alg}_2} = [-c, 0]$ with $c \geq 0$, $\textsc{Alg}_1$ dominates $\textsc{Alg}_2$. Relative interval analysis allows for a direct comparison between two algorithms without reference to $\textsc{Opt}$. In [67], relative interval analysis facilitates a distinction between paging algorithms $\textsc{Lru}$ and $\textsc{Fifo}$ on the one side and $\textsc{Fwf}$ on the other side; moreover, it also reflects the positive influence of lookahead on $\textsc{Lru}$.

Stochastic dominance (see, e.g., [133]) origins from statistics where it is used to establish an order relation between distributions of two random variables. By interpreting the objective value obtained by an algorithm as a random variable, this concept has been transferred to the analysis of online optimization algorithms in [90] where the objective value distributions of two algorithms $\textsc{Alg}_1$ and $\textsc{Alg}_2$ are related to each other using stochastic dominance[9]. Let $F_{\textsc{Alg}} : \mathbb{R} \to [0, 1]$ be the cumulative distribution function of the objective value of $\textsc{Alg}$:

- $\textsc{Alg}_1$ dominates $\textsc{Alg}_2$ stochastically at zeroth order if and only if $\textsc{Alg}_1[\sigma] \geq \textsc{Alg}_2[\sigma]$ for all $\sigma \in \Sigma$ (instance-wise dominance).
- $\textsc{Alg}_1$ dominates $\textsc{Alg}_2$ stochastically at first order ($\textsc{Alg}_1 \geq_{st} \textsc{Alg}_2$) if the probability for achieving an objective value larger than or equal to v is higher for $\textsc{Alg}_1$ than for $\textsc{Alg}_2$ for all $v \in \mathbb{R}$, i.e., $\textsc{Alg}_1$ has more mass on high values than $\textsc{Alg}_2$:

$$\begin{aligned} \textsc{Alg}_1 \geq_{st} \textsc{Alg}_2 \quad &:\Leftrightarrow \quad P(\textsc{Alg}_1[\sigma] \geq v) \geq P(\textsc{Alg}_2[\sigma] \geq v), \quad v \in \mathbb{R} \\ &\Leftrightarrow \quad F_{\textsc{Alg}_2}(v) \geq F_{\textsc{Alg}_1}(v), \quad v \in \mathbb{R}. \end{aligned}$$

- $\textsc{Alg}_1$ dominates $\textsc{Alg}_2$ stochastically at second order ($\textsc{Alg}_1 \geq_{st2} \textsc{Alg}_2$) if the cumulation of all probability densities for achieving an objective value larger than or equal to v is higher for $\textsc{Alg}_1$ than for $\textsc{Alg}_2$ for all $x \in \mathbb{R}$, i.e., $\textsc{Alg}_1$ has more mass on $[x, +\infty]$ than $\textsc{Alg}_2$:

$$\begin{aligned} \textsc{Alg}_1 \geq_{st2} \textsc{Alg}_2 \quad &:\Leftrightarrow \quad \int_{-\infty}^{x} P(\textsc{Alg}_1[\sigma] \geq v)dv \geq \int_{-\infty}^{x} P(\textsc{Alg}_2[\sigma] \geq v)dv, \quad x \in \mathbb{R} \\ &\Leftrightarrow \quad \int_{-\infty}^{x} (F_{\textsc{Alg}_2}(v) - F_{\textsc{Alg}_1}(v))dv \geq 0, \quad x \in \mathbb{R}. \end{aligned}$$

[9] Note that in a minimization problem $\textsc{Alg}_1[\sigma] \geq \textsc{Alg}_2[\sigma]$ means that $\textsc{Alg}_2$ is better than $\textsc{Alg}_1$ on σ.

Graphically, $\textsc{Alg}_1 \geq_{st} \textsc{Alg}_2$ yields $F_{\textsc{Alg}_1}(v) \leq F_{\textsc{Alg}_2}(v)$ and $\textsc{Alg}_1 \geq_{st2} \textsc{Alg}_2$ means that the area from $-\infty$ to v under $F_{\textsc{Alg}_2}$ is larger than that of $F_{\textsc{Alg}_1}$ for $v \in \mathbb{R}$. Figure 2.3 exemplifies stochastic dominance between the objective value distributions of three algorithms.

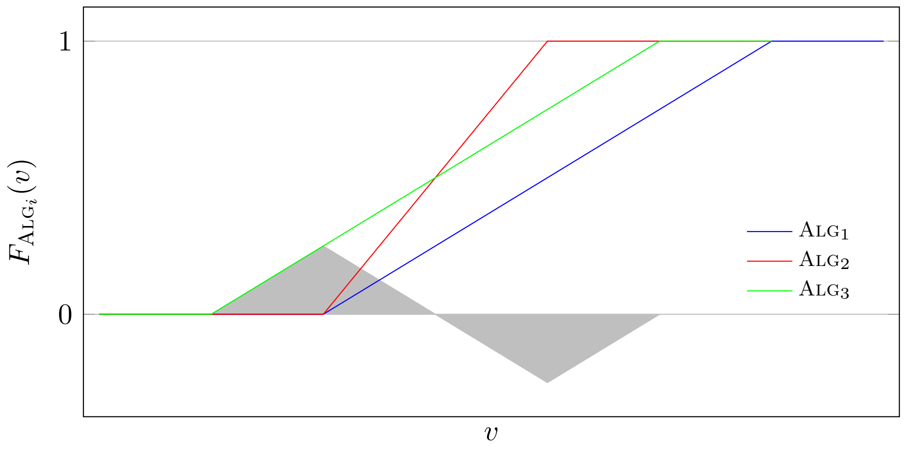


Figure 2.3: Stochastic dominance relations between $\textsc{Alg}_1, \textsc{Alg}_2, \textsc{Alg}_3$. $\textsc{Alg}_1 \geq_{st} \textsc{Alg}_2$ and $\textsc{Alg}_1 \geq_{st} \textsc{Alg}_3$, but neither $\textsc{Alg}_2 \geq_{st} \textsc{Alg}_3$ nor $\textsc{Alg}_3 \geq_{st} \textsc{Alg}_2$. $\textsc{Alg}_1 \geq_{st2} \textsc{Alg}_2$ and $\textsc{Alg}_1 \geq_{st2} \textsc{Alg}_3$, and from the shaded area we see that $\textsc{Alg}_2 \geq_{st2} \textsc{Alg}_3$.

If $\textsc{Alg}_1$ stochastically dominates $\textsc{Alg}_2$ at first order, then $\mathbb{E}(\textsc{Alg}_1) \geq \mathbb{E}(\textsc{Alg}_2)$. It is easy to see that stochastic dominance at a given order implies stochastic dominance at all subsequent orders. Unfortunately, no order of stochastic dominance admits a total ordering among all distributions and we cannot expect stochastic dominance to hold for arbitrary algorithms. However, when comparing an algorithm without lookahead $\textsc{Alg}_1$ to one with lookahead $\textsc{Alg}_2$ in a minimization problem, $\textsc{Alg}_1 \geq_{st} \textsc{Alg}_2$ would illustrate the benefit of lookahead because $\textsc{Alg}_2$ attains smaller objective values on more instances than $\textsc{Alg}_1$. In this case, $\textsc{Alg}_2$ is said to be stochastically better than $\textsc{Alg}_1$. In [90], stochastic dominance is analyzed for paging and bin coloring: Lru can be shown to be stochastically optimal for paging with locality of reference under three different models of locality; in bin coloring, the reasonable GreedyFit algorithm is shown to be stochastically better than the trivial OneBin algorithm, whereas competitive analysis fails to discriminate them.

Bijective analysis ([9]) is a special case of stochastic dominance where input sequences are uniformly distributed. The basic idea is to find a bijection $b : \Sigma_n \leftrightarrow \Sigma_n$ between the input sequences for $\textsc{Alg}_1$ and $\textsc{Alg}_2$ such that in a minimization problem, the objective value $\textsc{Alg}_1[\sigma]$ of $\textsc{Alg}_1$ on σ is never worse than that of $\textsc{Alg}_2$ on the image $b(\sigma)$ of σ, i.e., $\textsc{Alg}_2[b(\sigma)]$. Essentially, the approach consists in establishing an order of the elements in Σ_n such that $\textsc{Alg}_1$ outperforms $\textsc{Alg}_2$ on every pair $(\sigma, b(\sigma))$. $\textsc{Alg}_1$ is called no worse than $\textsc{Alg}_2$ according to bijective analysis ($\textsc{Alg}_1 \preceq \textsc{Alg}_2$) if for all $n \geq n_0 \geq 1$ with some $n_0 \in \mathbb{N}$ there exists $b : \Sigma_n \leftrightarrow \Sigma_n$ with $\textsc{Alg}_1[\sigma] \leq \textsc{Alg}_2[b(\sigma)]$ for all $\sigma \in \Sigma_n$. $\textsc{Alg}_1$ is called better than $\textsc{Alg}_2$ according to bijective analysis if $\textsc{Alg}_1 \preceq \textsc{Alg}_2$ and not $\textsc{Alg}_2 \preceq \textsc{Alg}_1$. Hence, $\textsc{Alg}_1$ does not have to outperform $\textsc{Alg}_2$ on each input sequence, but there has to be a relabeling of the input sequences such that this relation holds between the original and relabeled sequences. The authors applied bijective analysis to the paging and list update problem in [9] and [10]: $\textsc{Lru}$ is better than $\textsc{Fwf}$ according to bijective analysis. Moreover, lookahead is shown to be beneficial by $\textsc{Lru}(l) \preceq \textsc{Lru}$ where l is the lookahead size.

The authors of [9] themselves consider bijective analysis as too strong to establish a relation between algorithms in many problems and propose a substantially weaker concept called *average analysis* ([65]) which compares the average cost of two algorithms over all requests of the same length. $\textsc{Alg}_1$ is called no worse than $\textsc{Alg}_2$ according to average analysis if for all $n \geq n_0 \geq 1$ with some $n_0 \in \mathbb{N}$ we have

$$\sum_{\sigma \in \Sigma_n} \textsc{Alg}_1[\sigma] \leq \sum_{\sigma \in \Sigma_n} \textsc{Alg}_2[\sigma].$$

For paging with locality of reference as defined in concave analysis, $\textsc{Lru}$ is strictly separated from all other strategies as the unique optimal strategy according to average analysis.

We point out the advantages given in [10] for bijective analysis that can be transferred to any distributional performance measure:

- The performance measure is simple and intuitive, yet powerful.
- Algorithms can be compared directly without reference to a hypothetical algorithm.
- Typical properties of algorithms are likely to be uncovered.

We will adopt the idea of looking at distribution functions for assessing algorithm performance in our approach presented in the following section. We note that none of the distributional performance measures proposed in literature sticks to the idea of comparing algorithms on an instance-wise basis in form of a performance ratio.

2.3.2 Performance Comparison of Optimization Algorithms

The behavior of an algorithm with respect to the obtained objective value may differ strongly from instance to instance. We seek for an approach which summarizes the global behavior of an algorithm over all instances, but also does not lose sight of local quality, i.e., with respect to a particular instance. We wish to present algorithm behavior free of risk preferences such that the decision maker can choose the most suitable algorithm according to personal preferences, and we desire to analyze the impact of lookahead on algorithm performance.

Definition 2.39 (Objective value)**.**
Let $\Pi = (I, S, f, \text{opt})$ be an optimization problem, let $i \in I$ be an instance of Π, and let $\textsc{Alg}$ be an algorithm for Π choosing $s_{\textsc{Alg}}(i) \in S(i)$ on i.

a) $v_{\textsc{Alg}}(i) := f(i, s_{\textsc{Alg}}(i))$ is called objective value of $\textsc{Alg}$ with respect to i.

b) $\overline{v}_{\textsc{Alg}}(I) := \max_{i \in I}\{v_{\textsc{Alg}}(i)\}$ is called upper objective value of $\textsc{Alg}$ with respect to I.

c) $\underline{v}_{\textsc{Alg}}(I) := \min_{i \in I}\{v_{\textsc{Alg}}(i)\}$ is called lower objective value of $\textsc{Alg}$ with respect to I.

△

Since in online optimization, traditionally no probabilistic information on instance occurrences is given, choosing the maximum entropy distribution emulates the state of informational nescience best by minimizing the amount of a-priori information in the distribution ([97], [98]). The uniform distribution over I is the maximum entropy distribution among all distributions with support I (which follows from Langrangian relaxation by the definition of the entropy together with the constraint that the sum over all probabilities equals 1). For finite I, imposing a uniform distribution leads to counting results saying how many out of all instances yield a certain objective value ([90]). We define the counting distribution function[10] in order to subsume these frequency information of objective values over I:

Definition 2.40 (Counting distribution function of objective value)**.**
Let $\Pi = (I, S, f, \text{opt})$ be an optimization problem, let $i \in I$ be an instance of Π, let $\textsc{Alg}$ be an algorithm for Π, and let $v_{\textsc{Alg}}(i)$ be the objective value of $\textsc{Alg}$ on i.

a) If I is a discrete set, then the function $F_{\textsc{Alg}} : \mathbb{R} \to [0, 1]$ with

$$F_{\textsc{Alg}}(v) := \frac{\sum\limits_{i \in I} \mathbf{1}_{[-\infty, v]}(v_{\textsc{Alg}}(i))}{|I|}$$

is called counting distribution function of the objective value of $\textsc{Alg}$ over I.

b) If I is an uncountable set, then the function $F_{\textsc{Alg}} : \mathbb{R} \to [0, 1]$ with

$$F_{\textsc{Alg}}(v) := \frac{\int\limits_{i \in I} \mathbf{1}_{[-\infty, v]}(v_{\textsc{Alg}}(i))}{|I|}$$

is called counting distribution function of the objective value of $\textsc{Alg}$ over I.

△

[10] The indicator function $\mathbf{1}_A(x)$ is 1 if $x \in A$ and 0 otherwise.

Comparing two algorithms by means of their counting distribution functions of the objective value is done by examining their relative positions to each other. Assuming minimization in Figure 2.4 a), $\textsc{Alg}_1$ exhibits a more volatile behavior than $\textsc{Alg}_2$ with objective values ranging in a broader interval; likewise, chances for a low objective value are higher.

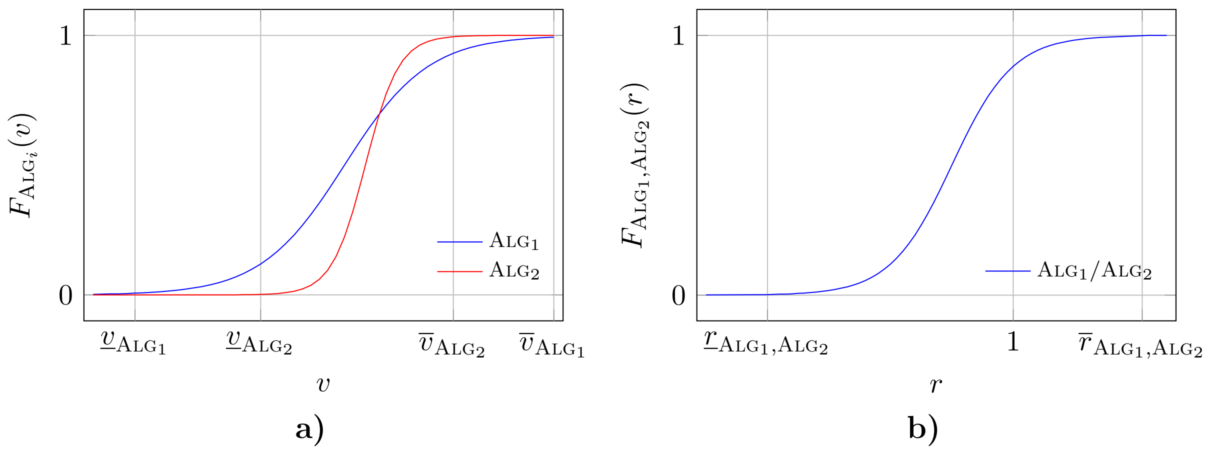


Figure 2.4: Counting distribution functions. **a)** Of objective value of $\textsc{Alg}_1$ and $\textsc{Alg}_2$. **b)** Of performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$.

The following definitions account for the relative performance of two algorithms to each other when both are restricted to operate on the same problem instance:

Definition 2.41 (Performance ratio)**.**
Let $\Pi = (I, S, f, \text{opt})$ be an optimization problem, let $i \in I$ be an instance of Π, and let $\textsc{Alg}_1$, $\textsc{Alg}_2$ be two algorithms for Π choosing $s_{\textsc{Alg}_1}(i), s_{\textsc{Alg}_2}(i) \in S(i)$ on i, respectively.

a) $r_{\textsc{Alg}_1,\textsc{Alg}_2}(i) := \frac{f(i,s_{\textsc{Alg}_1}(i))}{f(i,s_{\textsc{Alg}_2}(i))}$ is called performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$ with respect to i.

b) $\overline{r}_{\textsc{Alg}_1,\textsc{Alg}_2}(I) := \max_{i \in I}\{r_{\textsc{Alg}_1,\textsc{Alg}_2}(i)\}$ is called upper performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$ with respect to I.

c) $\underline{r}_{\textsc{Alg}_1,\textsc{Alg}_2}(I) := \min_{i \in I}\{r_{\textsc{Alg}_1,\textsc{Alg}_2}(i)\}$ is called lower performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$ with respect to I.

$\triangle$

$\overline{r}_{\textsc{Alg}_1,\textsc{Alg}_2}(I)$ coincides with the competitive ratio if Π is an online optimization problem, $\textsc{Alg}_1$ is an online algorithm and $\textsc{Alg}_2$ equals $\textsc{Opt}$. It coincides with the approximation ratio if Π is an offline optimization problem, $\textsc{Alg}_1$ is an offline algorithm and $\textsc{Alg}_2$ equals $\textsc{Opt}$. For online algorithms $\textsc{Alg}_1$ and $\textsc{Alg}_2$ with and without lookahead, respectively, the performance ratio expresses how much $\textsc{Alg}_1$ benefits from lookahead compared to $\textsc{Alg}_2$.

Definition 2.42 (Counting distribution function of performance ratio)**.**
Let $\Pi = (I, S, f, \mathrm{opt})$ be an optimization problem, let $i \in I$ be an instance of Π, let $\textsc{Alg}$ be an algorithm for Π, and let $r_{\textsc{Alg}_1,\textsc{Alg}_2}(i)$ be the performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$ on i.

a) If I is a discrete set, then the function $F_{\textsc{Alg}_1,\textsc{Alg}_2} : \mathbb{R} \to [0, 1]$ with

$$F_{\textsc{Alg}_1,\textsc{Alg}_2}(r) := \frac{\sum\limits_{i \in I} \mathbf{1}_{[-\infty,r]}(r_{\textsc{Alg}_1,\textsc{Alg}_2}(i))}{|I|}$$

is called counting distribution function of the performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$ over I.

b) If I is an uncountable set, then the function $F_{\textsc{Alg}_1,\textsc{Alg}_2} : \mathbb{R} \to [0, 1]$ with

$$F_{\textsc{Alg}_1,\textsc{Alg}_2}(r) := \frac{\int\limits_{i \in I} \mathbf{1}_{[-\infty,r]}(r_{\textsc{Alg}_1,\textsc{Alg}_2}(i))}{|I|}$$

is called counting distribution function of the performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_2$ over I.

△

Assuming minimization, comparing two algorithms $\textsc{Alg}_1$ and $\textsc{Alg}_2$ by means of their counting distribution function of the performance ratio is done by partitioning the set of all instances into those with performance ratio smaller than or equal to 1 (favoring $\textsc{Alg}_1$) and into those with performance ratio larger than 1 (favoring $\textsc{Alg}_2$). In Figure 2.4 b), $\textsc{Alg}_1$ attains an objective value at least as large as that of $\textsc{Alg}_2$ on the majority of the instances.

We mention advantages of our two-sided approach which consists of using distribution functions of the objective value and the performance ratio for assessing an algorithm's quality:

- The distribution function of the objective value gives a global view on the quality of an algorithm over all instances.
- The distribution function of the performance ratio of an algorithm pair offers a local view on the quality of both algorithms relative to each other on the same instance.
- Distribution-based analysis also yields information about ranges and variability.
- Algorithms with arbitrary lookahead levels can be compared to each other.

Example 2.43 (Counting distributions of objective value and performance ratio)**.**
Consider an optimization problem $\Pi = (I, S, f, \max)$ where the instance set I is defined by

$$I = \{x \in \mathbb{R}^2 \,|\, x_1 + x_2 \leq 5, \quad 0.75x_1 + x_2 \leq 4.5, \quad x_1 \leq 3, \quad x_1 \geq 0, \quad x_2 \geq 0\} \subset \mathbb{R}^2$$

as illustrated in Figure 2.5 with a total volume $|I| = 10$.

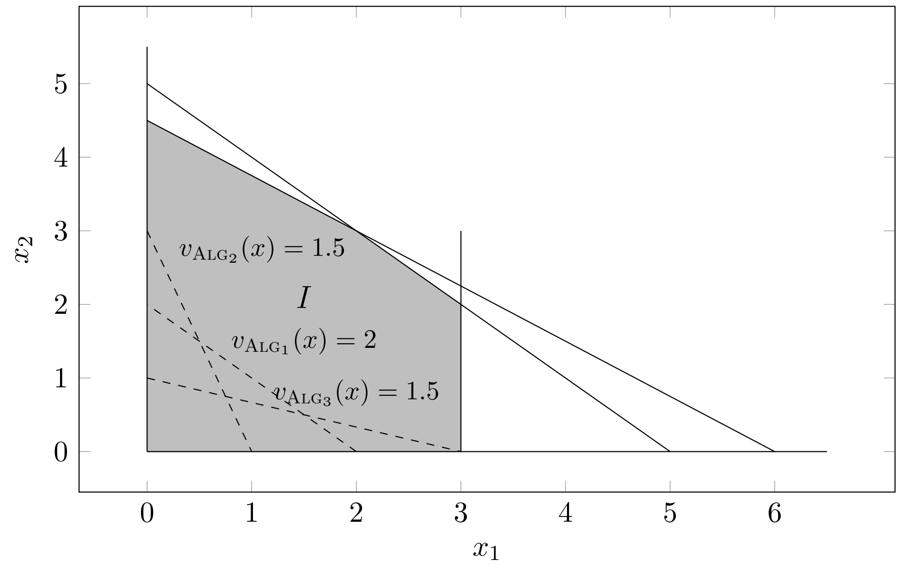


Figure 2.5: Graphical illustration of instance set I.

The set of feasible solutions for each $i \in I$ is given by

$$S(i) = S(x_1, x_2) = \{c_1x_1 + c_2x_2 \,|\, c_1, c_2 \in \mathbb{R}\} = \mathbb{R}$$

and a feasible solution $s \in S(i)$ is evaluated as its identity, i.e., $f(i, s) = s$. For some (intransparent) reason, we have three algorithms ALG_1, ALG_2, ALG_3 which choose

$$\begin{aligned} s_{\text{ALG}_1}(i) &= s_{\text{ALG}_1}(x_1, x_2) = x_1 + x_2, \\ s_{\text{ALG}_2}(i) &= s_{\text{ALG}_2}(x_1, x_2) = 1.5x_1 + 0.5x_2, \\ s_{\text{ALG}_3}(i) &= s_{\text{ALG}_3}(x_1, x_2) = 0.5x_1 + 1.5x_2, \end{aligned}$$

respectively. Which of the three algorithms is most promising when some instance $i \in I$ will be revealed to the decision maker?

In contrast to linear programming, we do not want to maximize $s_{\text{ALG}_j}(i)$ for $j = 1, 2, 3$ over I but rather would like to obtain the objective value distribution over I for each of the algorithms (cf. Figure 2.6 for ALG_1). One can think of a decision maker who has to choose one of the algorithms before being presented with some point $i = (x_1, x_2) \in I$. How should the decision maker behave in order to obtain a high objective value?

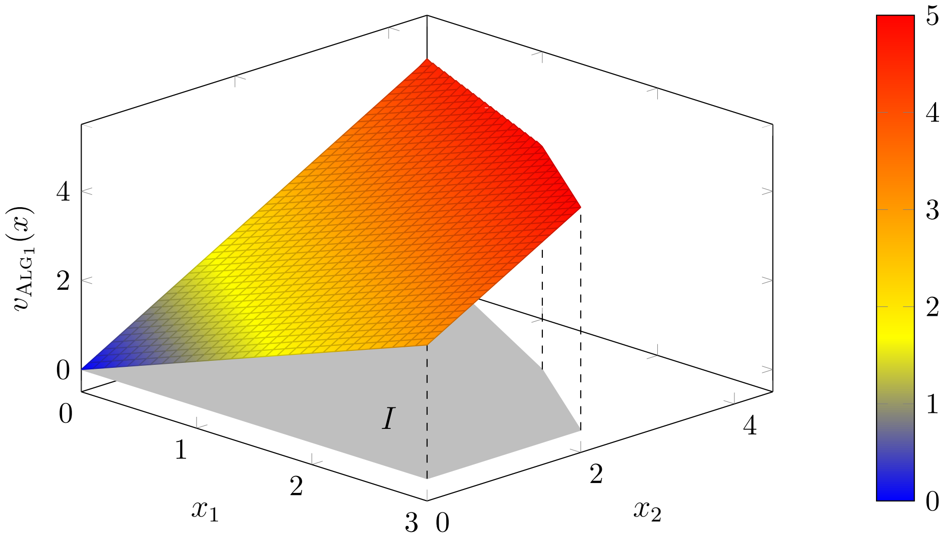


Figure 2.6: Objective value of ALG_1 over instance set I.

To find an expression for the counting distribution function $F_{\text{ALG}_j}(v)$ with $j = 1, 2, 3$, we first compute the area $V_v^j := |\{x \in I \mid f(x, s_{\text{ALG}_j}(x)) \le v\}|$ of all points $x \in I$ with objective function value of ALG_j smaller than or equal to v.

- $f(x, s_{\text{ALG}_1}(x)) = x_1 + x_2$:
  - $v \in [0, 3]$: $V_v^1 = \int_0^v (v - x_1)dx_1 = 0.5v^2$
  - $v \in [3, 4.5]$: $V_v^1 = \int_0^3 (v - x_1)dx_1 = 3v - 4.5$
  - $v \in [4.5, 5]$: $V_v^1 = \int_0^{4v-18} (4.5 - 0.75x_1)dx_1 + \int_{4v-18}^{3} (v - x_1)dx_1 = -2v^2 + 21v - 45$
- $f(x, s_{\text{ALG}_2}(x)) = 1.5x_1 + 0.5x_2$:
  - $v \in [0, 2.25]$: $V_v^2 = \int_0^{\frac{2}{3}v} (2v - 3x_1)dx_1 = \frac{2}{3}v^2$
  - $v \in [2.25, 4.5]$: $V_v^2 = \int_0^{\frac{8}{9}v-2} (4.5 - 0.75x_1)dx_1 + \int_{\frac{8}{9}v-2}^{\frac{2}{3}v} (2v - 3x_1)dx_1 = -\frac{2}{9}v^2 + 4v - 4.5$
  - $v \in [4.5, 5.5]$: $$V_v^2 = \int_0^2 (4.5 - 0.75x_1)dx_1 + \int_2^{v-2.5} (5 - x_1)dx_1 + \int_{v-2.5}^{3} (2v - 3x_1)dx_1 = -v^2 + 11v - 20.25$$

- $f(x, s_{\text{ALG}_3}(x)) = 0.5x_1 + 1.5x_2$:
  - $v \in [0, 1.5]$: $V_v^3 = \int\limits_0^{2v} (\frac{2}{3}v - \frac{1}{3}x_1)dx_1 = \frac{2}{3}v^2$
  - $v \in [1.5, 4.5]$: $V_v^3 = \int\limits_0^{3} (\frac{2}{3}v - \frac{1}{3}x_1)dx_1 = 2v - 1.5$
  - $v \in [4.5, 5.5]$: $V_v^3 = \int\limits_0^{7.5-v} (\frac{2}{3}v - \frac{1}{3}x_1)dx_1 + \int\limits_{7.5-v}^{3} (5 - x_1)dx_1 = -\frac{1}{3}v^2 + 5v - 8.25$
  - $v \in [5.5, 6.75]$: $V_v^3 = \int\limits_0^{10.8-1.6v} (\frac{2}{3}v - \frac{1}{3}x_1)dx_1 + \int\limits_{10.8-1.6v}^{2} (4.5 - 0.75x_1)dx_1 + \int\limits_2^3 (5 - x_1)dx_1 = -\frac{8}{15}v^2 + 7.2v - 14.3$

With V_v^j for $j \in \{1, 2, 3\}$ and I's total volume of 10, we have that the percentage of all $x \in I$ with objective value of ALG_j smaller than or equal to v is $\frac{V_v^j}{10}$. We can interpret this value as the probability that the objective value will be no more than v when we pick a point $x \in I$ at random, i.e., when we assume a uniform distribution over the elements of I. We obtain the counting distribution functions of the objective value (cf. Figure 2.7) as follows:

$$F_{\text{ALG}_1}(v) = \begin{cases} 0, & \text{if } v < 0 \\ \frac{1}{10}(0.5v^2), & \text{if } 0 \le v < 3, \\ \frac{1}{10}(3v - 4.5), & \text{if } 3 \le v < 4.5, \\ \frac{1}{10}(-2v^2 + 21v - 45), & \text{if } 4.5 \le v < 5, \\ 1, & \text{else.} \end{cases}$$

$$F_{\text{ALG}_2}(v) = \begin{cases} 0, & \text{if } v < 0, \\ \frac{1}{10}(\frac{2}{3}v^2), & \text{if } 0 \le v < 2.25, \\ \frac{1}{10}(-\frac{2}{9}v^2 + 4v - 4.5), & \text{if } 2.25 \le v < 4.5, \\ \frac{1}{10}(-v^2 + 11v - 20.25), & \text{if } 4.5 \le v < 5.5, \\ 1, & \text{else.} \end{cases}$$

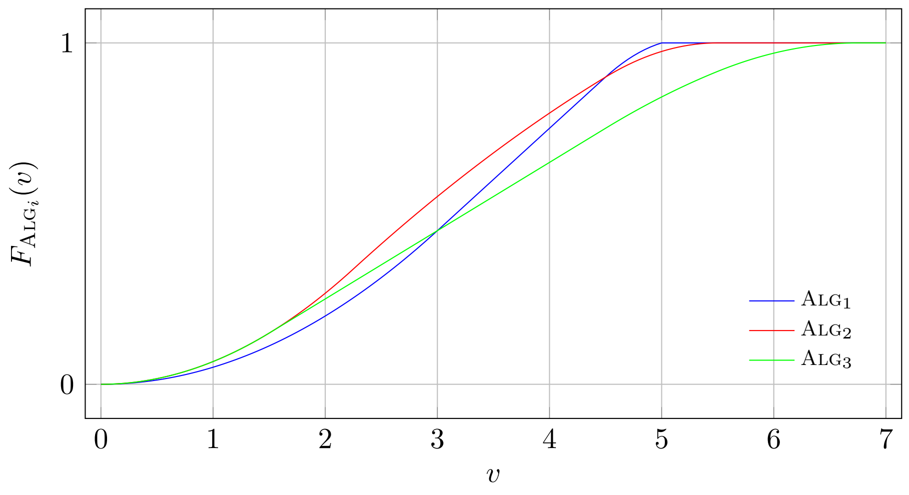


Figure 2.7: Distributions of objective value of ALG$_1$, ALG$_2$, ALG$_3$ over I.

$$F_{\text{ALG}_3}(v) = \begin{cases} 0, & \text{if } v < 0, \\ \frac{1}{10}(\frac{2}{3}v^2), & \text{if } 0 \le v < 1.5, \\ \frac{1}{10}(2v - 1.5), & \text{if } 1.5 \le v < 4.5, \\ \frac{1}{10}(-\frac{1}{3}v^2 + 5v - 8.25), & \text{if } 4.5 \le v < 5.5, \\ \frac{1}{10}(-\frac{8}{15}v^2 + 7.2v - 14.3), & \text{if } 5.5 \le v < 6.75, \\ 1, & \text{else.} \end{cases}$$

Because of $F_{\text{ALG}_2}(v) \ge F_{\text{ALG}_3}(v)$ for $v \in \mathbb{R}$, ALG$_3$ dominates ALG$_2$. No dominance relation involves F_{ALG_1}. However, when we compare F_{ALG_1} and F_{ALG_2}, we see that $F_{\text{ALG}_1}(v) \le F_{\text{ALG}_2}(v)$ for $v \in [0, 4.5]$ making ALG$_1$ comparably attractive.

We next seek to derive an expression for the counting distribution function of the performance ratio of one algorithm relative to another one. Exemplarily, Figure 2.8 shows the graph of the performance ratio $r_{\text{ALG}_1,\text{ALG}_2}(x)$ over all instances.

Considering performance ratio $r_{\text{ALG}_1,\text{ALG}_2}(i) = \frac{f(i,s_{\text{ALG}_1}(i))}{f(i,s_{\text{ALG}_2}(i))} = \frac{x_1+x_2}{1.5x_1+0.5x_2}$, we first recognize that every ray emanating from the origin consists of points with equal performance ratio: Let $x_2 = mx_1$ be such a ray, then it follows that

$$\frac{f(i, s_{\text{ALG}_1}(i))}{f(i, s_{\text{ALG}_2}(i))} = \frac{x_1 + mx_1}{1.5x_1 + 0.5mx_1} = \frac{x_1(1+m)}{x_1(1.5+0.5m)} = \text{const.}$$

As displayed in Figure 2.9, we subdivide I into three regions A_1 with $r_{\text{ALG}_1,\text{ALG}_2}(i) \in [\frac{2}{3}, \frac{10}{11}]$, A_2 with $r_{\text{ALG}_1,\text{ALG}_2}(i) \in [\frac{10}{11}, \frac{10}{9}]$, and A_3 with $r_{\text{ALG}_1,\text{ALG}_2}(i) \in [\frac{10}{9}, 2]$.

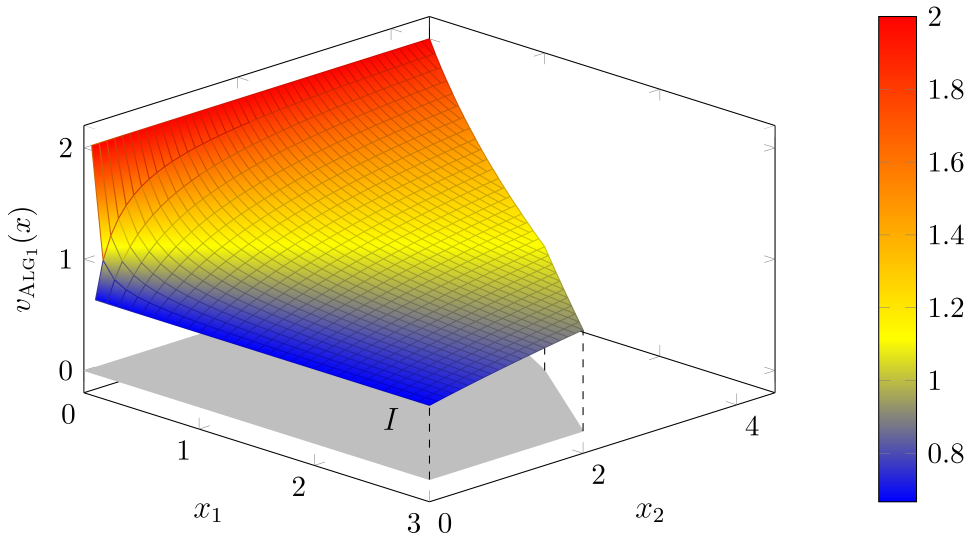


Figure 2.8: Performance ratio $r_{\text{ALG}_1,\text{ALG}_2}(x)$ over instance set I.

Observe that for two rays $x_2 = m_2x_1$ and $x_2 = m_1x_1$ with $m_2 > m_1$ it follows that all points on $x_2 = m_2x_1$ have a larger performance ratio than all points on $x_2 = m_1x_1$:

$$\frac{x_1 + m_2x_1}{1.5x_1 + 0.5m_2x_1} > \frac{x_1 + m_1x_1}{1.5x_1 + 0.5m_1x_1} \Leftrightarrow \frac{1 + m_2}{1.5 + 0.5m_2} > \frac{1 + m_1}{1.5 + 0.5m_1} \Leftrightarrow m_2 > m_1.$$

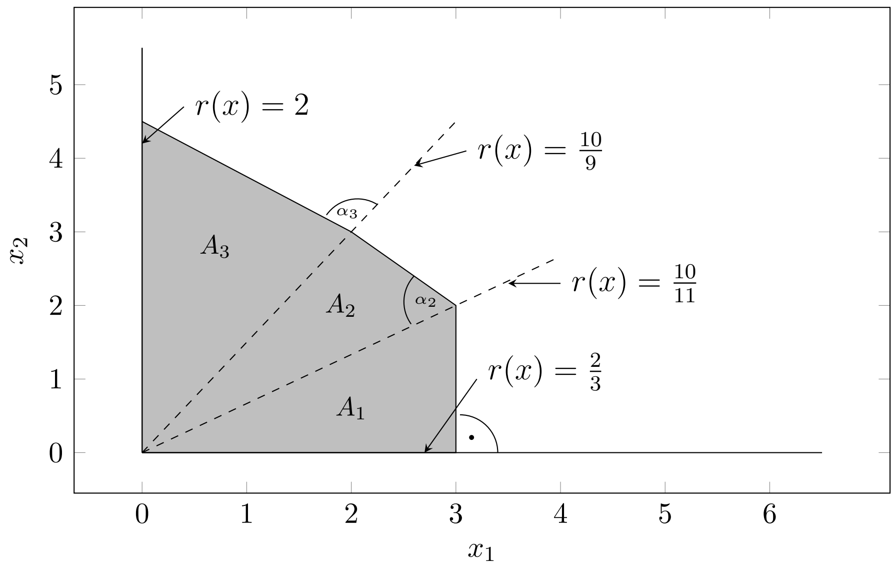


Figure 2.9: Instance set I with regions A_1, A_2, A_3.

In order to find an expression for $F_{\text{ALG}_1,\text{ALG}_2}(r)$, we have to characterize the amount of the volume of I where performance ratios not larger than r are attained for each $r \in \mathbb{R}$. We have from the previous discussion that $F_{\text{ALG}_1,\text{ALG}_2}(r) = 0$ for $r < \frac{2}{3}$ and $F_{\text{ALG}_1,\text{ALG}_2}(r) = 1$ for $r \geq 2$. For the remaining values of r, we analyze areas A_1, A_2 and A_3: By $A_i^{\leq}(r)$

($i = 1, 2, 3$) we denote the (triangular) subregion in A_i where each point x of this subregion has $r_{\text{ALG}_1,\text{ALG}_2}(x) \leq r$, by $x(r) = (x_1(r), x_2(r))$ we denote the point on the face of I with $r_{\text{ALG}_1,\text{ALG}_2}(x) = r$, and by $h(x(r))$ we denote the height of triangle $A_i^{\leq}(r)$ when the ray emanating from 0 intersects $x(r)$. As the bases of $A_1^{\leq}(r), A_2^{\leq}(r), A_3^{\leq}(r)$, we take $[0,(3,0)], [0,(3,2)], [0,(2,3)]$, respectively.

- For A_1, we have that $|A_1^{\leq}(\frac{2}{3})| = 0$ and $|A_1^{\leq}(\frac{10}{11})| = 3$. Exploiting the relation

$$r = \frac{3 + x_2}{4.5 + 0.5x_2} \Leftrightarrow x_2 = \frac{3 - 4.5r}{0.5r - 1} \Leftrightarrow x_2 = \frac{6 - 9r}{r - 2}$$

which follows from $x_1 = 3$ on the face of I in A_1 coinciding with the height of the triangle for computing $A_1^{\leq}(r)$, we obtain

$$|A_1^{\leq}(r)| = \frac{1}{2} \cdot 3 \cdot h(x(r)) = \frac{1}{2} \cdot 3 \cdot x_2(r) = \frac{3}{2}\Big(\frac{6 - 9r}{r - 2}\Big).$$

Relating $|A_1^{\leq}(r)|$ to $|I| = 10$, we get for $r \in [\frac{2}{3}, \frac{10}{11}]$ that

$$F_{\text{ALG}_1,\text{ALG}_2}(r) = \frac{3}{20}\Big(\frac{6 - 9r}{r - 2}\Big).$$

- For A_2, we have that $|A_2^{\leq}(\frac{10}{11})| = 0$ and $|A_2^{\leq}(\frac{10}{9})| = 2.5$. Using the values of the slopes of $x_2 = 5 - x_1$ and $x_2 = \frac{2}{3}x_1$ we have

$$\tan \alpha_2 = \left|\frac{-1 - \frac{2}{3}}{1 - \frac{2}{3}}\right| = 5 \Rightarrow \alpha_2 = \tan^{-1}(5) \approx 78.69°.$$

Exploiting the relation

$$r = \frac{5}{7.5 - x_2} \Leftrightarrow x_2 = 7.5 - \frac{5}{r} \Leftrightarrow x_2 = \frac{15}{2} - \frac{5}{r}$$

which follows from $x_1 = 5 - x_2$ on the face of I in A_2 needed to determine the height of the triangle for computing $A_2^{\leq}(r)$, we obtain

$$\begin{aligned}
|A_2^{\leq}(r)| &= \frac{1}{2} \cdot \sqrt{13} \cdot h(x(r)) \\
&= \frac{1}{2} \cdot \sqrt{13} \cdot \sin(\tan^{-1}(5)) \cdot \sqrt{((5 - x_2) - 3)^2 + (x_2 - 2)^2} \\
&= \frac{1}{2} \cdot \sqrt{13} \cdot \sin(\tan^{-1}(5)) \cdot \sqrt{2} \cdot (x_2 - 2) \\
&= \frac{5}{2} \cdot (x_2 - 2) = \frac{5}{2} \cdot \Big(\Big(\frac{15}{2} - \frac{5}{r}\Big) - 2\Big) = \frac{5}{2} \cdot \Big(\frac{11}{2} - \frac{5}{r}\Big).
\end{aligned}$$

Relating $|A_2^{\leq}(r)|$ to $|I| = 10$ and considering that $|A_1| = 3$, we get for $r \in [\frac{10}{11}, \frac{10}{9}]$ that

$$F_{\text{ALG}_1,\text{ALG}_2}(r) = 0.3 + \frac{5}{20}\Big(\frac{11}{2} - \frac{5}{r}\Big) = \frac{67}{40} - \frac{5}{4r}.$$

- For A_3, we have that $|A_3^{\leq}(\frac{10}{9})| = 0$ and $|A_3^{\leq}(2)| = 4.5$. Using the values of the slopes of $x_2 = 4.5 - 0.75x_1$ and $x_2 = 1.5x_1$ we have

$$\tan\alpha_3 = \Big|\frac{1.5 + 0.75}{1 - \frac{9}{8}}\Big| = 18 \Rightarrow \alpha_3 = \tan^{-1}(18) \approx 86.82°.$$

Exploiting the relation

$$r = \frac{36 - 2x_2}{54 - 9x_2} \Leftrightarrow x_2 = 6 - \frac{24}{9r - 2}$$

which follows from $x_1 = 6 - \frac{4}{3}x_2$ on the face of I in A_3 needed to determine the height of the triangle for computing $A_3^{\leq}(r)$, we obtain

$$\begin{aligned}|A_3^{\leq}(r)| &= \frac{1}{2}\cdot\sqrt{13}\cdot h(x(r))\\ &= \frac{1}{2}\cdot\sqrt{13}\cdot\sin(\tan^{-1}(18))\cdot\sqrt{((6 - \frac{4}{3}x_2) - 2)^2 + (x_2 - 3)^2}\\ &= \frac{1}{2}\cdot\sqrt{13}\cdot\sin(\tan^{-1}(18))\cdot\frac{5}{3}\cdot(x_2 - 3)\\ &= 3\cdot(x_2 - 3) = 3\Big(\Big(6 - \frac{24}{9r - 2}\Big) - 3\Big) = 9 - \frac{72}{9r - 2}.\end{aligned}$$

Relating $|A_3^{\leq}(r)|$ to $|I| = 10$ and considering that $|A_1| = 3, |A_1| = 2.5$, we get for $r \in [\frac{10}{9}, 2]$ that

$$F_{\text{ALG}_1,\text{ALG}_2}(r) = 0.55 + \frac{9}{10}\Big(1 - \frac{8}{9r - 2}\Big) = \frac{29}{20} - \frac{36}{5(9r - 2)}.$$

We obtain the counting distribution function for the performance ratio of ALG_1 relative to ALG_2 as

$$F_{\text{ALG}_1,\text{ALG}_2}(r) = \begin{cases} 0, & \text{if } r < \frac{2}{3},\\ \frac{3}{20}\big(\frac{6-9r}{r-2}\big), & \text{if } \frac{2}{3} \leq r < \frac{10}{11},\\ \frac{67}{40} - \frac{5}{4r}, & \text{if } \frac{10}{11} \leq r < \frac{10}{9},\\ \frac{29}{20} - \frac{36}{5(9r-2)}, & \text{if } \frac{10}{9} \leq r < 2,\\ 1, & \text{if } r \geq 2.\end{cases}$$

A similar analysis applies to determine the distribution functions of the performance ratio of $\textsc{Alg}_1$ relative to $\textsc{Alg}_3$ and of $\textsc{Alg}_2$ relative to $\textsc{Alg}_3$, respectively:

$$F_{\textsc{Alg}_1,\textsc{Alg}_3}(r) = \begin{cases} 0, & \text{if } r < \frac{2}{3}, \\ \frac{27}{20} - \frac{54-27r}{5(5r+2)}, & \text{if } \frac{2}{3} \le r < \frac{10}{11}, \\ \frac{73}{40} - \frac{5}{4r}, & \text{if } \frac{10}{11} \le r < \frac{10}{9}, \\ 1 - \frac{18-9r}{10(6r-4)}, & \text{if } \frac{10}{9} \le r < 2, \\ 1, & \text{if } r \ge 2. \end{cases}$$

$$F_{\textsc{Alg}_2,\textsc{Alg}_3}(r) = \begin{cases} 0, & \text{if } r < \frac{1}{3}, \\ \frac{27}{20} - \frac{81-27r}{5(5r+9)}, & \text{if } \frac{1}{3} \le r < \frac{9}{11}, \\ \frac{6}{5} - \frac{15-5r}{8(1+r)}, & \text{if } \frac{9}{11} \le r < \frac{11}{9}, \\ 1 - \frac{27-9r}{20(3r-1)}, & \text{if } \frac{11}{9} \le r < 3, \\ 1, & \text{if } r \ge 3. \end{cases}$$

Figure 2.10 plots the distribution functions of the performance ratio of the three algorithms relative to each other.

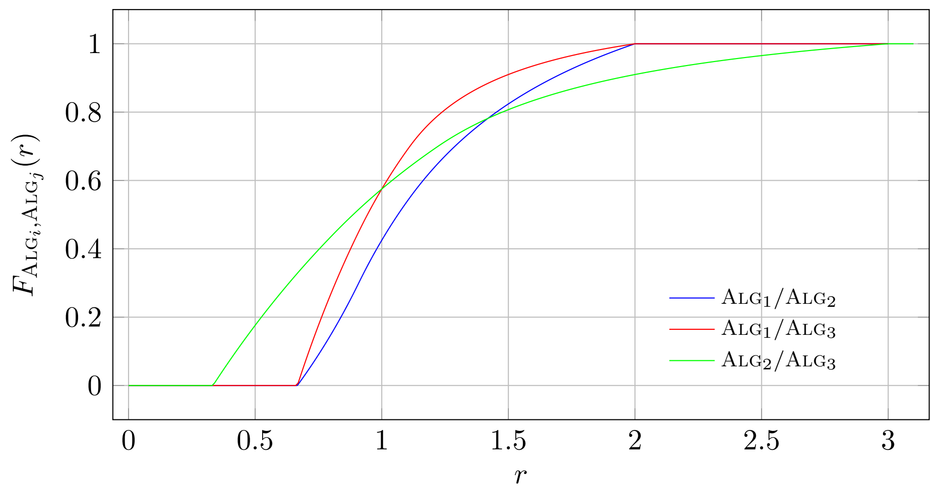


Figure 2.10: Distribution of performance ratio of $\textsc{Alg}_i$ relative to $\textsc{Alg}_j$ for $(i,j) = (1,2)$, $(i,j) = (1,3)$ and $(i,j) = (2,3)$ over I.

Because of maximization, it would be more desirable to choose $\textsc{Alg}_1$ over $\textsc{Alg}_2$: The performance ratio range $[\frac{2}{3}, 2]$ reveals that $\textsc{Alg}_1$ can be up to twice as good as $\textsc{Alg}_2$, whereas $\textsc{Alg}_2$ can only be $\frac{3}{2}$ as good as $\textsc{Alg}_1$; at the same time more than 50 % of the instances incur a ratio larger than 1. Consider $F_{\textsc{Alg}_1,\textsc{Alg}_3}$: On the one hand, $\textsc{Alg}_1$ loses to $\textsc{Alg}_3$ on having more instances with a ratio larger than 1; on the other hand, $\textsc{Alg}_1$ beats $\textsc{Alg}_3$ in potentially being twice as good, whereas $\textsc{Alg}_3$ can only be $\frac{3}{2}$ as good as $\textsc{Alg}_1$. For $F_{\textsc{Alg}_2,\textsc{Alg}_3}$, the ratio ranges in $[\frac{1}{3}, 3]$ and less than 50 % of the instances incur a ratio larger than 1. In this sense, $\textsc{Alg}_3$ should be favored over $\textsc{Alg}_2$. From the discussion it follows that $\textsc{Alg}_2$ is not to be recommended in the first place.

We remark that in general computing the volume of a polytope is $\#\mathcal{P}$-hard[11]. For further details, the interested reader is referred to [110], [111], [112]. $\Diamond$

2.4 Optimization Algorithms and Discrete Event Systems

So far we were only concerned with the outcome of an algorithm for a given instance of an online optimization problem but did not pay attention to the solution process that is undergone in order to obtain the outcome. Now we concentrate on the modeling of that solution process: Interpreting the release of an input element of input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ as the occurrence of an event allows us to recast the solution process in online optimization using the terminology from discrete event systems. In the following Chapters 2.4.1 to 2.4.4, different modeling concepts are discussed based on the standard works [43] and [126] on discrete event systems as well as on the guidelines of the Association of German Engineers (VDI, Verein Deutscher Ingenieure) on simulation in [159]; alongside the discussion we transfer these concepts to online optimization with lookahead.

2.4.1 Discrete Event Systems

Define a process as a transformation, transportation or storage of some energetic, material or informational resource, then a system is a fragment of the real world in which all processes relevant to the user take place. As such, a system is a reduction from the real world to those entities which influence these processes and satisfy the user's wish to duplicate them. Models are used to describe systems: A model is a formal description of a system which captures the relevant aspects of the system along with all related processes in a sufficient level of detail. Systems with time-dependent behavior are called dynamic systems and they are usually described by an input-output model.

[11] A counting problem Π (asking for the number of solutions to an instance with a given property) is in complexity class $\#\mathcal{P}$ if its associated decision problem is in $\mathcal{NP}$. A problem in $\#\mathcal{P}$ is at least as hard as its decision problem version in $\mathcal{NP}$. A counting problem Π is in complexity class $\#\mathcal{P}$-complete if it is in $\#\mathcal{P}$ and all other problems in $\#\mathcal{P}$ can be transformed to it in polynomial time. A problem Π is in complexity class $\#\mathcal{P}$-hard if it is at least as hard as any problem in $\#\mathcal{P}$.

Processes in dynamic systems are stimulated by an input function $v : \mathcal{T} \to \mathcal{V}$ from the set $\mathcal{T}$ of time instants to the set $\mathcal{V}$ of input values. In discrete time systems, we typically have $\mathcal{T} = \mathbb{N}_0$; in continuous time systems, usually $\mathcal{T} = \mathbb{R}^{\geq 0}$. The system responds to the values provided by v over time with an output function $w : \mathcal{T} \to \mathcal{W}$ from $\mathcal{T}$ to the set $\mathcal{W}$ of output values. Let $\mathcal{F}_v$ and $\mathcal{F}_w$ be the set of all input and output functions, respectively. Then we can view a (model of a) system as an operator $S : \mathcal{F}_v \to \mathcal{F}_w$ and S can be described by the input-output relation

$$(w(0), w(1), w(2), \ldots, w(k)) = S \circ (v(0), v(1), v(2), \ldots, v(k)).$$

It has proven useful to bypass the input-output relation by using the concept of a state. A state $s(k)$ of a system at time k comprises all information needed to determine $w(k')$ for $k' \geq k$ based on $v(k')$. The state space $\mathcal{S}$ of a system is the set of all possible states of the system. Using a state transition function $f : \mathcal{S} \times \mathcal{V} \to \mathcal{S}$ and an output function $h : \mathcal{S} \times \mathcal{V} \to \mathcal{W}$, we substitute the input-output relation with a recursive state space representation of the system in discrete time by

$$\begin{aligned} s(k+1) &= f(s(k), v(k)), \qquad s(0) = s_0, \\ w(k) &= h(s(k), v(k)) \end{aligned}$$

where s_0 is the initial state of the system.

Let an event be a spontaneous occurrence triggered by some external entity (event generator, nature) or by fulfillment of all conditions of a switching rule (event generation rule). Because events can be used to submit changes of the input function value to the system over time, the following definition is obtained:

Definition 2.44 (Discrete event system ([43], [126])).
A discrete event system (DES) is an event-driven system whose state trajectory depends only on the occurrence of discrete events over time. △

Figure 2.11 subsumes the causal relationships in a discrete event system. Note that events may result from the output function and feed back as an event to the input function.

In online optimization, the release of input elements occurs at discrete points in time and an online algorithm operates as part of a reactive planning system. Hence, the solution process in online optimization can be modeled as a DES. Because we associate an objective value to an input sequence processed by an algorithm, we embed an entry for the objective value of

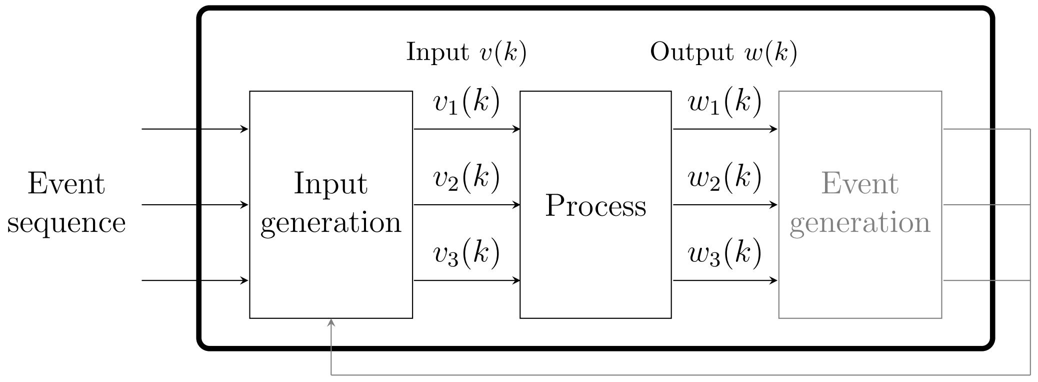


Figure 2.11: Event-based input-output model of discrete event systems. (Source: [126])

the current state into the state encoding scheme. Table 2.2 summarizes analogies between the solution procedure in online optimization and the behavior of discrete event systems.

Online optimization	Discrete event system
Input element	Event
Input sequence	Event sequence
Input element release	Event occurrence
Releases occur at discrete times	Events occur at discrete times
Algorithm reacts to input element releases	System reacts to event occurrences
Online algorithm	State transition function
Objective value	Entry in state encoding scheme

Table 2.2: Analogies between online optimization and discrete event systems.

As a result of their similarities, both online algorithms and discrete event systems can be tackled using the same arsenal of analytical methods.

Automata are the basic logical model to replicate discrete system behavior over time. They allow to track state transitions and resulting state trajectories over time. However, they do not provide information on frequencies for transitions or trajectories leading to a certain state.

Markov chains extend logical models with stochastic information. Their analysis gives information about state frequencies and yields mathematical expressions for quantities of interest. Recall that by imposing a uniform distribution over the set of input elements (as the maximum entropy distribution), we obtain deterministic counting results.

Discrete event simulation is applied for most real world problem settings since it is impossible to use exact methods like automata or Markov chains due to intractable complexity and state space explosion for increased problem size and level of detail.

Due to their logical character with regard to the order of events, both basic automata and Markov chains are restricted to the analysis of online optimization algorithms in the sequential model. In order to comply with the time stamp model, timed versions of automata and Markov chains are needed. These are obtained by additionally imposing a clock structure on the DES. We will not further discuss this topic in the sequel.

2.4.2 Automata

Automata theory provides a formal means to study the behavior of a discrete event system over time. The set of all events can be regarded as an alphabet and event sequences correspond to words which form the input to an automaton. The set of all words that can be processed by an automaton is called the language of this automaton. Informally, an automaton is a device capable of processing the words of its language. An automaton is the simplest representation of an event-driven state transition mechanism.

Definition 2.45 (Deterministic automaton ([43], [126])).
A deterministic automaton $\mathcal{A}$ is a quintuple $(\mathcal{S}, \mathcal{E}, f, s_0, \mathcal{S}_F)$ where $\mathcal{S}$ is the set of states, $\mathcal{E}$ is the set of events, $f : \mathcal{S} \times \mathcal{E} \to \mathcal{S}$ is the state transition function, $s_0 \in \mathcal{S}$ is the initial state and $\mathcal{S}_F \subseteq \mathcal{S}$ is the set of final states. △

A deterministic automaton can be represented by a graph: Nodes correspond to the states of the automaton; edges represent transitions between states and are labeled with the events which induce the state transition. The event-driven approach becomes apparent by using automata to process an event sequence resulting in a state sequence as seen in Figure 2.12.

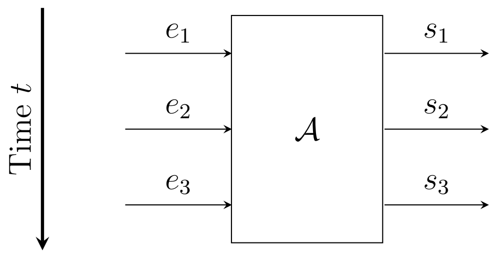


Figure 2.12: Automaton as a formal means to process elements of an event sequence. (Source: [126])

The behavior of an automaton for a given event sequence $(e_0, e_1, e_2, \ldots, e_K)$ with $K \in \mathbb{N}$ when starting in initial state $s_0 \in \mathcal{S}$ is specified by the sequence of state transitions

$$s_{k+1} = f(s_k, e_k), \qquad e_k \in \{e \in \mathcal{E} \mid f(s_k, e) \text{ is defined}\}, \quad k = 0, 1, \ldots, K.$$

Although instances in online optimization are uncertain, a deterministic online algorithm processes any instance deterministically. Thus, it suffices to restrict attention to deterministic automata. In order to describe the behavior of an online algorithm during input element processing by a deterministic automaton, we have to perform the following steps:

1. Define the set of input elements which are translated into event set $\mathcal{E}$.
2. Find a representation of the system state which is translated into state space $\mathcal{S}$.
3. Determine the behavior of the online algorithm when it encounters a given input element in a given state which is translated into state transition function f.
4. Determine the initial system state s_0 and, if necessary, the set of final states $\mathcal{S}_F$.

Analysis methods focus on properties that can be inferred from reachability between states: Given an automaton $\mathcal{A}$ with initial state s_i, it is asked for the set of reachable states $\mathcal{S}_R$ where state s_j is called reachable from s_i if there exists an event sequence such that the sequence of induced state transitions leads from s_i to s_j. The concept of reachability coincides with reachability in graph theory and can be reduced to finding a path between s_i and s_j in the graph representation of $\mathcal{A}$. Introduce the adjacency matrix $A = (a_{ij})$ with $i, j \in \mathcal{S}$ as

$$a_{ij} = \begin{cases} 1, & \text{if there exists } e \in \mathcal{E} \text{ with } f(i,e) = j \\ 0, & \text{else,} \end{cases}$$

and denote by $A^k = (a_{ij}^k)$ the kth power of A for $k \in \mathbb{N}$. Because in a connected graph with N nodes any node can be reached traversing at most $N-1$ edges, we have that by determining A^{N-1} all reachability relations between states of the automaton are known: s_j is reachable from s_i if and only if $a_{ij}^{N-1} \neq 0$.

With regard to the performance ratio analysis of online optimization algorithms, we consider the product coupling $\mathcal{A}_\times = \mathcal{A}_1 \times \mathcal{A}_2$ of two automata $\mathcal{A}_1, \mathcal{A}_2$. The product coupling of two automata is used when both automata are synchronized, i.e., they have to respond to a common sequence of events (cf. Figure 2.13). Comparing two online algorithms on the same input sequence, this condition is given whenever the input sequence does not depend on previous decisions.

Formally, the product coupling $\mathcal{A}_\times$ of $\mathcal{A}_1 = (\mathcal{S}_1, \mathcal{E}, f_1, s_{0,1}, \mathcal{S}_{F,1})$ and $\mathcal{A}_2 = (\mathcal{S}_2, \mathcal{E}, f_2, s_{0,2}, \mathcal{S}_{F,2})$ is given as $\mathcal{A}_\times = (\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{E}, f_\times, (s_{0,1}, s_{0,2}), \mathcal{S}_{F,1} \times S_{F,2})$ where a state is a pair $(s_1, s_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ and the state transition function $f_\times$ is recast as

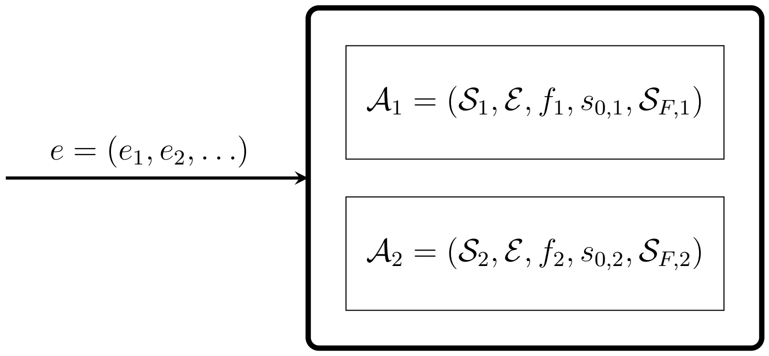

$e = (e_1, e_2, \ldots)$
$\mathcal{A}_1 = (\mathcal{S}_1, \mathcal{E}, f_1, s_{0,1}, \mathcal{S}_{F,1})$
$\mathcal{A}_2 = (\mathcal{S}_2, \mathcal{E}, f_2, s_{0,2}, \mathcal{S}_{F,2})$

Figure 2.13: Coupling of two automata. (Source: [126])

$$f_\times((s_1, s_2), e) = \begin{cases} \big(f_1(s_1, e), f_2(s_2, e)\big) & \text{if } f_1(s_1, e) \text{ and } f_2(s_2, e) \text{ are defined,} \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

Applying the product-coupling on two online algorithms permits a simultaneous tracking of the state evolution over time.

In preparation for the discussion of Markov chains, we introduce the notions of an autonomous deterministic automaton and an autonomous nondeterministic automaton.

Definition 2.46 (Autonomous deterministic automaton ([43], [126])).
An autonomous deterministic automaton $\mathcal{A}$ is a quadruple $(\mathcal{S}, f, s_0, \mathcal{S}_F)$ where $\mathcal{S}$, s_0, $\mathcal{S}_F$ are the same as in a deterministic automaton and $f : \mathcal{S} \to \mathcal{S}$ is the state transition function. △

State transitions of an autonomous automaton are not associated to the occurrence of events but rather happen without a certain reason out of nowhere.

Definition 2.47 (Autonomous nondeterministic automaton ([43], [126])).
An autonomous nondeterministic automaton $\mathcal{A}$ is a quadruple $(\mathcal{S}, f_{rel}, s_0, \mathcal{S}_F)$ where $\mathcal{S}$, s_0, $\mathcal{S}_F$ are the same as in an autonomus deterministic automaton and $f_{rel} : \mathcal{S} \to 2^{\mathcal{S}}$ is the state transition relation. △

The state transition relation $f_{rel}(\cdot)$ of a nondeterministic automaton gives a set of possible successor states from the power set $2^{\mathcal{S}}$ of $\mathcal{S}$ for some given state. Notice that we obtain an autonomous nondeterministic automaton from any deterministic automaton simply by omitting the set of events Σ in the definition of the state transition function. In the automaton graph this amounts to removing the edge labels (events) suggesting that the state transition is carried out as a result of some random mechanism rather than as a result of an event.

2.4.3 Markov Chains

Besides analyzing state reachability, Markov chains allow for analyzing the frequency of encountering a given state by incorporating probabilistic information on state transitions. This approach is particularly meaningful for representing the performance distribution of algorithms as it allows to track the objective value and performance ratio distribution.

Let $P(A)$ denote the probability of some event A and let S_k be a random variable for the state at time k in a discrete event system with time instants $\mathbb{N}_0$. A Markov chain is obtained from an autonomous nondeterministic automaton by replacing the state transition relation f_{rel} with a probability distribution $G_{prob} : \mathcal{S} \times \mathcal{S} \to [0,1]$ such that

$$g_{ss'} := G_{prob}(s, s') = P(S_{k+1} = s' \mid S_k = s)$$

gives the conditional probability that the system reaches state s' at time $k+1$ when it has already reached state s at time k. We have to impose that

$$\sum_{s' \in \mathcal{S}} g_{ss'} = 1, \qquad s \in \mathcal{S},$$

to ensure that $G_{prob}(s, \cdot)$ is a probability distribution. In a discrete event system, uncertainty is quantified by stochastic information on the occurrence of events leading to a state transition. When no stochastic information is given, imputing the uniform distribution yields deterministic counting results ([90]).

Definition 2.48 (Markov chain ([43], [126])).
A Markov chain $\mathcal{A}$ is a quadruple $(\mathcal{S}, G_{prob}, p_0, \mathcal{S}_F)$ where $\mathcal{S}$, $\mathcal{S}_F$ are the same as in an autonomous deterministic automaton, $G_{prob} : \mathcal{S} \times \mathcal{S} \to [0,1]$ is the state transition probability distribution and p_0 is a probability distribution on the initial state. △

A Markov chain is also called an autonomous stochastic automaton.

Denote by $p(k)$ the vector whose ith element $p_i(k) := P(S_k = i)$ gives the probability that the system takes on state i at time k. Using matrix $G = (g_{ij})$ with

$$g_{ij} := P(S_{k+1} = j \mid S_k = i),$$

we obtain the recursive relation

$$p(k+1) = Gp(k)$$

known as the Chapman-Kolmogorow equation. Hence, a Markov chain models a discrete time linear system and with the initial state probability distribution p_0 we have that $p_k = G^k p_0$ computes the state probabilities at time k. Using the uniform distribution over all events or state transitions, respectively, we obtain deterministic counting results with regard to how often a state is reached after a fixed amount of time has expired.

A system is said to have the Markov property if it holds that

$$P(S_{k+1} = s_{k+1} \mid S_k = s_k, \ldots, S_0 = s_0) = P(S_{k+1} = s_{k+1} \mid S_k = s_k)$$

for all $k \geq 0$. The Markov property states that the successor state solely depends on the current state, but on none of the states prior to that. It can be shown that a system has to fulfill the Markov property in order to establish a recursion in the form of the Kolmogorow-Chapman equation which gives rise to the term *Markov chain* as a synonym for *autonomous stochastic automaton*. The Markov property has also coined the notion of a memoryless system due to the system's oblivion to past states before the current state.

Classical Markov chain analysis is concerned with finding the steady state distribution, if it exists. We are interested in a Markov chain's transient behavior which yields information about the state distribution after a given number of time steps ([90]). In most Markov chain models of online optimization, no steady state distribution exists due to the objective value being a part of the state encoding. We use Markov chains to obtain frequency information concerning states with a prescribed objective value. For the performance ratio of an algorithm relative to another, the state space of an associated Markov chain would need to comprise the Cartesian product of the state space used for the analysis of the objective value of one algorithm as outlined in Chapter 2.4.2.

2.4.4 Discrete Event Simulation

According to [159], "simulation is the reproduction of a system along with its dynamic processes in an executable model in order to retrieve results which are transferable to reality". An execution of the simulation with specified input parameters is called simulation run or replication. Results on system performance are derived from first collecting data over a sufficient number of simulation runs which then serve as the computational basis so as to determine (point, interval or distributional) estimates for several performance measures of interest. In an experiment, we define a sufficiently large replication set to increase the likelihood for results of high representativeness. Besides experiment design, we have to take into account in the model design that each involved stochastic process is associated with a separate stream of random numbers such that any sort of dependency is ruled out.

Whenever a real world system shall be analyzed but exact analysis is out of reach due to a high degree of complexity and a large number of dependent stochastic processes, simulation represents an appropriate tool to provide estimates for desired performance measures. Although the approximation character in the outcomes may be deprecating, there are quite a number of advantages:

- Simulation may be the only way to analyze the behavior of complex systems with multiple dependent random variables and manifold types of events such as input element releases, machine breakdowns, occurrence of erroneous data or erroneous operations.
- Sufficiently many simulation runs allow to obtain statistically sound results.
- Simulation allows to track manifold quantities of interest as intended by multicriteria approaches in optimization.
- Simulation as an abstract model of reality is comparatively risk-free and cheap.

The event calendar is the pivotal element of a discrete event simulation model. It can be thought of as a list of pending events that are fed into the discrete event system over the course of a simulation run. Hence, the event calendar at time t is a list of events $e_k, e_{k+1}, \dots, e_{k_{\max}}$

$$L_t = \big((e_k, t_k), (e_{k+1}, t_{k+1}), \dots, (e_{k_{\max}}, t_{k_{\max}})\big)$$

with elements ordered according to non-decreasing occurrence times $t_k, t_{k+1}, \dots, t_{k_{\max}}$. We take care of state trajectories by using a state transition function $f : \mathcal{S} \times \mathcal{E} \to \mathcal{S}$ where $\mathcal{S}$ corresponds to the state space and $\mathcal{E}$ corresponds to the event set.

A run of a discrete event simulation model comprises the steps displayed in Algorithm 2.1 and any implementation of a discrete event simulation software amounts to a generic implementation of these steps. A simulation study then consists of building a model for the simulation and executing Algorithm 2.1 with respect to the simulation model for a sufficient number of times with independent seeds. Clearly, the main contribution in a simulation study consists in building the right model for the right purpose. Therefore, modeling requires continuous surveillance using validation and verification methods ([141]).

There are two possible hierarchical relations between simulation and optimization ([135], [119]): First, optimization appears as a subroutine module in the simulation; second, simulation is used as a multi-criteria evaluation function for an optimization method. Due to incremental decision making in online optimization, we are concerned with the former

Algorithm 2.1 Discrete event simulation run

Input: Seed value list S, initial state s, initial event list E, set of statistics, ending condition

```
1: Initialize random number generators with seed values from $S$
2: Initialize system state with $s$
3: Initialize event calendar with initial event list $E$
4: while ending condition not fulfilled do
5:     Remove first entry $(e, t)$ from event calendar
6:     Advance to time $t$
7:     Update system state according to state transition function $s := f(s, e)$
8:     Update statistics
9:     Delete $(e, t)$ and all infeasible events from event calendar
10:    Generate new events and add them to event calendar
11:    Reorder elements in the event calendar by non-decreasing event times
12: end while
```

Output: Terminal state s, values of statistics

relation where decisions from an optimization module are iteratively requested within the simulation model at the corresponding decision points of the function logic in the real world model. In Algorithm 2.1, a decision elicited from an optimization method is manifested in the discrete event simulation run by a state transition according to that decision in line 7.

Real world systems from production and logistics are inherently complex and contain optimization problems exhibiting an online character at several control points. Discrete event simulation combined with methods from online optimization represents a powerful tool in the design (or reorganization) phase of such systems as it allows to evaluate the performance of algorithms and to identify the most promising algorithm with respect to the user's requirements on system performance. As a major disadvantage, it is much more difficult to gain structural insights on why a system behaves the way it does from simulation experiments than from exact analysis methods. After the simulation model is finished, it is often just used as a black box to evaluate sample state trajectories of the system.

2.5 Concluding Discussion

This chapter provided a clear definition of the term *online optimization with lookahead* in order to ensure a common understanding of this optimization concept not only throughout this thesis but also throughout different problem settings in general. We clarified the relations between the different optimization paradigms and examined their implications on computational complexity as well as on algorithm analysis. It became evident that – due to the inevitability of failure in online optimization – algorithms in this field should not be judged based only on the almighty yardstick of offline optimization. Therefore, we reviewed the vast majority of alternative performance measures proposed in literature alongside their advantages and disadvantages. This led to the conclusion that more comprehensive methods of performance analysis are indispensable, especially regarding algorithms for online optimization with lookahead. We suggested a combined usage of the counting distribution functions for the performance ratio and for the objective function value which together submit a thorough picture of an algorithm's behavior in a given problem context. From the analogy between discrete event systems and the solution process in online optimization it became clear that exact analysis is out of scope whenever problem sizes become too large and dependencies of random variables too complex. Overall, we gathered a set of potential analysis tools for algorithms in online optimization with lookahead as displayed in Figure 2.14.

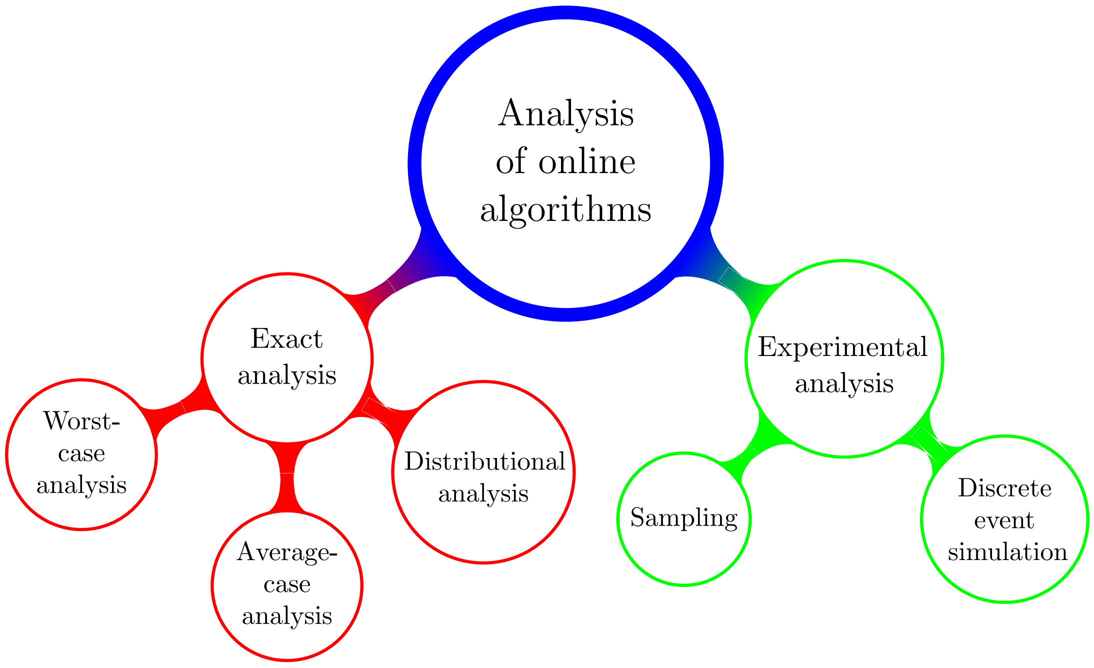


Figure 2.14: Analysis methods for online algorithms.

Since algorithms as solution methods for arising instances of online optimization problems are not addressed explicitly in the analysis methods of discrete event systems, we next develop a model of discrete event systems that explicitly takes into account sequential decision-making by algorithms.

3 A Modeling Framework for Online Optimization with Lookahead

In this chapter, we consider how a system whose function logic iteratively requires solving subproblems of an online optimization problem instance can be tackled formally. We propose a generic framework in order to harmonize concepts and notation throughout problems and applications. Its instantiations specify the constraints and the potential workflow that the system under consideration has to undergo while processing an input sequence using an algorithm with lookahead capabilities. The framework intends to facilitate systematic algorithm analysis based on a common understanding of the solution process paradigm and lookahead mechanism. In contrast to discrete event systems, our framework explicitly accounts for an integration of an algorithm's decision making and available lookahead.

3.1 Modeling Prototypes

A first attempt to model online optimization problems generically were request answer games as introduced at the beginning of the previous millenium's last decade by Ben-David et al. ([21]). A sequence of requests is presented online and each time a new request arrives, an adequate answer has to be given incurring some cost.

Formally, a request answer game over time horizon $n \in \mathbb{N}$ is a triple $(\mathcal{R}, \mathcal{A}, \mathcal{C})$ where $\mathcal{R}$ is a set of requests, $\mathcal{A}$ is a set of answers and $\mathcal{C} = \{c_n \,|\, n \in \mathbb{N}\}$ is a set of cost functions with $c_n : \mathcal{R}^n \times \mathcal{A}^n \to \mathbb{R} \cup \{\infty\}$. In this model, a deterministic online algorithm $\textsc{Alg} = (\textsc{Alg}_1, \textsc{Alg}_2, \ldots, \textsc{Alg}_n)$ is a sequence of functions with $\textsc{Alg}_i : \mathcal{R}^i \to \mathcal{A}$ for $i = 1, 2, \ldots, n$. The output of an algorithm upon request sequence $r = (r_1, r_2, \ldots, r_n) \in \mathcal{R}^n$ is a sequence of answers $a = (a_1, a_2, \ldots, a_n) \in \mathcal{A}^n$ with $a_i = \textsc{Alg}_i(r_1, r_2, \ldots, r_i)$ for $i = 1, 2, \ldots, n$. The costs incurred by $\textsc{Alg}$ upon processing r are given by $c_n(r, a)$.

Although request and answer games cover many online optimization problems, the concept could not be established as a modeling standard due to its lacking ability to account for the rationale in an algorithm's decision making. Clearly, the concept of a state is needed for this purpose. Request and answer games are mainly used in order to distinguish between several types of adversaries in competitive analysis of randomized online algorithms ([21], [32]).

Another proposition for a generic model of online optimization origins from the priority programme *Online Optimization of Large Scale Systems* of the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) from 1996 to 2001 ([84]). On an informal basis, Grötschel et al. ([85]) propose the sequential model and the time stamp model of online optimization as refurbished in Definitions 2.14 and 2.15, respectively.

In both models, an input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ is revealed over time and an algorithm has to serve each of the input elements. In the sequential model, σ_{i+1} is presented only when σ_i has just been served, i.e., input elements are processed according to the first-in first-out priority rule. In the time stamp model, each input element σ_i comes along with a release time τ_i which is independent of an algorithm's previous actions. The release time represents the earliest point in time where σ_i may be served. Hence, in complete opposite to the sequential model, the order of service is not fixed initially. Under the time stamp model, decisions have a tentative character as long as they have not been executed and may be revoked until then. As a result of an inclusion of the factor time, different solution strategies arise for an algorithm: Waiting may become lucrative and decisions may be made at arbitrary points in time.

Both for the sequential and time stamp model, no extension to the lookahead case was suggested. In particular, incorporation of lookahead would require a more concise statement about the restrictions on the service order and on the earliest possible times for service which would arise due to lookahead. A subdivision of lookahead into an informational and a processual component as outlined in Chapter 2.1.3 would be the logical consequence.

For classical disciplines in optimization, the working schemes of exact algorithms are known: In mixed integer linear programming, the branch and bound method is established and research concentrates on improving or extending the method by valid inequalities or branch and cut procedures. In dynamic programming, value iteration, policy iteration or linear programming are known to work as a result of the Bellman equation and research focuses on how to apply the methods in practice, e.g., by approximative approaches. In contrast, no online optimization algorithm can be exact; as a result, there is no basic starting point for further research as outlined for the other fields. We believe that the introduction of a formal modeling framework is at least a first step to find a common basis of understanding.

3.2 Modeling Framework Components

We recognize that previous models of online optimization lack an integration of lookahead and a representation of the solution process for a given problem instance. In particular, the sequential character of decision making by an algorithm in form of reactive planning upon notification of new input elements needs a dedicated formulation. In the following sections, we build a generic modeling framework for online optimization problems with lookahead; it comprises four building blocks:

- The *basic modeling elements* provide an abstract view on the infrastructure of information flows needed to model the solution process.
- The *lookahead type* specifies the instance revelation rule that an algorithm under lookahead has to obey in contrast to the reference online case.
- The *processing characteristics* specify the processing rules that an algorithm under lookahead has to obey in its decision making on how to serve the input elements in contrast to the reference online case.
- The *algorithm execution mode* defines the time instants at which algorithm evaluation is scheduled over the course of the solution process.

Instantiations of the four building blocks are combined in a generic process model which describes the interaction of the modeling elements over time as a result of the specific forms of lookahead type, implied processing characteristics and prescribed algorithm execution mode. The process model is of a discrete event nature because an algorithm has to respond to a sequence of input elements arriving at discrete points in time ([68]).

3.2.1 Basic Modeling Elements

We first identify the constituting elements which are needed in each model of an online optimization problem with lookahead. To this end, we consider the problem as part of a dynamic system: We let the system under consideration be the abstracted collection of all real world entities needed to describe the optimization problem, and we embed the system in a time horizon with current time denoted by t.

Input element and input sequence

The smallest unit of information is given by an input element from the set of all possible input elements Σ_1 (which is the set of all input sequences of length 1). The input appears in form of an input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ consisting of input elements $\sigma_i \in \Sigma_1$ for $i = 1, 2, \ldots$ These are invisible at first, but then revealed successively according to the lookahead mechanism. Each input element awaits some processing during the solution process. The processing of an input element can be understood as the result of a decision (e.g., in form of a decision variable assignment) on what to do with it in order to comply with all restrictions and to contribute to the optimization goal.

An input element σ_i is a data record containing all relevant information needed by an algorithm to decide on the processing of σ_i. On the most granular level, it is associated with the following atomic parts:

- Release time $\tau_i := \tau(\sigma_i)$ in the reference online optimization problem
- Release time $\tau'_i := \tau'(\sigma_i)$ in the online optimization problem with lookahead
- Processing time interval $[\underline{T}_i, \overline{T}_i] := [\underline{T}(\sigma_i), \overline{T}(\sigma_i)]$ in the reference online optimization problem
- Processing time interval $[\underline{T}'_i, \overline{T}'_i] := [\underline{T}'(\sigma_i), \overline{T}'(\sigma_i)]$ in the online optimization problem with lookahead
- Input element information $r_i := r(\sigma_i)$

The only requirement on release times is $\tau'_i \leq \tau_i$. The processing time intervals $[\underline{T}_i, \overline{T}_i]$ and $[\underline{T}'_i, \overline{T}'_i]$ give the times at which input element σ_i is allowed to be (physically) processed in the absence and presence of lookahead, respectively. Many combinatorial online optimization problems have $\underline{T}_i = \overline{T}_i$ and $\underline{T}'_i = \overline{T}'_i$; in this case, we can also use T_i and T'_i as the time instants where processing is due to be executed. r_i carries the essential information of σ_i which is needed for decision making (e.g., the size of an item in bin packing or the location of a request in the traveling salesman problem).

We associate two time-dependent variables with each input element σ_i:

- Processing status $p_i(t) := p(\sigma_i, t) \in \{\text{unprocessed, processing, finished}\}$
- Action $a_i(t) := a(\sigma_i, t)$

Processing status $p_i(t)$ gives the state of σ_i at time t and rules membership to the sets of still unprocessed, currently processing or already finished input elements. Action $a_i(t)$ is a data record with all information concerning how σ_i is, was, or is planned to be processed. $a_i(t)$ is determined by an algorithm; if no action has been determined yet at time t, we set $a_i(t) = \textsc{Null}$. Whenever $a_i(t) \neq \textsc{Null}$, we have that $a_i(t)$ contains at least the elements $t_i^{a,start}$ and $t_i^{a,finish}$ denoting the time instants when processing of σ_i is (planned to be) started and finished, respectively. Decisions made previously may be revoked until the current time t reaches $t_i^{a,start}$. The set of all possible actions for an input element is denoted by $\mathcal{A}$.

There are no additional sources of incomplete information with respect to σ_i once it has been notified, i.e., it is given in its entirety as soon as its existence emerges. Each input element experiences two steps with respect to its information: First, all information of an input element is announced at its release time. Second, this information is exploited by an algorithm to determine how an input element will be processed. The processing itself, of course, takes place between $t_i^{a,start}$ and $t_i^{a,finish}$ without any further ado.

We remark that there are different types of events besides the arrival of new input elements in real world applications such as system breakdowns, erroneous inputs or other kinds of unforeseeable disturbances. In these cases, it must be ensured that input sequence σ can accommodate input elements of different event classes.

Lookahead set

At each time t, the processing statuses of the known input elements establish a partition of all available input elements into sets U_t, P_t and F_t of unprocessed, currently processing and finished input elements, respectively. In the reference online optimization problem, we have

- $U_t := \{\sigma_i \,|\, \tau_i \leq t, p_i(t) = \text{unprocessed}\}$
- $P_t := \{\sigma_i \,|\, \tau_i \leq t, p_i(t) = \text{processing}\}$
- $F_t := \{\sigma_i \,|\, \tau_i \leq t, p_i(t) = \text{finished}\}$

In an online optimization problem with lookahead, τ_i is replaced with τ_i'. The following equivalences rule membership of σ_i to these sets:

- $p_i(t) = \text{unprocessed} \;\Leftrightarrow\; a_i(t) = \textsc{Null} \,\vee\, t_i^{a,start} > t$
- $p_i(t) = \text{processing} \;\Leftrightarrow\; t_i^{a,start} \leq t < t_i^{a,finish}$
- $p_i(t) = \text{finished} \;\Leftrightarrow\; t_i^{a,finish} \leq t$

The lookahead set L_t at time t is the collection of those input elements which have been revealed, but still require (some amount of) processing, i.e., $L_t := U_t \cup P_t$. Thus, L_t depends on previous decisions and actions caused by an algorithm and not on a distinction between the release times in the cases of absent and present lookahead. Although it might be tempting to define lookahead as the set difference between the input elements known in the reference online case and those known in the lookahead case, this approach has no practical benefit as it is oblivious to the input elements which are at an algorithm's disposal. Note that an input element can stay in L_t arbitrarily long and suffer starvation if its processing is continuously rejected in favor of another input element.

State space

For a better representation of the dependencies between an algorithm's decision and the current state, we decompose each state s from the set of all states $\mathcal{S}$ into three components $s = (s^{in}, s^{sys}, s^{obj})$ with input state s^{in}, system state s^{sys} and objective state s^{obj}. All information concerning the input of the problem at time t is collected in input state $s_t^{in} = (U_t, P_t, F_t)$; the set of all input states is denoted by $\mathcal{S}^{in}$. Because s_t^{in} contains all σ_i with $\tau_i \leq t$ or $\tau_i' \leq t$, it also contains all action variables $a_i(t)$ for these input elements. To describe external circumstances which have to be taken into account by an algorithm, the system state s_t^{sys} at time t is a data record containing all information to describe the system configuration (e.g., bin configurations in bin packing or the current server position in the traveling salesman problem) at time t; the set of all system states is denoted by $\mathcal{S}^{sys}$. In the context of optimization, we have to use valued states to keep track of the current objective value during the solution process. At time t, we extract all information relevant to the future development of the objective value (e.g., the plain current objective value) in objective state s_t^{obj}; the set of all objective states is denoted by $\mathcal{S}^{obj}$. The set of all states is $\mathcal{S} = \mathcal{S}^{in} \times \mathcal{S}^{sys} \times \mathcal{S}^{obj}$; concerning the solution process of an online optimization problem with lookahead, we are interested in the state trajectory evolution $(s_t)_{t \geq 0}$ with $s_t \in \mathcal{S}$ for $t \geq 0$.

Event space

Because in online optimization an algorithm has to respond to arriving input elements, we classify the arrival of a new input element as an event. Likewise, finished processing may change the objective state such that the end of an input element's processing also represents an event. In total, all state transitions which are of interest in monitoring the solution process of an online optimization problem are triggered by the occurrence of an event. We subsume all possible events in the event set $\mathcal{E}$. Events occur at discrete time instants and have a duration of zero. In our framework, events are the only source of uncertainty which an algorithm has to cope with. Note that both the sequential and time stamp model of online optimization only know two types of events referring either to finished processing of an input element or to notification of a new input element; both types coincide in the sequential model because a new element only becomes available when a known one finishes processing.

Algorithm

Input element processing is controlled by the decisions of an algorithm, ultimately causing changeovers of input elements between sets U_t, P_t and F_t as imputed by the algorithm's decisions. Recalling that in online optimization the overall solution is composed of a sequence of partial solutions, an algorithm $\textsc{Alg}$ is used to successively produce partial solutions by determining the values of the input elements' action variables. Let $n_t := |U_t \cup P_t| = |L_t|$ be the number of unfinished known input elements, then an online algorithm $\textsc{Alg}$ is a family of functions $\textsc{Alg} := (\textsc{Alg}_t)_{t \geq 0}$ where $\textsc{Alg}_t : \mathcal{S} \times \mathcal{E} \to \mathcal{A}^{n_t}$ is a function determining for each of the n_t known and yet unfinished input elements an action from the action space $\mathcal{A}$ based on current state $s \in \mathcal{S}$ and occurring event $e \in \mathcal{E}$. Hence, our definition of an algorithm generalizes that of the request answer games from Chapter 3.1 in terms of the dependency on the current state and the multidimensionality of the codomain. $\textsc{Alg}_t$ is evaluated at each time instant t where an event occurs.

State transition

The state transition function $f : \mathcal{S} \times \mathcal{E} \to \mathcal{S}$ determines for a given state $s \in \mathcal{S}$ and an occurring event $e \in \mathcal{E}$ the successor state $s' \in \mathcal{S}$. It only needs to be evaluated at the discrete time instants where an event occurs because for all other times the state trajectory is assumed to advance deterministically, i.e., it can be precomputed.

3.2.2 Lookahead Type

The lookahead type specifies the mechanism under which the membership of input elements to lookahead set L_t is governed for all t. According to the taxonomy from Chapter 2.1.3, it corresponds to the instance revelation rule of online optimization with lookahead. Which lookahead type is employed depends on the application under consideration and on the technical possibilities. We give some frequently used types of lookahead in literature, harmonize them with our notation and introduce property lookahead as a generalization of all instance revelation rules which allow for a property-related characterization of all known input elements. Recall that a key characteristic in the sequential model is the processing-dependent release of input elements: Only when an input element finishes processing, a new one becomes known; in the time stamp model, input elements become known independently of previous decisions and an online algorithm is allowed to wait and revoke decisions as long as they have not been implemented. Despite of that, also the sequential model can incorporate temporal aspects in the specification of input element σ_i. A similar classification of lookahead types as described subsequently has also been given by Tinkl ([155]).

Request lookahead

Request lookahead of size $k \in \mathbb{N}$ ([155]) is defined to have access to a fixed number k of unprocessed or currently processing input elements, or to all of the remaining unprocessed and currently processing input elements if there are less than k of them. The first of these input elements is also known in the pure online situation, but the remaining $k-1$ input elements or all remaining input elements if there are less than k of them are known due to the lookahead capability. Request lookahead construes the lookahead set dependent on the processing statuses of the input elements and not in an independent process of release. We obtain the release times recursively from

$$\tau_i := \begin{cases} 0, & \text{if } i = 1, \\ \min\{t_j^{a,finish} \mid \sigma_j \in L_{\tau_{i-1}}\}, & \text{if } i = 2, 3, \dots \end{cases}$$

in the case without lookahead and

$$\tau_i' := \begin{cases} 0, & \text{if } i = 1, \dots, k, \\ \min\{t_j^{a,finish} \mid \sigma_j \in L_{\tau_{i-1}'}\}, & \text{if } i = k+1, k+2, \dots \end{cases}$$

in the case with lookahead. Request lookahead origins from applications where time is not modeled explicitly such that request lookahead is understood only in the sequential model. Request lookahead is not possible in the time stamp model because an arbitrary number of input elements may be released at any time contradicting the requirement to hold at most k input elements. For applications which adhere to capacities, such as storage devices or inventory spaces, request lookahead is realistic, whereas for applications which deal with immaterial requests, such as emergency calls or customer demands, it is unrealistic.

Under request lookahead, input elements are revealed in a rolling time horizon. However, one can also think of situations where information is released rather in batches or blocks than steadily over time: A new batch is only given when all elements of the previous one have finished processing. This gives rise to a batched version of this lookahead variant. In batched request lookahead of size k, input elements are always revealed in blocks of k elements if there are more than k elements left, otherwise the remaining input elements are revealed. In this case, we have

$$\tau_i' := \begin{cases} 0, & \text{if } i = 1, \dots, k, \\ \max\{t_j^{a,finish} \mid \sigma_j \in L_{\tau'_{ck}}\}, & \text{if } i = ck+1, ck+2, \dots, ck+k \text{ with } c \in \mathbb{N}. \end{cases}$$

Time lookahead

Time lookahead of length $D \in \mathbb{R}^{>0}$ ([155]) essentially makes input element σ_i known D time units earlier in the online setting with lookahead than in the setting without lookahead. Hence, except for input sequences which are released before $t = D$, there is a fixed offset D between an input element's release time in the cases of present and absent lookahead. Release times τ_i in the pure online setting are assumed to be independent of an algorithm's processing and we obtain that $\tau_i' := \max\{\tau_i - D, 0\}$. Time lookahead can be seen as an artificial preponement of the moment at which an input element is notified by D time units. It is not admissible in the sequential model because the generation of input elements is an independent process. A drawback of time lookahead is that arbitrarily many input elements may reside in the lookahead set: When processing of a single input element extends over time, the workload may collapse in the long run. Thus, it is reasonable to see time lookahead in connection with stability enforcing constraints similar to those in queuing systems where the occupation rate is used to measure server utilization. It deserves mentioning that artificially diminishing the release time of an input element is conceptually different from allowing an algorithm to wait in order to accumulate input elements as we nominally make information available earlier in time. Though, the consequences and positive effects of delaying processing decisions to a later point in time are similar because the planning basis, i.e., the available data for a set of decisions, is extended in both situations (cf. also [7]).

Collective property lookahead

Collective property lookahead has been first introduced as a general concept in [155] although in bin packing (see, e.g., [83]) and paging (see, e.g., [2]) it had already been instantiated before. Let *c-prop* be a time variant property of an arbitrary subset $\sigma_{\leq j}(t)$ of the elements in input sequence σ where

$$\sigma_{\leq j}(t) := \{\sigma_i \mid i \in \{1, 2, \dots, j\},\ p_i(t) \neq \text{finished}\}.$$

Write $\textit{c-prop}_j(t) := \textit{c-prop}(\sigma_{\leq j}, t)$ if $\sigma_{\leq j}(t)$ fulfills *c-prop* at time t and $\neg\textit{c-prop}_j(t)$ otherwise. In addition, we assume for $\sigma_i, \sigma_j, \sigma_k \in L_t$ that $\neg\textit{c-prop}_k(t)$ holds for all $k > j$ whenever $\neg\textit{c-prop}_j(t)$ and that $\textit{c-prop}_i(t)$ holds for all $i < j$ whenever $\textit{c-prop}_j(t)$. Without a collective property lookahead device for *c-prop*, any algorithm is oblivious to *c-prop* and σ_i is released according to the instance revelation rule in the online case; collective property lookahead makes all input elements that comply with *c-prop* visible to an algorithm. Collective property lookahead is defined to have access at time t to a largest possible subsequence of unfinished input elements such that all input elements of this subsequence collectively fulfill property *c-prop* at time t. The first time t where $\textit{c-prop}_i(t)$ is fulfilled is set to be the preponed release time of σ_i. Thus, in the online setting with lookahead, we have $\tau_i' := \min\{t \mid \textit{c-prop}_i(t)\}$. The instance revelation rule of the pure online setting has to ensure that $\tau_i' \leq \tau_i$. Collective property lookahead can be admissible both in the sequential model and time stamp model depending on the definition of *c-prop* itself. Collective property lookahead is used when input elements collectively influence which part of the input sequence is seen, e.g., due to their combined size or weight.

Property lookahead

We introduce this type of lookahead as a generalization of all lookahead types that can be described explicitly. Let *prop* be a time variant property of each input element. Write $\textit{prop}_i(t) := \textit{prop}(\sigma_i, t)$ if σ_i fulfills *prop* at time t and $\neg\textit{prop}_i(t)$ otherwise. Without a property lookahead device for *prop*, any algorithm is oblivious to *prop* and σ_i is released according to the instance revelation rule in the online case; property lookahead makes all input elements that comply with *prop* visible to an algorithm. The first time t where $\textit{prop}_i(t)$ is fulfilled is set to be the preponed release time of σ_i. Thus, in the online setting with lookahead, we have $\tau_i' := \min\{t \mid \textit{prop}_i(t)\}$. The instance revelation rule of the pure online setting has to ensure that $\tau_i' \leq \tau_i$. Property lookahead can be admissible both in the sequential model and time stamp model depending on the definition of *prop* itself. Property lookahead is used whenever there is some device that allows to recognize input elements which fulfill the property. Note that property lookahead is a generalization of all lookahead types that rely on some rule which governs membership of input elements to the lookahead set.

Free lookahead

Free lookahead is applied whenever it is observed that input elements become known earlier than in the reference online setting, but the mechanism of preponed release of input elements is unknown or hidden to the user. The fundamental feature of free lookahead is that there is no relation known between τ_i and τ_i' other than $\tau_i' \leq \tau_i$.

3.2.3 Processing Mode and Order

According to Chapter 2.1.3, when input elements are processed, they are subject to the rules specified in (processing) rule sets P and P' of the online optimization problem in absence and presence of lookahead, respectively. Unfortunately, in literature it is never specified how P and P' exactly look like but tacitly assumed according to the problem context. We provide a classification which verbally expresses the essential difference between P and P'. By definition, the decisions of an algorithm on processing start and end times have to obey $\underline{T}_i \leq t_i^{a,start} \leq t_i^{a,finish} \leq \overline{T}_i$ in the pure online setting; in the online case with lookahead, $\underline{T}_i$ and $\overline{T}_i$ have to be replaced by $\underline{T}_i'$ and $\overline{T}_i'$, respectively.

Under lookahead, we have to deal with the questions whether we have to adhere to the order in the input element sequence also in processing (processing order) and whether more than one input element can be processed at a time (processing mode).

We explain combinations of the four possible processing modes

- single processing,
- parallel processing,
- limited parallel processing,
- property processing

with the two possible processing orders

- in-order processing,
- random-order processing.

Single in-order processing

In single in-order processing, we have to obey the given order of input element releases also when processing the input elements and we have to process them one after another. For the decisions of an algorithm on input element σ_i it has to hold that $t_i^{a,finish} \leq t_{i+1}^{a,start}$.

Single random-order processing

In single random-order processing, we are allowed to choose any available unfinished input element for being processed next at any time, but have to respect that only one input element can be processed at a time. For the decisions of an algorithm on two input elements σ_i and σ_j with $i \neq j$ it has to hold that $[t_i^{a,start}, t_i^{a,finish}) \cap [t_j^{a,start}, t_j^{a,finish}) = \emptyset$.

Parallel in-order processing

In parallel in-order processing, we have to obey the given order of input element releases also when processing the input elements, but we may process more than one input element at a time. For the decisions of an algorithm on input element σ_i it has to hold that $t_i^{a,start} \leq t_{i+1}^{a,start}$.

Parallel random-order processing

In parallel random-order processing, we are allowed to choose any available unfinished input element for being processed next at any time and we may process more than one input element at a time. There are no temporal restrictions on processing times which have to hold for the decisions of an algorithm on the input elements.

Limited parallel in-order processing

Limited parallel in-order processing is similar to parallel in-order processing, but additionally imposes that at most m input elements can be processed at a time. For the decisions of an algorithm on input element σ_i it has to hold that $t_i^{a,start} \leq t_{i+1}^{a,start}$ and additionally for $t \geq 0$ that $|\{\sigma_j \mid t \in [t_j^{a,start}, t_j^{a,finish})\}| \leq m$.

Limited parallel random-order processing

Limited parallel random-order processing is similar to parallel random-order processing, but additionally imposes that at most m input elements can be processed at a time, i.e., for the decisions of an algorithm it has to hold for $t \geq 0$ that $|\{\sigma_j \mid t \in [t_j^{a,start}, t_j^{a,finish})\}| \leq m$.

Property processing

In property processing, input elements eligible to be processed at time t are marked. Let *proc* be a time-dependent property of each input element; write $proc_i(t) := proc(\sigma_i, t)$ if σ_i fulfills *proc* at time t and $\neg proc_i(t)$ otherwise. Processing start times are coordinated such that $proc_i(t)$ is fulfilled when $t_i^{a,start} = t$ is chosen. For the decisions of an algorithm on input element σ_i it has to hold that $t_i^{a,start} \in \{t \geq 0 \,|\, proc_i(t)\}$. All previous processing modes and orders can be emulated by property processing using an adequate specification of *proc*.

3.2.4 Processing Accessibility

We address the question of when input elements are ready to be processed once they have been disclosed. Exact specifications of instance revelation rules r and r' as well as of rule sets P and P' according to the taxonomy in Chapter 2.1.3 would resolve this issue automatically. However, these specifications are rarely given explicitly in publications such that we provide a classification which verbally expresses the processing permissions arising by lookahead. The task is to specify whether earlier disclosure of input element information only means that the information is known earlier or also that the input element itself is ready to be processed earlier (cf. also [155]). In the first case, there can only be an informational benefit of lookahead, but there may additionally be a processual benefit in the latter case.

Immediate accessibility

Processing input element σ_i is possible directly upon receiving it, i.e., we have $\underline{T}_i := \tau_i$ in the online setting without lookahead and $\underline{T}'_i := \tau'_i$ in the online setting with lookahead.

Regular accessibility

Processing input element σ_i is possible only when the regular earliest processing time is reached, i.e., we have $\underline{T}'_i := \underline{T}_i$.

Delayed accessibility

Processing input element σ_i is not possible directly upon receiving its information, but may be possible at some time before the regular earliest processing time, i.e., we have $\underline{T}_i > \tau_i$ in the online setting without lookahead and $\underline{T}'_i > \tau'_i$ in the online setting with lookahead. A typical case is to impose a fixed offset t_{off} between information release and earliest possible processing, i.e., $\underline{T}_i = \tau_i + t_{off}$ and $\underline{T}'_i = \tau'_i + t_{off}$.

3.2.5 Algorithm Execution Mode

The algorithm execution mode controls at which time instants an algorithm is executed in order to determine the values of the action variables of known input elements. Recall that algorithm execution and action execution (physical processing) are different: Algorithm execution refers to the computational steps needed to determine actions for input elements, whereas action execution is the realization of the decisions which have been determined by algorithm execution before. Action execution runs automatically, but algorithm execution needs initiation.

We present three intuitive algorithm execution modes which are applicable to algorithms in online optimization problems with absent or present lookahead. Computing time of algorithms is not a scarce resource in the online optimization paradigm by definition; the only scarce resource is information ([32]). However, when exact algorithms are applied to instances of $\mathcal{NP}$-hard optimization problems, we have to ensure that prescribed real-time requirements are not violated.

Cyclic Execution

Algorithm execution is carried out cyclically, i.e., at all times $t_i^{exec} = i \cdot t^{cycle}$ for $i = 0, 1, 2, \ldots$ where t^{cycle} is the base time interval between two algorithm executions. It is possible that an arbitrary large number of new input elements accumulates during two algorithm executions or that no new input element arrives at all. Execution may still be worthwhile because actions different from Null may be determined for known input elements which have not yet been assigned an action so far.

Full Buffer Execution

Algorithm execution is performed every time that the number of elements in the lookahead set reaches a prescribed limit $c \in \mathbb{N}$, i.e., at all times in

$$\left\{t^{exec} \geq 0 \;\middle|\; |L_{t_{exec}}| = c, |L_{t_{exec}-\epsilon}| < c\right\}$$

with sufficiently small $\epsilon > 0$. At the end of the input sequence, it has to be assured that none of the input elements σ_i exhibits $a_i(t) = \textsc{Null}$ so as to guarantee that each input element will be processed.

Discrete Event Execution

Execution of an algorithm is triggered by events occurring at discrete time instants. From a technical point of view, an event detecting device must be installed in order to monitor incoming events. Release of a new input element and finished processing of an input element serve as typical events in basic online optimization problems. In more complex settings, additional events such as any type of system breakdown or failure can also be considered as events. Denote by $(t_i^e)_{i \in \mathbb{N}}$ the sequence of time instants at which events are notified, then algorithm execution takes place at times $t_i^{exec} = t_i^e + \epsilon$ for $i \in \mathbb{N}$ with sufficiently small $\epsilon > 0$. Note that practically all types of relevant algorithm execution modes can be traced back to the case of discrete event execution by appropriate definition of the set of events.

3.3 A Classification Scheme

Similar to the classification scheme of Graham et al. for scheduling problems ([81]), we provide a classification scheme for the modeling of online optimization problems with lookahead that takes into account their characteristics as identified in the previous sections. Given an instance of an online optimization problem with lookahead, the components of the modeling framework need to be specified in order to implement a solution procedure: Basic modeling elements as introduced in Chapter 3.2.1 are specific to the problem under investigation such that an associated modeling effort is unavoidable; restrictions concerning the release process, the processing characteristics and the algorithm execution mode from Chapters 3.2.2 to 3.2.5 work in a similar way for different problem settings, and they characterize the form of the instance revelation rule substitution and rule set substitution that comes along with lookahead.

We propose a four-position classification scheme of the form $\alpha \mid \beta \mid \gamma \mid \delta$ for online optimization problems with lookahead in order to quickly indicate the qualitative characteristics of a problem under lookahead as compared to the online problem without lookahead:

Lookahead type α Frequently used entries for α are:

- *req* for request lookahead
- *req-b* for request lookahead in batches
- *time* for time lookahead
- *col* for collective property lookahead
- *prop* for property lookahead
- *free* for free lookahead

Processing mode and order β Frequently used entries for β are:

- *sngl/ord* for single in-order processing
- *sngl/rnd* for single random-order processing
- *prl/ord* for parallel in-order processing
- *prl/rnd* for parallel random-order processing
- *prl-ltd/ord* for parallel limited in-order processing
- *prl-ltd/rnd* for parallel limited random-order processing
- *pp* for property processing

Processing accessibility γ Frequently used entries for γ are:

- *im* for immediate accessibility
- *reg* for regular accessibility
- *dly* for delayed accessibility

Algorithm execution mode δ Frequently used entries for δ are:

- *cyc* for cyclic algorithm execution
- *full* for full buffer algorithm execution
- *discr* for discrete event algorithm execution

Reference	Proprietary name	α	β	γ	δ
Routing and transportation					
Allulli et al. [7], [8]	–	time	sngl/rnd	reg	discr
Ausiello et al. [11]	–	time	sngl/rnd	reg	discr
Bosman and La Poutré [34]	oracle	free	prl-ltd/rnd	reg	discr
Jaillet and Lu [95]	advanced information	prop	sngl/rnd	reg	discr
Jaillet and Wagner [96]	advanced information	prop	sngl/rnd	reg	discr
Tinkl [155]	lookahead (La) by order	req	sngl/rnd	im	cyc
Scheduling					
Coleman [51]	–	req	sngl/rnd	reg	discr
Li et al. [123]	–	time	prl/rnd	reg	discr
Mandelbaum and Shabtay [129]	adaptive La	req	prl-ltd/ord	reg	cyc
Mandelbaum and Shabtay [129]	non-adaptive La	req-b	prl-ltd/ord	reg	cyc
Mao and Kincaid [130]	–	req	sngl/rnd	reg	discr
Motwani et al. [134]	finite La	req	sngl/ord	im	cyc
Yang et al. [161]	head-of-the-line	req	sngl/rnd	im	cyc
Zheng et al. [168]	–	time	sngl/ord	reg	discr
Zheng et al. [169]	–	time	sngl/rnd	reg	discr
Data structures					
Albers [2], [3]	weak La	req	sngl/ord	reg	cyc
Albers [3]	strong LA	col	sngl/ord	reg	cyc
Breslauer [41]	natural LA	col	sngl/ord	reg	cyc
Kiniwa et al. [113]	–	req	sngl/rnd	im	cyc
Torng [156]	–	req	sngl/ord	reg	cyc
Yeh et al. [163]	–	req-b	sngl/rnd	im	cyc
Young [164]	resource-bounded La	col	sngl/ord	reg	cyc
Data transfer					
Dooly et al. [64]	oracle	req	sngl/ord	reg	cyc
Imrek and Németh [93]	–	time	sngl/ord	reg	discr
Packing					
Grove [83]	–	col	sngl/rnd	im	cyc
Gutin et al. [86]	–	req-b	sngl/rnd	im	cyc
Lot sizing					
Ahlroth et al. [1]	–	time	prl/ord	im	cyc
Metrical task systems					
Ben-David and Borodin [20]	–	req	prl-ltd/ord	reg	cyc
Koutsoupias and Papadimitriou [115]	–	req	sngl/ord	reg	cyc
Graph theory					
Chung et al. [45]	window index	col	sngl/ord	reg	cyc
Halldorssón and Szegedy [87]	–	req	sngl/ord	reg	cyc
Halldorssón and Szegedy [87]	buffer	req-b	sngl/ord	reg	cyc
Irani [94]	–	req	sngl/ord	reg	cyc

Table 3.1: Classification of lookahead concepts in papers on online optimization with lookahead.

As an example, the sequential model of online optimization endowed with request lookahead as encountered in the paging problem may be represented both by $req \mid sngl/ord \mid im \mid discr$ and $req \mid sngl/ord \mid im \mid cyc$.

In literature, lookahead is mainly defined with respect to the topic of a paper; often an explicit specification of processing characteristics is missing but implicitly assumed. Table 3.1 on the previous page classifies the lookahead concepts and their requirements on processing and algorithm execution as described in papers on online optimization with lookahead.

Finally, we relate the classification scheme to Definitions 2.13 and 2.21 of online optimization problems in absence and presence of lookahead, respectively: The lookahead type α indicates the instance revelation rule substitution from r to r'. The processing mode and order β as well as the processing accessibility γ are established by rule set P' which constitutes the conditions for the processing of the input elements such that compliance with the feasible set of the problem under lookahead is ensured. The algorithm execution mode δ as part of the solution routine is unaffected by the problem formulation.

3.4 Discrete Event Process Model

The generic process model for online optimization with lookahead describes the interaction of the basic modeling elements over time in order to reproduce the function logic of a dynamic system. Because we basically impute a discrete-event-triggered algorithm execution, we also speak of a discrete event process model. In order to keep track of the solution process in an instance of an online optimization problem with lookahead, we need instantiations of the lookahead type, the processing mode and order, and the processing accessibility as specified by the (instance revelation rule and the rule set of the) application.

The system is assumed to operate on an event-driven basis. Starting with initial state $s_0 \in \mathcal{S}$, Figure 3.1 schematically illustrates how the system's state trajectory $(s_0, s_1, s_2, \ldots)$ evolves as a result of events arriving in the event sequence $(e_0, e_1, e_2, \ldots)$ and related computations of algorithm $\textsc{Alg}$: Upon arrival of an event $e \in \mathcal{E}$, the system proceeds from its current state $s \in \mathcal{S}$ to successor state $s' \in \mathcal{S}$ by evaluating the state transition function $f(s, e)$. However, computing the successor state at time t as $f(s, e)$ implicitly requires a preceding evaluation of $\textsc{Alg}_t(s, e) \in \mathcal{A}^{n_t}$ where n_t is the number of known unfinished input elements at time t because the successor state also contains a specification of the action variables for the yet unfinished input elements. Upon reaching the successor state, the system awaits the arrival of a new event causing the system to undergo the same sequence of steps again.

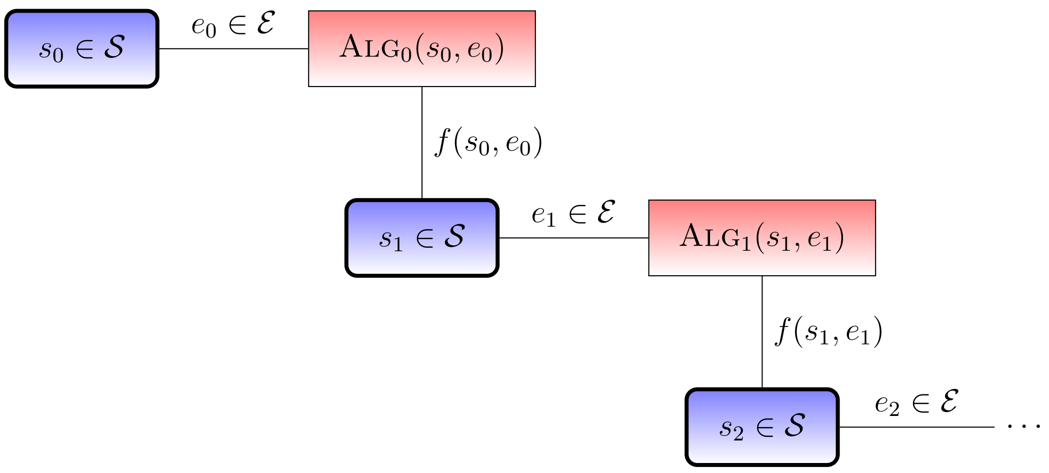


Figure 3.1: State trajectory with associated state transition function and algorithm evaluations in the process model for online optimization with lookahead.

The computational effort of an algorithm caused by a state transition typically depends on the type of the event e that is encountered: The arrival of a new input element suggests to solve a snapshot optimization problem in order to determine the action variable for the new input element and to redetermine the action variables for still unfinished input elements; finished processing of an input element will not cause an algorithm to spend excessive computational resources since no additional action for a new input element needs to be determined and all remaining actions are expected to remain the same as they have been computed previously on the same informational basis.

Information about the evolution of the objective value which is incurred by processing the elements of the input sequence can be tracked by monitoring the objective state component s^{obj} of each attained state $s = (s^{in}, s^{sys}, s^{obj})$. Typically, when an event corresponds to a finished processing of an input element, a change in the objective value can be observed as a result of the associated state transition, whereas when an event corresponds to a new input element release, no immediate change in the objective value will occur since processing of this input element is still pending.

3.5 Relation to Markov Chains

In a Markov chain, state transitions occur at discrete time instants $i \in \mathbb{N}$ irrespective of all past states other than the current state resulting in an associated random (stochastic) process exhibiting the memoryless property. That is why Markov chains are an equivalent type of description for autonomous stochastic automata (see also Chapter 2.4.3). A Markov chain can be modeled by identifying the state space $\mathcal{S}$, the initial state probabilities $p_0(s)$ which give the probability that the initial state of the system is $s \in \mathcal{S}$, the state transition probabilities $g_{ss'}$ which give the probability that the successor state of $s \in \mathcal{S}$ is $s' \in \mathcal{S}$, and a set $\mathcal{S}_F \subseteq \mathcal{S}$ of terminal states for which we would like to obtain frequency information. For a fixed initial state $s_0 \in \mathcal{S}$, the state of the Markov chain is assumed to evolve autonomously according to the state transition probabilities; after $n \in \mathbb{N}$ transition steps, a probability distribution over the states reached after n transitions starting from s_0 is obtained. Our discrete event process model is somewhat different from this perspective because it neither specifies initial state probabilities nor state transition probabilities. Moreover, upon receiving a new event from event set $\mathcal{E}$, an algorithm decides deterministically which successor state will be attained. However, with the sequence of occurring events $(e_0, e_1, e_2, \ldots)$, we can identify a source of randomness also in the discrete event process model. We translate our nescience of probabilities for occurring events into the assumption of a uniform distribution for the occurring events: According to the principle of maximum entropy, the largest entropy distribution, which is the discrete uniform distribution in this case, should be chosen representatively in order to ensure that no unjustified assumptions are introduced ([97], [98]). As a major side effect of this approach, we can use a Markov chain to model the evolution of the state trajectory in online optimization with lookahead: The state space $\mathcal{S}$ of the Markov chain coincides with the state space of the discrete event process model, the distribution of the initial state probabilities is degenerated by $p_0(s_0) = 1$ for $s_0 \in \mathcal{S}$ and $p_0(s) = 0$ for $s \in \mathcal{S}$ with $s \neq s_0$, and the state transition probabilities are chosen as $g_{ss'} := \frac{e(s,s')}{|\mathcal{E}|}$ where $e(s,s') := \big| \{e \in \mathcal{E} \mid f_{\text{ALG}}(s,e) = s'\} \big|$ is the number of events in $\mathcal{E}$ leading from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by applying algorithm ALG. The choice of $g_{ss'}$ in this Markov chain accounts for the discrete uniform distribution over the elements of event set $\mathcal{E}$. We note that because of the objective state component s^{obj} in state $s = (s^{in}, s^{sys}, s^{obj}) \in \mathcal{S}$ holding information about the valuation of a state, we implicitly model a valued Markov chain with the above specifications.

A requirement to adopt Markov chains to online optimization with lookahead is that the state space $\mathcal{S}$ is a finite or countable set. Hence, for time lookahead or continuous state space information this method of evaluation would be inapplicable and a discretization of the respective dimensions is required first. However, analyzing an algorithm for an online optimization problem with lookahead by means of an associated Markov chain suffers the same computational burdens as the analysis of any Markov chain when the size of the state space explodes for increased values of the problem parameters ([18]).

We note that the set up of a Markov decision process ([140]) is different from ours: State transitions occur probabilistically once a control action has been chosen, whereas in our setting they occur deterministically based on an algorithm's deterministic decision. Markov decision processes are used as a modeling formalism to determine an optimal strategy, i.e., the decisions of an optimal algorithm with respect to some expected objective value, using dynamic programming. Stochastic assumptions concerning transition probabilities depending on the control action are given a-priori: $p(s, a, s')$ with $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ is the probability that the successor state of s is s' if action a is chosen. In contrast to this, our analysis merely intends to evaluate the quality of a given algorithm in a setting of complete nescience of stochastic information. In particular, we do not seek for an optimal algorithm.

3.6 Instantiations of the Framework

We instantiate the generic modeling framework for online optimization with lookahead in three exemplary applications by specifying its components.

Online Bin Packing with Lookahead

Consider a version of the online bin packing problem whose lookahead and processing characteristics are classified according to

$$req \mid sngl/ord \mid reg \mid discr.$$

Lookahead type Request lookahead of size k.

Processing Mode and Order Items have to be packed in their order of appearance one after another. As a result, the advantage accrued by lookahead is merely informational. However, as seen in Example 2.37 and confirmed by the computational results in Chapter 5.3, the value of permuting the item order in physical processing turns out to be 0 anyway when an unbounded number of open bins is allowed.

Processing Accessibility Although lookahead makes items known earlier in time, their processing time in the lookahead setting coincides with the processing time in the setting without lookahead due to required sequentiality. This type of restriction may arise due to the physical dimensions of the objects which have to be packed.

Algorithm Execution Mode Since assignment decisions are invariant as long as no event (arrival of new item or packing of item into a bin) occurs, they only need to be (re-) determined by evaluating Alg whenever an event takes place.

Input element and input sequence Denote the set of items by Σ_1. An element $\sigma_i \in \Sigma_1$ of the input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ corresponds to an item and comprises the following data:

- Release time $\tau_i = i - 1$ in the reference problem without lookahead
- Release time $\tau'_i = \max\{0, i-k\}$ in the problem with lookahead
- Processing time $T_i = i - \epsilon$ in the reference problem without lookahead
- Processing time $T'_i = i - \epsilon$ in the problem with lookahead
- Input element information $r_i = s_i$ where s_i is the size of item σ_i

Lookahead set The lookahead set essentially contains those k items which have not yet been put into a bin, i.e., $L_t = \{\sigma_i \,|\, p_i(t) = \text{unprocessed}, \tau_i \leq t\}$ in the setting without lookahead and $L_t = \{\sigma_i \,|\, p_i(t) = \text{unprocessed}, \tau'_i \leq t\}$ in the setting with lookahead.

State space The state space is given as $\mathcal{S} = \{(U, c_1, c_2, \ldots, c_m, m)\}$ where U is the set of items which have not yet been physically assigned to a bin with $U \in \Sigma_1 \cup \emptyset$ in the case without lookahead and $U \in \bigcup_{k' \leq k} \big\{\{\sigma_1, \sigma_2, \ldots, \sigma_{k'}\} \,|\, \sigma_i \in \Sigma_1, i = 1, 2, \ldots, k'\big\}$ in the case with lookahead and $(c_1, c_2, \ldots, c_m)$ are the fill levels of the m bins used.

Event space The set of events comprises two event types. Events of the first type are of the form $e_1(\sigma_i) :=$ *Arrival of new item* σ_i and events of the second type are of the form $e_2(\sigma_i, j) :=$ *Physical assignment of item* σ_i *to bin* j. Hence, the event set is given as $\mathcal{E} = \{e_1(\sigma_i) \,|\, \sigma_i \in \Sigma_1\} \cup \{e_2(\sigma_i, j) \,|\, \sigma_i \in \Sigma_1, j \in \mathbb{N}\}$.

Algorithm An algorithm has to decide into which bin an item has to be put such that no bin capacity is ever exceeded. Thus, with $n_t = |L_t|$ as the number of unprocessed items at time t, $\text{ALG}_t : \mathcal{S} \times \mathcal{E} \to \{\mathbb{N} \cup \text{NULL}\}^{n_t}$ is a function which assigns each item in L_t either the number of the bin into which it is planned to be put at its respective processing time or leaves the destination bin undetermined.

State transition function Let the current state at time t be $s = (U, c_1, c_2, \ldots, c_m, m)$ and let an event of the form $e = e_1(\sigma_i)$ occur at t, then the successor state s' computed by the state transition function is $f(s, e) = (U', c_1, c_2, \ldots, c_m, m)$ where U' corresponds to U expanded with σ_i. Upon event $e = e_2(\sigma_i, j)$, the successor state s' is $f(s, e) = (U', c'_1, c'_2, \ldots, c'_{m'}, m')$ where U' corresponds to U diminished by σ_i, $c'_j = c_j + s_i$, $c'_k = c_k$ for $k = 1, \ldots, m$ with $k \neq j$ and $m' \in \{m, m+1\}$ gives the number of occupied bins after σ_i has been packed.

Online Traveling Salesman Problem with Lookahead

Consider a version of the online traveling salesman problem whose lookahead and processing characteristics are classified according to

$$time \mid sngl/rnd \mid im \mid discr.$$

Due to the infinite state space as a result of continuous time, the elements of the state space are uncountable and Markov chain analysis is computationally prohibitive. Nonetheless, the discrete event process model can be utilized in sample-based analysis or simulation.

Lookahead type Time lookahead of length D.

Processing Mode and Order Locations have to be visited one after another, but not necessarily in their order of appearance. As a result, the advantage from lookahead is both related to earlier information and possibilities of earlier processing. As seen in Example 2.37 and confirmed by the computational results in Chapter 5.4, algorithms benefit from lookahead by visiting spatially proximate locations in temporal proximity.

Processing Accessibility Positive effects in terms of increased degrees of freedom due to permutability of locations in the visiting sequence are further enhanced by immediate processing accessibility of the locations, i.e., once a location is known, it is allowed to be visited immediately. Unrestricted visitation opportunities of this kind may arise whenever the requesting entity has no temporal preferences which is the case for physical or virtual objects, but not for animate beings.

Algorithm Execution Mode Because sequencing decisions are invariant as long as no event (arrival of new request or finished service) occurs, they only need to be (re-) determined by evaluating ALG whenever an event takes place. Due to continuous time, the state trajectory is a continuous path in the state space. However, events can be ordered such that discrete event algorithm execution applies, and the evolution of the state trajectory can be (analytically) reproduced based on the event occurrences and associated decisions of the algorithm.

Input element and input sequence Denote the set of locations by Σ_1. An element $\sigma_i \in \Sigma_1$ of the input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ corresponds to a location to be visited and comprises the following data:

- Release time $\tau_i \in [0, \infty)$ in the reference problem without lookahead
- Release time $\tau_i' = \max\{0, \tau_i - D\}$ in the problem with lookahead

- Processing time $T_i \in [\tau_i, \infty)$ in the reference problem without lookahead
- Processing time $T_i' \in [\tau_i', \infty)$ in the problem with lookahead
- Input element information $r_i = x_i$ where x_i gives the spatial position of σ_i

Lookahead set Set $p(\sigma_i) :=$ processing if location σ_i is currently approached and $p(\sigma_i) :=$ unprocessed otherwise. The lookahead set contains all known locations which have not yet been reached by the server, i.e., $L_t = \{\sigma_i \,|\, p_i(t) \in \{\text{unprocessed, processing}\}, \tau_i \leq t\}$ in the case without lookahead and $L_t = \{\sigma_i \,|\, p_i(t) = \{\text{unprocessed, processing}\}, \tau_i' \leq t\}$ in the case with lookahead.

State space Let n be the maximum number of locations to be visited by the server. The state space is given as $\mathcal{S} = \{(U, P, t, x_s, t_{last})\}$ where $U \in \bigcup\limits_{n' \leq n} \big\{\{\sigma_1, \sigma_2, \ldots, \sigma_{n'}\} \,|\, \sigma_i \in \Sigma_1, i = 1, 2, \ldots, n'\big\}$ is the set of unprocessed locations, $P \in \Sigma_1 \cup \emptyset$ is the currently approached location, t is the current time, x_s is the current server location and t_{last} is the last time instant where a request location has been visited.

Event space The server moves autonomously to its destinations as prescribed in the action variables of the locations. The set of events only comprises two event types. Events of the first type are of the form $e_1(\sigma_i) :=$ *Arrival of new location* σ_i and events of the second type are of the form $e_2(\sigma_i) :=$ *Server has reached location* σ_i. Hence, the event set is given as $\mathcal{E} = \{e_1(\sigma_i) \,|\, \sigma_i \in \Sigma_1\} \cup \{e_2(\sigma_i) \,|\, \sigma_i \in \Sigma_1\}$.

Algorithm An algorithm has to decide upon the order in which the locations in L_t have to be visited. It suffices at each time if the server knows which location to approach next; note that the current destination may be revised upon announcement of a new location. Thus, with $n_t = |L_t|$ as the number of unvisited locations at time t, $\textsc{Alg}_t : \mathcal{S} \times \mathcal{E} \to \{\mathbb{N} \cup \textsc{Null}\}^{n_t}$ is a function which assigns each location in L_t either the position in the visiting order of the unvisited locations or leaves the position undetermined.

State transition function Let the current state at time t be $s = (U, P, t, x_s, t_{last})$ and let an event of the form $e = e_1(\sigma)$ occur at t, then the successor state s' computed by the state transition function is $f(s, e) = (U', P', t, x_s, t_{last})$ where P' contains the location from $U \cup P \cup \{\sigma_i\}$ which $\textsc{Alg}_t$ decides to approach next and U' contains the remaining unvisited locations. Upon event $e = e_2(\sigma_i)$ at time t, the successor state s' is $f(s, e) = (U', P', t, x_s', t_{last}')$ where $x_s' = x_i$, $t_{last}' = t$, and for U' and P' there are two cases. First, if all known locations have been visited, then $U' = P' = \emptyset$. Second, if at least one location is still to be visited, then P' contains the location determined by $\textsc{Alg}_t(s, e)$ to be visited next and U' contains the remaining unvisited locations.

Online Linear Programming with Lookahead

Consider for $m, n \in \mathbb{N}$ a linear program with n-dimensional decision variable $x \in [0,1]^n$ of the form

$$\max\ c^{\intercal}x \qquad \text{s.t. } Ax \leq b, \quad 0 \leq x_i \leq 1, \quad i = 1, \dots, n$$

where $c \in \mathbb{R}^n, b \in \mathbb{R}^m$, and $A = (a_{ij})$ is a (m,n)-matrix with $a_{ij} \in \mathbb{R}$ for all $i = 1, \dots, m$ and $j = 1, \dots, n$. In the online linear programming problem ([162]), the number of variables n and the right-hand side b are given in advance, but the objective function coeffients c_j and the columns $a_j = (a_{1j}, \dots, a_{mj})$ of A are revealed one after another for $j = 1, \dots, n$. The goal is to assign a value from $[0,1]$ to x_j when (c_j, a_j) is disclosed. In the case of lookahead, more than one variable with unassigned value may be known at a time.

Consider a version of the online linear programming problem whose lookahead and processing characteristics are classified according to

$$req \mid prl/rnd \mid im \mid cyc.$$

Lookahead type Request lookahead of size k.

Processing Mode and Order Processing a revealed variable amounts to irrevocably set its value.

Processing Accessibility Because variables are abstract entities, there are no restrictions on their processing and variables can be assigned values as soon as their respective column data is known.

Algorithm Execution Mode Because value assignment decisions are invariant as long as no event (release of a new column or value assignment to a variable) occurs, they only need to be (re-) determined by evaluating ALG whenever a new variable's data is revealed (which is assumed to happen cyclically).

Input element and input sequence Denote the set of variables by Σ_1. An element $\sigma_i \in \Sigma_1$ of the input sequence $\sigma = (\sigma_1, \sigma_2, \dots)$ corresponds to a variable x_i and comprises the following data:

- Release time $\tau_i = i - 1$ in the reference problem without lookahead
- Release time $\tau_i' = \max\{0, i - k\}$ in the problem with lookahead
- Processing time $T_i = i - \epsilon$ in the reference problem without lookahead
- Processing time $T_i' = i - \epsilon$ in the problem with lookahead

- Input element information $r_i = (c_i, a_i)$ where c_i is the objective function coefficient of x_i and a_i is the vector of the coefficients $(a_{1i}, \dots, a_{mi})$ in the ith column of the coefficient matrix A

Lookahead set The lookahead set essentially contains those k variables which have not yet been assigned an irrevocable value, i.e., $L_t = \{\sigma_i \,|\, p_i(t) = \text{unprocessed}, \tau_i \leq t\}$ in the setting without lookahead and $L_t = \{\sigma_i \,|\, p_i(t) = \text{unprocessed}, \tau_i' \leq t\}$ in the setting with lookahead.

State space The state space is given as $\mathcal{S} = \{(U, F, x, o)\}$ where U is the set of variables which have not yet been assigned an irrevocable value with $U \in \Sigma_1 \cup \emptyset$ in the case without lookahead and $U \in \bigcup\limits_{k' \leq k} \big\{\{\sigma_1, \sigma_2, \dots, \sigma_{k'}\} \,|\, \sigma_i \in \Sigma_1, i = 1, 2, \dots, k'\big\}$ in the case with lookahead, F is the tuple of variables which have already been assigned a final value with $F \in \bigcup\limits_{n' \leq n} \big\{\{\sigma_1, \sigma_2, \dots, \sigma_{n'}\} \,|\, \sigma_i \in \Sigma_1, i = 1, 2, \dots, n'\big\}$ for $n' = 0, 1, \dots, n$, $x \in \big\{\mathbb{R} \cup \{\textsc{Null}\}\big\}^n$ is the vector of values assigned to the variables and

$$o := \sum_{\substack{i=1,\\ \sigma_i \in F}}^{n} c_i x_i$$

is the objective value incurred so far.

Event space The set of events comprises two event types. Events of the first type are of the form $e_1(\sigma_i) :=$ *Release of a new column* σ_i and events of the second type are of the form $e_2(\sigma_i, v) :=$ *The variable of column* σ_i *has been assigned the irrevocable value* v. Hence, the event set is given as $\mathcal{E} = \{e_1(\sigma_i) \,|\, \sigma_i \in \Sigma_1\} \cup \{e_2(\sigma_i, v) \,|\, \sigma_i \in \Sigma_1, v \in [0,1]\}$.

Algorithm An algorithm has to decide which values to assign to the revealed variables and in which order of the variables the value assignment will take place. Thus, with $n_t = |L_t|$ as the number of variables without fixed value assignment at time t, $\textsc{Alg}_t : \mathcal{S} \times \mathcal{E} \to \{[0,1] \cup \textsc{Null}\}^{n_t} \times \{\mathbb{N} \cup \textsc{Null}\}^{n_t}$ is a function which assigns each variable in L_t a pair (v, p) where v gives the planned value of the variable and p gives the planned position in the sequence of upcoming assignments; if no value and no position for a variable have been determined yet, the pair $(\textsc{Null}, \textsc{Null})$ can be used tentatively. Algorithms may strongly rely on the theory of linear programming and the concepts introduced for column generation in order to check profitability of a new column ([61], [162]), e.g., by making use of the value of the dual variables from the preceding snapshot linear program in order to decide on the value of the next variables.

State transition function Let the current state at time t be $s = (U, F, x, o)$ and let an event of the form $e = e_1(\sigma_i)$ occur at t, then the successor state s' computed by the state transition function is $f(s, e) = (U', F, x, o)$ where U' corresponds to U expanded with the additional column σ_i that has been released at time t. Upon event $e = e_2(\sigma_i, v)$, the successor state s' is $f(s, e) = (U', F', x', o')$ where U' corresponds to U diminished by σ_i, F' corresponds to F expanded with σ_i, x' contains the same elements as x except for the element corresponding to σ_i, the element corresponding to σ_i in x' is set to v and $o' = o + c_i v$.

3.7 Concluding Discussion

The discussion of previous attempts to model online optimization problems in a general fashion showed that these suffer from several shortcomings with respect to practical purposes. Amongst them, the lack of integration of lookahead and of the solution routine itself indicated the need for a more comprehensive framework for online optimization with lookahead. Taking advantage of the analogies between the solution process in sequential decision making and discrete event systems trajectories, we derived a general framework for online optimization with lookahead which can easily be instantiated in arbitrary applications. We accounted for the multitude of different lookahead forms encountered in theory and practice by introducing a classification scheme which allows for a quick categorization of a lookahead setting on hand, including the restrictions that arise for any solution process due to the provision of lookahead. Table 3.1 on page 89 gave a classification of the lookahead settings encountered in publications on online optimization with lookahead according to the proposed classification scheme. Furthermore, as a result of the close relationship between discrete event systems and Markov chains, we transferred Markov chain analysis to online optimization with lookahead. The final section of this chapter exemplified three instantiations of the modeling framework.

4 Theoretical Analysis of Algorithms for Online Optimization with Lookahead

In this chapter, we investigate the question of what can be achieved by the provision of additional lookahead relative to the pure online case where no lookahead is available from a theoretical point of view. The exact analysis comprises the derivation of exact expressions for the counting distribution functions of the objective value and the performance ratio (see Definitions 2.40 and 2.42). We stick to base cases of three academic problems:

- Online Ski Rental with Lookahead
- Online Bin Packing with Lookahead
- Online Traveling Salesman Problem with Lookahead

In the first problem, we allow for arbitrary additional lookahead, whereas for the two latter problems only one additional input element is considered. Theoretical analysis is possible because the implications of lookahead on the objective can be traced back to combinatorics.

4.1 Online Ski Rental with Lookahead

In the ski rental problem ([103], [128]), a novice to skiing needs to procure a pair of skis for an unknown number of days. As long as the skis have not been bought, the skier has to choose before each period whether to rent the skis for that day and thereby repeatedly incur renting costs, or to buy the skis and incur buying costs on that day with no additional future costs. The ski rental problem can be generalized to a decision maker in need of a resource for a time horizon of unknown length with two possibilities for procurement as long as the resource has not yet been bought: Either by paying a one-time cost (buy option) and using the resource without additional future costs, or by paying a pay-per-period cost granting right of use on that day (renting option).

In the offline version, the skier knows the number of skiing days in advance.

Problem 4.1 (Offline Ski Rental).

INSTANCE	Number n of skiing days, buying cost B, daily renting costs r.
TASK	Decide on which day $i \in \{1, 2, \dots n\} \cup \{\infty\}$ to buy skis where $i = \infty$ means that they are never bought such that total skiing costs are minimized.

In the online version, the skier knows at the beginning of each day whether skiing will occur on this day or not, i.e., whether the end of the skiing period has been reached or not.

Problem 4.2 (Online Ski Rental).

INSTANCE	Sequence of bits $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n, \sigma_{n+1})$ with $\sigma_1 = \dots = \sigma_n = 1$ and $\sigma_{n+1} = 0$ where σ_i indicates whether the skier will go skiing on day i ($\sigma_i = 1$) or not ($\sigma_i = 0$) for $i = 1, \dots, n+1$, buying cost B, daily renting costs r, instance revelation rule $\tilde{r}$ for the online case.
TASK	Sequentially decide for each $i = 1, 2, \dots, n$ without knowing σ_j for $j > i$ whether to rent or buy skis on day i such that total skiing costs are minimized.

In the online version with lookahead of size $l \in \mathbb{N}$, the skier knows at the beginning of each day whether skiing will occur on this day and the consecutive $l-1$ days or not, i.e., whether the end of the skiing period has been or will be reached during this and the next $l-1$ days.

Problem 4.3 (Online Ski Rental with Lookahead).

INSTANCE	Sequence of bits $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n, \sigma_{n+1})$ with $\sigma_1 = \dots = \sigma_n = 1$ and $\sigma_{n+1} = 0$ where σ_i indicates whether the skier will go skiing on day i ($\sigma_i = 1$) or not ($\sigma_i = 0$) for $i = 1, \dots, n+1$, buying cost B, daily renting costs r, instance revelation rule $\tilde{r}'$ for the lookahead case.
TASK	Sequentially decide for each $i = 1, 2, \dots, n$ without knowing σ_j for $j > i + l - 1$ whether to rent or buy skis on day i such that total skiing costs are minimized.

The instance revelation rule in the online case is

$$\tilde{r} := \text{At the start of day } i\text{, } \sigma_i \text{ is revealed}$$

and in the lookahead case

$$\tilde{r}' := \text{At the start of day } i\text{, unknown elements of } \sigma_i, \sigma_{i+1}, \dots, \sigma_{\min\{i+l-1, n+1\}} \text{ are revealed.}$$

The rule set is trivial and coincides in both cases with

$$P = P' := \{\text{At any day } i\text{, go skiing if } \sigma_i = 1\text{, otherwise do not go skiing}\}.$$

According to the modeling framework from Chapter 3, the lookahead setting is

$$req \mid sngl/ord \mid reg \mid discr \qquad \text{or equivalently} \qquad req \mid sngl/ord \mid reg \mid cyc.$$

The input information of an input element is given by σ_i. In the online case, we have $\tau_i = i$ and $T_i = i + \epsilon$; in the lookahead case, we have $\tau_i' = \max\{1, i - l + 1\}$ and $T_i' = \max\{1, i - l + 1\} + \epsilon$ with sufficiently small $\epsilon > 0$.

The offline problem is solved easily: Buy at the beginning of the first period if $B \leq rn$, otherwise rent the pair of skis in all periods yielding costs of $rn < B$. Thus, we have

$$\text{OPTIMAL}[(\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_{n+1})] = \min\{B, rn\}.$$

In Algorithm 4.1, we denote the time of purchase by k and interpret $k = \infty$ as the option to never buy.

Algorithm 4.1 Optimal offline algorithm OPTIMAL for ski rental

Input: Input sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_{n+1})$, renting cost $r > 0$, buying cost B

```
1: if $B \leq rn$ ($\Leftrightarrow n \geq \frac{B}{r}$) then
2:     $k := 1, C := B$
3: else
4:     $k := \infty, C := rn$
5: end if
```

Output: Time of purchase k, optimal skiing costs C

A generic algorithm for the online problem buys in a prescribed period k if $n \geq k$. We will denote this algorithm by BUY_k (cf. Algorithm 4.2 and [128]) and its payment profile is given by

$$\text{BUY}_k[(\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_{n+1})] = \begin{cases} rn & \text{if } n < k, \\ r(k-1) + B & \text{otherwise.} \end{cases}$$

Because of $r\lfloor \frac{B}{r} \rfloor \leq B \leq r\lceil \frac{B}{r} \rceil$, $\lceil \frac{B}{r} \rceil$ gives the minimum number of days for which buying induces costs not higher than for renting. Thus, in the lookahead case with $l \geq \lceil \frac{B}{r} \rceil$ (recall that additional $l - 1$ periods can be overseen besides the current period), we see at the first day whether buying at the first day excels renting and any reasonable algorithm would buy at the first day if $\sigma_{\lceil \frac{B}{r} \rceil} = 1$. The payment profile of such an algorithm coincides with that of OPTIMAL. Thus, there is no need for further discussion of the case $l \geq \lceil \frac{B}{r} \rceil$ and we turn to the case $l < \lceil \frac{B}{r} \rceil$ where we cannot see in the first period whether buying will excel renting because the overseen time horizon is too short.

Algorithm 4.2 Online algorithm $\textsc{Buy}_k$ for ski rental

Input: First element σ_1 of input sequence σ, renting cost $r > 0$, buying cost B, intended time of purchase k

```
1: $i := 1, \tilde{k} := \infty, C := 0$
2: while $\sigma_i = 1$ and $i \leq k$ do
3:   if $i = k$ then
4:     $\tilde{k} := k, C := r(k-1) + B$
5:   else
6:     $C := ri$
7:     additional input: $(i+1)$st element $\sigma_{i+1}$ of input sequence $\sigma$
8:     $i := i + 1$
9:   end if
10: end while
```

Output: Realized time of purchase $\tilde{k}$, skiing costs C

Algorithm 4.3 Online algorithm $\textsc{ConditionalBuy}_{k,l}$ with lookahead for ski rental

Input: Lookahead l (in days), first l elements $\sigma_1, \sigma_2, \ldots, \sigma_l$ of input sequence σ, renting cost $r > 0$, buying cost B, intended time of purchase k

```
1: $i := 1, \tilde{k} := \infty, C := 0$
2: if $l < \lceil \frac{B}{r} \rceil$ then
3:   while $\sigma_i = 1$ and $i \leq k$ do
4:     if $i = k$ and $\sigma_{i+l-1} = 1$ then
5:       $\tilde{k} := k, C := r(k-1) + B$
6:     else
7:       $C := ri$
8:       if $\sigma_{i+l-1} = 1$ then
9:         additional input: $(i+l)$th element $\sigma_{i+l}$ of input sequence $\sigma$
10:      end if
11:      $i := i + 1$
12:    end if
13:  end while
14: else
15:   if $\sigma_{\lceil \frac{B}{r} \rceil} = 1$ then
16:     $\tilde{k} := 1, C := B$
17:   else
18:     $C := r \cdot \max\{i \in \{1, \ldots, l\} \mid \sigma_i = 1\}$
19:   end if
20: end if
```

Output: Realized time of purchase $\tilde{k}$, skiing costs C

On the previous page, we introduce generic online algorithm $\textsc{ConditionalBuy}_{k,l}$ (cf. Algorithm 4.3) for the lookahead setting; it exploits lookahead by the following decision rule: Buy in period k if $n \geq k+l-1$. The algorithm coincides with $\textsc{Buy}_k$ for $l = 1$ and its payment profile is given by

$$\textsc{ConditionalBuy}_{k,l}[(\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_{n+1})] = \begin{cases} \min\{B, rn\} & \text{if } l \geq \lceil \frac{B}{r} \rceil, \\ rn & \text{if } l < \lceil \frac{B}{r} \rceil, n < k+l-1, \\ r(k-1)+B & \text{otherwise.} \end{cases}$$

We compare algorithms $\textsc{Optimal}$, $\textsc{Buy}_k$ and $\textsc{ConditionalBuy}_{k,l}$ in order to evaluate the benefit of lookahead in the ski rental problem. The costs of all mentioned algorithms are summarized in Table 4.1 depending on the relation between $k, k+l-1$ and $\lceil \frac{B}{r} \rceil$, and we can already conclude that lookahead in the ski rental problem induces stochastic dominance of all orders.

n	$\textsc{Optimal}$	$\textsc{Buy}_k$	$\textsc{ConditionalBuy}_{k,l}$
$1, 2, \ldots, k-1$	rn	rn	rn
$k, k+1, \ldots, k+l-2$	rn	$r(k-1)+B$	rn
$k+l-1, k+l, \ldots, \lceil \frac{B}{r} \rceil - 1$	rn	$r(k-1)+B$	$r(k-1)+B$
$\lceil \frac{B}{r} \rceil, \lceil \frac{B}{r} \rceil + 1, \ldots, n_{max}$	B	$r(k-1)+B$	$r(k-1)+B$

a)

n	$\textsc{Optimal}$	$\textsc{Buy}_k$	$\textsc{ConditionalBuy}_{k,l}$
$1, 2, \ldots, k-1$	rn	rn	rn
$k, k+1, \ldots, \lceil \frac{B}{r} \rceil - 1$	rn	$r(k-1)+B$	rn
$\lceil \frac{B}{r} \rceil, \lceil \frac{B}{r} \rceil + 1, \ldots, k+l-2$	B	$r(k-1)+B$	rn
$k+l-1, k+l, \ldots, n_{max}$	B	$r(k-1)+B$	$r(k-1)+B$

b)

n	$\textsc{Optimal}$	$\textsc{Buy}_k$	$\textsc{ConditionalBuy}_{k,l}$
$1, 2, \ldots, \lceil \frac{B}{r} \rceil - 1$	rn	rn	rn
$\lceil \frac{B}{r} \rceil, \lceil \frac{B}{r} \rceil + 1, \ldots, k-1$	B	rn	rn
$k, k+1, \ldots, k+l-2$	B	$r(k-1)+B$	rn
$k+l-1, k+l, \ldots, n_{max}$	B	$r(k-1)+B$	$r(k-1)+B$

c)

Table 4.1: Costs in the ski rental problem with $l < \lceil \frac{B}{r} \rceil$. **a)** $k < k+l-1 \leq \lceil \frac{B}{r} \rceil$. **b)** $k \leq \lceil \frac{B}{r} \rceil < k+l-1$. **c)** $\lceil \frac{B}{r} \rceil \leq k < k+l-1$. In **a)**, the third line vanishes for $\lceil \frac{B}{r} \rceil = k+l-1$; in **b)** and **c)**, the second line vanishes for $\lceil \frac{B}{r} \rceil = k$.

Note that $\textsc{ConditionalBuy}_{k,l}$ never achieves higher costs than $\textsc{Buy}_k$ because of

$$\begin{aligned} rn \leq r(k-1) + B \text{ for } n = k, \dots, k+l-2 \quad &\Leftrightarrow \quad rn \leq r(k-1) + B \text{ for } n = k+l-2 \\ &\Leftrightarrow \quad r(l-1) \leq B \\ &\Leftrightarrow \quad l \leq \frac{B}{r} + 1 \end{aligned}$$

where the last inequality is certainly true when $l < \lceil \frac{B}{r} \rceil$. For $l \geq \lceil \frac{B}{r} \rceil$, recall that $\textsc{Opt}$ and $\textsc{ConditionalBuy}_{k,l}$ behave identically.

We now analyze the counting distribution functions of the objective value and the performance ratio for the set of input sequences corresponding to at most $n_{\max} \in \mathbb{N}$ skiing days, i.e., for

$$\Sigma_{\leq n_{\max}} = \big\{\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n, \sigma_{n+1}) \,|\, n \in \{1, 2, \dots, n_{\max}\}\big\}.$$

The dominance relations between $\textsc{Optimal}$, $\textsc{ConditionalBuy}_{k,l}$ over $\textsc{Buy}_k$ can be seen in the resulting counting distribution functions for the objective value in Figure 4.1 and Table 4.2 (for a simplified exposition we only give the values at the breakpoints) where for all $v \in \mathbb{R}$ we have $F_{\textsc{Optimal}}(v) \geq F_{\textsc{ConditionalBuy}_{k,l}}(v) \geq F_{\textsc{Buy}_k}(v)$.

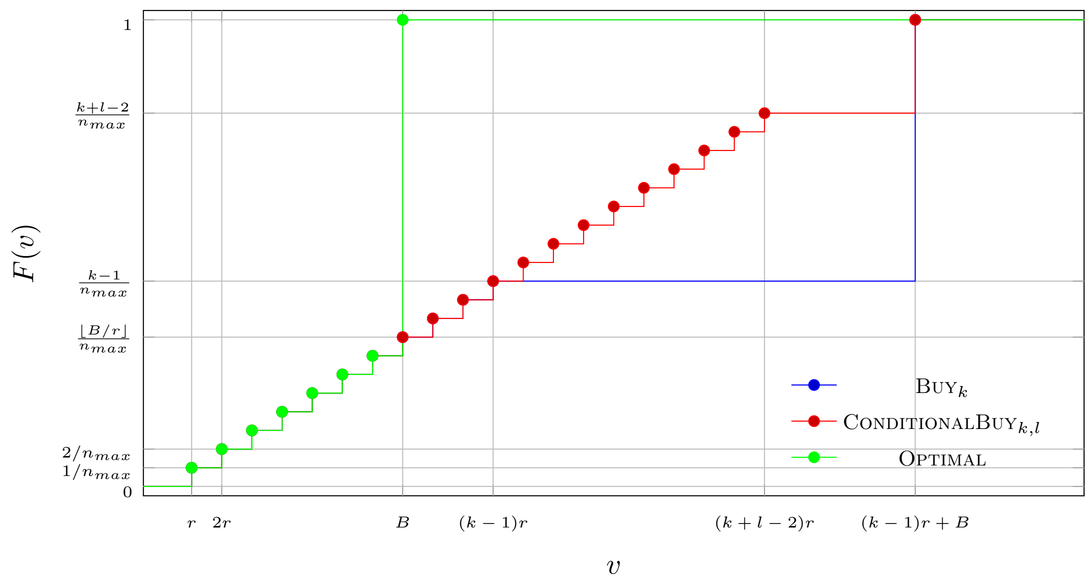


Figure 4.1: Counting distribution functions of costs in the ski rental problem with $k \geq \frac{B}{r}$.

We note that no special relation has to be enforced on B, r and k: Choosing $k < \frac{B}{r}$ may be justified by the fact that if buying is carried out, then it should be beneficial in the outset. Choosing $k \geq \frac{B}{r}$ may be justified by the fact that the decision maker wants to try skiing for a certain time first before buying skis.

v	$F_{\text{Buy}_k}(v)$	$F_{\text{ConditionalBuy}_{k,l}}(v)$	$F_{\text{Optimal}}(v)$
r	$\frac{1}{n_{max}}$	$\frac{1}{n_{max}}$	$\begin{cases} \frac{1}{n_{max}} & \text{if } r < B, \\ 1 & \text{else} \end{cases}$
$2r$	$\frac{2}{n_{max}}$	$\frac{2}{n_{max}}$	$\begin{cases} \frac{2}{n_{max}} & \text{if } 2r < B, \\ 1 & \text{else} \end{cases}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(k-1)r$	$\frac{k-1}{n_{max}}$	$\frac{k-1}{n_{max}}$	$\begin{cases} \frac{k-1}{n_{max}} & \text{if } (k-1)r < B, \\ 1 & \text{else} \end{cases}$
kr	$\frac{k-1}{n_{max}}$	$\frac{k}{n_{max}}$	$\begin{cases} \frac{k}{n_{max}} & \text{if } kr < B, \\ 1 & \text{else} \end{cases}$
$(k+1)r$	$\frac{k-1}{n_{max}}$	$\frac{k+1}{n_{max}}$	$\begin{cases} \frac{k+1}{n_{max}} & \text{if } (k+1)r < B, \\ 1 & \text{else} \end{cases}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(k+l-2)r$	$\frac{k-1}{n_{max}}$	$\frac{k+l-2}{n_{max}}$	$\begin{cases} \frac{k+l-2}{n_{max}} & \text{if } (k+l-2)r < B, \\ 1 & \text{else} \end{cases}$
$(k+l-1)r$	$\frac{k-1}{n_{max}}$	$\frac{k+l-2}{n_{max}}$	$\begin{cases} \frac{k+l-1}{n_{max}} & \text{if } (k+l-1)r < B, \\ 1 & \text{else} \end{cases}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(k-1)r+B$	1	1	1

Table 4.2: Counting distribution functions of costs in the ski rental problem with $l < \lceil \frac{B}{r} \rceil$.

When we consider an instance-wise comparison of Buy_k and $\text{ConditionalBuy}_{k,l}$, we find that for all but $l-1$ input sequences there is no difference in the objective value, resulting in performance ratio $\frac{\text{Buy}_k[\sigma]}{\text{ConditionalBuy}_{k,l}[\sigma]} = 1$ for these input sequences σ. The remaining $l-1$ input sequences exhibit improved performance of $\text{ConditionalBuy}_{k,l}$ over Buy_k due to lookahead. Performance ratios for these inputs range in the interval

$$\left[\frac{r(k-1)+B}{r(k+l-2)}, \frac{r(k-1)+B}{rk}\right] = \left[1 + \frac{B+r-rl}{r(k+l-2)}, 1 + \frac{B-r}{rk}\right]$$

which can be seen by considering the extreme cases $n = k+l-2$ and $n = k$. The resulting values of the counting distribution function for the performance ratio of Buy_k relative to $\text{ConditionalBuy}_{k,l}$ are graphically illustrated in Figure 4.2 and shown in Table 4.3.

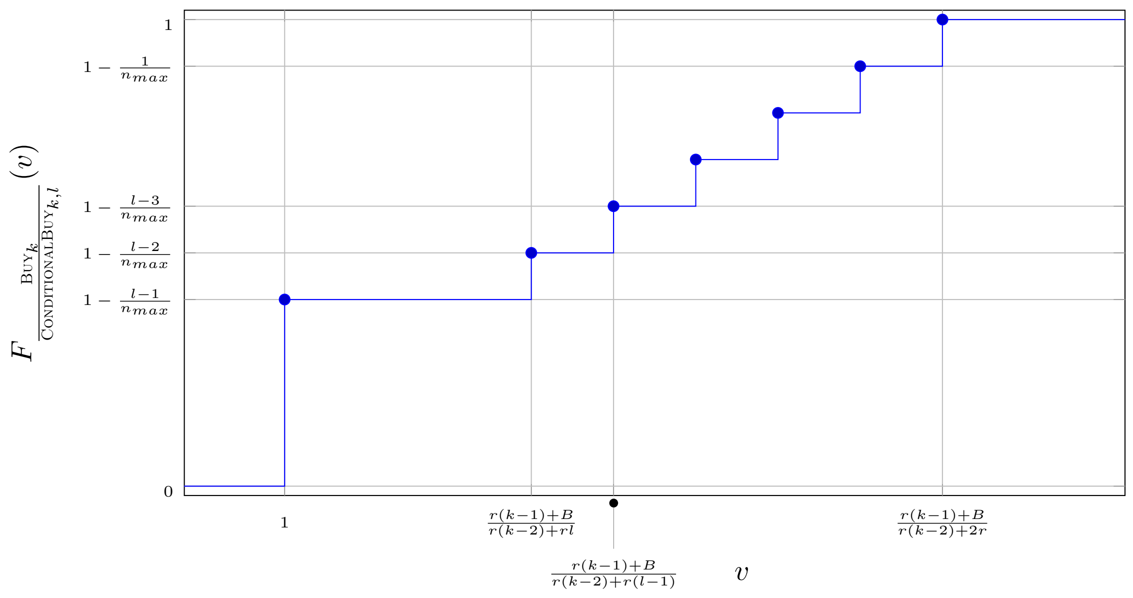


Figure 4.2: Counting distribution function of performance ratio of costs in the ski rental problem.

v	$F_{\frac{\text{Buy}_k}{\text{ConditionalBuy}_{k,l}}}(v)$
1	$1-\frac{l-1}{n_{max}}$
$\frac{r(k-1)+B}{r(k-2)+rl}$	$1-\frac{l-2}{n_{max}}$
$\frac{r(k-1)+B}{r(k-2)+r(l-1)}$	$1-\frac{l-3}{n_{max}}$
$\vdots$	$\vdots$
$\frac{r(k-1)+B}{r(k-2)+3r}$	$1-\frac{1}{n_{max}}$
$\frac{r(k-1)+B}{r(k-2)+2r}$	1

Table 4.3: Counting distribution function of performance ratio of costs in the ski rental problem.

Similarly, we instance-wise compare Buy$_k$ and Optimal as well as ConditionalBuy$_{k,l}$ and Optimal. Since Buy$_k$ and ConditionalBuy$_{k,l}$ coincide for $l = 1$, it suffices to analyze ConditionalBuy$_{k,l}$. According to Table 4.1, we have to consider the cases $k+l-1 \leq \lceil \frac{B}{r} \rceil$ and $k+l-1 > \lceil \frac{B}{r} \rceil$. For $k+l-1 \leq \lceil \frac{B}{r} \rceil$, we have that $k+l-2$ of the n_{max} input sequences lead to performance ratio 1 and $n_{max} - \lceil \frac{B}{r} \rceil + 1$ input sequences lead to performance ratio $\frac{r(k-1)+B}{B}$. Input sequences $\sigma = (\sigma_1, \ldots, \sigma_n, \sigma_{n+1})$ with $n \in \{k+l-1, \ldots, \lceil \frac{B}{r} \rceil - 1\}$ lead to performance ratio $\frac{r(k-1)+B}{rn}$. Thus, for these input sequences the performance ratio ranges in the interval

$$\left[\frac{r(k-1)+B}{r(\lceil \frac{B}{r} \rceil - 1)}, \frac{r(k-1)+B}{r(k+l-1)}\right].$$

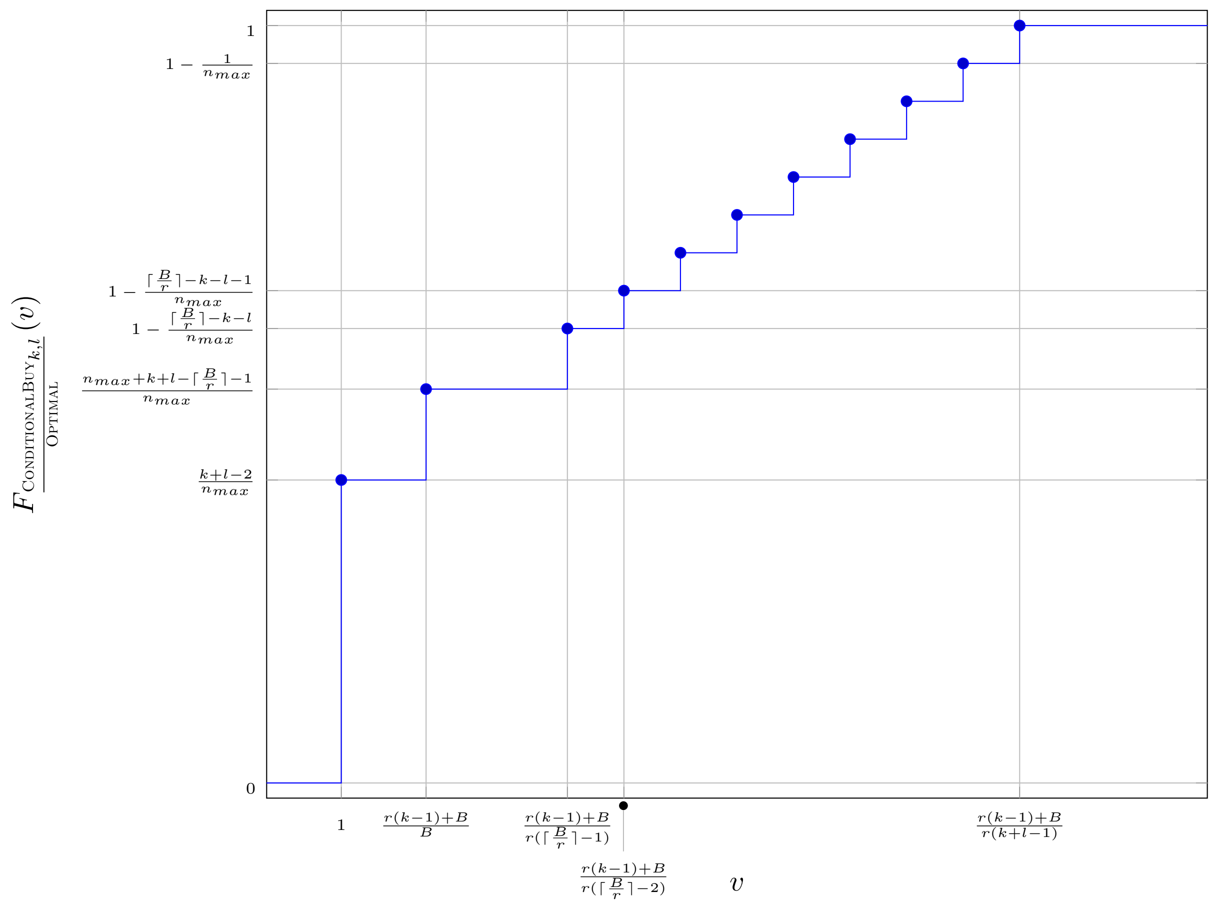


Figure 4.3: Counting distribution function of performance ratio of costs in the ski rental problem with $k+l-1 \leq \lceil \frac{B}{r} \rceil$.

v	$\frac{F_{\text{ConditionalBuy}_{k,l}}}{\text{Optimal}}(v)$
1	$\frac{k+l-2}{n_{max}}$
$\frac{r(k-1)+B}{B}$	$\frac{n_{max}+k+l-\lceil \frac{B}{r}\rceil-1}{n_{max}}$
$\frac{r(k-1)+B}{r(\lceil \frac{B}{r}\rceil-1)}$	$1-\frac{\lceil \frac{B}{r}\rceil-k-l}{n_{max}}$
$\frac{r(k-1)+B}{r(\lceil \frac{B}{r}\rceil-2)}$	$1-\frac{\lceil \frac{B}{r}\rceil-k-l-1}{n_{max}}$
$\vdots$	$\vdots$
$\frac{r(k-1)+B}{r(k+l-2)}$	$1-\frac{1}{n_{max}}$
$\frac{r(k-1)+B}{r(k+l-1)}$	1

a)

v	$\frac{F_{\text{ConditionalBuy}_{k,l}}}{\text{Optimal}}(v)$
1	$\frac{\lceil \frac{B}{r}\rceil-1}{n_{max}}$
$\frac{r\lceil \frac{B}{r}\rceil}{B}$	$\frac{\lceil \frac{B}{r}\rceil}{n_{max}}$
$\frac{r(\lceil \frac{B}{r}\rceil+1)}{B}$	$\frac{\lceil \frac{B}{r}\rceil+1}{n_{max}}$
$\vdots$	$\vdots$
$\frac{r(k+l-3)}{B}$	$\frac{k+l-3}{n_{max}}$
$\frac{r(k+l-2)}{B}$	$\frac{k+l-2}{n_{max}}$
$\frac{r(k-1)+B}{B}$	1

b)

Table 4.4: Counting distribution functions of performance ratio of costs in the ski rental problem. **a)** $k+l-1 \leq \lceil \frac{B}{r} \rceil$ and **b)** $k+l-1 > \lceil \frac{B}{r} \rceil$.

The resulting counting distribution function of the performance ratio is illustrated in Figure 4.3 and Table 4.4 a) on the previous page.

For $k+l-1 > \lceil \frac{B}{r} \rceil$, we have that $\lceil \frac{B}{r} \rceil - 1$ of the n_{max} input sequences lead to performance ratio 1 and $n_{max} - k - l + 2$ input sequences lead to performance ratio $\frac{r(k-1)+B}{B}$. Input sequences $\sigma = (\sigma_1, \ldots, \sigma_n, \sigma_{n+1})$ with $n \in \{\lceil \frac{B}{r} \rceil, \ldots, k+l-2\}$ lead to performance ratio $\frac{rn}{B}$. Thus, for these input sequences the performance ratio ranges in the interval

$$\left[\frac{r \lceil \frac{B}{r} \rceil}{B}, \frac{r(k+l-2)}{B} \right].$$

The resulting counting distribution function of the performance ratio is illustrated in Figure 4.4 and Table 4.4 b).

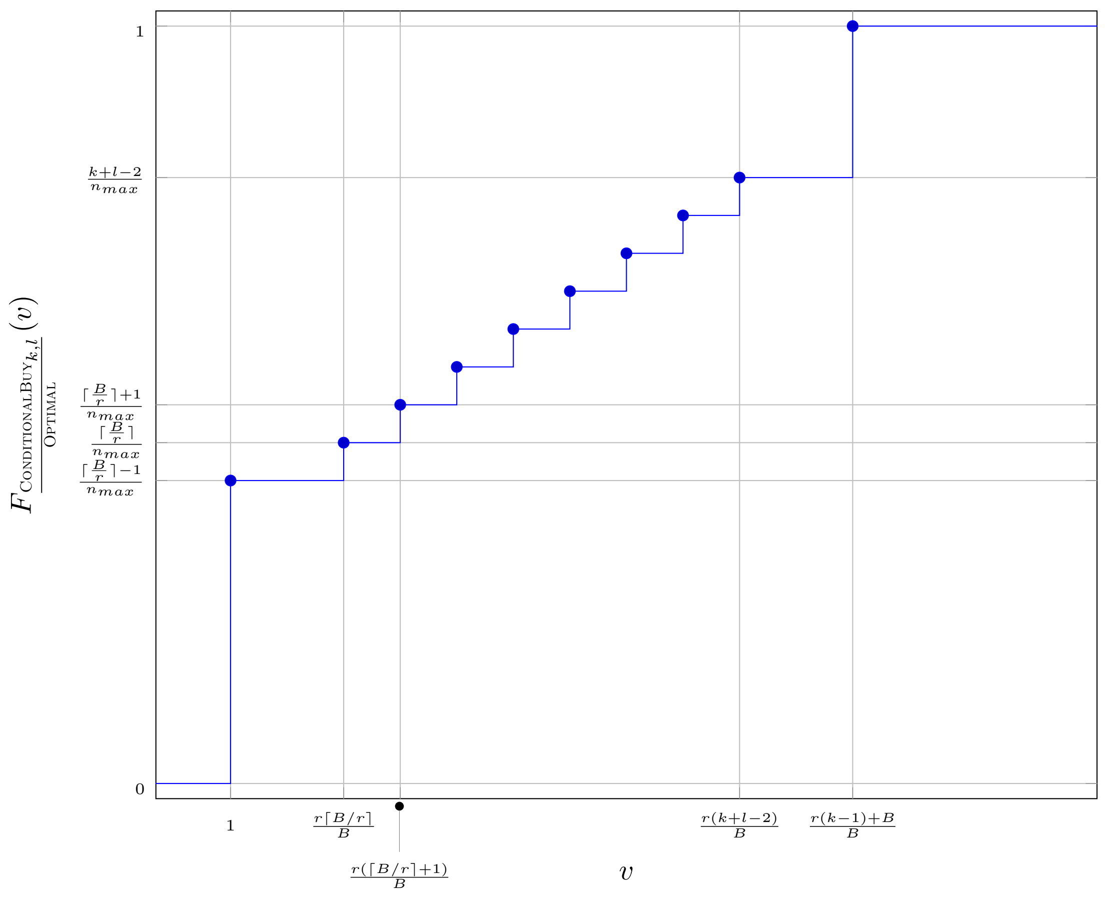


Figure 4.4: Counting distribution function of performance ratio of costs in the ski rental problem with $k+l-1 > \lceil \frac{B}{r} \rceil$.

For $l_1 > l_2$, $\text{ConditionalBuy}_{k,l_1}$ instance-wise outperforms $\text{ConditionalBuy}_{k,l_2}$, i.e., no algorithm of this family ever fails to interpret lookahead to its advantage. The explanation for this can be traced back to the one-shot decision character: Because there is only one decision, there is no possibility that its profitability could be undone by a future decision.

In this section, we underlined by exact analysis that in the ski rental problem it is possible to incur considerable improvement (for typical parameter values of B, r, k and l) when additional lookahead information concerning the plan on skiing over the next days is provided. The improvement is due to the avoidance of mispurchases when a short time later the skier ends the skiing trip.

4.2 Online Bin Packing with Lookahead

The bin packing problem is a fundamental combinatorial problem from the class of cutting and packing ([55]). It has attracted research attention as an abstract problem of packing a set of small objects into a set of larger objects under some objective function.

Problem 4.4 (Offline Bin Packing).

INSTANCE	Bin capacity C, set of items $\{\sigma_i \,|\, i \in \{1, \dots, n\}\}$ where item σ_i has size s_i for $i = 1, \dots, n$.
TASK	Assign each item to a bin such that the sum of item sizes assigned to a bin does not exceed C and the number of bins used is minimal.

In the online version, the items have to be packed one after another without knowledge of items other than the next one.

Problem 4.5 (Online Bin Packing).

INSTANCE	Bin capacity C, sequence of items $(\sigma_1, \dots, \sigma_n)$ where item σ_i has size s_i for $i = 1, \dots, n$, instance revelation rule r for the online case.
TASK	Sequentially assign each item σ_i to a bin without having any information about items σ_j with $j > i$ such that the sum of item sizes assigned to a bin does not exceed C and the number of bins used is minimal.

In the online version with lookahead, the items have to be packed one after another with limited knowledge about future items.

Problem 4.6 (Online Bin Packing with Lookahead).

INSTANCE	Bin capacity C, sequence of items $(\sigma_1, \dots, \sigma_n)$ where item σ_i has size s_i for $i = 1, \dots, n$, instance revelation rule r' for the lookahead case.
TASK	Sequentially assign an unpacked item which is known as part of the lookahead information to a bin such that the sum of item sizes assigned to a bin does not exceed C and the number of bins used is minimal.

Different types of lookahead are possible in bin packing, and we make the type to be investigated subsequently more concrete. The instance revelation rule in the online case is

$$r := \text{At the beginning, } \sigma_1 \text{ is known; a new item is revealed when a known one is packed}$$

and in the lookahead case with $l \in \mathbb{N}$

$r' :=$ At the beginning, $\sigma_1, \dots, \sigma_l$ are known; a new item is revealed when a known one is packed.

The rule set in the online case is trivial, i.e.,

$$P := \{\text{Pack the known item}\};$$

permutations are allowed in the lookahead case, i.e.,

$$P' := \{\text{Pack one of the known items}\}.$$

According to the modeling framework from Chapter 3, the lookahead setting is

$$req \mid sngl/rnd \mid im \mid discr \qquad \text{or equivalently} \qquad req \mid sngl/rnd \mid im \mid cyc.$$

The input information of an input element is given by s_i. In the online case, we have $\tau_i = i$ and $T_i = i + \epsilon$; in the lookahead case, we have $\tau_i = \max\{1, i - l\}$ and $\underline{T}'_i = \max\{1, i - l\} + \epsilon$, $\overline{T}'_i = \infty$ with sufficiently small $\epsilon > 0$.

We now compare the pure online setting with the setting enhanced by lookahead: Online algorithm BESTFIT puts the known item into the fullest open bin that can accommodate it, if any; otherwise a new bin is opened and the item put in it (cf. Algorithm 4.4 and [55]). Online algorithm BESTFIT$_l$ with request lookahead l first sorts the known items in order of non-increasing sizes and fictively packs them using BESTFIT; the largest known item is then put into the fullest open bin that can accommodate it, if any; otherwise a new bin is opened and the largest known item put in it (cf. Algorithm 4.5 and [55]).

Under additional lookahead of one item ($l = 2$), we derive exact expressions for the counting distribution functions of the objective value and the performance ratio for the case of two item sizes $0.5 + \epsilon$ and $0.5 - \epsilon$ with arbitrary $\epsilon \in (0, \frac{1}{6})$ and $C = 1$. Under these conditions at most two items fit into a single bin. Despite the simple setting, the exact analysis is rather intricate and relies upon the combinatorial structure imposed to the problem. We conclude that there is no general recipe for exact analysis in online optimization with lookahead.

Algorithm 4.4 Online algorithm BestFit for bin packing

```
Input: First element $\sigma_1$ of input sequence $\sigma$ with size $s_1$, bin capacity $C$
 1: $i := 1, J := \emptyset$
 2: while $i \leq n$ do
 3:    Determine the fill levels $f_j$ of all open bins $j \in J$
 4:    if $J' := \{j \in J \mid f_j + s_i \leq C\} \neq \emptyset$ then
 5:       Determine $j' \in J'$ such that $f_{j'} = \max\{f_j \mid j \in J'\}$
 6:    else
 7:       Open a new bin $j'$, $J := J \cup \{j'\}$
 8:    end if
 9:    Assign item $\sigma_i$ to bin $j'$
10:    if $i < n$ then
11:       additional input: $(i+1)$st element $\sigma_{i+1}$ of input sequence $\sigma$ with size $s_{i+1}$
12:    end if
13:    $i := i + 1$
14: end while
Output: Number of bins used $|J|$
```

Algorithm 4.5 Online algorithm BestFit$_l$ with lookahead for bin packing

```
Input: First $l$ elements $\sigma_1, \dots, \sigma_l$ of input sequence $\sigma$ with sizes $s_1, \dots, s_l$, bin capacity $C$
 1: $i := 1, J := \emptyset$
 2: while $i \leq n$ do
 3:    Determine the fill levels $f_j$ of all open bins $j \in J$
 4:    Sort the unpacked known items obtaining list $\sigma_{(1)}, \sigma_{(2)}, \dots$ with $s_{(1)} \geq s_{(2)} \geq \dots$
 5:    if $J' := \{j \in J \mid f_j + s_{(1)} \leq C\} \neq \emptyset$ then
 6:       Determine $j' \in J'$ such that $f_{j'} = \max\{f_j \mid j \in J'\}$
 7:    else
 8:       Open a new bin $j'$, $J := J \cup \{j'\}$
 9:    end if
10:    Assign item $\sigma_{(1)}$ to bin $j'$
11:    if $i \leq n - l$ then
12:       additional input: $(i+l)$th element $\sigma_{i+l}$ of input sequence $\sigma$ with size $s_{i+l}$
13:    end if
14:    $i := i + 1$
15: end while
Output: Number of bins used $|J|$
```

From now on, we refer to BestFit as Bf and to BestFit$_2$ as Bfd because BestFit$_2$ emulates BestFit with the additional feature that the two known items are always sorted by decreasing size. We set $C := 1$, let $\epsilon \in (0, \frac{1}{6})$ and use the following additional notation:

- Large / small item: Item of size $0.5 + \epsilon$ / $0.5 - \epsilon$
- $n^l(\sigma)$ / $n^s(\sigma)$: Number of large / small items in σ
- Bf(n, m) / Bfd(n, m): Number of item sequences of length n which need m bins under BestFit / BestFit$_2$
- C_i: ith Catalan number given by $C_i = \binom{2i}{i} - \binom{2i}{i+1} = \frac{1}{i+1}\binom{2i}{i}$ [12]

Theorem 4.1. *For any item sequence* $\sigma = (\sigma_1, \sigma_2, \ldots)$ *with* $\sigma_i \in \{0.5 + \epsilon, 0.5 - \epsilon\}$ *it holds that* Bf$[\sigma]$ − Bfd$[\sigma] \in \{0, 1\}$.

Proof. The first difference in the packings of Bf and Bfd occurs when item subsequence $(0.5 - \epsilon, 0.5 - \epsilon, 0.5 + \epsilon)$ appears and there is no bin to accommodate any of these items. Bfd packs the first and third item into a single bin at full capacity and keeps the second (small) item unpacked in the lookahead until the end of the sequence (since items are homogenous), whereas Bf packs the first two items in a bin at capacity $1 - 2\epsilon$ and the third item in a second bin. Thus, Bfd leads with one bin less used, but also one small item less packed. Bfd also loses its lookahead power as it holds the small item in the lookahead and will not change orders of two lookahead items ever again. Thus, both Bfd and Bf will process the remaining items in parallel, but starting from a different bin configuration. The number of upcoming *new* bins for the remaining items by Bf can only be the same or one less than that of Bfd (without considering the left-over small item) because items have to be packed in the same order and Bf can pack one small item without opening a new bin, whereas Bfd has to open a new bin immediately. Finally, Bfd has to pack the left-over small item: When the number of new bins in the previous step is the same for Bfd and Bf and in the packing of Bfd there is room for a small item, Bfd will end up with one bin less than Bf, otherwise Bfd will have to open a new bin resulting in a tie for the numbers of bins used. □

From Theorem 4.1 it follows that Bfd dominates Bf in the sense that for each item sequence it produces the same number of bins or even needs one bin less. As a result, Bfd never makes a decision which will have worse consequences than that of Bf in the given setting.

[12] The Catalan numbers (see, e.g., [146]) are a sequence of natural numbers discovered by Eugene Charles Catalan (1814-1894) which appear in an enormous number of counting problems. The Catalan numbers $(C_i)_{i \in \mathbb{N}_0}$ start with $1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \ldots$

Definition 4.1 (Condensation of an input sequence)**.**
Let $\sigma = (\sigma_1, \sigma_2, \ldots)$ be an input sequence with $\sigma_i \in \{0.5+\epsilon, 0.5-\epsilon\}$. The condensation σ^c of σ is the input sequence that arises by repetitively removing all pairs $(0.5-\epsilon, 0.5-\epsilon)$ starting in an odd position if the number of large items encountered previously is not larger than the number of small items encountered previously. △

A pair of removed small items in Definition 4.1 is also referred to as a condensed pair or as a condensation *in* an input sequence.

Example 4.2 (Condensation of an input sequence)**.**
Consider for $\epsilon = 0.1$ the item sequences $\sigma_1 = (0.4, 0.4, 0.6, 0.6, 0.4, 0.6)$ and $\sigma_2 = (0.6, 0.4, 0.4, 0.6, 0.4, 0.4, 0.6, 0.4, 0.4, 0.4)$. The condensation of σ_1 is $\sigma_1^c = (0.6, 0.6, 0.4, 0.6)$; the condensation of σ_2 is $\sigma_2^c = (0.6, 0.4, 0.4, 0.6, 0.6, 0.4)$. ♢

The condensation σ^c of an input sequence σ can be computed by Algorithm 4.6.

Algorithm 4.6 Determining the condensation of an item sequence

Input: Input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$ with $\sigma_j \in \{0.5-\epsilon, 0.5+\epsilon\}$

```
1: $\sigma^c := \sigma, i := 1$
2: while $i \le |\sigma^c| - 1$ do
3:     if $\sigma_i^c = \sigma_{i+1}^c = 0.5-\epsilon$ and $n^l(\sigma_j^c)_{j=1,\ldots,i-1} \le n^s(\sigma_j^c)_{j=1,\ldots,i-1}$ then
4:         Delete $\sigma_i^c$ and $\sigma_{i+1}^c$ from $\sigma^c$ by removal (successive elements are shifted forwards)
5:         $i := i - 2$
6:     end if
7:     $i := i + 2$
8: end while
```

Output: Condensation σ^c of σ

Theorem 4.2. *For any item sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ with $\sigma_i \in \{0.5+\epsilon, 0.5-\epsilon\}$ and $n \in \mathbb{N}$ it holds that $\textsc{Bf}[\sigma] - \textsc{Bfd}[\sigma] = 1$ if and only if there is an odd $j \in \mathbb{N}$ such that the following conditions are satisfied:*

i) $n^l((\sigma_1, \ldots, \sigma_{j-1})) = n^s((\sigma_1, \ldots, \sigma_{j-1})^c)$

ii) $(\sigma_j, \sigma_{j+1}, \sigma_{j+2}) = (0.5-\epsilon, 0.5-\epsilon, 0.5+\epsilon)$

iii) $n^s((\sigma_{j+3}, \ldots, \sigma_n)) = n^s((\sigma_{j+3}, \ldots, \sigma_n)^c)$

iv) $n^l((\sigma_{j+3}, \ldots, \sigma_n)) \ge n^s((\sigma_{j+3}, \ldots, \sigma_n)) + 1$

Proof. $\Rightarrow$: Let $\text{BF}[\sigma] - \text{BFD}[\sigma] = 1$ for $\sigma = (\sigma_1, \ldots, \sigma_n)$. Then σ can be split into $\sigma^i = (\sigma_1, \ldots, \sigma_{j-1})$ and $\sigma^{ii} = (\sigma_j, \ldots, \sigma_n)$ such that $\text{BF}[\sigma^i] - \text{BFD}[\sigma^i] = 0$, $\text{BF}[\sigma^{ii}] - \text{BFD}[\sigma^{ii}] = 1$ and both algorithms produce the *same* bin configurations (albeit in a different order) for σ^i. Among all these splits there exists one with longest σ^i which we refer to as σ^i from now on.

Recall that the first difference in the packings of BF and BFD occurs when item subsequence $(0.5 - \epsilon, 0.5 - \epsilon, 0.5 + \epsilon)$ appears and no open bin can accommodate any of these items. By definition, σ^i immediately precedes this subsequence. If $|\sigma^i|$ was odd, any algorithm would leave a bin with space at least $0.5 - \epsilon$ after packing the odd number of items in σ^i. Hence, the first (small) item of the subsequence could also be added contradicting that no open bin can accommodate any of the items. Thus, $|\sigma^i|$ is even and j is odd.

From Definition 4.1, it follows that $n^l((\sigma^i)) \geq n^s((\sigma^i)^c)$ because any pair of small items that would lead to more small than large items immediately after this pair has been deleted in $n^s((\sigma^i)^c)$ and $|\sigma^i|$ is even. For $n^l((\sigma^i)) = n^s((\sigma^i)^c)$, each large item has a matching small item which comes after or immediately before the large item. Thus, the configuration determined by BF is composed of $n^l((\sigma^i))$ completely filled bins and $\frac{|\sigma^i| - 2n^l((\sigma^i))}{2}$ bins with two small items. Clearly, this number of bins is optimal. From the proof of Theorem 4.1, both BFD and BF attain the optimal number of bins by the same bin configurations for $n^l((\sigma^i)) = n^s((\sigma^i)^c)$. For $n^l((\sigma^i)) > n^s((\sigma^i)^c)$, we show by contradiction that σ^i cannot be a longest possible subsequence such that BFD and BF produce the same bin configurations: Assume that σ^i is a longest possible subsequence such that BFD and BF produce the same bin configurations and $n^l((\sigma^i)) > n^s((\sigma^i)^c)$. Then there is at least one bin containing a large item without a matching small item. An additional small item will be put into such a bin, an additional large item will need a new bin, but the configurations of both algorithms will remain the same contradicting the definition of σ^i. Thus, $n^l((\sigma^i)) = n^s((\sigma^i)^c)$ which is *i)*.

According to *i)* and the definition of σ^i, there must be an odd j such that BFD starts to exhibit an advantage over BF on $\sigma^{ii} = (\sigma_j, \sigma_{j+1}, \sigma_{j+2}, \ldots)$ after σ^i has been packed resulting in the same bin configurations with no space left by both algorithms. Only $(\sigma_j, \sigma_{j+1}) = (0.5 - \epsilon, 0.5 - \epsilon)$ potentially produces a difference. To make this happen, BFD need not pack these two items into the same bin, whereas BF has to. This happens if and only if $\sigma_{j+2} = 0.5 + \epsilon$: BFD will not pack σ_{j+1} immediately, but delay it until the end of the item sequence, whereas σ_{j+2} will be matched with σ_j. This establishes *ii)*.

To see *iii)*, note that the processing of BFD on $\sigma^{ii} = (0.5 - \epsilon, 0.5 - \epsilon, 0.5 + \epsilon, \sigma_{j+3}, \ldots, \sigma_n)$ is emulated by BF on $\tilde{\sigma}^{ii} = (0.5 - \epsilon, 0.5 + \epsilon, \sigma_{j+3}, \ldots, \sigma_n, 0.5 - \epsilon)$. Assume there is a condensed pair of small items in the subsequence starting with σ_{j+3}; if there is more than one condensation, consider the first one. Let $\sigma_{j'}$ be the first small item of this condensation. BF produces a bin with two small items for σ_j and σ_{j+1}, but not for $\sigma_{j'}$ and $\sigma_{j'+1}$ since two small items starting in an even position of the *original* sequence cannot be put in the same bin by BF. BFD processes $(\sigma_j, \sigma_{j+1}, \sigma_{j+2}, \ldots, \sigma_{n-1}, \sigma_n)$ as $(\sigma_j, \sigma_{j+2}, \ldots, \sigma_{n-1}, \sigma_n, \sigma_{j+1})$ emulated by BF, i.e., it does not produce a bin with two small items for σ_j and σ_{j+2}, but for $\sigma_{j'}$ and $\sigma_{j'+1}$ since in $(\sigma^i, \tilde{\sigma}^{ii})$ these items *are* condensed items. Between σ_{j+3} and $\sigma_{j'}$, neither algorithm produces another bin with two small items since we consider the *first* condensation in the subsequence starting from σ_{j+3}. Hence, both BF and BFD produce one additional bin with two small items for $\tilde{\sigma}^i = (\sigma_1, \ldots, \sigma_{j-1}, \sigma_j, \sigma_{j+1}, \sigma_{j+2}, \ldots, \sigma_{j'})$ as compared to σ^i, and σ^i could not have been the longest possible first part among all splits of σ.

From *i)*, *ii)*, *iii)*, we know that in BFD's processing there is no bin with two small items from σ_j onwards, whereas BF creates such a bin for (σ_j, σ_{j+1}). Hence, in order to pack σ_{j+1} at the end of BFD's processing into an already open bin and to save a bin as compared to BF, we need an open bin with a large item only. This is the case if and only if in the subsequence starting from σ_{j+3} at least one more large item exists, i.e., *iv)*.

$\Leftarrow$: We have that for item sequence σ there is an odd $j \in \mathbb{N}$ such that conditions *i)* to *iv)* are fulfilled. In the sequel, a bin is called matched if it contains a large and small item, otherwise it is called unmatched. From *iii)*, we know that BFD will not produce a bin with two small items from σ_j onwards, whereas BF creates such a one for (σ_j, σ_{j+1}). From *iv)*, we conclude that the number of matched bins in BFD is two higher than in BF. From the pigeonhole principle, it follows that the number of unmatched bins in BF is three higher than in BFD. Thus, $\text{BF}[(\sigma_j, \sigma_{j+1}, \ldots, \sigma_n)] - \text{BFD}[(\sigma_j, \sigma_{j+1}, \ldots, \sigma_n)] = 1$. *i)* guarantees that in the bin configurations induced both by BFD and BF there is a matching small item for any large item such that there is no bin with a large item only after $(\sigma_1, \ldots, \sigma_{j-1})$ have been processed. Since $|(\sigma_1, \ldots, \sigma_{j-1})|$ is even, there is no bin with a small item only after $(\sigma_1, \ldots, \sigma_{j-1})$ has been packed. Hence, the initial position for processing $(\sigma_j, \sigma_{j+1}, \ldots, \sigma_n)$ is the same for BFD and BF and can be viewed as restarting with no bins used so far. In particular, $\text{BF}[(\sigma_1, \sigma_2, \ldots, \sigma_{j-1})] - \text{BFD}[(\sigma_1, \sigma_2, \ldots, \sigma_{j-1})] = 0$. □

We are in a position to characterize the number of item sequences of given length which lead to a a saving of one bin by applying BFD instead of BF.

Theorem 4.3.

a) The number of item sequences σ of odd length $|\sigma| = 2n+1$ for $n \in \mathbb{N}$ with $\text{BF}[\sigma] - \text{BFD}[\sigma] = 1$ *is given by*

$$\sum_{p=1}^{n-1}\Big(2^{2(p-1)}-\sum_{i=1}^{p-1}C_i\cdot 2^{2(p-1-i)}\Big)\Big(2^{2(n-p)}-\sum_{i=1}^{n-p}C_i\cdot 2^{2(n-p-i)}-C_{n-p+1}\Big).$$

b) The number of item sequences σ of even length $|\sigma| = 2n$ for $n \in \mathbb{N}$ with $\text{BF}[\sigma] - \text{BFD}[\sigma] = 1$ *is given by*

$$\sum_{p=1}^{n-1}\Big(2^{2(p-1)}-\sum_{i=1}^{p-1}C_i\cdot 2^{2(p-1-i)}\Big)\Big(2^{2(n-p)-1}-\sum_{i=1}^{n-p-1}C_i\cdot 2^{2(n-p-i)-1}-C_{n-p}\Big).$$

Proof.

a) We compute the number of item sequences σ which can be brought into the form $\sigma = (\sigma_1,\ldots,\sigma_{j-1},\sigma_j,\sigma_{j+1},\sigma_{j+2},\sigma_{j+3},\ldots,\sigma_{2n+1})$ fulfilling *i)* to *iv)* from Theorem 4.2 with odd j. We can choose j to be every odd number with $|(\sigma_j,\sigma_{j+1},\sigma_{j+2},\ldots,\sigma_{2n+1})| \geq 4$. The largest j fulfilling this condition is $j = 2(n-1)-1$ such that five items $(\sigma_{2(n-1)-1},\sigma_{2(n-1)},\sigma_{2n-1},\sigma_{2n},\sigma_{2n+1})$ remain. Hence, in the first sum, p runs from 1 to $n-1$ indicating that j runs from $2\cdot 1-1 = 1$ to $2\cdot(n-1)-1$ with even j's omitted.

We first consider the first pair of parentheses: For fixed p, the number of item (sub-) sequences of length $2(p-1)$ satisfying condition *i)* from Theorem 4.2, i.e.,

$$n^l((\sigma_1,\ldots,\sigma_{j-1})) = n^s((\sigma_1,\ldots,\sigma_{j-1})^c) \qquad (\star)$$

with $j-1 = 2(p-1)$ is $n_0 = 2^{2(p-1)} - \sum_{i=1}^{p-1} C_i\cdot 2^{2(p-1-i)}$ which can be seen as follows: It holds that $n_0 = 2^{2(p-1)} - n_{viol}$ where n_{viol} is the number of sequences of length $2(p-1)$ violating $(\star)$. Since by definition $n^l((\sigma_1,\ldots,\sigma_{j-1})) \geq n^s((\sigma_1,\ldots,\sigma_{j-1})^c)$, we have $n_{viol} = |\Sigma^{viol}|$ with $\Sigma^{viol} := \{(\sigma_1,\ldots,\sigma_{j-1})\,|\,n^l((\sigma_1,\ldots,\sigma_{j-1})) > n^s((\sigma_1,\ldots,\sigma_{j-1})^c)\}$. From the pigeonhole principle, we can only have $n^l((\sigma_1,\ldots,\sigma_{j-1})) = n^s((\sigma_1,\ldots,\sigma_{j-1})^c) + d$ where $d = 2,4,6,\ldots$ for $(\sigma_1,\ldots,\sigma_{j-1}) \in \Sigma^{viol}$ because $|(\sigma_1,\ldots,\sigma_{j-1})^c|$ is even. Thus, a pair of large items has to occur at positions $2(p-1-i)+1$ and $2(p-1-i)+2$ for $i \in \{1,2,\ldots,p-1\}$ which will be responsible for $n^l((\sigma_1,\ldots,\sigma_{j-1})) > n^s((\sigma_1,\ldots,\sigma_{j-1})^c)$ under conditions

- $n^l((\sigma_{2(p-1-i)+3},\ldots,\sigma_{2(p-1)})) = n^s((\sigma_{2(p-1-i)+3},\ldots,\sigma_{2(p-1)}))$ and
- $n^s((\sigma_{2(p-1-i)+3},\ldots,\sigma_{2(p-1)})) = n^s((\sigma_{2(p-1-i)+3},\ldots,\sigma_{2(p-1)})^c)$

regardless of $\sigma_1,\ldots,\sigma_{2(p-1-i)}$. Thus, we can choose $\sigma_1,\ldots,\sigma_{2(p-1-i)}$ freely giving the factor $2^{2(p-1-i)}$. Note that the first condition is necessary to ensure that the two large items $\sigma_{2(p-1-i)+1}$ and $\sigma_{2(p-1-i)+2}$ can be held responsible for $n^l((\sigma_1, \dots, \sigma_{j-1})) > n^s((\sigma_1, \dots, \sigma_{j-1})^c)$; the second condition is sufficient to make them responsible for $n^l((\sigma_1, \dots, \sigma_{j-1})) > n^s((\sigma_1, \dots, \sigma_{j-1})^c)$ because if there was a condensed pair of small items, then at least one pair of two large items would follow which would be responsible for $n^l((\sigma_1, \dots, \sigma_{j-1})) > n^s((\sigma_1, \dots, \sigma_{j-1})^c)$. For fixed i, it remains to show that the number of item sequences of length $2(i-1)$ fulfilling the two itemized conditions is C_i.

To this end, we make use of the concept of (recurring) unit-sloped paths ([131]):

Definition 4.3 ((Recurring) unit-sloped path)**.**
A unit-sloped path of length $2i$ is a path in $\mathbb{R}^2$ from $(0,0)$ to $(2i, s_{2i})$ consisting only of line segments between $(k-1, s_{k-1})$ and (k, s_k) for $k = 1, 2, \dots, 2i$ where $s_k = s_{k-1} + 1$ or $s_k = s_{k-1} - 1$ and $s_0 = 0$. A recurring unit-sloped path of length $2i$ is a unit-sloped path of length $2i$ that ends in $(2i, 0)$, i.e., it has $s_{2i} = 0$. △

Note that if we restrict $s_k \geq 0$ for all $k = 0, 1, 2, \dots, 2i$, this definition coincides with that of the well-known Dyck path (see, e.g., [62], [63]). Figure 4.5 shows an example for a recurring unit-sloped path and a Dyck path, respectively.

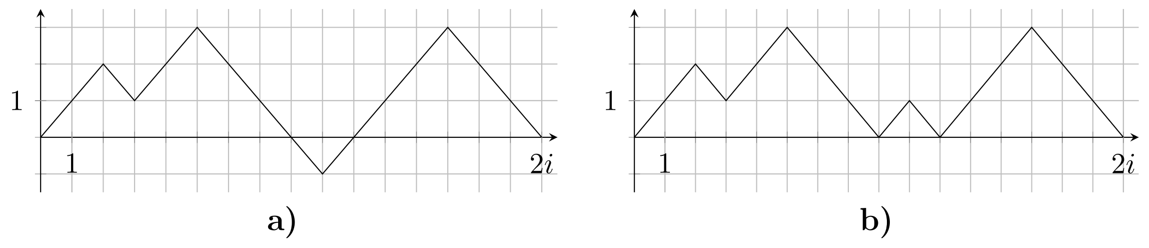


Figure 4.5: Recurring unit sloped paths. **a)** General path. and **b)** Dyck path.

Lemma 4.4.

a) The number of recurring unit-sloped paths of length $2i$ which have $s_k \geq 0$ for all $k = 0, 1, 2, \dots, 2i$ is C_i.

b) The number of recurring unit-sloped paths of length $2i$ which have $s_k \geq -1$ for all $k = 0, 1, 2, \dots, 2i$ is C_{i+1}.

Proof. See Appendix A.1.1. □

In order to fulfill the first itemized condition, observe the correspondences between the appearance of a large item and an up-move ($s_k = s_{k-1} + 1$) and between the appearance of a small item and a down-move ($s_k = s_{k-1} - 1$) in a unit-sloped path. Because the number of large items equals the number of small items, this path is also recurring.

The second itemized condition expresses that there are never two small items such that the number of large items minus the number of small items up to an arbitrary position in the item sequence would drop down to -2. Hence, the number of item sequences of length $2(i-1)$ fulfilling both conditions is equal to the number of recurring unit-sloped paths with $s_k \geq -1$ for all k. According to Lemma 4.4 b), this number is C_i.

We now consider the second pair of parentheses: For fixed p, the number of item sequences of length $2(n-p)$ which fulfill conditions *iii)* and *iv)* from Theorem 4.2, i.e.,

- $n^s((\sigma_{j+3}, \ldots, \sigma_{2n+1})) = n^s((\sigma_{j+3}, \ldots, \sigma_{2n+1})^c)$ and
- $n^l((\sigma_{j+3}, \ldots, \sigma_{2n+1})) \geq n^s((\sigma_{j+3}, \ldots, \sigma_{2n+1})) + 1$

with $j = 2p-1$ can be characterized as $2^{2(n-p)} - n_1 - n_2$ where n_1 is the number of item sequences of length $2(n-p)$ with at least one condensation and n_2 is the number of item sequences of length $2(n-p)$ without condensation that has the same number of small and large items.

For n_1, let $i \in \{1, \ldots, n-p\}$ be fixed such that the first condensation consists of $\sigma_{2(p+i)}$ and $\sigma_{2(p+i)+1}$, then we need to have $n^l(\sigma_{j+3}, \ldots, \sigma_{2(p+i)-1}) = n^s(\sigma_{j+3}, \ldots, \sigma_{2(p+i)-1})$ and no condensation is allowed in $(\sigma_{j+3}, \ldots, \sigma_{2(p+i)-1})$. Structurally, we obtain the same two conditions as the two itemized conditions for the first pair of parentheses, i.e.,

- $n^l((\sigma_{2(p+1)}, \ldots, \sigma_{2(p+i)-1})) = n^s((\sigma_{2(p+1)}, \ldots, \sigma_{2(p+i)-1}))$ and
- $n^s((\sigma_{2(p+1)}, \ldots, \sigma_{2(p+i)-1})) = n^s((\sigma_{2(p+1)}, \ldots, \sigma_{2(p+i)-1})^c)$

regardless of $\sigma_{2(p+i)+2}, \ldots, \sigma_{2n+1}$ because once a condensation occurs the rest is irrelevant. Since $\sigma_{2(p+i)+2}, \ldots, \sigma_{2n+1}$ are free, the factor $2^{2(n-p-i)}$ follows, and from Lemma 4.4, the number of item sequences of length $2(i-1)$ fulfilling the two itemized conditions is C_i. Thus, we obtain $n_1 = \sum_{i=1}^{n-p} C_i \cdot 2^{2(n-p-i)}$. Also from Lemma 4.4, it follows that $n_2 = C_{n-p+1}$ gives the number of item sequences of length $2(n-p)$ where the number of large items equals the number of small items and no condensations occur.

b) The proof is analogous to part a) with the only difference that the decomposition into $\sigma = (\sigma_1, \ldots, \sigma_{j-1}, \sigma_j, \sigma_{j+1}, \sigma_{j+2}, \sigma_{j+3}, \ldots, \sigma_{2n+1})$ now has odd $|(\sigma_{j+3}, \ldots, \sigma_{2n})|$. Thus, for a given $p \in \{1, \ldots, n-1\}$, the subsequence $(\sigma_{j+3}, \ldots, \sigma_{2n})$ now only has $2(n-p)-1$ elements, and the number of sequences of (longest possible even) length $2(n-p-1)$ without condensation and balanced number of small and large items is C_{n-p}.

□

The previous result yielded the *number* of item sequences of given length for which BFD saves a bin as compared to BF. Although this cannot be used directly for the counting distribution functions of the objective value or performance ratio, many proof ideas are reused subsequently.

Lemma 4.5.

a) The number $a_{n,k}$ of unit-sloped paths of length $2n$ with $s_{k'} \geq -1$ for all $k' = 0, 1, 2, \dots, 2n$ which end at position $(2n, 2k)$, i.e., at height $2k$, is given by

$$a_{n,k} = \frac{k+1}{n+1}\binom{2n+2}{n-k}.$$

b) The number of item sequences σ of length $2n$ without condensations where $\mathrm{BF}[\sigma] = m$ for $m \in \{n, n+1, \dots, 2n\}$ is given by $a_{n,m-n}$.

Proof.

a) The proof is an immediate consequence of the bijection between Dyck paths of length $2n+2$ and path pairs of length n given in [63] and the included remark concerning the relaxation of the restriction of path pairs having to end in the same point. To this end, we first modify the bijection by omitting the appended u-step at the beginning and the appended d-step at the end of the Dyck path in order to facilitate recurring unit-sloped paths that are allowed to hit the level of -1. Moreover, since the number of path pairs of length n having endpoints $k\sqrt{2}$ apart is $a_{n,k}$[13], it follows from the bijection that the number of unit-sloped paths ending at height $2k$ is $a_{n,k}$.

b) A total number of m bins with $m \in \{n, n+1, \dots, 2n\}$ is obtained when in an item sequence without condensations $m - n$ out of the n pairs of successive items are pairs of large items for which no matching small items can be found afterwards. Each such item sequence corresponds to a unit-sloped path of length $2n$ with $s_k \geq -1$ for $k = 0, 1, 2, \dots, 2n$ ending at height $2(m-n)$ because each pair of large items contributes an amount of 2 to the total height achieved at the end of the path, and the result follows.

$\square$

[13] It also holds that $a_{n,k} = \sum_{i_1+i_2+\ldots+i_k=n} C_{i_1} C_{i_2} \cdots C_{i_k}$, i.e., $a_{n,k}$ is the sum of products of Catalan numbers which is why the resulting table for $n, k \in \mathbb{N}$ with $k \leq n$ is called Catalan triangle ([146]).

Theorem 4.6.

a) The number of item sequences σ of length $2n$ where $\mathrm{BF}[\sigma] = m$ is given by

$$\mathrm{BF}(2n, m) = \begin{cases} \sum_{k=m-n}^{n} a_{n,k} = \sum_{k=m-n}^{n} \frac{k+1}{n+1}\binom{2n+2}{n-k} & \text{if } n \leq m \leq 2n, \\ 0 & \text{otherwise.} \end{cases}$$

b) The number of item sequences σ of length $2n+1$ where $\mathrm{BF}[\sigma] = m$ is given by

$$\mathrm{BF}(2n+1, m) = \begin{cases} 2\sum_{k=m-n}^{n} a_{n,k} + a_{n,m-1-n} & \text{if } n+1 < m \leq 2n+1, \\ 3\sum_{k=0}^{n} a_{n,k} - a_{n,0} & \text{if } m = n+1, \\ 0 & \text{otherwise.} \end{cases}$$

Proof.

a) Since for $2n$ items at least n and at most $2n$ bins are needed, $\mathrm{BF}(2n, m) = 0$ for $m < n$ and $m > 2n$. For the remaining m, we perform a reverse induction on m. The base case $m = 2n$ is valid because the only item sequence which needs $2n$ bins has $2n$ large items and it holds that

$$\sum_{k=2n-n}^{n} a_{n,k} = a_{n,n} = \frac{n+1}{n+1}\binom{2n+2}{0} = 1.$$

For the inductive step, let $\mathrm{BF}(2n, m) = \sum_{k=m-n}^{n} a_{n,k}$ be valid for some m with $2n \geq m > n$. We show that $\mathrm{BF}(2n, m-1) = \sum_{k=m-1-n}^{n} a_{n,k}$. Because of $m > n$, there must be a pair of large items starting at an odd position for which no matching small items follow in every item sequence with objective value m since otherwise these large items could be matched with small items and would fit into a bin contradicting $m > n$. Hence, we obtain for any item sequence with objective value m an item sequence with objective value $m - 1$ by replacing the first pair of large items starting at an odd position for which no matching small items follow with a pair of small items which in turn lead to a condensation. As a result, we have $\mathrm{BF}(2n, m-1) = \mathrm{BF}(2n, m) + |\Sigma_{add}|$ where Σ_{add} are the additional item sequences leading to objective value $m - 1$ which have *not* resulted from establishing a condensation in an item sequence with objective value

m. These item sequences can be mapped to a unit-sloped path of length $2n$ not going below level -1 and ending at height $2(m-1-n)$. From Lemma 4.5, we have that $|\Sigma_{add}| = a_{n,m-n-1}$. Together with the induction hypothesis we conclude that

$$\textsc{Bf}(2n, m-1) = \textsc{Bf}(2n, m) + |\Sigma_{add}| = \sum_{k=m-n}^{n} a_{n,k} + a_{n,m-n-1} = \sum_{k=m-n-1}^{n} a_{n,k}.$$

b) Since for $2n+1$ items at least $n+1$ and at most $2n+1$ bins are needed, $\textsc{Bf}(2n+1, m) = 0$ for $m < n+1$ and $m > 2n+1$. Notice that whenever $m > n+1$ for an item sequence of length $2n+1$, we have $m > n$ for the same item sequence where the last item is deleted. Thus, there must be a pair of large items beginning at an odd position in the truncated sequence from the same reasoning as in part a) of the proof. Objective value m with $n+1 < m \leq 2n+1$ for an item sequence of length $2n+1$ can be attained in two ways: First, Bf needed $m-1$ bins after $2n$ items and the $2n+1$st item leads to the mth bin. Second, Bf needed m bins after $2n$ items and the $2n+1$st item needs no new bin. In the first case, we have $\textsc{Bf}(2n, m-1)$ item sequences which must incur a new bin upon appending a large item; appending a small item would leave the objective value at m because there are at least two large items which could be matched with the small item. In the second case, $\textsc{Bf}(2n, m)$ item sequences will not incur a new bin upon appending a small item as this item can be matched with one of the large items; appending a large item would lead to objective value $m+1$ since after $2n$ items there can never be a bin with a small item only.

We obtain

$$\begin{aligned}\textsc{Bf}(2n+1, m) &= \textsc{Bf}(2n, m-1) + \textsc{Bf}(2n, m)\\ &= \sum_{k=m-1-n}^{n} a_{n,k} + \sum_{k=m-n}^{n} a_{n,k} = 2\sum_{k=m-n}^{n} a_{n,k} + a_{n,m-1-n}.\end{aligned}$$

Objective value $n+1$ can be attained in three ways: First, Bf needed n bins after $2n$ items and the $2n+1$st item is large leading to the $n+1$st bin. Second, Bf needed n bins after $2n$ items and the $2n+1$st item is small leading to the $n+1$st bin. Third, Bf needed $n+1$ bins after $2n$ items and the $2n+1$st item is small, but does not lead to a new bin. The first case is trivial. In the second case, we seek for the same item sequences because neither of them can exhibit a pair of large items starting in an odd position. In the third case, we seek for the item sequences of length $2n$ with objective value $n+1$ which have at least one pair of large items beginning at an odd position such that the appended small item does not incur a new bin. These item sequences are counted by $\text{BF}(2n, n+1)$. We obtain

$$\begin{aligned}\text{BF}(2n+1, n+1) &= \text{BF}(2n, n) + \text{BF}(2n, n) + \text{BF}(2n, n+1)\\ &= \sum_{k=0}^{n} a_{n,k} + \sum_{k=0}^{n} a_{n,k} + \sum_{k=1}^{n} a_{n,k} = 3\sum_{k=0}^{n} a_{n,k} - a_{n,0}.\end{aligned}$$

□

We now give another formula to compute the numbers $\text{BF}(2n, m)$ and $\text{BF}(2n+1, m)$.

Theorem 4.7.

a) The number of item sequences σ of length $2n$ where $\text{BF}[\sigma] = m$ is given by

$$\text{BF}(2n, m) = \begin{cases} \binom{2n+1}{m+1} = \binom{2n}{m} + \binom{2n}{m+1} & \text{if } n \le m \le 2n,\\ 0 & \text{otherwise.}\end{cases}$$

b) The number of item sequences σ of length $2n+1$ where $\text{BF}[\sigma] = m$ is given by

$$\text{BF}(2n+1, m) = \begin{cases} \binom{2n+2}{m+1} = \binom{2n+1}{m} + \binom{2n+1}{m+1} & \text{if } n+1 < m \le 2n+1,\\ \binom{2n+3}{n+2} - \binom{2n+1}{n+1} & \text{if } m = n+1,\\ 0 & \text{otherwise.}\end{cases}$$

Proof.

a) We show by two-dimensional induction on n and m that

$$\sum_{k=m-n}^{n} a_{n,k} = \binom{2n+1}{m+1}.$$

Recall that $n = 1, 2, \ldots$ and $m = n, n+1, \ldots, 2n$. The base case $n = 1$ and $m = n = 1$ is valid because it holds that

$$\sum_{k=0}^{1} a_{1,k} = a_{1,0} + a_{1,1} = 2 + 1 = 3 = \binom{2\cdot 1+1}{1+1} = \binom{3}{2} = 3.$$

In the first inductive step (on n with fixed $m = n$), we show that $\sum_{k=0}^{n} a_{n,k} = \binom{2n+1}{n+1}$ holds. From [146], we know that $\frac{1}{2}\binom{2(n+1)}{n+1} = \sum_{k=0}^{n} a_{n,k}$. The result follows from

$$\frac{1}{2}\binom{2(n+1)}{n+1} = \frac{1}{2}\frac{(2n+2)!}{(n+1)!(n+1)!} = \frac{(2n+2)(2n+1)!}{2(n+1)n!(n+1)!} = \binom{2n+1}{n+1}.$$

In the second inductive step (on m with arbitrary n), we show that $\sum_{k=m-n}^{n} a_{n,k} = \binom{2n+1}{m+1}$ implies $\sum_{k=m+1-n}^{n} a_{n,k} = \binom{2n+1}{m+2}$. This can be seen by the following calculations:

$$\begin{aligned}
\sum_{k=m+1-n}^{n} a_{n,k} = \sum_{k=m-n}^{n} a_{n,k} - a_{n,m-n} &= \binom{2n+1}{m+1} - \frac{m-n+1}{n+1}\binom{2n+2}{2n-m} \\
&= \frac{(2n+1)!}{(m+1)!(2n-m)!} - \frac{m-n+1}{n+1}\frac{(2n+2)!}{(2n-m)!(m+2)!} \\
&= \frac{(2n+1)!(n+1)(m+2) - (m-n+1)(2n+2)!}{(2n-m)!(m+2)!(n+1)} \\
&= \frac{(2n+1)!}{(m+2)!(2n-m-1)!}\frac{(n+1)(m+2) - (m-n+1)(2n+2)}{(2n-m)(n+1)} \\
&= \frac{(2n+1)!}{(m+2)!(2n-m-1)!} \cdot 1 = \binom{2n+1}{m+2}.
\end{aligned}$$

b) For $m = n+1$, we have

$$\begin{aligned}
3\sum_{k=0}^{n} a_{n,k} - a_{n,0} &\overset{a)}{=} 3\binom{2n+1}{n+1} - \frac{1}{n+1}\binom{2n+2}{n} \\
&= 3\frac{(2n+1)!}{(n+1)!n!} - \frac{1}{n+1}\frac{(2n+2)!}{n!(n+2)!} \\
&= \frac{(2n+1)!}{(n+2)!(n+1)!}(3(n+2)(n+1) - (2n+2)) \\
&= \frac{(2n+1)!}{(n+2)!(n+1)!}(3n^2 + 7n + 4)
\end{aligned}$$

and

$$\begin{aligned}\binom{2n+3}{n+2}-\binom{2n+1}{n+1} &= \frac{(2n+3)!}{(n+2)!(n+1)!}-\frac{(2n+1)!}{(n+1)!n!}\\ &= \frac{(2n+1)!}{(n+2)!(n+1)!}((2n+3)(2n+2)-(n+2)(n+1))\\ &= \frac{(2n+1)!}{(n+2)!(n+1)!}(3n^2+7n+4).\end{aligned}$$

which together yields the desired relation for $m = n + 1$.

For $n + 1 < m \leq 2n + 1$, we have

$$\begin{aligned}2\sum_{k=m-n}^{n} a_{n,k} + a_{n,m-n-1} &\overset{a)}{=} 2\binom{2n+1}{m+1}+\frac{m-n}{n+1}\binom{2n+2}{2n-m+1}\\ &= 2\frac{(2n+1)!}{(m+1)!(2n-m)!}+\frac{m-n}{n+1}\frac{(2n+2)!}{(2n-m+1)!(m+1)!}\\ &= \frac{(2n+2)!}{(m+1)!(2n-m+1)!}\left(\frac{2(2n-m+1)}{2n+2}+\frac{m-n}{n+1}\right)\\ &= \frac{(2n+2)!}{(m+1)!(2n-m+1)!}\left(\frac{2n+2}{2n+2}\right)=\binom{2n+2}{m+1}.\end{aligned}$$

□

Theorem 4.8. *The number of item sequences σ of length n where* $\text{BF}[\sigma] = m$ *and* $\text{BFD}[\sigma] = m - 1$ *is given by* $\binom{n}{m+1}$ *for* $m = \lceil \frac{n}{2} \rceil + 1, \dots, n - 1$.

Proof. From [143], we have for $1 \leq q \leq p \leq 2q - 1$ the following expression for the binomial coefficient:

$$\binom{p}{q} = \sum_{i\geq 0} C_i \binom{p-1-2i}{q-1-i}$$

For item sequence σ, assume that $|\sigma| = 2n$ and that σ fulfills the conditions stated in Theorem 4.2. We further decompose σ into three parts: For some $i \in \{1, \dots, n-1\}$, let $(\sigma_1, \sigma_2, \dots, \sigma_{2(i-1)})$ correspond to a recurring unit-sloped path with $s_k \geq -1$ for all $k = 1, 2, \dots, 2(i-1)$, let $\sigma_{2i-1}, \sigma_{2i}$ be the first two small items which would be condensed, and let $(\sigma_{2i+1}, \sigma_{2(i+1)}, \ldots, \sigma_{2n})$ be the rest of σ. Note that $\sigma_{2i-1}, \sigma_{2i}$ are not necessarily responsible for the saving which can be accrued by BFD over BF.

According to Lemma 4.4, the number of item sequences fulfilling the properties of the first subsequence is C_i. The number of item subsequences $(\sigma_{2i+1}, \sigma_{2(i+1)}, \ldots, \sigma_{2n})$ leading to overall m bins for BF and overall $m-1$ bins for BFD is $\binom{2n-2i}{m-i}$ which can be seen as follows: For $(\sigma_1, \sigma_2, \ldots, \sigma_{2(i-1)})$, $i-1$ bins were needed such that for $(\sigma_{2i-1}, \sigma_{2i}, \ldots, \sigma_{2n})$, $m-i+1$ and $m-i$ additional bins will be needed by BF and BFD, respectively. Since $\sigma_{2i-1}, \sigma_{2i}$ are small items, $m-i$ bins will be needed by BF for $(\sigma_{2i+1}, \sigma_{2(i+1)}, \ldots, \sigma_{2n})$ and in addition, conditions *ii*), *iii*) and *iv*) of Theorem 4.2 have to hold for some $j \geq 2i-1$ to realize the saving of BFD. Because condition *iv*) of Theorem 4.2 has to hold, objective value m is attained by BF when $m-i$ out of the $2n-2i$ items $(\sigma_{2i+1}, \sigma_{2(i+1)}, \ldots, \sigma_{2n})$ are large. However, this choice does not necessarily fulfill conditions *ii*) and *iii*) of Theorem 4.2 because it may be that $(\sigma_{2i+1}, \sigma_{2i+2}) = (0.5-\epsilon, 0.5+\epsilon)$ or that condensations can be found in $(\sigma_{2i+2}, \sigma_{2i+3}, \ldots, \sigma_{2n})$. These two cases are resolved as follows: Whenever $(\sigma_{2i+1}, \sigma_{2i+2}) = (0.5-\epsilon, 0.5+\epsilon)$, we find a bijective mapping from σ to some σ' where σ' is identical to σ except for $(\sigma_{2i+1}, \sigma_{2i+2})$ now being $(\sigma_{2i+1}, \sigma_{2i+2}) = (0.5-\epsilon, 0.5-\epsilon)$. Hereby, another condensation is established which erases responsibility of the first condensation for the saving of BFD and claims responsibility itself (by setting $j = 2i+1$). Whenever we have another condensation in $(\sigma_{2i+2}, \sigma_{2i+3}, \ldots, \sigma_{2n})$, we recognize that it erases responsibility for the saving of BFD of a previous condensation and claims responsibility itself (by setting j accordingly). Thus, for some i such that $(\sigma_{2i-1}, \sigma_{2i})$ is the first condensation, there are $\binom{2n-2i}{m-i}$ item subsequences $(\sigma_{2i+1}, \sigma_{2(i+1)}, \ldots, \sigma_{2n})$ fulfilling conditions *ii*), *iii*) and *iv*) of Theorem 4.2.

Lemma 4.9. *For $n \leq m \leq 2n$ it holds that*

$$\sum_{i\geq 1} C_i \binom{2n-2i}{m-i} = \sum_{i\geq 1} C_{i-1} \binom{2n-2i+1}{m-i+1}.$$

Proof. See Appendix A.1.2. □

We complete the proof for even length $2n$ of σ with the following calculation:

$$\begin{aligned}\sum_{i\geq 1} C_i \binom{2n-2i}{m-i} &= \sum_{i\geq 1} C_{i-1} \binom{2n-2i+1}{m-i+1} = \sum_{i\geq 1} \frac{1}{i} \binom{2i-2}{i-1} \binom{2n-2i+1}{m-i+1} \\ &= \sum_{i\geq 0} \frac{1}{i+1} \binom{2i}{i} \binom{2n-2i-1}{m-i} = \sum_{i\geq 0} C_i \binom{2n-2i-1}{m-i} = \binom{2n}{m+1}.\end{aligned}$$

where the the last equality follows for $n < m \leq 2n-1$ from the proof of Lemma 9 in [143] and condition *iv)* from Theorem 4.2. For odd length $2n+1$ of σ, the proof is analogous with $2n-2i$ replaced by $2n-2i+1$. □

Corollary 4.10. *The number of item sequences σ of length n with* $\textsc{Bf}[\sigma] = m$ *and* $\textsc{Bfd}[\sigma] = m$ *is given by* $\binom{n}{m}$ *for* $m = \lceil \frac{n}{2} \rceil + 1, \dots, n$.

Proof. From Theorem 4.1, we know that $\textsc{Bfd}[\sigma] \in \{m, m-1\}$ whenever $\textsc{Bf}[\sigma] = m$; from the previous Theorem 4.8, we know that $|\{\sigma \mid |\sigma| = n, \textsc{Bfd}[\sigma] = m-1, \textsc{Bf}[\sigma] = m\}| = \binom{n}{m+1}$ for $m = \lceil \frac{n}{2} \rceil + 1, \dots, n-1$. Hence, from Theorem 4.7 it immediately follows for these m that

$$|\{\sigma \mid |\sigma| = n, \textsc{Bfd}[\sigma] = m\}| = \binom{n+1}{m+1} - \binom{n}{m+1} = \binom{n}{m}.$$

Clearly, $|\{\sigma \mid |\sigma| = n, \textsc{Bf}[\sigma] = n\}| = |\{\sigma \mid |\sigma| = n, \textsc{Bfd}[\sigma] = n\}| = 1$. □

We are in a position to state the central relation between the objective values attained by Bfd and Bf.

Theorem 4.11. *The number of item sequences σ of length n where* $\textsc{Bfd}[\sigma] = m$ *is given by*

$$\textsc{Bfd}(n,m) = \begin{cases} 1 & \text{if } m = n, \\ \textsc{Bf}(n,m) + \textsc{Bf}(n,m+1) - 2\binom{n}{m+1} & \text{if } m = \lceil \frac{n}{2} \rceil + 1, \dots, n-1, \\ \textsc{Bf}(n,m) + \textsc{Bf}(n,m+1) - \binom{n}{m+1} & \text{if } m = \lceil \frac{n}{2} \rceil, \\ 0 & \text{if } m \leq \lceil \frac{n}{2} \rceil - 1. \end{cases}$$

Proof. Because of item sizes in $\{0.5-\epsilon, 0.5+\epsilon\}$, at most two items can be packed in a bin so that packing n items in less than $\lceil \frac{n}{2} \rceil$ bins is infeasible. Likewise, each item of σ is packed separately if and only if each item is large, and there is only one such item sequence σ.

The number of item sequences of length n for which Bfd attains objective value m can be computed as $n_1 + n_2 - n_3 - n_4$ where n_1 is the number of item sequences σ of length n with $\textsc{Bf}[\sigma] = m$, n_2 is the number of item sequences σ of length n with $\textsc{Bf}[\sigma] = m+1$, n_3 is the number of item sequences σ of length n with $\textsc{Bf}[\sigma] = m$, $\textsc{Bfd}[\sigma] = m-1$, and n_4 is the number of item sequences σ of length n with $\textsc{Bf}[\sigma] = m+1$, $\textsc{Bfd}[\sigma] = m+1$.

From Theorem 4.8, we have that the number of all item sequences σ of length n with $\textsc{Bf}[\sigma] = m$ and $\textsc{Bfd}[\sigma] = m-1$ is $n_3 = \binom{n}{m+1}$ for $m = \lceil \frac{n}{2} \rceil + 1, \dots, n-1$; from Corollary 4.10, we have that the number of all item sequences σ of length n with $\textsc{Bf}[\sigma] = m+1$ and $\textsc{Bfd}[\sigma] = m+1$ is $n_4 = \binom{n}{m+1}$ for $m = \lceil \frac{n}{2} \rceil, \dots, n-1$, and the result follows. □

From the previous results, we immediately obtain expressions for the counting distribution functions of the objective value $F_{\text{BF}}(v)$ and $F_{\text{BFD}}(v)$, respectively. We now restrict ourselves to item sequences of even length because from the previous results analogous conclusions can be drawn immediately for item sequences of odd length.

Corollary 4.12. *The counting distribution functions of the objective value $F_{\text{BF}}(v)$ and $F_{\text{BFD}}(v)$ of* Bf *and* Bfd *for item sequences of length $2n$ are given by*

$$F_{\text{BF}}(v) = \begin{cases} 0 & \textit{if } v < n, \\ 2^{-2n} \sum\limits_{m=n}^{\lfloor v \rfloor} \binom{2n+1}{m+1} & \textit{if } n \leq v < 2n, \\ 1 & \textit{if } v \geq 2n, \end{cases}$$

and

$$F_{\text{BFD}}(v) = \begin{cases} 0 & \textit{if } v < n, \\ 2^{-2n} \left(\sum\limits_{m=n}^{\lfloor v \rfloor} \left(\binom{2n+1}{m+1} + \binom{2n+1}{m+2} - 2\binom{2n}{m+1} \right) + \binom{2n}{m+1} \right) & \textit{if } n \leq v < 2n, \\ 1 & \textit{if } v \geq 2n. \end{cases}$$

Proof. Direct consequence of Theorems 4.7 and 4.11. □

The following corollary expresses Bfd's dominance over Bf. As could already be seen from Theorem 4.1, stochastic dominance of all orders is established.

Theorem 4.13. *For item sequences of length $2n$, we have $F_{\text{BFD}}(v) \geq F_{\text{BF}}(v)$ for all $v \in \mathbb{R}$ and $F_{\text{BFD}}(v) - F_{\text{BF}}(v) > F_{\text{BFD}}(v+1) - F_{\text{BF}}(v+1)$ for all $v < 2n - 1$.*

Proof. For $v < n$ and $v \geq 2n - 1$, $F_{\text{BF}}(v) = F_{\text{BFD}}(v)$. For $n \leq v < 2n - 1$

$$\begin{aligned} F_{\text{BFD}}(v) - F_{\text{BF}}(v) &= \left(\sum_{m=n}^{\lfloor v \rfloor} \left(\binom{2n+1}{m+2} - 2\binom{2n}{m+1} \right) + \binom{2n}{n+1} \right) \cdot (2^{-2n}) \\ &= \left(\sum_{m=n}^{\lfloor v \rfloor} \left(\binom{2n+1}{m+2} - \binom{2n}{m+1} - \binom{2n}{m+1} \right) + \binom{2n}{n+1} \right) \cdot (2^{-2n}) \\ &= \left(\sum_{m=n}^{\lfloor v \rfloor} \left(\binom{2n}{m+2} - \binom{2n}{m+1} \right) + \binom{2n}{n+1} \right) \cdot (2^{-2n}) \\ &= \left(\binom{2n}{\lfloor v \rfloor + 2} - \binom{2n}{n+1} + \binom{2n}{n+1} \right) \cdot (2^{-2n}) = \binom{2n}{\lfloor v \rfloor + 2} \cdot (2^{-2n}) > 0. \end{aligned}$$

The second part follows immediately from Pascal's triangle as a result of $\binom{2n}{v+2} > \binom{2n}{v+3}$ for $v = n, n+1, \ldots, 2n-3$. □

Theorem 4.14. *For item sequences of length* $2n$*, we have* $\sup_{v\in\mathbb{R}} |F_{\text{BFD}}(v) - F_{\text{BF}}(v)| \to 0$ *as* $n \to \infty$*.*

Proof. Using the formula of Stirling ($n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$; see, e.g., [114]), we get for $n \to \infty$ that

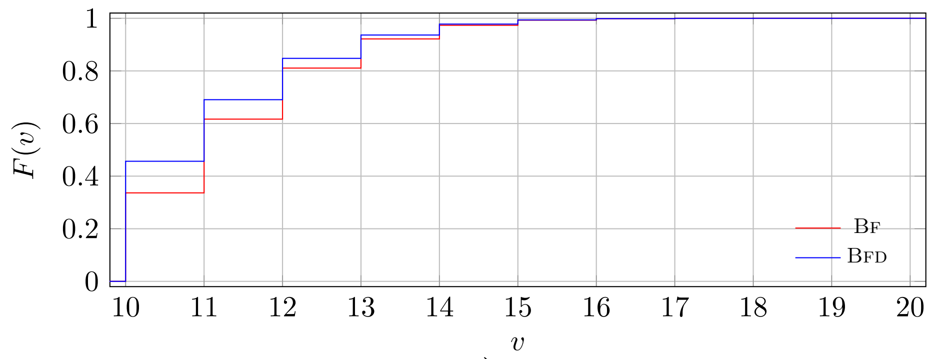
$$\begin{aligned}
F_{\text{BFD}}(n) - F_{\text{BF}}(n) &= \binom{2n}{n+2} \cdot 2^{-2n} \\
&= \frac{(2n)!}{(n+2)!(n-2)!} \cdot 2^{-2n} \\
&\approx \frac{\sqrt{4\pi n}(\frac{2n}{e})^{2n}}{\sqrt{2\pi(n+2)}(\frac{n+2}{e})^{n+2}\sqrt{2\pi(n-2)}(\frac{n-2}{e})^{n-2}} \cdot 2^{-2n} \\
&= \frac{\sqrt{n}(\frac{2n}{e})^{2n}}{\sqrt{\pi(n^2-4)}(\frac{n+2}{e})^{n+2}(\frac{n-2}{e})^{n-2}} \cdot 2^{-2n} \\
&= \frac{\sqrt{n}(2n)^{2n}}{\sqrt{\pi(n^2-4)}(n+2)^{n+2}(n-2)^{n-2}} \cdot 2^{-2n} \\
&= \frac{\sqrt{n}2^{2n}n^{2n}}{\sqrt{\pi(n^2-4)}(n+2)^{n+2}(n-2)^{n-2}} \cdot 2^{-2n} \in \Theta(\frac{1}{\sqrt{n}}).
\end{aligned}$$

In addition, $\binom{2n}{v} > \binom{2n}{v+1}$ for $v \in \mathbb{N}$ with $v \geq n$, i.e., $F_{\text{BFD}}(v) - F_{\text{BF}}(v)$ monotonously decreases for $v \geq n$. Since also $F_{\text{BFD}}(v) = F_{\text{BF}}(v)$ for $v < n$ and $v \geq 2n-1$, the result follows. □

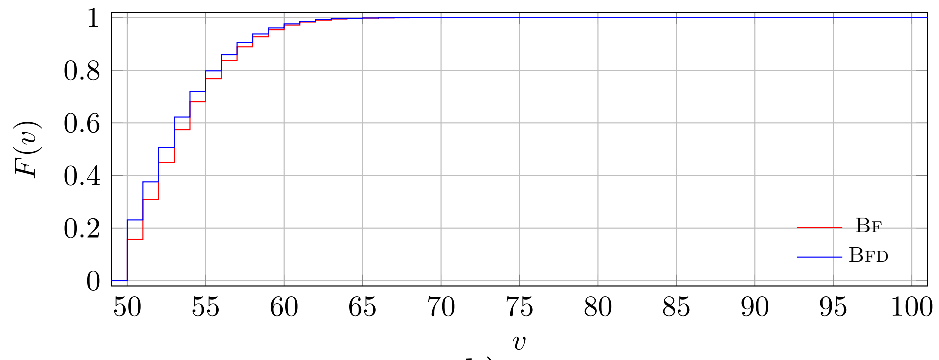
As a consequence of Theorem 4.14, one additional lookahead item is worthless in the limit for BFD in comparison to BF. Taking into account Theorem 4.1, this result was to be expected. The reason for this ineffectiveness lies in the total forfeiture of the power of the lookahead capability once a small item occurs and occupies the lookahead set.

Figure 4.6 exemplarily plots $F_{\text{BF}}(v)$ and $F_{\text{BFD}}(v)$ for all item sequences of length $n = 10, 50, 100, 500$. We observe a rather small lookahead effect as a result of the conditions in Theorem 4.2 which are collectively found only in a minor fraction of all item sequences.

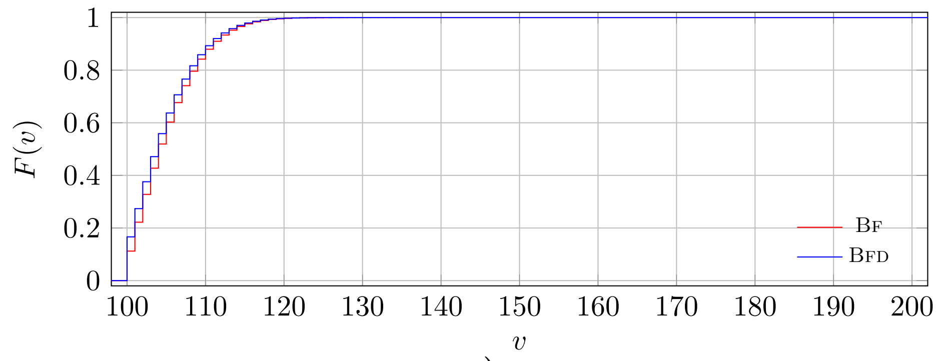
We next derive an expression for the counting distribution function of the performance ratio $F_{\frac{\text{BF}}{\text{BFD}}}(v)$ of BF relative to BFD.



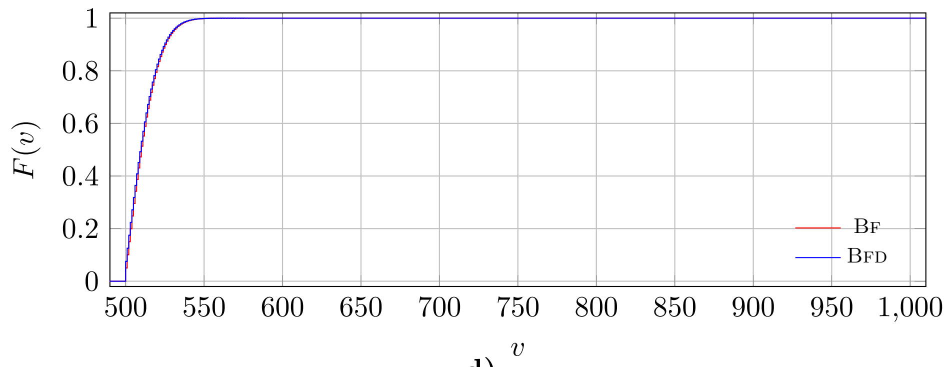
a)



b)



c)



d)

Figure 4.6: Counting distribution functions of costs in the bin packing problem. **a)** $2n = 20$. **b)** $2n = 100$. **c)** $2n = 200$. **d)** $2n = 1000$.

Corollary 4.15. *The counting distribution function of the performance ratio $F_{\frac{\text{Bf}}{\text{Bfd}}}(v)$ of* Bf *relative to* Bfd *for item sequences of length $2n$ is given by*

$$F_{\frac{\text{Bf}}{\text{Bfd}}}(v) = \begin{cases} 0 & \textit{if } v < 1, \\ \frac{1}{2} + \frac{\frac{1}{2}\binom{2n}{n} + \binom{2n}{n+1}}{2^{2n}} & \textit{if } 1 \le v < \frac{2n-1}{2n-2} = 1 + \frac{1}{2n-2}, \\ \frac{1}{2} + \frac{\frac{1}{2}\binom{2n}{n} + \binom{2n}{n+1} + \sum_{i=0}^{n-2-k} \binom{2n}{2n-i}}{2^{2n}} & \textit{if } \frac{n+k+1}{n+k} = 1 + \frac{1}{n+k} \le v < \frac{n+k}{n+k-1} = 1 + \frac{1}{n+k-1} \\ & \textit{for } k = 1, \dots, n-2, \\ 1 & \textit{else.} \end{cases}$$

Proof. According to Theorem 4.8, we have:

- $\binom{2n}{n+2}$ is the number of sequences of length $2n$ with $\text{Bf}[\sigma] = n+1$, $\text{Bfd}[\sigma] = n$
- $\binom{2n}{n+3}$ is the number of sequences of length $2n$ with $\text{Bf}[\sigma] = n+2$, $\text{Bfd}[\sigma] = n+1$
- ...
- $\binom{2n}{2n}$ is the number of sequences of length $2n$ with $\text{Bf}[\sigma] = 2n-1$, $\text{Bfd}[\sigma] = 2n-2$

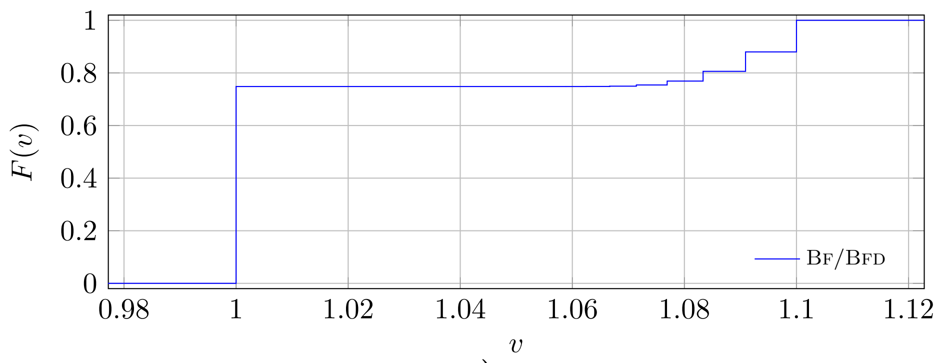
Apart from these item sequences, no other sequences change their objective due to application of Bfd instead of Bf. Thus, the performance ratio ranges in $[1, \frac{n+1}{n}]$ and the number of item sequences of length $2n$ which leave the number of bins unchanged in both algorithms is

$$2^{2n} - \binom{2n}{n+2} - \binom{2n}{n+3} - \dots - \binom{2n}{2n} = \binom{2n}{0} + \binom{2n}{1} + \dots + \binom{2n}{n+1} = 2^{2n-1} + \frac{1}{2}\binom{2n}{n} + \binom{2n}{n+1}.$$

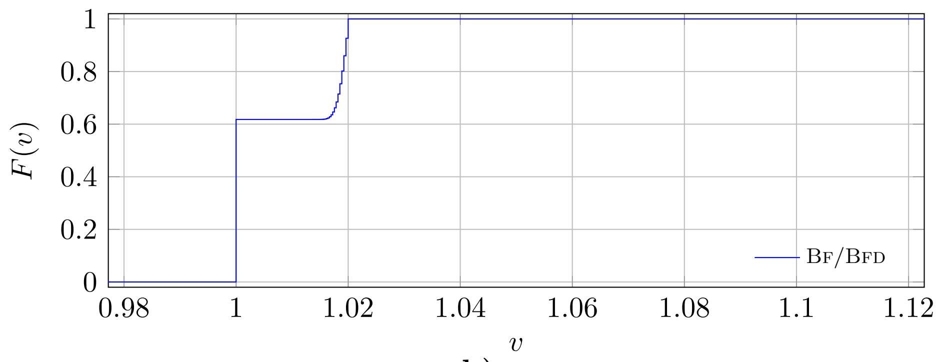
Exploiting this relation, we immediately get the given expression for the counting distribution function of the performance ratio as $F_{\frac{\text{Bf}}{\text{Bfd}}}(v)$. □

In Figure 4.7, exemplary plots of $F_{\frac{\text{Bf}}{\text{Bfd}}}(v)$ are given for input sequences of length $n = 10, 50, 100, 500$ confirming that the lookahead effect is also relatively small with respect to an instance-wise comparison of both algorithms to each other.

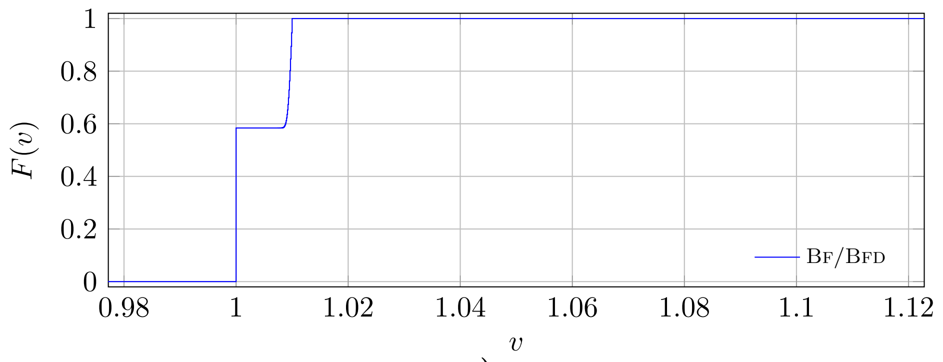
The only pair of items where algorithm Bfd changes the packing order in contrast to the revelation order is $\sigma^{sub} = (0.5 - \epsilon, 0.5 + \epsilon)$. Assume now that Bfd is prohibited to permute the item order during packing. Define algorithm Bfd′ which fictively operates as Bfd, i.e., it does not put the small item of σ^{sub} into an open bin with a small item but in a new bin because it reserves the space in the already open bin for the large item of σ^{sub}. Since Bfd′ emulates Bfd and leads to the same outcome, it is seen easily that the value of being allowed to permute the items is zero. The attainable benefit out of the additional lookahead item is entirely due to the information about the size of this item.



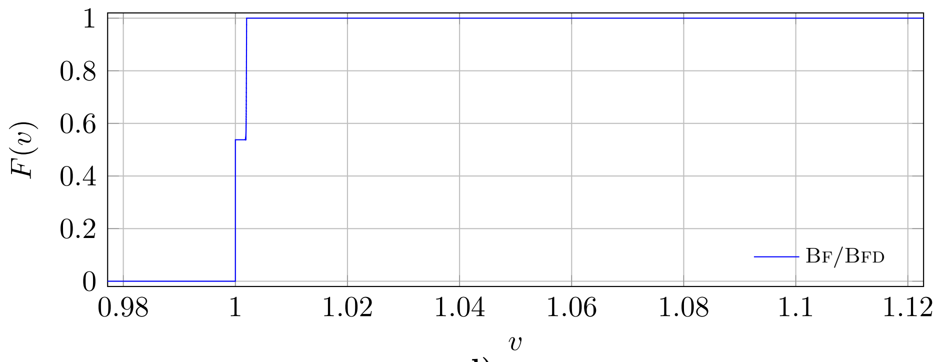
a)



b)



c)



d)

Figure 4.7: Counting distribution functions of performance ratio of costs in the bin packing problem. **a)** $2n = 20$. **b)** $2n = 100$. **c)** $2n = 200$. **d)** $2n = 1000$.

The derivation of expressions for the counting distribution functions of the objective value and of the performance ratio for the case of two item sizes showed that the improvement to be expected from an additional lookahead item is very small as a result of a number of (restrictive) conditions that have to be fulfilled by the structure of an item sequence (cf. Theorem 4.2). We also found that the lookahead effect vanishes in the sense of converging distribution functions for Bf and Bfd as the number of items tends to infinity; vice versa, the impact of lookahead is more likely to be observed on item sequences which are of rather short length. Moreover, it is irrelevant whether one is allowed to change the order of the items during their packing or not.

4.3 Online Traveling Salesman Problem with Lookahead

The traveling salesman problem (TSP) lies at the core of nearly every transportation or routing problem as it seeks to find a round trip (also called tour) for a given set of locations to be visited (also called requests) such that some cost function depending on the total travel distance is minimized ([122]). Hence, the TSP is a pure sequencing problem.

For input sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$, let a request σ_i with $i \in \mathbb{N}$ correspond to a point x_i in a space $\mathcal{M}$ with metric $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$, then the TSP consists of visiting the points of all requests with a server in a tour of minimum length starting and ending in some distinguished origin $o \in \mathcal{M}$.

A permutation $\pi(\sigma) = (\pi_1(\sigma), \pi_2(\sigma), \ldots, \pi_n(\sigma))$ of the set $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ of requests represents a tour $(o, x_{\pi_1(\sigma)}, x_{\pi_2(\sigma)}, \ldots, x_{\pi_n(\sigma)}, o)$, i.e., a feasible solution to an instance of the TSP. The value of

$$D(\pi(\sigma)) := d(o, x_{\pi_1(\sigma)}) + \sum_{i=1}^{n-1} d(x_{\pi_i(\sigma)}, x_{\pi_{i+1}(\sigma)}) + d(x_{\pi_n(\sigma)}, o)$$

is called the tour length of $\pi(\sigma)$.

Problem 4.7 (Offline Traveling Salesman).

Instance	Metric space $(\mathcal{M}, d)$, origin $o \in \mathcal{M}$, set of requests $\sigma = \{\sigma_i \mid i \in \{1, \ldots, n\}\}$ where σ_i is a request at $x_i \in \mathcal{M}$.
Task	Find a permutation $\pi(\sigma)$ of σ with minimum tour length.

Because we neglect temporal aspects, the online versions are considered in the sequential model. The online version without lookahead is trivial because the requests have to be visited in their order of appearance.

Problem 4.8 (Online Traveling Salesman).

INSTANCE	Metric space $(\mathcal{M}, d)$, origin $o \in \mathcal{M}$, sequence of requests $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ where σ_i is a request at $x_i \in \mathcal{M}$, instance revelation rule r for the online case.
TASK	Sequentially set $\pi_i(\sigma) := \sigma_i$ when σ_i is revealed.

In the online version with lookahead, there is limited knowledge about a number of requests to be visited.

Problem 4.9 (Online Traveling Salesman with Lookahead).

INSTANCE	Metric space $(\mathcal{M}, d)$, origin $o \in \mathcal{M}$, sequence of requests $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ where σ_i is a request at $x_i \in \mathcal{M}$, instance revelation rule r' for the lookahead case.
TASK	Sequentially choose $\pi_i(\sigma)$ as one of the unvisited requests which are known as part of the lookahead information such that the tour length of $\pi(\sigma)$ is minimized.

As a result of the irrelevance of the factor time, the instance revelation rule in the online case is

$r :=$ At the beginning, σ_1 is known; a new request is revealed when the known one is visited

and in the lookahead case with $l \in \mathbb{N}$

$r' :=$ At the beginning, $\sigma_1, \ldots, \sigma_l$ are known; a new request is revealed when a known one is visited.

The rule set in the online case is trivial and does not give any degrees of freedom to the decision maker, i.e.,

$$P := \{\text{Visit the known request}\};$$

permutations are allowed in the lookahead case, i.e.,

$$P' := \{\text{Visit one of the known requests}\}.$$

According to the modeling framework from Chapter 3, the lookahead setting is

$$req \mid sngl/rnd \mid im \mid discr \qquad \text{or equivalently} \qquad req \mid sngl/rnd \mid im \mid cyc.$$

The input information of an input element is given by x_i. In the online case, we have $\tau_i = i$ and $T_i = i + \epsilon$; in the lookahead case, we have $\tau_i = \max\{1, i - l\}$ and $\underline{T}'_i = \max\{1, i - l\} + \epsilon$, $\overline{T}'_i = \infty$ with sufficiently small $\epsilon > 0$. Hence, the server is assumed to travel at infinite speed.

We now compare the pure online setting with the setting enhanced by lookahead: Online algorithm FirstComeFirstServed has no choices, i.e., the server has to visit the requests in their order of appearance in a first-come first-served manner (cf. Algorithm 4.7). Online algorithm NearestNeighbor$_l$ with request lookahead l always moves the server to the closest known point in terms of distance from its current location (cf. Algorithm 4.8).

Algorithm 4.7 Online algorithm FirstComeFirstServed for the TSP

```
Input: Metric space $(\mathcal{M}, d)$, origin $o \in \mathcal{M}$, first element $\sigma_1$ of input sequence $\sigma$ with request
    at $x_1$
1: $D := d(o, x_{\sigma_1}), i := 1, \pi_1(\sigma) := \sigma_1$
2: while $i < n$ do
3:    additional input: $(i+1)$st element $\sigma_{i+1}$ of input sequence $\sigma$ with request at $x_{i+1}$
4:    $D := D + d(x_{\sigma_i}, x_{\sigma_{i+1}}), \pi_{i+1}(\sigma) := \sigma_{i+1}$
5:    $i := i + 1$
6: end while
7: $D := D + d(x_{\sigma_n}, o)$
Output: (Identity) Permutation $\pi(\sigma)$, total distance $D$ of $\pi(\sigma)$
```

Algorithm 4.8 Online algorithm NearestNeighbor$_l$ with lookahead for the TSP

```
Input: Metric space $(\mathcal{M}, d)$, origin $o \in \mathcal{M}$, first $l$ elements $\sigma_1, \sigma_2, \ldots, \sigma_l$ of input sequence
    $\sigma$ with requests at $x_1, x_2, \ldots, x_l$
1: $x := o, D := 0, i := 1, J := \{1, 2, \ldots, l\}$
2: while $i \leq n$ do
3:    $k := \arg\min_{j \in J} d(x, x_{\sigma_j}),\ J := J \backslash \{k\},\ x' := x_{\sigma_k}$
4:    $D := D + d(x, x'), \pi_i(\sigma) := \sigma_k$
5:    if $i \leq n - l$ then
6:       additional input: $(i+l)$th element $\sigma_{i+l}$ of input sequence $\sigma$ with request at $x_{i+l}$
7:       $J := J \cup \{i + l\}$
8:    end if
9:    $i := i + 1, x := x'$
10: end while
11: $D := D + d(x, o)$
Output: Permutation $\pi(\sigma)$, total distance $D$ of $\pi(\sigma)$
```

Under additional lookahead of one request ($l = 2$), we derive exact expressions for the counting distribution functions of the objective value and the performance ratio for the case of a metric space consisting of two points only, i.e., $\mathcal{M} = \{0, 1\}$ with $d(0, 1) = d(1, 0) = 1$, $d(0, 0) = d(1, 1) = 0$ and $o = 0$.

From now on, we refer to FirstComeFirstServed as Fcfs and to NearestNeighbor$_2$ as Nn. We use the following additional terminology:

- A pass is the transition between two successive requests.
- A change pass is a pass between two requests (σ_i, σ_{i+1}) with $x_i = 1, x_{i+1} = 0$ or $x_i = 0, x_{i+1} = 1$.
- A remain pass is a pass between two requests (σ_i, σ_{i+1}) with $x_i = x_{i+1}$.
- A request subsequence $(\sigma_i, \sigma_{i+1}, \sigma_{i+2})$ is a pass pair.
- A free point at a given time is a point which is not occupied by the server at that time.

The main advantage of Nn over Fcfs is its ability to crack pass pairs $(0, 1, 0)$ or $(1, 0, 1)$ in order to build pass pairs $(0, 0, 1)$ or $(1, 1, 0)$. Note that because of the return to o, only even overall tour lengths are possible.

Theorem 4.16.

a) The number of request sequences σ of length n where Fcfs$[\sigma] = m$ *and m is even is given by*

$$\textsc{Fcfs}(n, m) = \binom{n+1}{m}.$$

b) The number of request sequences σ of length n where Nn$[\sigma] = m$ *and m is even is given by*

$$\textsc{Nn}(n, m) = \binom{n+1}{2m-2} + \binom{n+1}{2m}.$$

Proof. A sequence of n points starting and ending in o has $n + 2$ requests including the two dummy requests at o and encounters $n + 1$ passes to either the current or the free point.

a) For Fcfs in order to result in objective value m, a request sequence has to exhibit exactly m change passes out of the $n + 1$ passes.

b) For $m = 0$, the formula obviously holds. For $m > 0$, first observe that each resulting sequence after being processed by Nn exhibits a last change from 0 to 1 and a last change from 1 to 0 in the visiting order of requests which together contribute a total of 2 to the objective value. We conclude that a contribution of $m - 2$ is due to all previous passes that do not involve the 1 of the last change from 0 to 1. There are two cases how this 1 could be obtained: Either it was at the same position originally and remained there also under processing of Nn (case 1), or it had been shifted to that position as a result of the processing of Nn (case 2).

Denote by σ the original request sequence and by σ' the visiting order under Nn. For $n = 9$ and $m = 4$, we give an example of case 1 by

- $\sigma = (\ 0\ ,\ 1\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 1\ ,\ 0\ )$,
- $\sigma' = (\ 0\ ,\ 0\ ,\ 1\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 1\ ,\ 0\ )$

and of case 2 by

- $\sigma = (\ 0\ ,\ 1\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 1\ ,\ 1\ ,\ 0\ )$,
- $\sigma' = (\ 0\ ,\ 0\ ,\ 1\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 1\ ,\ 1\ ,\ 0\ )$.

For each of the $m-2$ changes incurred by NN, of course, there must also have been a corresponding change pass in the original sequence (marked red). Additionally, through the processing of NN, this change pass can only be responsible for a change if in the processing order of NN the (potentially shifted) destination of the original change pass is succeeded by another pass which has to be a remain-pass (marked green).

In the first case, $m-2$ change passes along with their affirmative remain-passes in the transformed sequence and two additional change passes (marked blue) incur changes, i.e., $2(m-2)+2 = 2m-2$ out of the $n+1$ passes have to be chosen. In the second case, because of the last change from 0 to 1 in the transformed sequence (which resulted from a shifted change pass, also marked red) and its affirmative remain pass, $m-2+1 = m-1$ change passes along with their affirmative remain-passes in the transformed sequence and two additional passes (marked blue), whereof the first one is a pass from 1 to 0 ensuring that the 1 will be shifted to the right (cf. definition of case 2), incur changes, i.e., $2(m-1)+2 = 2m$ out of the $n+1$ passes have to be chosen.

□

From the previous result, we immediately obtain expressions for the counting distribution functions of the objective value $F_{\text{FCFS}}(v)$ and $F_{\text{NN}}(v)$, respectively. Note that FCFS and NN have possible tour lengths in $\{0, 2, 4, \ldots, 2\lceil \frac{n}{2} \rceil\}$ and $\{0, 2, 4, \ldots, 2\lceil \frac{n}{4} \}\rceil$, respectively, for $n+2$ requests including the first and last request to o (cf. proof of Theorem 4.18).

Corollary 4.17. *The counting distribution functions of the objective value $F_{\text{FCFS}}(v)$ and $F_{\text{NN}}(v)$ of FCFS and NN for request sequences of length $n+2$ (including the first and last request to o) are given by*

$$F_{\text{FCFS}}(v) = \begin{cases} 0 & \text{if } v < 0, \\ \sum_{i=0}^{n'} \binom{n+1}{2i} \cdot 2^{-n} & \text{if } 2n' \leq v < 2n'+2 \text{ for } n' = 0, 1, \ldots, \lceil \frac{n}{2} \rceil - 1, \\ 1 & \text{if } v \geq 2\lceil \frac{n}{2} \rceil, \end{cases}$$

$$F_{\mathrm{NN}}(v) = \begin{cases} 0 & \textit{if } v < 0, \\ \displaystyle\sum_{i=0}^{n'} \left(\binom{n+1}{4i} + \binom{n+1}{4i-2} \right) \cdot 2^{-n} & \textit{if } 2n' \leq v < 2n' + 2 \textit{ for } n' = 0, 1, \ldots, \lceil \frac{n}{4} \rceil - 1, \\ 1 & \textit{if } v \geq 2\lceil \frac{n}{4} \rceil. \end{cases}$$

Figure 4.8 exemplarily plots $F_{\mathrm{FCFS}}(v)$ and $F_{\mathrm{NN}}(v)$ for all item sequences of length $n = 5, 10, 50, 100, 500, 1000$. We observe a significant lookahead effect as a result of permuting request triples with two sucessive change passes such that these turn into one change pass and one remain pass.

We next derive an expression for the counting distribution function of the performance ratio $F_{\frac{\mathrm{NN}}{\mathrm{FCFS}}}(v)$ of NN relative to FCFS.

Theorem 4.18. *Let $m_{\mathrm{NN}}(\sigma)$ and $m_{\mathrm{FCFS}}(\sigma)$ denote the objective values of algorithms* NN *and* FCFS *on request sequence σ, respectively, and let $n_{\frac{\mathrm{NN}}{\mathrm{FCFS}}}(n, a, b)$ be the number of request sequences σ of length $n + 2$ (including the first and last request to o) with $m_{\mathrm{NN}}(\sigma) = a$ and $m_{\mathrm{FCFS}}(\sigma) = b$ for $a = 0, 2, 4, \ldots, 2\lceil \frac{n}{4} \rceil$ and $b = 0, 2, 4, \ldots, 2\lceil \frac{n}{2} \rceil$, then it holds that*

$$n_{\frac{\mathrm{NN}}{\mathrm{FCFS}}}(n, a, b) = \begin{cases} 1 & \textit{if } b = a = 0 \\ \binom{n+3-a}{a} & \textit{if } b = a \neq 0 \\ \binom{n+1-a}{2a} & \textit{if } b = 3a \\ \binom{n+2-a}{2a-1} + (a-1)\binom{n+1-a}{2a-1} & \textit{if } b = 3a - 2 \\ \binom{a-2}{\frac{b-a}{2}}\binom{n+3-a}{\frac{a+b}{2}} + \binom{a-2}{\frac{b-a}{2}-1}\binom{n+2-a}{\frac{a+b}{2}} + \binom{a-1}{\frac{b-a}{2}-1}\binom{n+1-a}{\frac{a+b}{2}} & \textit{if } b = a+2, a+4, \ldots, \\ & 3a - 4 \\ 0 & \textit{else.} \end{cases}$$

Proof. Notice that NN can never be worse than FCFS because whenever NN changes the order, a saving occurs without future drawbacks. NN needs at least a third of the distance of FCFS which can be seen by $(0, 1, 0, 1, 0, 1, 0)$ which requires two units from NN and six units from FCFS. There are no sequences with a larger percentage of savings because at the end of this sequence only two requests on 0 are seen by NN, i.e., there is no value of lookahead in this moment, and modifying the above sequence by additional requests cannot improve the advantage of NN over FCFS any further. For a request sequence with n points (apart from the dummy requests at the beginning and end), b can attain values up to $2\lceil \frac{n}{2} \rceil$ which is seen by the worst-case sequences for FCFS: Sequences of odd length consist only of change passes; sequences of even length consist only of change passes except for one remain pass. For a request sequence with n points (apart from the dummy requests at the beginning and

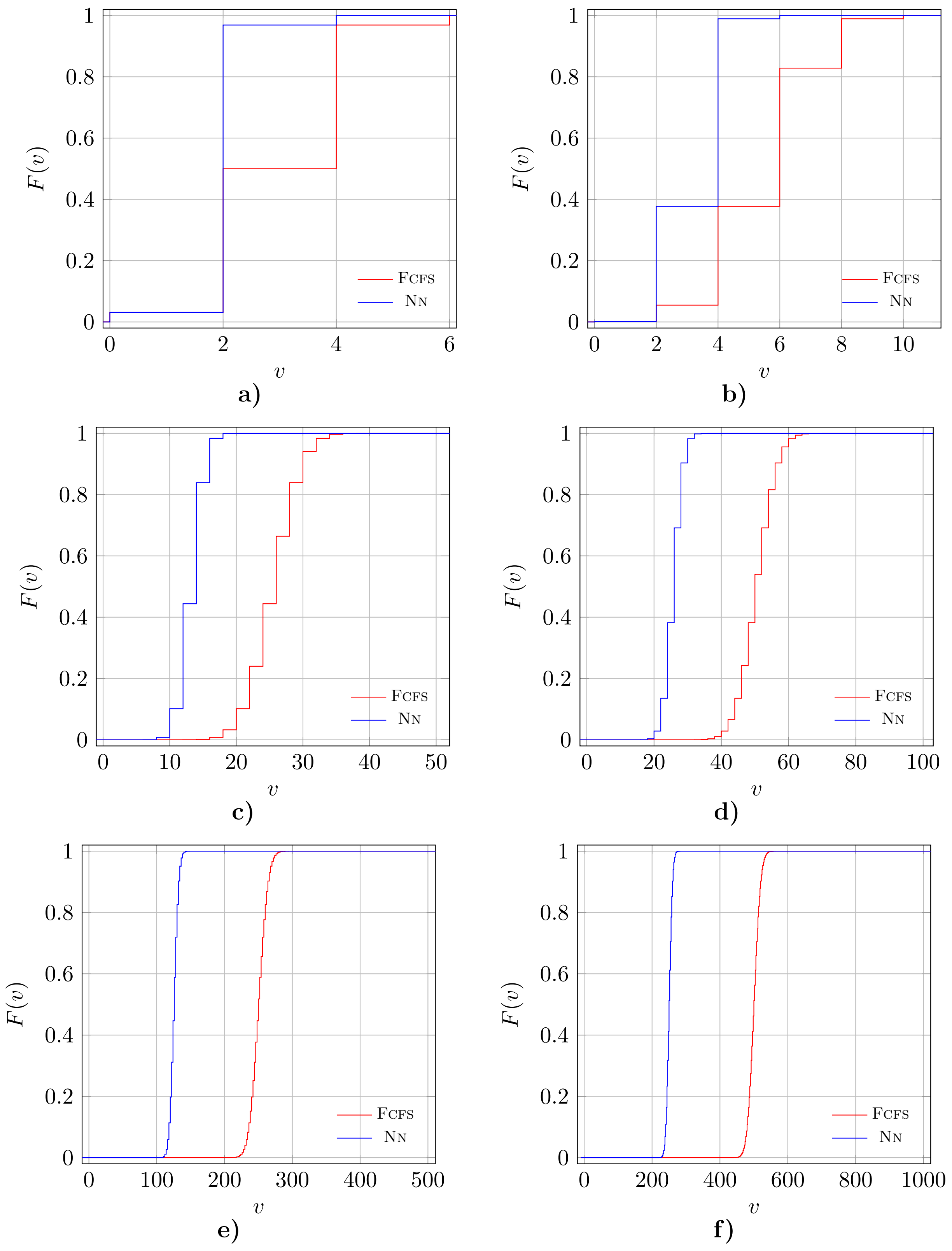


Figure 4.8: Counting distribution functions of costs in the TSP. **a)** $n = 5$. **b)** $n = 10$. **c)** $n = 50$. **d)** $n = 100$. **e)** $n = 500$. **f)** $n = 1000$.

end), a can attain values up to $2\lceil \frac{n}{4} \rceil$ which is seen by the worst-case sequences for NN of the forms $(0,(1,1,0,0)^c,1,0)$, $(0,(1,1,0,0)^c,1,1,0)$ or $(0,(1,1,0,0)^c,1,1,0,0)$ for $c \in \mathbb{N}$ where $(1,1,0,0)^c$ means that $(1,1,0,0)$ is requested c times in a row. Hence, every four requests contribute at most two units to the objective value (except for the last, the two last or the three last requests). The only sequence with $a = b = 0$ has $\sigma_i = 0$ for $i = 1, \dots, n$.

For $a = b \neq 0$, the visiting order of the points in σ is identical for FCFS and NN because NN has a lookahead of one additional request. In particular, we know that the pass immediately following each of the first $a-2$ change passes has to be a remain pass since otherwise NN would have reorganized the order. (The last two change passes do not have to exhibit this structure because these changes cannot be extinguished by NN due to the forced return to the origin.) Thus, since for $a-2$ passes we know the type of the immediate successor pass, we only have to choose the a change passes out of $n+1-(a-2) = n+3-a$ passes.

For $b = 3a$, it follows from the definition of the request lookahead mechanism with one additional request that the request sequence has to consist only of disjunct subsequences of the form $(0^{c_0}, 1, 0^{c_1}, 1^{c_2}, 0, 1^{c_3}, 0^{c_4})$ with $c_i \in \mathbb{N}$ for $i = 0,1,2,3,4$ where x^c stands for a request on $x \in \{0,1\}$ for c times in a row. The requests of each such subsequence will be visited in the order $(0^{c_0}, 0^{c_1}, 1, 1^{c_2}, 1^{c_3}, 0, 0^{c_4})$ by NN producing two moves, whereas FCFS needs six. In particular, we know that in the original sequence after the first pass from 0 to 1 a pass to 0 immediately follows and that the pass from 0 to 1^{c_3} is immediately preceded by a 1. Hence, for each pass in σ that leads to one of the a moves of NN, there is also another associated pass known in σ. Thus, we choose out of $n+1-a$ passes rather than out of $n+1$ passes. Within each such subsequence, we have to choose the ends of 0^{c_0}, 0^{c_1}, 1^{c_2}, 1^{c_3} by selecting c_0, c_1, c_2, c_3. Since there are $\frac{a}{2}$ such subsequences for objective value a, in total we have to choose $4 \cdot \frac{a}{2} = 2a$ out of $n+1-a$ passes.

In the sequel, we call a subsequence $(0,1,0)$ or $(1,0,1)$ a change-change pass pair (c/c-pair) and a subsequence $(0,1,1)$ or $(1,0,0)$ a change-remain pass pair (c/r-pair).

For a difference of $b-a$ in the outcome, one has to encounter exactly $\frac{b-a}{2}$ non-overlapping c/c-pairs before the final return to o. Hence, for $b = 3a-2$, we need $\frac{3a-2-a}{2} = a-1$ non-overlapping c/c-pairs. Moreover, one additional (isolated) change pass has to occur within σ because $a-1$ is odd and the server has to return to o. For each of the $a-1$ c/c-pairs $(0,1,0)$ or $(1,0,1)$, we also have to specify the position until which the sequence continues with 0 and 1, respectively. Hence, we have to choose $(a-1)+1+(a-1) = 2a-1$ passes. Now there are two cases which lead to $\binom{n+2-a}{2a-1} + (a-1)\binom{n+1-a}{2a-1}$ sequences exhibiting $b = 3a-2$:

- The isolated change pass occurs after all non-overlapping c/c-pairs. Then we have to choose the non-overlapping c/c-pairs and the isolated change pass out of $n+1-(a-1) = n+2-a$ passes.

- The isolated change pass occurs prior to all or within the sequence of non-overlapping c/c-pairs. Then we have to choose the non-overlapping c/c-pairs and the isolated change pass out of $n+1-(a-1)-1 = n+1-a$ passes because we know that the isolated change pass is succeeded by a remain pass since otherwise it would not be an isolated change pass. Clearly, there are $a-1$ positions to locate the change pass prior to all or within the sequence of the $a-1$ non-overlapping c/c-pairs.

For $b = a+2i$ with $i = 1, 2, \dots, a-2$, exactly i non-overlapping c/c-pairs occur before the final return to o because each such pair gives a saving of two. We conclude that at least $(a-2)-i$ pairs have to be c/r-pairs and two additional change passes are left over (which may come in arbitrary form, i.e., as a c/c-pair or as two c/r-pairs). We distinguish whether these two left-over change passes appear at the end of σ (case 1) or not (cases 2 and 3):

- First, assume the two left-over change passes to appear at the end of σ, i.e., when all of the $a-2$ c/c- and c/r-pairs have occurred. Hence, we choose out of $n+1-(a-2) = n+3-a$ rather than out of $n+1$ passes. Recall that for a difference of $b-a$ one has to encounter exactly $\frac{b-a}{2}$ non-overlapping c/c-pairs before the final return to o. In total, in order to obtain objective value pair (a,b) one has to choose $\frac{b-a}{2}+a = \frac{a+b}{2}$ change passes. Since the two left-over change passes appear at the end of σ, we have that the $\frac{b-a}{2}$ non-overlapping c/c-pairs appear somewhere among the $a-2$ known c/c- and c/r-pairs, i.e., we have to choose $\frac{b-a}{2}$ out of $a-2$.

- Second, assume one additional c/r-pair (containing a left-over change pass) is inserted before the last of the *known* $a-2$ c/c- and c/r-pairs, i.e., within the first $a-1$ c/c- and c/r-pairs. Then we have to choose $\frac{a+b}{2}$ change passes out of $n+1-(a-2)-1 = n+2-a$ passes. Due to the additional c/r-pair, there are only $\frac{b-a}{2}-1$ c/c-pairs among the *first* $a-2$ c/c- and c/r-pairs, i.e., we have to choose $\frac{b-a}{2}-1$ out of $a-2$.

- Third, assume two additional c/r-pairs (each containing a left-over change pass) are inserted before the last of the *known* $a-2$ c/c- and c/r-pairs, i.e., within the first a c/c- and c/r-pairs. Then we have to choose $\frac{a+b}{2}$ change passes out of $n+1-(a-2)-2 = n+1-a$ passes. Due to the additional c/r-pairs, there are only $\frac{b-a}{2}-2$ or $\frac{b-a}{2}-1$ c/c-pairs among the *first* $a-2$ c/c- and c/r-pairs, i.e., we have to choose $\frac{b-a}{2}-2$ out of $a-2$ or $\frac{b-a}{2}-1$ out of $a-2$, i.e., $\binom{a-2}{\frac{b-a}{2}-2} + \binom{a-2}{\frac{b-a}{2}-1} = \binom{a-1}{\frac{b-a}{2}-1}$.

□

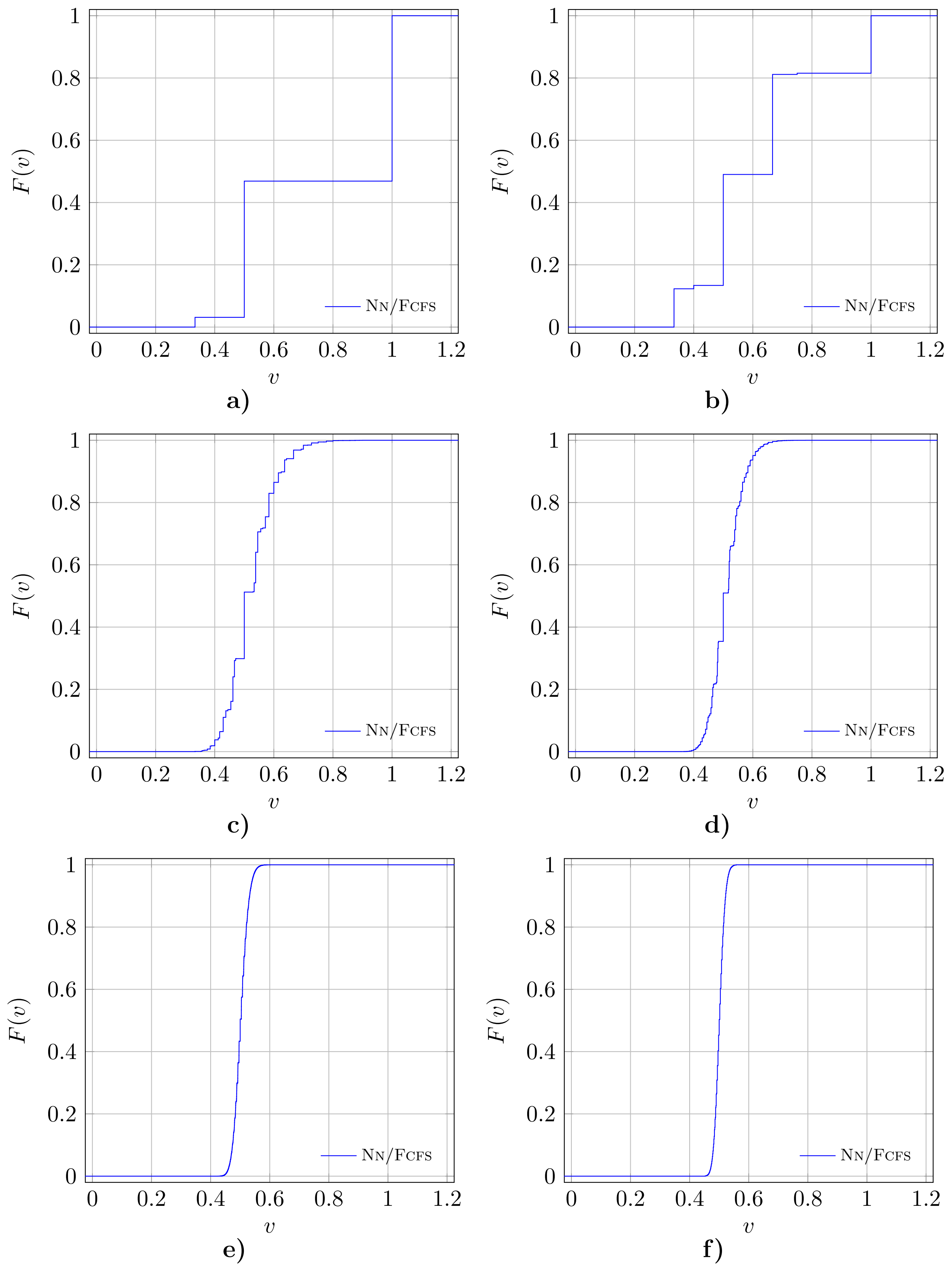


Figure 4.9: Counting distributions functions of performance ratio of costs in the TSP. **a)** $n = 5$. **b)** $n = 10$. **c)** $n = 50$. **d)** $n = 100$. **e)** $n = 500$. **f)** $n = 1000$.

Corollary 4.19. *The counting distribution function of the performance ratio* $F_{\frac{\text{NN}}{\text{FCFS}}}(v)$ *of* FCFS *and* NN *for request sequences of length* $n+2$ *(including the first and last request to* 0*) is given by*

$$F_{\frac{\text{NN}}{\text{FCFS}}}(v) = 2^{-n} \cdot \sum_{j=0,2,\dots,2\lceil\frac{n}{2}\rceil} \sum_{\substack{i=0,2,\dots,\min\{j,2\lceil\frac{n}{4}\rceil\},\\ \frac{i}{j}\le v}} n_{\frac{\text{NN}}{\text{FCFS}}}(n,i,j)$$

with $n_{\frac{\text{NN}}{\text{FCFS}}}(n,i,j)$ *from Theorem 4.18.*

In Figure 4.9 on the previous page, exemplary plots of $F_{\frac{\text{NN}}{\text{FCFS}}}(v)$ are given for input sequences of length $n = 5, 10, 50, 100, 500, 1000$ confirming that the lookahead effect is also enormous with respect to an instance-wise comparison of both algorithms to each other.

For the online TSP with two locations, we showed that the provision of only one additional location already leads to huge improvements in the total distance to be traveled by the server as a result of the immediate savings that can be accrued without future obligations whenever the second of the two known locations coincides with the current server position and the first one does not. The benefit is exclusively due to the allowance to visit the known requests in any order. Hence, the lookahead effect is entirely made up of its processual component.

4.4 Concluding Discussion

In this chapter, we carried out an exact theoretical analysis of algorithm performance in online optimization with lookahead for three academic problem classes. In each of the settings, we characterized the improvement attainable by additional information in terms of the resulting counting distribution functions of the objective value and of the performance ratio. The analysis reproduced an exact image of algorithm behavior over all input sequences (including also competitive analysis results).

In all three problems, we found that lookahead has a positive influence on solution quality, albeit the magnitude of the impact differs strongly from one problem to another. Explanations and reasons for these differences in the value of information could mainly be identified in the proofs of the results: The ski rental problem – representing the class of rent-or-buy problems – exhibits a remarkable lookahead effect as a result of improved one-shot decision making. The bin packing problem – representing the class of packing problems – only admits small improvements because a number of restrictive conditions on combined item sizes would need to be fulfilled such there is potential for saving a bin at all. In contrast to that, in the

TSP – representing the class of routing problems – the largest improvement is observed because permuting the visiting order directly pays off in form of savings in the travel distance without future drawbacks. Note that permutations were allowed in bin packing, too; yet, the considered algorithm could not capitalize from it.

Table 4.5 sums up at a coarse-grained level the qualitative findings of the exact analysis in the three problem settings from this chapter: While in the ski rental and bin packing problem the lookahead effect (column $\Delta f^{r,r',P,P'}_{\mathrm{ALG},\mathrm{ALG}'}$) was entirely made up of its informational component (column $\Delta f^{r,r'}_{\mathrm{ALG}}$), the benefit attained in the TSP was facilitated by the rule set substitution (column $\Delta f^{P,P'}_{\mathrm{ALG},\mathrm{ALG}'}$) in form of the allowance to visit the locations in an arbitrary order. Observe from the last column (*Deterioration*) that in no problem it was possible to incur a deterioration upon obtaining additional lookahead information.

The derivation of the expressions for the counting distribution functions of the performance ratio and of the objective value showed that an exact analysis of combinatorial online optimization problems is strongly interconnected to the combinatorial structure which is injected to the problem setting by the processing of an algorithm and the choice of problem parameters. Clearly, these structures are recognizable by human thinking capabilities only when algorithms apply simple decision rules. However, even under this assumption the analysis became quite involved. We recognize that for larger lookahead and general settings it is virtually impossible to track the effects of lookahead in the processing of an algorithm over all input sequences. Hence, we are led to conducting experimental analysis in more realistic problem settings which will be our agenda for the next two chapters.

Problem	Type/Size	Attribute	Rule	$\Delta f^{r,r',P,P'}_{\text{Alg},\text{Alg}'}$	$\Delta f^{r,r'}_{\text{Alg}}$	$\Delta f^{P,P'}_{\text{Alg},\text{Alg}'}$	Deterioration
Ski rental	request lookahead/ arbitrary	–	–	large	large	zero	no
Bin packing	request lookahead/ 2	2 item sizes	permutations	small	small	zero	no
			no permutations	small	small	zero	no
TSP	request lookahead/ 2	2 locations	permutations	large	zero	large	no
			no permutations	zero	zero	zero	no

Table 4.5: Qualitative summary of the exact results from Chapter 4.

5 Experimental Analysis of Algorithms for Online Optimization with Lookahead

Complementing the theoretical results derived in the previous chapter, we now conduct a series of numerical experiments on five standard online optimization problems in order to examine the impact of lookahead on solution quality for algorithms specifically tailored to take advantage of lookahead. We observe significant differences in the magnitude of the lookahead effect depending on the respective problem settings and give explanations for varying algorithm behavior under additional lookahead. The study of this chapter enables us to preestimate the magnitude of the lookahead effect based on the problem characteristics provided. We consider two additional problems alongside those of the previous chapter:

- Online Paging with Lookahead
- Online Scheduling with Lookahead

For each problem, three different types of analysis are applied in order to ensure a comprehensive view on algorithm behavior under different information regimes:

1. *Average results* portray the overall behavior of algorithms under different amounts of lookahead for a prescribed number of independent random input sequences.
2. *Distributional results* submit a more precise picture by including frequency information on observed objective values and performance ratios, respectively.
3. *Markov chain results* serve as an exact underpinning for the sampled results. Because of the curse of dimensionality, we restrict ourselves to small parameter values.

Computational experiments were performed on a personal computer with AMD Phenom II X6 1100T 3.31 GHz processor and 16 GB RAM under Microsoft Windows 7 (64-bit). All algorithms were implemented in C++ and arising instances of IP and MIP formulations were solved using IBM ILOG CPLEX 12.5 with a prescribed time limit of 120 seconds. A detailed summary of statistical key figures for each experiment is given in Appendix A.2.

5.1 Online Ski Rental with Lookahead

A specification of the online ski rental problem with lookahead and corresponding algorithms is given in Chapter 4.1. Because exact expressions for the counting distribution functions of the objective value and the performance ratio have been derived, we only present a brief (exact) numerical analysis for fixed parameters $n_{max} = 100$, $B = 50$, $r = 1$ to see the magnitude of the lookahead impact. Recall that an input sequence σ corresponds to a prescribed number of skiing days $n \in \{1, 2, \dots, n_{max}\}$. For better readability, we denote algorithm $\textsc{ConditionalBuy}_{k,l}$ as $\textsc{Alg}_{k,l}$; the optimal offline algorithm is called $\textsc{Opt}$. Observe that $\textsc{Alg}_{k,l}$ coincides with $\textsc{Opt}$ for $l \geq \lceil \frac{B}{r} \rceil = \lceil \frac{50}{1} \rceil = 50$ and with $\textsc{Buy}_k$ for $l = 1$. Algorithms are tested with variable purchase time $k \in \{1, 25, 50, 75, 100\}$ and variable lookahead sizes $l \in \{1, 5, 10, 20, \dots, 100\}$.

5.1.1 Average Results

From Figure 5.1, we see that lookahead leads to drastic reductions in the average costs required for skiing over all input sequences when each of the $n_{max} = 100$ days is equally probable to be the last skiing day. Comparing the online and offline situation, average cost reductions vary between 24.5 % and 39.8 % depending on the appointed day of purchase.

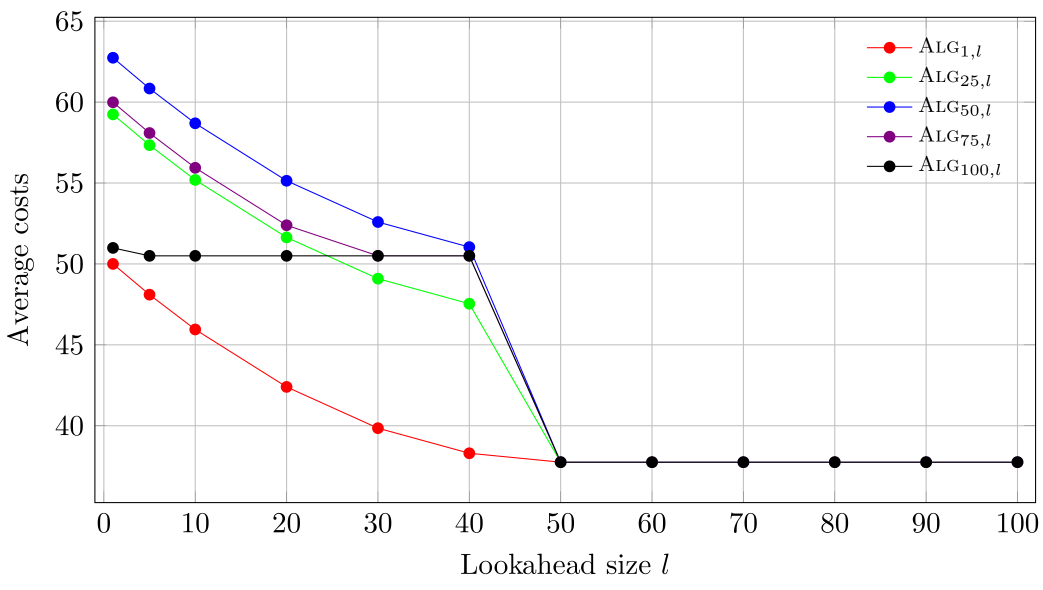


Figure 5.1: Average costs for different lookahead sizes and $n_{\max} = 100$ in the ski rental problem.

For fixed k, consider two lookahead levels l_1, l_2 with $l_1 < l_2$ and $l_1, l_2 < \lceil \frac{B}{r} \rceil = 50$. From the cost profile of $\textsc{Alg}_{k,l}$ it follows that $\textsc{Alg}_{k,l_1}[\sigma] \geq \textsc{Alg}_{k,l_2}[\sigma]$ for all input sequences σ.

Hence, additional lookahead proves exclusively beneficial and the average savings is

$$\sum_{\sigma} \frac{\textsc{Alg}_{k,l_1}[\sigma] - \textsc{Alg}_{k,l_2}[\sigma]}{n_{max}} = \sum_{t=k+l_1-1}^{k+l_2-2} \frac{r(k-1)+B-rt}{n_{max}}$$
$$= \sum_{t=k+l_1-1}^{k+l_2-2} \frac{k+49-t}{100} \geq 0.$$

For fixed lookahead l, the relation of the average costs incurred by $\textsc{Alg}_{k_1,l}$ and $\textsc{Alg}_{k_2,l}$ with $k_1 < k_2$ is determined by the value

$$\sum_{\sigma} \frac{\textsc{Alg}_{k_1,l}[\sigma] - \textsc{Alg}_{k_2,l}[\sigma]}{n_{max}} = \frac{\sum\limits_{t=k_1+l-1}^{k_2+l-2} (r(k_1-1)+B-rt) - \sum\limits_{t=k_2+l-1}^{n_{max}} r(k_2-k_1)}{n_{max}}$$
$$= \frac{\sum\limits_{t=k_1+l-1}^{k_2+l-2} (k_1+49-t) - (102-k_2-l)(k_2-k_1)}{100}$$

whose sign depends on the choices of k_1 and k_2. For the given parameters, the average of $\textsc{Alg}_{1,l}$, i.e., $k_1 = 1$, is not larger than that of $\textsc{Alg}_{k_2,l}$ for all $k_2 \in \{2, \ldots, 100\}$ and all l as indicated by the red line in Figure 5.1: The savings of $\textsc{Alg}_{1,l}$ for all $\sigma = \{1\}^n$ with $n \geq k_2 + l - 1$ cannot be exceeded by its excess expenditures for all $\sigma = \{1\}^n$ with $k_1 + l - 1 = l \leq n \leq k_2 + l - 2$. This can be seen by

$$\sum_{t=l}^{k_2+l-2} (50-t) - (102-k_2-l)(k_2-1) = 50(k_2-1) - \sum_{t=l}^{k_2+l-2} t - (102-k_2-l)(k_2-1)$$
$$= \frac{k_2}{2}(k_2-103)+51$$

which is smaller than 0 for $k_2 \leq 100$. We remark that the order relation between the average of $\textsc{Alg}_{k,l}[\sigma]$ for varying k and fixed l is strongly affected by the skier's expectation for the length of the skiing trip: Buying early should pay off whenever skis are expected to be in need for a long time; vice versa, buying late should pay off whenever skis are expected to be in need only shortly. Incorporating these considerations by weighting of input sequences, superiority of $\textsc{Alg}_{1,l}$ as displayed in Figure 5.1 is obtained by attributing equal weights $\frac{1}{n_{\max}}$ to all input sequences. Because of the parameter choices, this weighting already suffices to suggest earliest possible buying. Weightings which favor ski trips of short duration tend to shift lower costs to algorithms which buy late; vice versa, weightings which favor ski trips of long duration tend to shift lower costs to algorithms which buy early.

5.1.2 Distributional Results

Figure 5.2 displays the counting distribution functions of the objective value for $\textsc{Alg}_{k,l}$ and benchmarks them with the objective value distribution of $\textsc{Opt}$. Since for $l \geq 50$, the distributions of $\textsc{Alg}_{k,l}$ coincide with that of $\textsc{Opt}$, their plots are omitted.

Note that it is not possible for any of the algorithms to incur a deterioration in the objective value when more information is provided. In each of the diagrams, this can be seen from the perfect (stochastic) ordering of the counting distribution functions for prescribed purchase time k. Since $\textsc{Alg}_{1,l}$ is the only family of algorithms which considers buying at the outset of the dynamic problem, it is the only algorithm family which approximates $\textsc{Opt}$ for increasing l. For all other $\textsc{Alg}_{k,l}$ with $k > 1$, a gap remains for increasing l with $l < 50$ due to non-optimal buying decisions before day k. Clearly, for $l \geq 50$ the plot of any algorithm $\textsc{Alg}_{k,l}$ would snap to that of $\textsc{Opt}$. For fixed lookahead l, the range of occurring objective values broadens to larger values for increased k, thereby confirming the previously described superiority of $\textsc{Alg}_{1,l}$ in the given parameter constellation.

Although the average objective values and the objective value distributions suggest to buy at the very beginning of the planning horizon, the counting distribution functions of the performance ratio of each algorithm relative to $\textsc{Opt}$ in Figure 5.3 show that there is a high risk of incurring large multiples of the optimal objective value for these algorithms. For instance, on the input sequence σ corresponding to one skiing day, $\textsc{Alg}_{1,1}[\sigma]$ charges costs of 50, whereas $\textsc{Opt}$ only incurs a unit cost. Hence, the performance ratio is 50, emphasizing the potential risk of buying too early. This result is also confirmed by the comparatively large coefficients of variation for the performance ratios when k is small and the reduction in the confidence interval width for increasing k as shown in Table A.2 of Appendix A.2.1. Similar conclusions as found previously are discovered regarding the benefit of lookahead within an algorithm family $\textsc{Alg}_{k,l}$ with fixed k and variable l. Additional information leads to a considerable risk mitigation in form of a diminished competitive ratio that is attained by $\textsc{Alg}_{k,l}$ for increasing values of l. Note that in general no deterministic algorithm can be better than $(2 - \frac{1}{B} = 1.98)$-competitive ([107]).

In case of comparing to the pure online versions $\textsc{Alg}_{k,1}$ of the algorithms rather than to $\textsc{Opt}$, the variability of performance ratios for small k becomes evident in the immense width of their range for $k = 1$ as compared to larger values of k (see Figure 5.4 and Table A.3 of Appendix A.2.1). From this point of view, we also find that additional lookahead exclusively leads to lower costs as reflected by all performance ratios being smaller than 1 and confidence intervals shifting to regions with smaller values for increasing l.

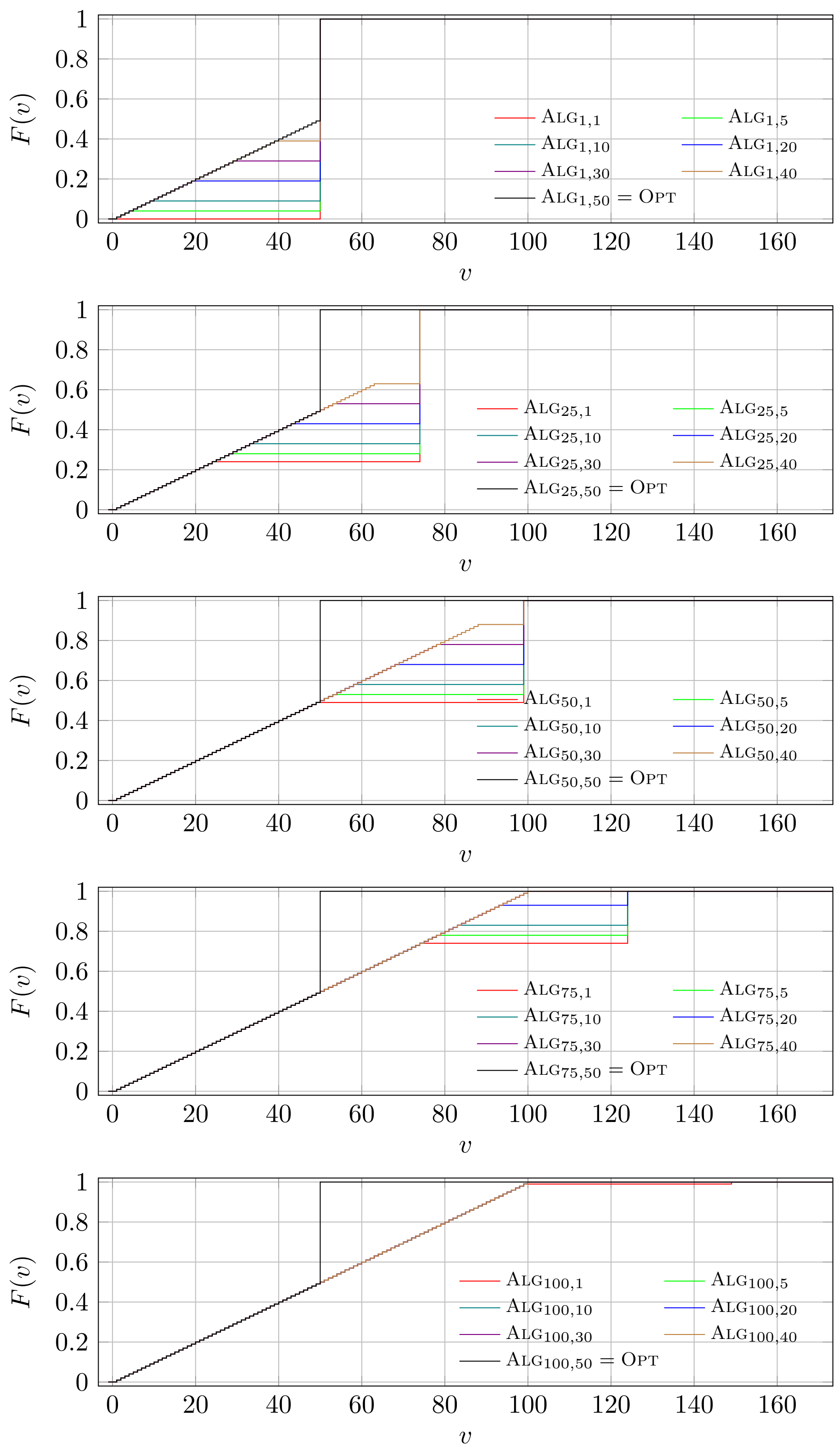


Figure 5.2: Counting distribution functions of costs for $n_{\max} = 100$ in the ski rental problem.

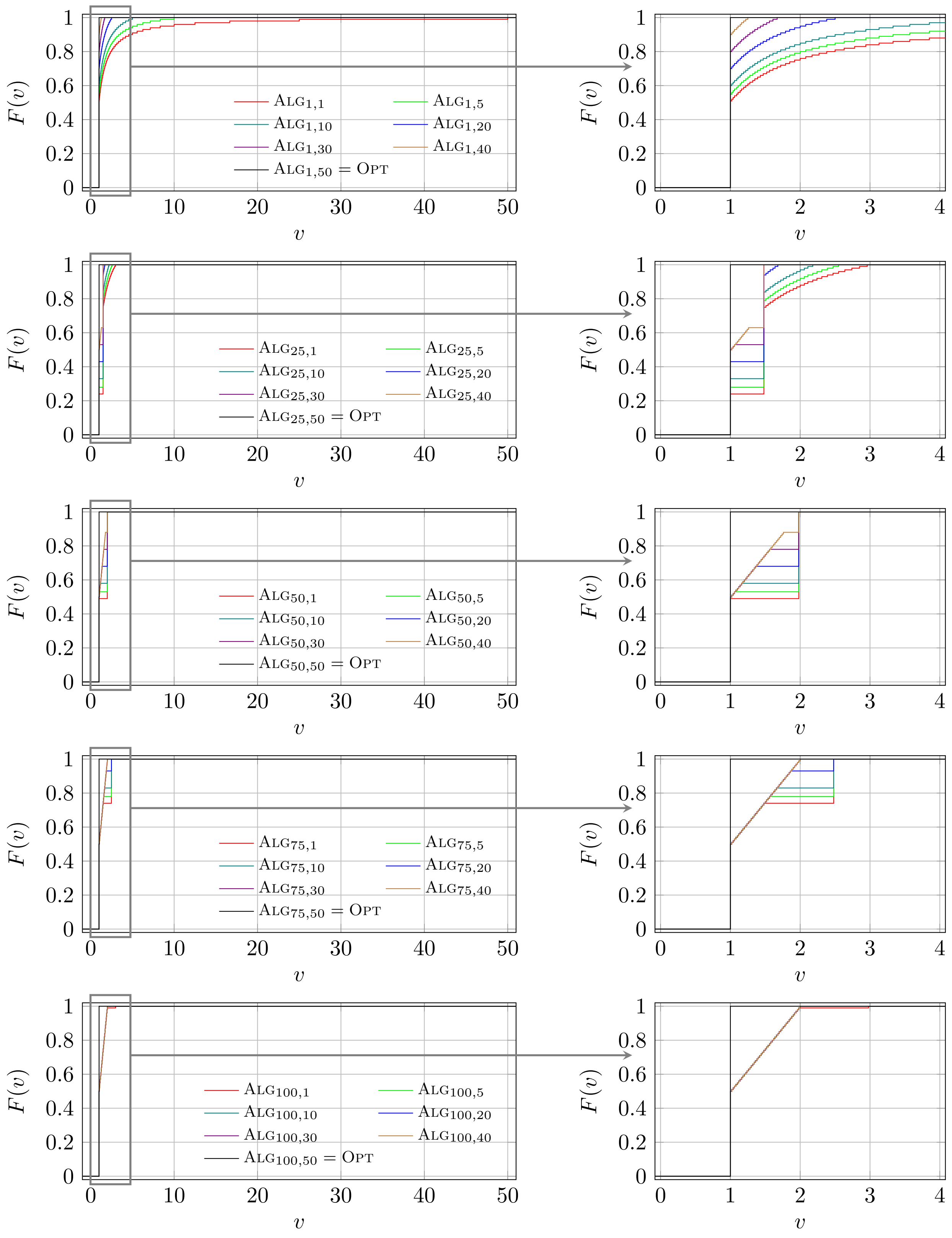


Figure 5.3: Counting distribution functions of performance ratio of costs relative to OPT for $n_{\max} = 100$ in the ski rental problem.

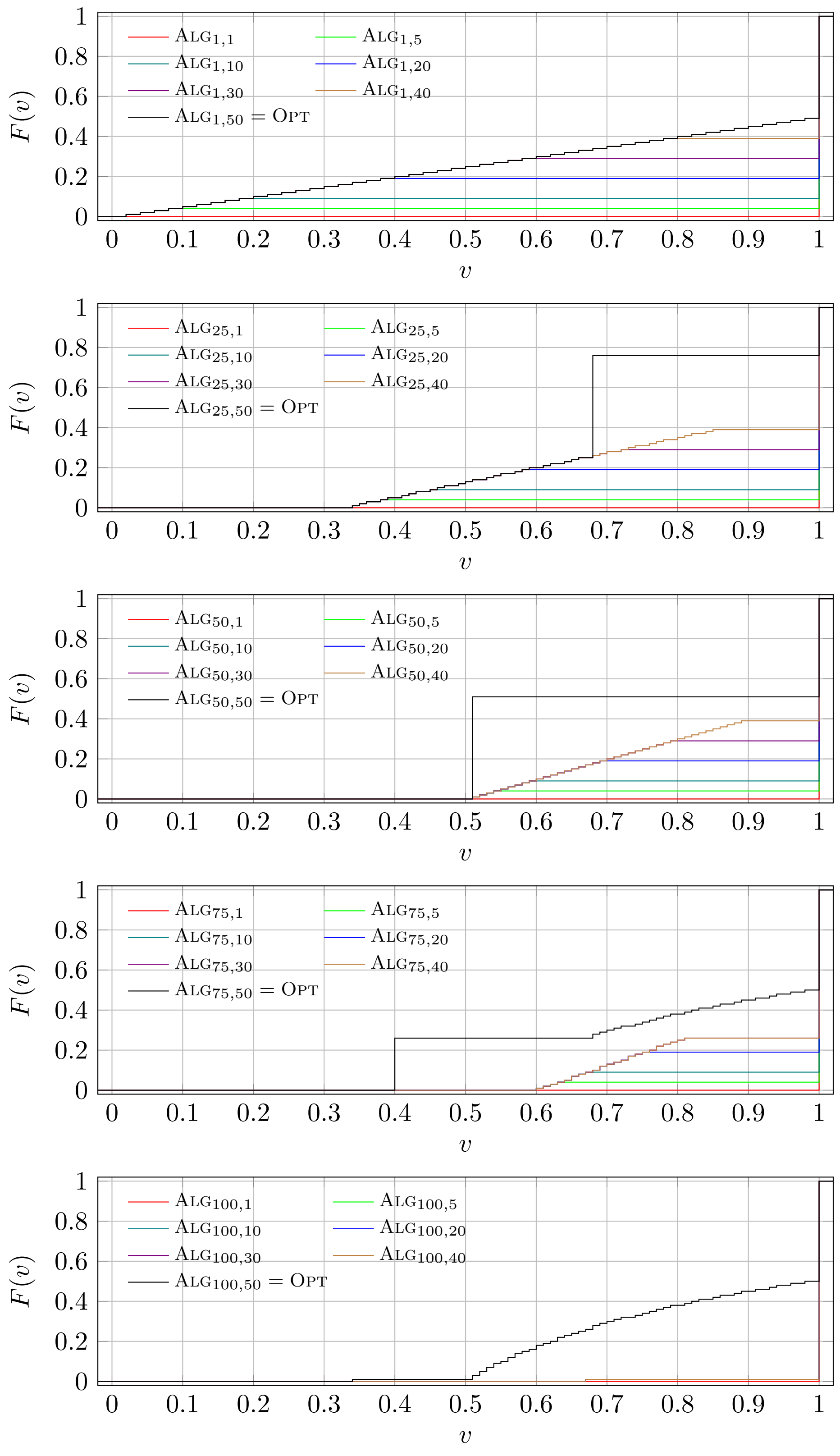


Figure 5.4: Counting distribution functions of performance ratio of costs relative to the online version for $n_{\max} = 100$ in the ski rental problem.

Since there is only one input sequence of a given length n and an expression for the costs on a given input sequence is known for all algorithms, Markov chain analysis becomes obsolete; all results are already included in the theoretical analysis in Chapter 4.1.

We conclude this section by pointing out that the computational experiments on the ski rental problem merely serve as an illustration for the results obtained by the exact analysis in Chapter 4.1. For $n_{max} = 100$, the exclusive benefit from lookahead is apparent from the clear-cut gaps between the counting distribution function plots of successive lookahead levels. Advantages arise by the fact that the ski rental problem is a one-shot decision problem where more information never leads to worse decisions.

5.2 Online Paging with Lookahead

Paging deals with the organization of a two-level memory system for sequential requests on elements (pages) of an alphabet $\mathcal{A}$ ([74], [164]). The large and slow main memory holds all pages of $\mathcal{A}$, whereas the small and fast cache memory can only hold $k < |\mathcal{A}|$ pages (or pointers to them). A requested page currently not in the cache produces a page fault, and whenever a page fault occurs, the page has to be brought to the cache first before it can be accessed. The goal is to minimize the number of page faults. For fixed $n \in \mathbb{N}$, the set of all input sequences is given by

$$\Sigma_n = \big\{(\sigma_1, \sigma_2, \ldots, \sigma_n) \,|\, \sigma_i \in \mathcal{A}, i = 1, \ldots, n\big\}$$

and comprises all page sequences of length n. Typically, pages must be brought to the cache in their order of appearance in the input sequence – regardless of the type and size of lookahead. In the online version, only σ_i with $i = 1, \ldots, n$ is known when σ_i is requested. In the online version with lookahead of size l, σ_i with $i = 1, \ldots, n$ and $l-1$ successive requests $\sigma_{i+1}, \ldots, \sigma_{i+l-1}$ are known if $i + l - 1 \leq n$, otherwise σ_i and additional $n - i$ successive requests $\sigma_{i+1}, \ldots, \sigma_n$ are known. In the offline version, all requested pages are known at the beginning. Subsequently, a page that is already in the cache is called a cache page.

In our experiment, we draw $m = 1000$ independent request sequences where each one of them consists of $n = 100$ pages. The cache has a capacity of $k = 10$ pages and the alphabet corresponds to the 26 standard letters, i.e., $\mathcal{A} = \{A, B, \ldots, Z\}$. Lookahead sizes l are chosen from $\{1, 5, 10, 20, 40, \ldots, 100\}$. In accordance with mainstream research on paging (see, e.g., [4], [33]), also our analysis takes into account different levels of locality of reference in the input sequence:

- No locality at all, i.e., at any time each page is equally probable.
- Locality by generating input sequences according to an access graph, i.e., page sequences are generated by traversing the edges of a graph whose vertices are labeled with the pages. Pages could, for instance, be understood as the subpages of an internet homepage, i.e., the access graph amounts to the homepage navigation scheme.
- Locality by assigning each page a fixed probability for occurring next based on relative letter frequencies in texts as elicited by statistical evaluations.

We restrict the discussion to the first two cases and refer to Appendix A.2.2 for more information on the third case.

A decision by an algorithm is required only when a page fault occurs and merely consists of selecting the cache page to be evicted in order to make way for the current request.

Online algorithms

- FirstInFirstOut (Fifo): In case of a page fault, evict a cache page which entered the cache first ([149]).
- LastInFirstOut (Lifo): In case of a page fault, evict a cache page which entered the cache last ([149]).
- LeastFrequentlyUsed (Lfu): In case of a page fault, evict a cache page which has been requested least frequently ([149]).
- LeastRecentlyUsed (Lru): In case of a page fault, evict a cache page which has been requested least recently ([149]).

Online algorithms with lookahead of size l

- FirstInFirstOut$_l$ (Fifo$_l$): In case of a page fault, if there are cache pages which are not requested in the lookahead, evict a cache page which entered the cache first among these pages; otherwise evict a cache page whose next request is farthest ahead.
- LastInFirstOut$_l$ (Lifo$_l$): In case of a page fault, if there are cache pages which are not requested in the lookahead, evict a cache page which entered the cache last among these pages; otherwise evict a cache page whose next request is farthest ahead.
- LeastFrequentlyUsed$_l$ (Lfu$_l$): In case of a page fault, if there are cache pages which are not requested in the lookahead, evict a cache page which has been requested least frequently among these pages; otherwise evict a cache page whose next request is farthest ahead.

- LeastRecentlyUsed$_l$ (Lru$_l$): In case of a page fault, if there are cache pages which are not requested in the lookahead, evict a cache page which has been requested least recently among these pages; otherwise evict a cache page whose next request is farthest ahead.

Optimal offline algorithm

- Optimal (Opt): In case of a page fault, evict a cache page whose next request is farthest ahead (Belady's optimal replacement algorithm in [17]).

Note that Fifo$_l$, Lifo$_l$, Lfu$_l$ and Lru$_l$ all coincide with Opt for $l = n = 100$ and with Fifo, Lifo, Lfu and Lru, respectively, for $l = 1$.

Besides regular request lookahead, we also include batched request lookahead of size l into our experiment to find out whether a steady supply with input elements is really needed or whether input element release in blocks suffices to obtain satisfying results: During the processing of a batch of input elements, the number of pages seen by the algorithm gradually reduces by one in each processing step. Since data is often organized in blocks or packages, batched lookahead is realistic for practical applications. If algorithms operate under batched lookahead, we indicate them with an added suffix B in the algorithm name.

5.2.1 Average Results

Figure 5.5 displays that each algorithm class performs equally good under the same lookahead regime when each page is equally probable to be the next one encountered in an input sequence. Because of this assumption, the probability for a page fault is identical for each possible current cache configuration, i.e., irrespective of an algorithm's previous organization of the cache. Lookahead leads to a substantial decrease in the number of page faults: Between the extreme cases of online and offline optimization, average page fault reductions of 36.7 % are observed. However, the marginal benefit of lookahead is strictly decreasing, i.e., the first lookahead units are the most valuable. Because all algorithms collapse into Opt for $l = 100$, every algorithm yields the optimal number of average page faults in this case. Applying regular lookahead instead of batched lookahead pays off by up to 15.4 % for $l = 20$.

When a (non-trivial) stochastic model about page occurrences is imputed by locality of reference in form of an access graph, algorithms behave differently as can be seen in Figure 5.6. Algorithm Lru$_l$ now outperforms the other algorithms for fixed l because construing input sequences according to the access graph implies that recently requested pages are more likely for being next than pages requested a long time ago as a result of their (shortest path)

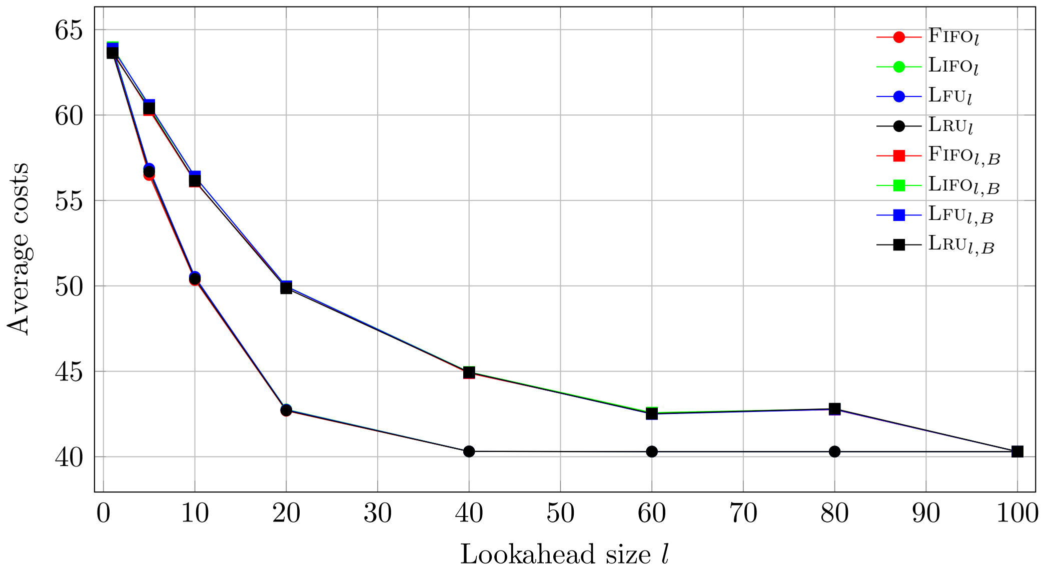


Figure 5.5: Average costs for different lookahead sizes and $n = 100$ in the paging problem when each page is equally probable.

distance to the vertex labeled with the current page. This is in line with $\textsc{Lru}_l$'s behavior of keeping recent requests and evicting requests from long ago. Likewise, $\textsc{Fifo}_l$ fares relatively good as it also sticks to deleting older pages first. Quite contrary, both $\textsc{Lifo}_l$ and $\textsc{Lfu}_l$ counteract the rationale of locality of reference imputed by an access graph: Because they evict pages that just joined the cache or yet have a small frequency count since they have just joined the cache, they make page faults much more likely as the evicted pages are near to the current page in terms of the distance in the access graph.

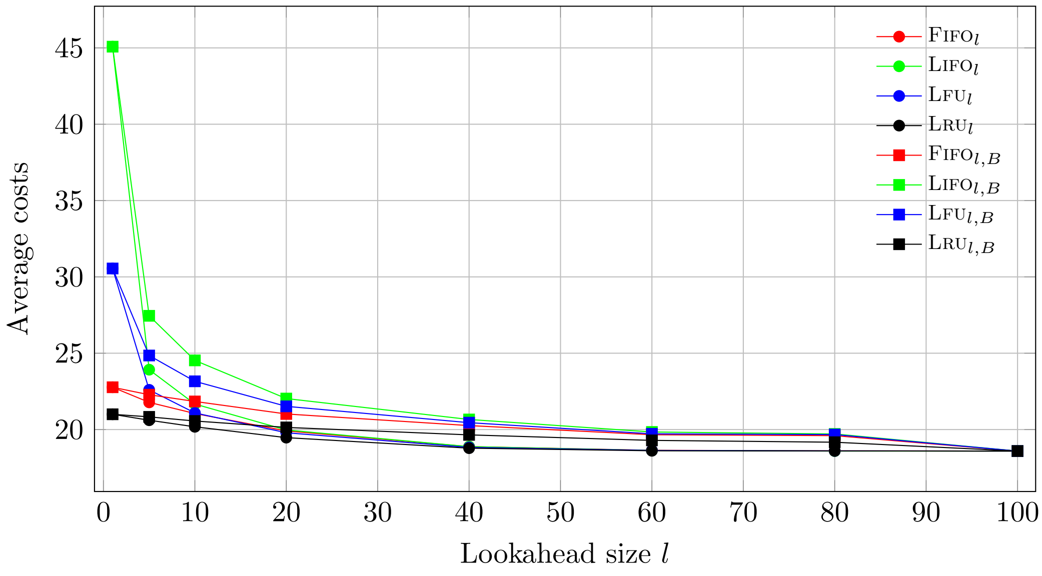


Figure 5.6: Average costs for different lookahead sizes and $n = 100$ in the paging problem when page sequences are generated according to an access graph.

The positive effect of lookahead is retained also under locality of reference, albeit for Lru$_l$ and Fifo$_l$ its magnitude is diminished strongly as a result of the already satisfying performance of the pure online algorithms. For these two algorithms, the lookahead effect is up to 11.5 % and 18.4 %, respectively, when online and offline optimization are compared. We conclude that locality of reference makes the future more predictive such that suitable algorithms can capitalize already from this circumstance substantially. For these algorithms, it is nearly irrelevant whether lookahead is provided on a rolling time horizon or in batches.

We remark that in locality of reference with preassigned page probabilities, Lfu$_l$ excels the other algorithms for fixed l because it bases its decision on frequency counts which are expected to comply with prescribed page probabilities that have been derived from letter frequencies. This result is in sharp contrast to competitive analysis where Lfu and Lifo are known to be non-competitive, whilst Fifo and Lru are $(k = 10)$-competitive.

The impact of lookahead on the given algorithms is exclusively beneficial in case of regular lookahead which is seen by two arguments: First, evicting a page seen in the lookahead would lead to a guaranteed page fault within the time window of the lookahead size, whereas evicting one that is not seen does not lead to a page fault in the same time window. Second, the worst-case concerning the obtained cache configuration would be that the evicted page would be needed immediately after the lookahead time window. However, the needed cache configuration could then be produced with one page fault in total by reinserting the evicted page in the cache again. In summary, evicting a page not seen in the lookahead can never be worse than evicting a page seen in the lookahead, and it maintains the chance of avoiding unnecessary page faults. In case of batched lookahead, there may be instances with degradations in the objective value although more lookahead was provided as seen in the last column of Table A.7 in Appendix A.2.2. This effect is due to the variable size of the information preview window under batched lookahead where algorithms operate under successively tightened lookahead within each batch.

5.2.2 Distributional Results

Because algorithms behave identically when each page is equally probable and due to the small width of confidence intervals as seen in Tables A.4 to A.6 of Appendix A.2.2, we omit a discussion of this case and concentrate on the results found under locality of reference in the access graph model.

In Figure 5.7, the stable and superior performance of Lru$_l$ and Fifo$_l$ is seen by the narrow interval of observed objective intervals for each lookahead level and the relative closeness of

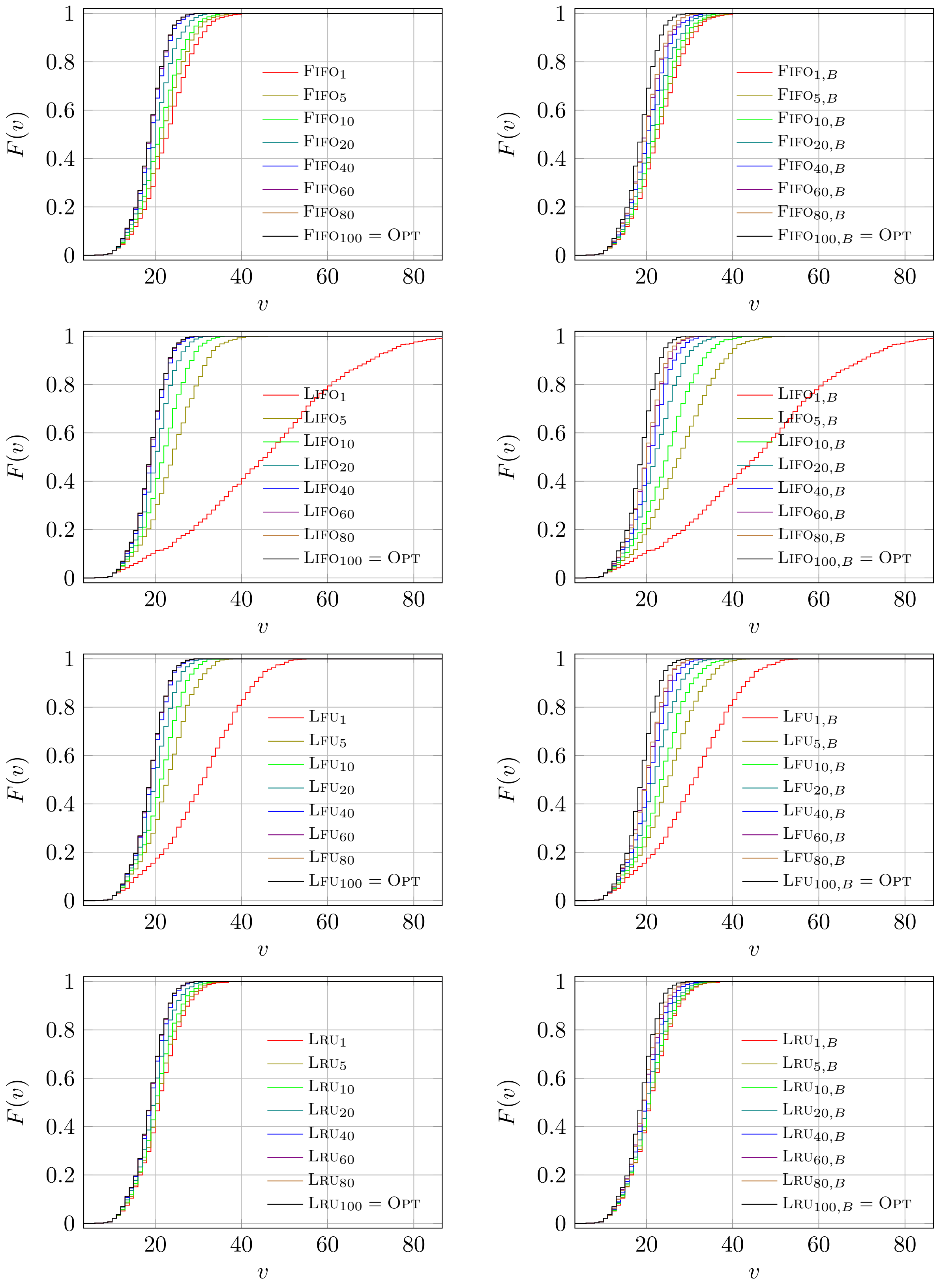


Figure 5.7: Empirical counting distribution functions of costs for $n = 100$ in the paging problem when page sequences are generated according to an access graph.

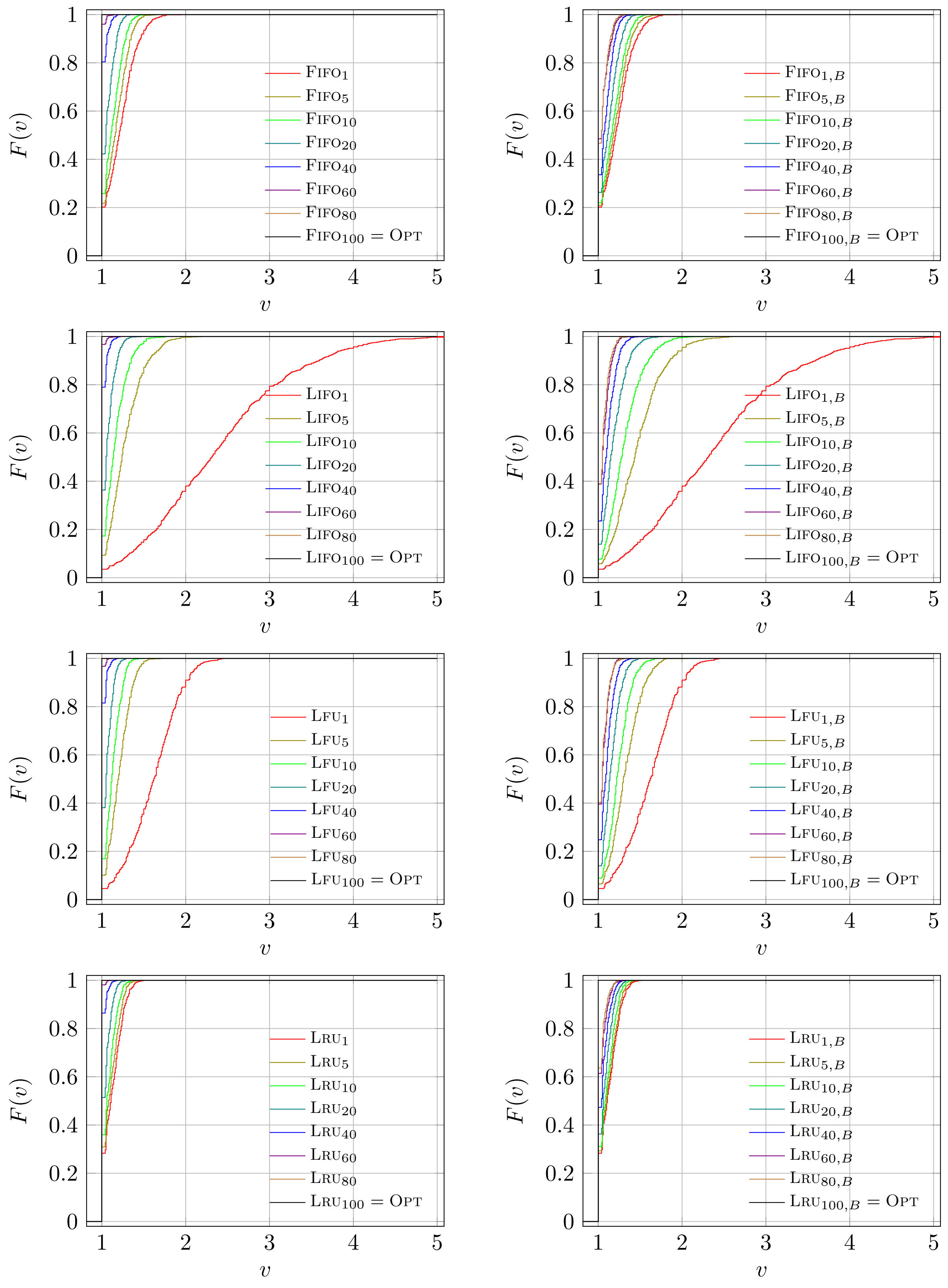


Figure 5.8: Empirical counting distribution functions of performance ratio of costs relative to OPT for $n = 100$ in the paging problem when page sequences are generated according to an access graph.

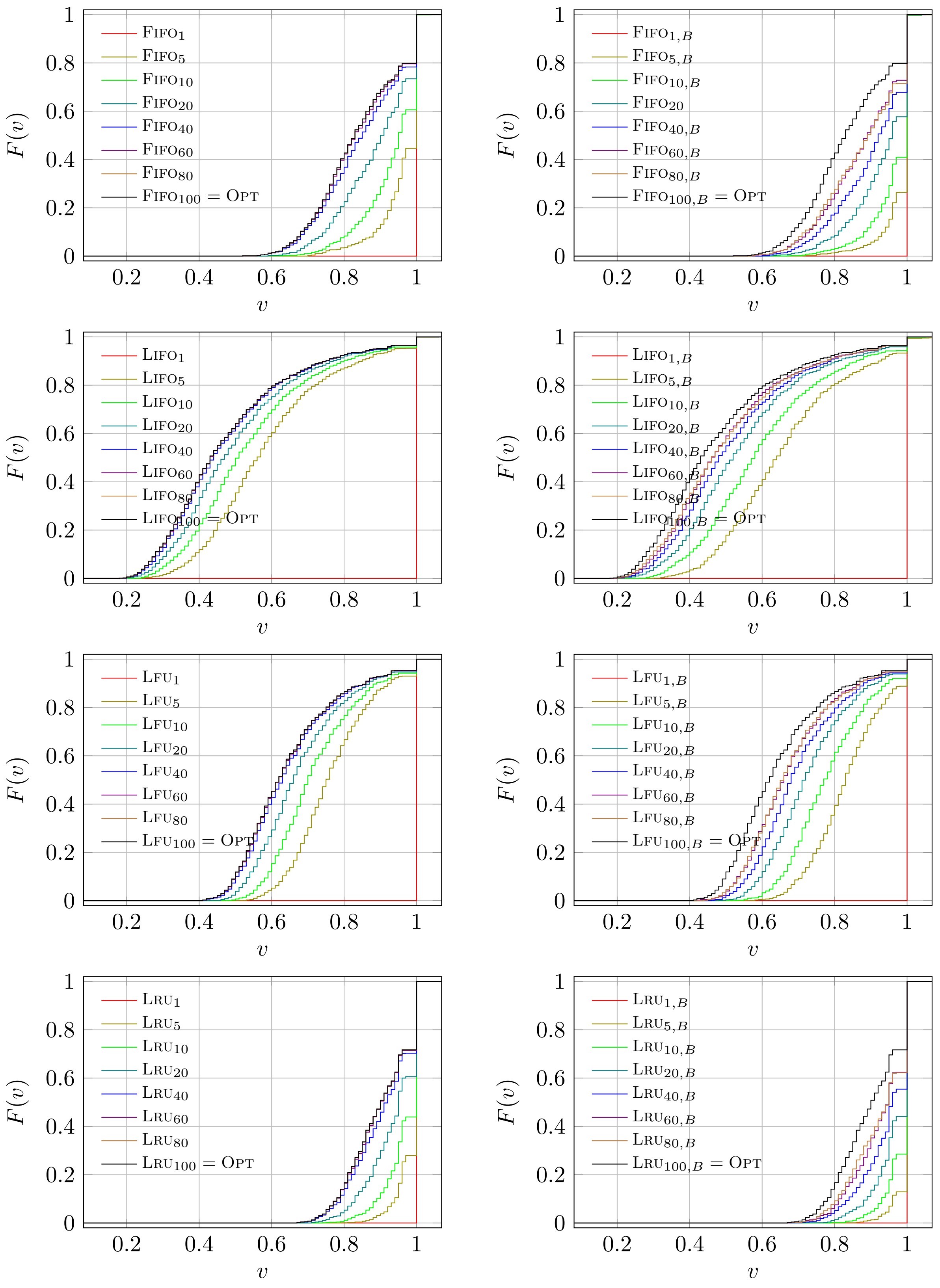


Figure 5.9: Empirical counting distribution functions of performance ratio of costs relative to the online version for $n = 100$ in the paging problem when page sequences are generated according to an access graph.

the plots to each other for all pairs of lookahead levels. This is also confirmed by the widths and positions of the confidence intervals in Table A.7 of Appendix A.2.2. Lookahead leads to improvement as seen by the ordering of the empirical counting distribution functions, although not as drastically as for $\textsc{Lifo}_l$ and $\textsc{Lfu}_l$. For these algorithms, stable behavior is reached only if there is a (real) lookahead of size larger than 1. Accordingly, additional lookahead leads to a much higher percentage of decrease in the objective value than for $\textsc{Lru}_l$ and $\textsc{Fifo}_l$.

Experimental competitive ratios as displayed in Figure 5.8 and indicated in the column for the maximum observed value in Table A.8 of Appendix A.2.2 are consistently smaller than 2 for $\textsc{Fifo}_1$ and $\textsc{Lru}_1$; hence, they are much better than the theoretical value of $k = 10$ would suggest. Already for $\textsc{Lru}_1$ and $\textsc{Fifo}_1$ the fraction of admissible input sequences with the minimum number of page faults amounts to 20.2 % and 28 %, respectively. Yet, even under locality of reference, instances are encountered where $\textsc{Lru}_1$ and $\textsc{Fifo}_1$ miss optimality by 50 % and 94 %, respectively. For $l = 80$, the worst observed performance ratio drops to 1.06 for both algorithms. For high lookahead values, the empirical counting distribution functions of the performance ratios are moving closer to 1 and become steeper, i.e., with a fair amount of lookahead the deviation from optimality can be preestimated. From the (local) optimality of evicting a cache page which lies farthest in the future, it follows that for l tending to $n = 100$ the number of page sequences with performance ratio larger than 1 is non-increasing.

Comparing algorithms to their respective online version as illustrated in Figure 5.9 approves the exclusive benefit of additionally provided information under regular lookahead because all performance ratios encountered are smaller than 1. For $\textsc{Lru}_l$ and $\textsc{Fifo}_l$, there are instances where full lookahead leads to a reduction to only 67 % and 52 % of the number of page faults from the online case, respectively. For batched lookahead, improvements are slightly smaller; instances sporadically experience an objective value degradation upon additional information.

5.2.3 Markov Chain Results

The elements of the state space $\mathcal{S}$ subsume the cache configuration $c = (c_1, \ldots, c_k)$ whose elements are ordered lexicographically, the lookahead pages $p = (p_1, \ldots, p_l)$, the page attribute configurations $a = (a_1, \ldots, a_{|\mathcal{A}|})$ and the number of page faults v so far, i.e.,

$$\mathcal{S} = \big\{(c, p, a, v) \,|\, c \in \mathcal{A}^k, p \in \mathcal{A}^l, a \in \mathbb{N}^{|\mathcal{A}|}, v \in \mathbb{N}_0\big\}.$$

Page attribute a_j with $j \in \{1, 2, \ldots, |\mathcal{A}|\}$ holds information concerning page j relevant to successive algorithm computations. For instance, a_j may hold the last time that j entered the cache, the number of requests on j so far or the number of requests that lie between the current and the last request on j. Upon arrival of a new page, each algorithm Alg decides which page to evict from the cache – if necessary – in order to bring p_1 into the cache. Hence, Alg is a function

$$\textsc{Alg} : \mathcal{S} \times \mathcal{A} \to \{0, 1, 2, \ldots, k\}$$

which gives the cache position to be evicted where $\textsc{Alg}(\cdot) = 0$ means that no page needs to be evicted. The successor state of $s = (c, p, a, v)$ upon arrival of page p_{new} is given by $s' = (c', p', a', v')$ where c' arises from c by removing element $c_{\textsc{Alg}(s, p_{new})}$, appending element p_1 and ordering lexicographically, p' arises from p by removing p_1 and appending p_{new},

$$a'_j = \begin{cases} a_j & \text{if } \textsc{Alg} \in \{\textsc{Fifo}, \textsc{Lifo}\}, \textsc{Alg}(s, p_{new}) = 0, \\ a_j & \text{if } \textsc{Alg} \in \{\textsc{Fifo}, \textsc{Lifo}\}, \textsc{Alg}(s, p_{new}) \neq 0, p_1 \neq j, \\ \max\{a_i\} + 1 & \text{if } \textsc{Alg} \in \{\textsc{Fifo}, \textsc{Lifo}\}, \textsc{Alg}(s, p_{new}) \neq 0, p_1 = j, \\ a_j & \text{if } \textsc{Alg} = \textsc{Lfu}, p_1 \neq j, \\ a_j + 1 & \text{if } \textsc{Alg} = \textsc{Lfu}, p_1 = j, \\ a_j + 1 & \text{if } \textsc{Alg} = \textsc{Lru}, p_1 \neq j, \\ 0 & \text{if } \textsc{Alg} = \textsc{Lru}, p_1 = j. \end{cases}$$

for $j = 1, \ldots, |\mathcal{A}|$ and

$$v' = \begin{cases} v & \text{if } \textsc{Alg}(s, p_{new}) = 0, \\ v + 1 & \text{otherwise.} \end{cases}$$

Finally, we have to specify the initial state of the cache.

Since all algorithms include past information into their decision making, the Markov property is lost in the ordinary sense. We artificially make Markov chain analysis applicable by integrating all necessary information from the past in form of the page attributes a into the state space. For this trick, we have to pay the price of an increased state space size, namely

$$|\mathcal{S}| = \binom{|\mathcal{A}|}{k} \cdot |\mathcal{A}|^l \cdot (n+1)^{|\mathcal{A}|} \cdot (n+1).$$

For $\textsc{Alg} \in \{\textsc{Fifo}_l, \textsc{Lifo}_l, \textsc{Lfu}_l, \textsc{Lru}_l\}$ and selected lookahead level l, we now determine the successor for each state and each new page request to obtain the one-step frequency matrix

$F^{\text{ALG}} = (f_{ij}^{\text{ALG}})$ of the Markov chain. Entry f_{ij}^{ALG} gives the number of page requests which lead from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ by applying ALG in i. We get the n-step frequency matrix $F^{\text{ALG},n} = (f_{ij}^{\text{ALG},n})$ as the nth power of F^{ALG}, i.e., $F^{\text{ALG},n} = (F^{\text{ALG}})^n$. Entry $f_{ij}^{\text{ALG},n}$ gives the number of permutations of n page requests, i.e., the number of input sequences of length n which lead from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ by applying ALG n times in a row starting in i.

In our numerical experiment, we use the parametrization $n = 5$, $\mathcal{A} = \{A, B, C\}$, $k = 2$ and $l \in \{1, 2, \ldots, 5\}$. For this small exemplary setting, the state space size already amounts to $\binom{3}{2} \cdot 3^5 \cdot 6^3 \cdot 6 = 944\,784$ elements for $l = 5$ which is mainly due to the artificial extension of the state space in order to include information concerning the past behavior of the algorithms. Since we consider no locality of reference, we have that $\text{FIFO}_l, \text{LIFO}_l, \text{LFU}_l, \text{LRU}_l$ all lead to identical objective value distributions for fixed l. Therefore, we speak of ALG_l in the sequel and refer to each of the algorithms.

In consonance with Figure 5.5, also Figure 5.10 affirms the positive impact of lookahead on the number of page faults encountered. From the area between the plots of two successive lookahead levels, we find that already for relatively short input sequences it is seen that the first lookahead units prove most valuable.

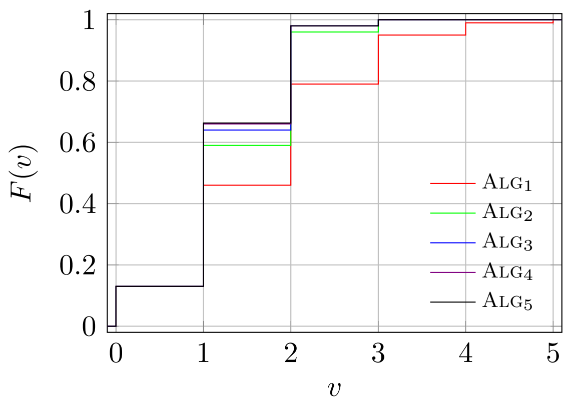


Figure 5.10: Exact distribution functions of costs for $n = 5$ in the paging problem when pages are equally distributed.

We conclude that the positive effect of lookahead has its roots in the immediate reduction of the number of page faults by not deleting a page whenever it is seen in the lookahead as compared to the case where it may be deleted as it cannot be seen due to a more limited information preview window. Hence, lookahead can never be harmful in the paging problem. Observe that there is no change in the rule set of paging under additional lookahead due to the sequential character of page requests; all improvements are a result of advanced information.

5.3 Online Bin Packing with Lookahead

A specification of the online bin packing problem with lookahead and algorithms BestFit and BestFit$_l$ is given in Chapter 4.2. In the numerical analysis, we take into account further algorithms and consider two cases with respect to the rule set: Item permutations may be allowed when items have small physical dimensions such that sorting is done easily, or item permutations may be forbidden when items are physically too large to be rearranged. In applications, the restriction of a bounded number of open bins at a time is encountered frequently due to capacity or space limitations. The latter problem is called the bounded-space bin packing problem with lookahead. We obtain four problem settings:

- Classical bin packing with item permutations
- Classical bin packing without item permutations
- Bounded-space bin packing with item permutations
- Bounded-space bin packing without item permutations

For fixed $n \in \mathbb{N}$, the set of all input sequences of length n is given by

$$\Sigma_n = \big\{(\sigma_1, \sigma_2, \ldots, \sigma_n) \,|\, \sigma_i := s_i \in (0,1], i = 1, \ldots, n\big\}$$

and comprises all item sequences of length n where σ_i is identified with the size of the ith item. In the online version, only σ_i with $i = 1, \ldots, n$ is known when σ_i has to be packed. When σ_i with $i = 1, \ldots, n$ has to be packed in the online version with lookahead of size l under order preservation, also $l-1$ successive items $\sigma_{i+1}, \ldots, \sigma_{i+l-1}$ are known if $i+l-1 \leq n$, otherwise σ_i and additional $n-i$ successive items $\sigma_{i+1}, \ldots, \sigma_n$ are known. When known items may be packed in arbitrary order, then at the ith packing time, the l unpacked items from $\sigma_1, \sigma_2, \ldots, \sigma_{i+l-1}$ with $i = 1, \ldots, n$ are known if $i + l - 1 \leq n$, otherwise the $n - i + 1$ unpacked items from $\sigma_1, \sigma_2, \ldots, \sigma_n$ are known. In the offline version, all items are known at the beginning.

We select two settings for our numerical experiments which both feature $m = 1000$ independently drawn item sequences: In the first setting, we have $n = 25$ items per item sequence; in the second setting, an item sequence consists of $n = 100$ items. Item sizes are drawn from $(0, 1]$ and each bin has unit capacity. In the bounded-space version, only $k = 3$ bins are allowed to be open at the same time. For $n = 25$, lookahead sizes are $l \in \{1, 5, 10, \ldots, 25\}$; for $n = 100$, we have $l \in \{1, 5, 10, 20, 40, \ldots, 100\}$. We discuss results for $n = 25$ and refer the reader to Appendix A.2.3 for $n = 100$.

We consider both regular and batched request lookahead of size l in order to see whether a steady information about future items leads to mentionable improvements or whether it suffices to release the items in batches in order to obtain satisfactory results. If algorithms operate under batched lookahead, we indicate them with an added suffix B in the algorithm name.

A decision by an algorithm is required for every item that has to be packed and consists of selecting the bin in which this item should be put.

5.3.1 Classical Bin Packing

Because a bin once opened is never closed again, each of the following algorithms can be used irrespective of whether item permutations are allowed or not: An item to bin assignment determined by any of the algorithms can be realized also in case of forbidden permutations by fictively rearranging all items assigned to the same bin such that their order in a bin complies with their release order.

Online algorithms

- FirstFit (Ff): If there is at least one open bin that can accommodate the item to be packed, put the item in the bin that was opened first among these bins; otherwise open a new bin and put the item in ([55]).
- BestFit (Bf): If there is at least one open bin that can accommodate the item to be packed, put the item in the fullest among these bins; otherwise open a new bin and put the item in ([55]).

Online algorithms with lookahead of size l

- FirstFit$_l$ (Ff$_l$): Sort the items in the lookahead by non-increasing size and fictively pack them with Ff. If the item to be packed is put in a new bin, open a new bin and put the item in; otherwise put the item in the bin from the fictive assignment ([55]).
- BestFit$_l$ (Bf$_l$): Sort the items in the lookahead by non-increasing size and fictively pack them with Bf. If the item to be packed is put in a new bin, open a new bin and put the item in; otherwise put the item in the bin from the fictive assignment ([55]).
- Optimal$_l$ (Opt$_l$): In Figure 5.11, set N, N^o, s and f according to the current bin configuration and items seen in the lookahead. Solve the resulting IP formulation in Figure 5.12. If the item to be packed is put in a new bin, open a new bin and put the item in; otherwise put the item in the bin from the obtained assignment.

Sets

I	set of items with $I = \{1, \ldots, N\}$
J	set of potential new bins with $J = \{1, \ldots, N\}$
J^o	set of already used bins with $J^o = \{1, \ldots, N^o\}$
L	set of fill levels ℓ with $L = \{0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$

Parameters

N	number of items and potential new bins
N^o	number of bins already used
s_i	size of item $i \in I$
f_{j^o}	fill level of already used bin $j^o \in J^o$
w_ℓ	weight of fill level $\ell \in L$ with $w_{0.7} = 0.01$, $w_{0.75} = 0.02$, $w_{0.8} = 0.04$, $w_{0.85} = 0.08$, $w_{0.9} = 0.16$, $w_{0.95} = 0.32$

Variables

$$x_{ij} = \begin{cases} 1 & \text{if item } i \in I \text{ is put in bin } j \in J, \\ 0 & \text{else} \end{cases}$$

$$x^o_{ij^o} = \begin{cases} 1 & \text{if item } i \in I \text{ is put in already used bin } j^o \in J^o, \\ 0 & \text{else} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if bin } j \in J \text{ is used}, \\ 0 & \text{else} \end{cases}$$

$$\alpha_{j\ell} = \begin{cases} 1 & \text{if bin } j \in J \text{ has fill level in } [\ell, \ell + 0.05) \text{ for } \ell \in L \text{ with } \ell \neq 0.95, \\ & \text{or } [\ell, \ell + 0.05] \text{ for } \ell = 0.95, \\ 0 & \text{else} \end{cases}$$

$$\alpha^o_{j^o\ell} = \begin{cases} 1 & \text{if already used bin } j^o \in J^o \text{ has fill level in } [\ell, \ell + 0.05) \text{ for } \ell \in L \\ & \text{with } \ell \neq 0.95, \text{or } [\ell, \ell + 0.05] \text{ for } \ell = 0.95, \\ 0 & \text{else} \end{cases}$$

Figure 5.11: Sets, parameters and variables in the IP formulation of the bin packing problems (see also [72]).

$$
\begin{aligned}
\min \quad & \sum_{j \in J} y_j + |J^o| && && (5.1)\\
\text{s.t.} \quad & \sum_{j \in J} x_{ij} + \sum_{j^o \in J^o} x^o_{ij^o} &= 1 && i \in I && (5.2)\\
& \sum_{i \in I} s_i x_{ij} &\leq y_j && j \in J && (5.3)\\
& \sum_{i \in I} s_i x^o_{ij^o} &\leq 1 - f_{j^o} && j^o \in J^o && (5.4)\\
& x_{ij}, x^o_{ij^o}, y_j &\in \{0,1\} && i \in I,\ j \in J,\ j^o \in J^o && (5.5)
\end{aligned}
$$

Figure 5.12: IP formulation of the bin packing problem.

- Optimal$'_l$ (Opt$'_l$): In Figure 5.11, set N, N^o, s and f according to the current bin configuration and items seen in the lookahead. Solve the IP formulation in Figure 5.12 after extending it with expressions from Figure 5.13 as follows: Replace Objective Function (5.1) by (5.6) and include Constraints (5.7) to (5.11). If the item to be packed is put in a new bin, open a new bin and put the item in; otherwise put the item in the bin from the obtained assignment.

$$
\begin{aligned}
\min \quad & \sum_{j \in J} y_j + |J^o| - \tfrac{1}{|J|+|J^o|}\Big(\sum_{j \in J}\sum_{\ell \in L} w_\ell \alpha_{j\ell} + \sum_{j^o \in J^o}\sum_{\ell \in L} w_\ell \alpha^o_{j^o\ell}\Big) && && (5.6)\\
& \sum_{\ell \in L} \alpha_{j\ell} \leq 1 && j \in J && (5.7)\\
& \sum_{\ell \in L} \alpha^o_{j^o\ell} \leq 1 && j^o \in J^o && (5.8)\\
& \ell \alpha_{j\ell} \leq \sum_{i \in I} s_i x_{ij} && j \in J,\ \ell \in L && (5.9)\\
& \ell \alpha^o_{j^o\ell} \leq \sum_{i \in I} s_i x^o_{ij^o} && j^o \in J^o,\ \ell \in L && (5.10)\\
& \alpha_{j\ell}, \alpha^o_{j^o\ell} \in \{0,1\} && j \in J,\ j^o \in J^o,\ \ell \in L && (5.11)
\end{aligned}
$$

Figure 5.13: Objective function and additional constraints for the modified IP formulation of the bin packing problem (see also [72]).

Optimal offline algorithm

- Optimal (Opt): In Figure 5.11, set $N := n$, $N^o := 0$, $f := 0$ and s according to all items seen. Solve the resulting IP formulation in Figure 5.12. Pack the items according to the obtained assignment. If necessary, sort items of a bin first.

While $\textsc{Opt}_l$ only considers the number of bins in the objective, $\textsc{Opt}'_l$ secondarily searches for an item to bin assignment with bins as full or empty as possible, but not with medium fill levels. This is done by promoting bins with large fill levels in Objective Function 5.6 in such a way that the total number of bins used stays unaffected. Constraints 5.7 to 5.11 ensure that at most one fill level is attained according to the combined item sizes in a bin. Thereby, the initial position for successive items is improved because in case of large item sizes, empty bins are more valuable. This approach of using a surrogate objective function in order to improve the initial position for future steps is due to Esen ([72]).

5.3.1.1 Average Results

Average algorithm behavior is displayed in Figures 5.14 and 5.15 for the cases of allowed and forbidden item permutations, respectively. The positive effect of allowing item permutations is virtually non-existent as compared to forbidden item permutations as can be seen from the nearly identical shapes of the corresponding plots in both diagrams.

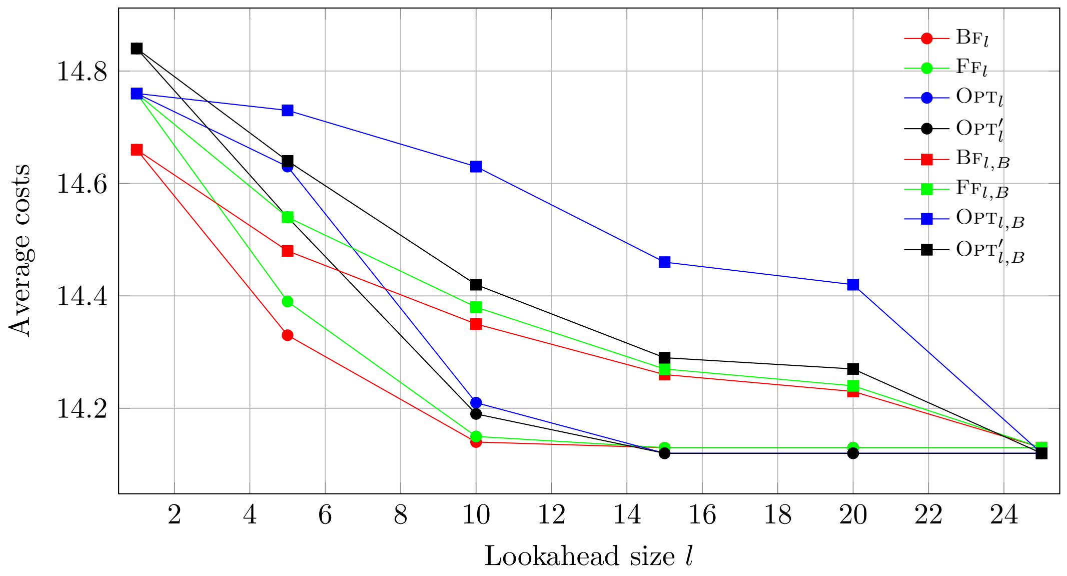


Figure 5.14: Average costs for different lookahead sizes and $n = 25$ in the classical bin packing problem when item permutations are allowed.

Recall that in Example 2.37 there was no value of permuting items at all in online bin packing with two item sizes. Moreover, as pointed out earlier, all algorithms under allowed item permutations can also be applied under forbidden item permutations to first determine a fictive assignment before packing an item. However, due to new items, the obtained bin configurations are not necessarily identical; yet, in both cases algorithms combine items to fit a bin, and the overall character of the bin configurations is approximately the same.

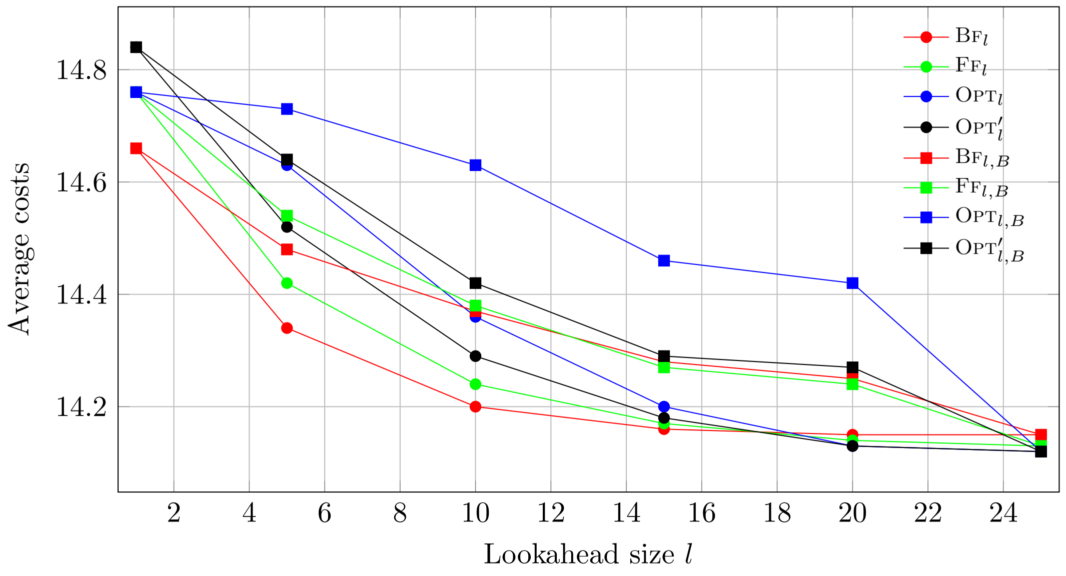


Figure 5.15: Average costs for different lookahead sizes and $n = 25$ in the classical bin packing problem when item permutations are forbidden.

From Tables A.13 and A.14 of Appendix A.2.3, we conclude that it may be slightly beneficial to be allowed to pack large items earlier, but we consider the overall effect as negligible, e.g., $\textsc{Opt}_{15}$ needs an average of 14.12 bins in case of allowed item permutations as opposed to 14.2 bins in case of no item permutations which is a relative advantage of less than 1 %. For this reason, we will restrict our discussion only to the case of allowed item permutations from now on.

In accordance with the theoretical analysis in Chapter 4.2, also for arbitrary item sizes only small gains can be achieved by additional lookahead. Between the pure online and offline situations only 0.53 to 0.7 bins could be saved on average depending on the algorithm used which amounts to an improvement of 3.6 % to 4.7 %. Moreover, we find that relatively simple algorithms like $\textsc{Bf}_l$ and $\textsc{Ff}_l$ outperform exact reoptimization methods for all lookahead sizes l with $l < 15$. $\textsc{Bf}_l$ produces the best overall results in this information regime. Algorithms $\textsc{Opt}_l$ and $\textsc{Opt}'_l$ beat the rule-based algorithms only for sufficiently large lookahead $l \geq 15$ by a rather slight margin. Hence, we infer that exact reoptimization is not needed in bin packing problems when a too large part of the future is unseen. The stability granted by $\textsc{Bf}_l$ and $\textsc{Ff}_l$ for small to medium lookahead sizes which is achieved by packing large items first proves advantageous over the "local" optimality of exact solutions to snapshot problems; these solutions are fragile once the situation changes due to newly announced lookahead items. Optimality of partial solutions becomes important only if their benefit cannot be undone by future decisions on upcoming items. Algorithms collectively show a decreasing marginal benefit from lookahead, i.e., the first lookahead units are most valuable and sufficient to drive item to bin assignments towards an optimal solution. In order to take full advantage of possible improvements due to additional information, it is recommended to use the regular type of lookahead rather than the batched type: For instance, BF_{10} shows an improvement of 3.5 % compared to the online case, whilst $\text{BF}_{10,B}$ accounts for an improvement of 2.1 %. From the percentage of item sequences with degraded objective value in case of additional lookahead in Tables A.13 and A.14 of Appendix A.2.3, we find that exclusive benefit from lookahead cannot be guaranteed. However, instances with a deterioration are encountered rarely. For $n = 100$ items per sequence, the picture remains the same and improvements are between 3.8 % and 4.9 %; for details, see Appendix A.2.3.

5.3.1.2 Distributional Results

The distributional results with respect to obtained objective values as shown in Figure 5.16 are affirmative to the minor positive effect of lookahead. This can be seen from the relative closeness of the plots of two successive lookahead levels to each other. For medium to large lookahead, differences in the counting distribution functions are even hardly perceivable. The small lookahead impact is further witnessed by the position of the confidence intervals (cf. Tables A.13 and A.14 of Appendix A.2.3): For two successive lookahead levels, they lie intimately close to each other on the objective value axis, sometimes even overlapping. Moreover, the width of all confidence intervals is smaller than 0.3 irrespective of the algorithm and lookahead level used which implies that results can be considered representative for typical item sequences. From the left column in Figure 5.16, we see that under regular request lookahead the effect of the first lookahead units is stronger than those of successive lookahead levels, whereas for algorithms using the batched type of lookahead on the right column the impact of further lookahead units appears more evenly distributed.

The most striking conclusion that can be drawn from the empirical counting distribution functions of the performance ratios relative to OPT in Figure 5.17 is that there is an immense fraction of item sequences which already lead to a performance ratio of 1 even if the algorithm that has to compete with OPT is not supplied any lookahead item at all. In this category, BF performs best with nearly half of all item sequences leading to an optimal number of bins used. The worst deviation from optimality as a result of informational nescience is approximately 23 % attained on an item sequence by OPT_1. Confidence intervals of the performance ratios (see Tables A.15 and A.16 of Appendix A.2.3) reduce to one point if rounded to two decimal places for all algorithms and lookahead levels. The largest encountered performance ratio in a confidence interval is 1.05 obtained for $\text{BF}_{1,B}$, FF_1, $\text{FF}_{1,B}$, OPT_1, $\text{OPT}_{1,B}$, OPT'_1 and $\text{OPT}'_{1,B}$. More than 90 % of all experimental competitive ratios

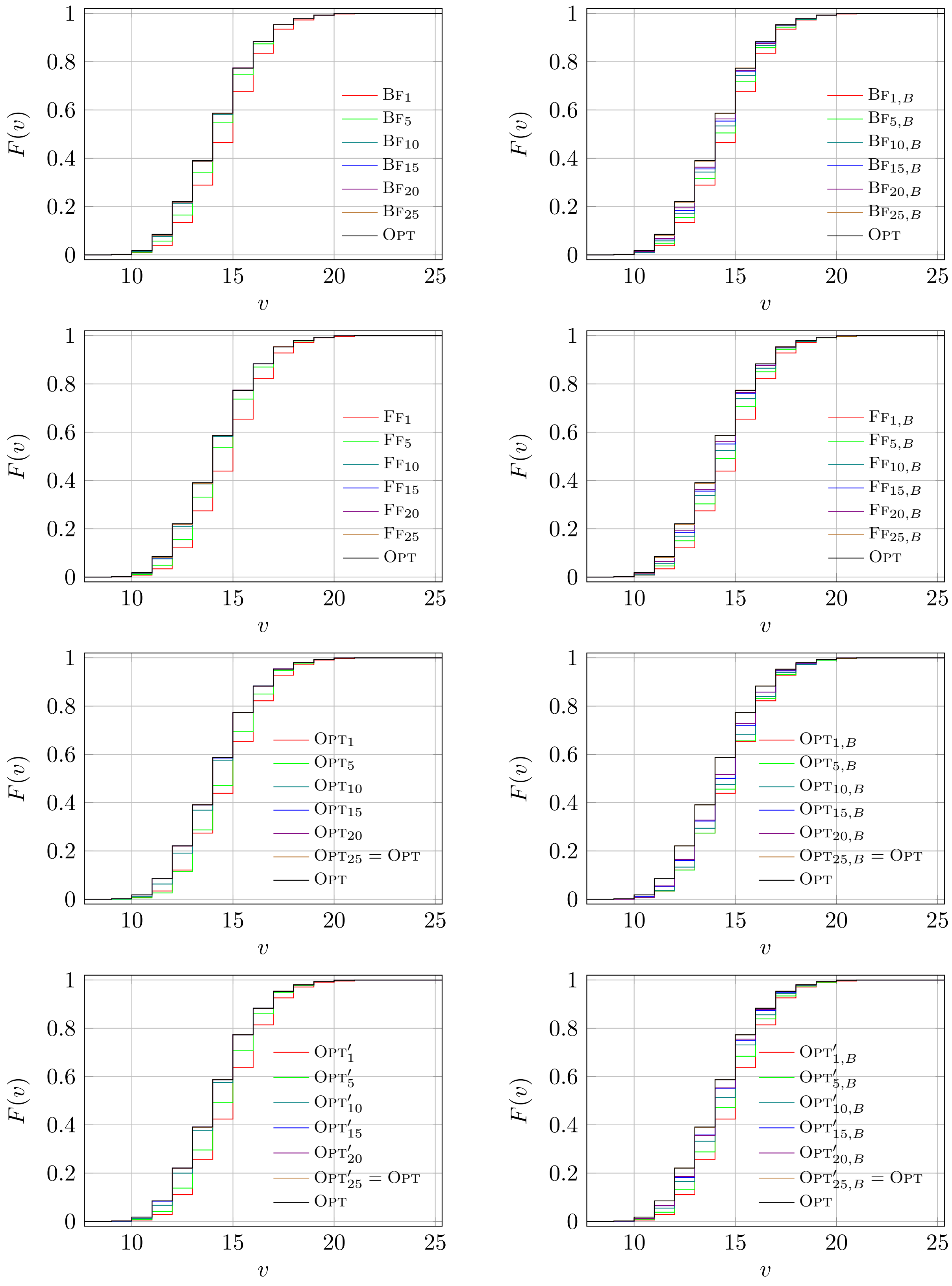


Figure 5.16: Empirical counting distribution functions of costs for $n = 25$ in the classical bin packing problem when item permutations are allowed.

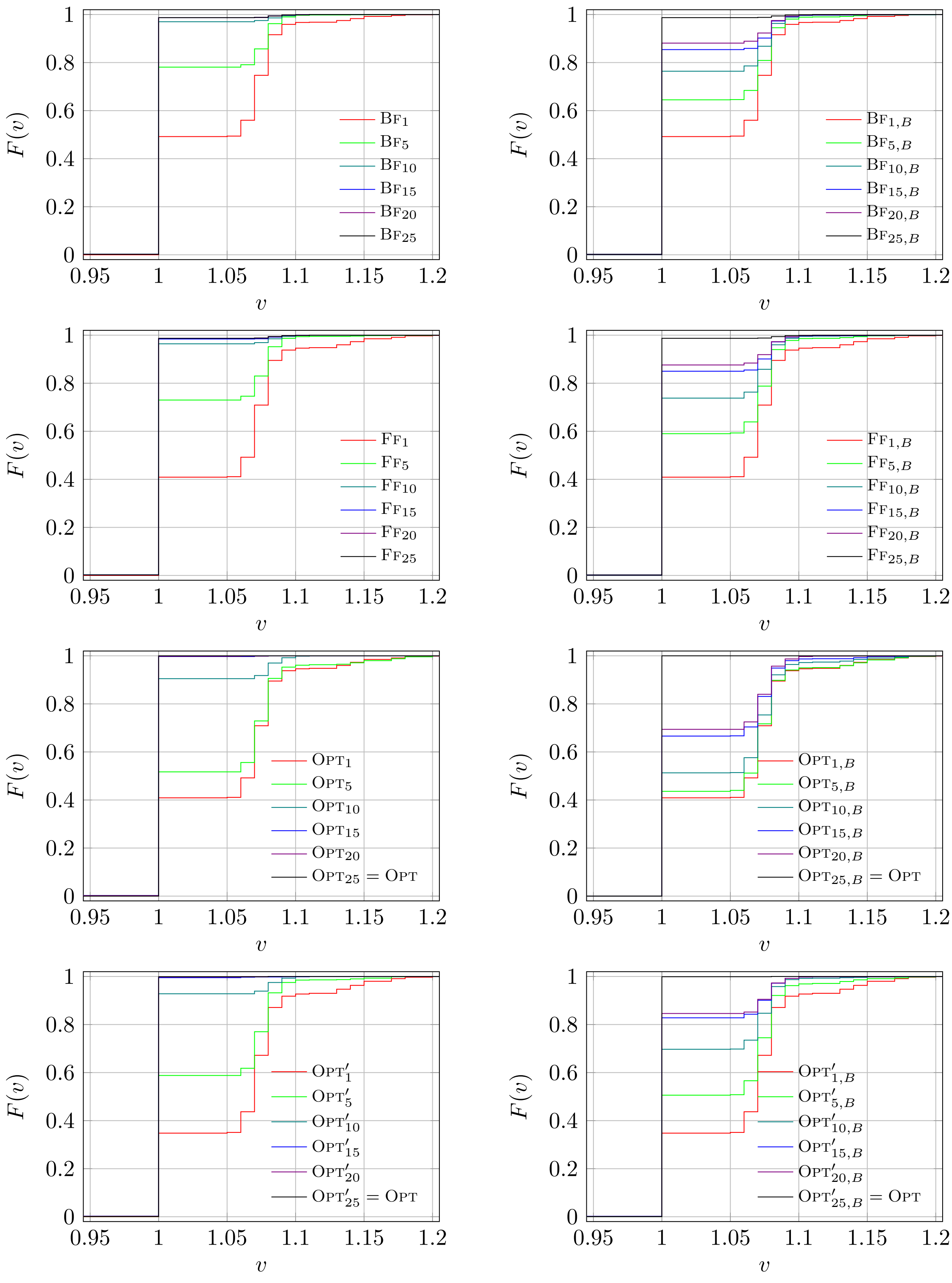


Figure 5.17: Empirical counting distribution functions of performance ratio of costs relative to OPT for $n = 25$ in the classical bin packing problem when item permutations are allowed.

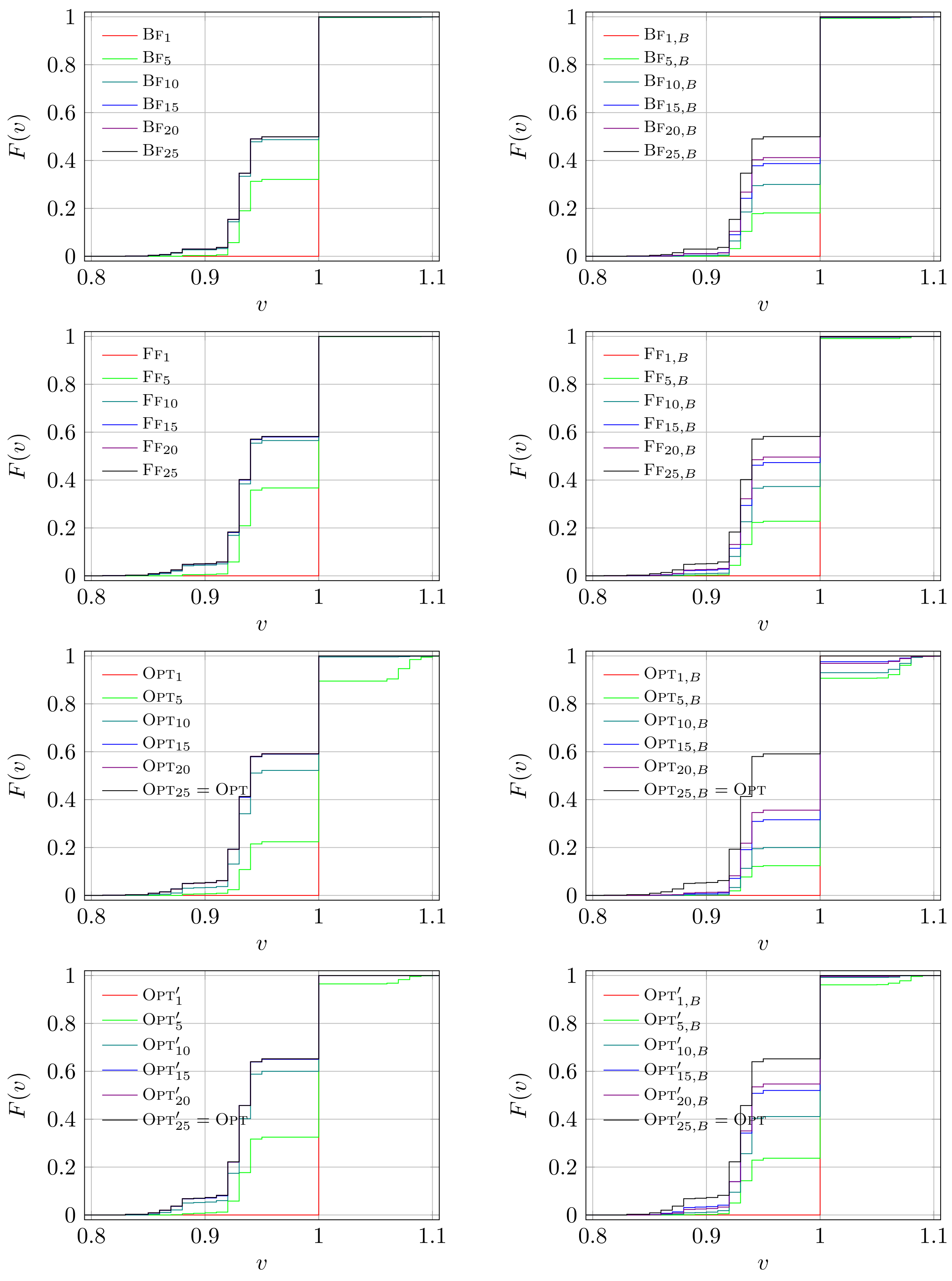


Figure 5.18: Empirical counting distribution functions of performance ratio of costs relative to the online version for $n = 25$ in the classical bin packing problem when item permutations are allowed.

are smaller than 1.1 for all algorithms and lookahead levels which additionally shows that lookahead benefits are of a rather small magnitude. The satisfactory behavior of the online algorithms can be explained by two reasons: First, since any bin is left open forever, a solid utilization of each bin is probable in the long term also in the online setting and the implications of bad decisions are either unnoticeable or rather small. Second, it is known that the competitive ratio of $\textsc{Bf}$ and $\textsc{Ff}$ is $\frac{17}{10}$ and that of $\textsc{Bf}_n$ and $\textsc{Ff}_n$ is $\frac{11}{9}$ ([14], [147]). Thus, algorithms endowed with lookahead cannot deviate more from optimality than suggested by these ratios, especially not under representative item sequences that look significantly different from pathologic worst-case sequences for which the above ratios were derived. Note that some few of the $m = 1000$ instances could not be solved to optimality in the prescribed time limit of 120 seconds. However, their proportion is so tiny such that performance ratios smaller than 1 are nearly unnoticeable in Figure 5.17 (cf. also Tables A.15 and A.16 of Appendix A.2.3).

Figure 5.18 yields plots of the empirical counting distributions functions for the performance ratio that result from comparing the algorithms to their respective online versions without lookahead. We derive two distinctive features of online bin packing with lookahead from these diagrams: First, it is seen from the plots of $\textsc{Ff}_{5,B}$ and all exact reoptimization approaches that additional information may also lead to a deterioration in the objective value if item sequences were leading algorithms to disadvantageous decisions that cannot necessarily be corrected later. Second, the largest part of the item sequences leads to performance ratios in the range between 0.9 and 1 irrespective of the chosen algorithm, lookahead type and batching mode. The counting distributions of attained performance ratios for the respective algorithms mostly differ within this group of item sequences depending on the lookahead level l. Confidence intervals and coefficients of variations of the performance ratios are negligibly small and collectively have lower bounds larger than 0.95 as listed in Tables A.17 and A.18 of Appendix A.2.3. Hence, there is also no great potential for improving the outcome of online algorithms by providing additional lookahead with respect to the performance ratio on typical item sequences.

We left out an experiment on Markov chains for the classical bin packing problem because of state space considerations: Since a bin once opened remains open for the rest of the packing process, the state space representation would have to account for that by including each possible bin configuration of up to n bins. In the bounded-space bin packing problem, this issue is mitigated due to the limited number of bins. Therefore, we include a Markov chain analysis in the following section on numerical experiments for the bounded-space bin packing problem.

5.3.2 Bounded-Space Bin Packing

Recall that $k \in \mathbb{N}$ gives the number of maximum allowed open bins and that whenever a new bin would need to be opened, one of the k open bins has to be closed first. A fictive assignment of the l lookahead items to bins based on an item list sorted by non-increasing item sizes only guarantees feasibility under allowed item permutations. When the item appearance order has to be respected also during packing, such a fictive assignment only guarantees feasibility for the first k items of the list of items sorted by non-increasing sizes: From the pigeonhole principle, it follows for $l \leq k$ that for each of the l items which requires a new bin, one of the already open bins will not be occupied with any of the $l - 1$ other items. For $l > k$, it could occur that the list assignment would put an item in a bin which had to be closed already before. The fullest of the k open bins is closed whenever a new bin is required. As a result, the algorithms with lookahead which rely on reordering items according to their sizes can exploit only $\min\{k, l\}$ lookahead units.

Online algorithms

- FirstFitBounded (Ffb): If there is at least one open bin that can accommodate the item to be packed, put it in the bin that was opened first among these bins; otherwise close the fullest open bin if k bins are open, open a new bin and put the item in ([55]).
- BestFitBounded (Bfb): If there is at least one open bin that can accommodate the item to be packed, put the item in the fullest among these bins; otherwise close the fullest open bin if k bins are open, open a new bin and put the item in ([55]).

Online algorithms with lookahead of size l

- FirstFitBounded$_l$ (Ffb$_l$):
  Item permutations allowed
  Sort the items in the lookahead by non-increasing size and fictively pack them with Ff. If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the fictive assignment.
  Item permutations forbidden
  Sort the first $\min\{k, l\}$ items in the lookahead by non-increasing size and fictively pack them with Ff. If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the fictive assignment.

- BestFitBounded$_l$ (Bfb$_l$):
  Item permutations allowed
  Sort the items in the lookahead by non-increasing size and fictively pack them with Bf. If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the fictive assignment.
  Item permutations forbidden
  Sort the first $\min\{k, l\}$ items in the lookahead by non-increasing size and fictively pack them with Bf. If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the fictive assignment.

- OptimalBounded$_l$ (Optb$_l$):
  Item permutations allowed
  In Figure 5.11, set N, N^o, s and f according to the current configuration of open bins and items seen in the lookahead. Solve the resulting IP formulation in Figure 5.12. If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the obtained assignment.
  Item permutations forbidden
  In Figures 5.11 and 5.19, set N, N^o, s, f and k according to the current configuration of open bins and items seen in the lookahead. Solve the resulting IP formulation in Figure 5.20. If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the obtained assignment.

- OptimalBounded$'_l$ (Optb$'_l$):
  Item permutations allowed
  In Figure 5.11, set N, N^o, s and f according to the current bin configuration and items seen in the lookahead. Solve the IP formulation in Figure 5.12 after extending it with expressions from Figure 5.13 as follows: Replace Objective Function (5.1) by (5.6) and include Constraints (5.7) to (5.11). If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the obtained assignment.
  Item permutations forbidden
  In Figures 5.11 and 5.19, set N, N^o, s, f and k according to the current configuration of open bins and items seen in the lookahead. Solve the IP formulation in Figure 5.20 after extending it with expressions from Figure 5.13 as follows: Replace Objective

Sets

T	set of time instants with $T = \{1, \dots, N\}$

Parameters

k	maximum number of open bins at a time

Variables

$$x_{ijt} = \begin{cases} 1 & \text{if item } i \in I \text{ is put in bin } j \in J \text{ at time } t \in T, \\ 0 & \text{else} \end{cases}$$

$$x^o_{ij^ot} = \begin{cases} 1 & \text{if item } i \in I \text{ is put in already used bin } j^o \in J^o \text{ at time } t \in T, \\ 0 & \text{else} \end{cases}$$

$$y_{jt} = \begin{cases} 1 & \text{if bin } j \in J \text{ is open at time } t \in T, \\ 0 & \text{else} \end{cases}$$

$$y^o_{j^ot} = \begin{cases} 1 & \text{if already used bin } j^o \in J^o \text{ is still open at time } t \in T, \\ 0 & \text{else} \end{cases}$$

$$z^{open}_{jt} = \begin{cases} 1 & \text{if bin } j \in J \text{ is opened at time } t \in T, \\ 0 & \text{else} \end{cases}$$

$$z^{close}_{jt} = \begin{cases} 1 & \text{if bin } j \in J \text{ is closed at time } t \in T, \\ 0 & \text{else} \end{cases}$$

$$z^{o,close}_{j^ot} = \begin{cases} 1 & \text{if already used bin } j^o \in J^o \text{ is closed at time } t \in T, \\ 0 & \text{else} \end{cases}$$

Figure 5.19: Additional sets, parameters and variables in the IP formulation of the bounded-space bin packing problem when item permutations are forbidden.

Function (5.12) by (5.6) and include Constraints (5.7) to (5.11). If the item to be packed is put in a new bin, close the fullest open bin if k bins are open, open a new bin and put the item in; otherwise put the item in the bin from the obtained assignment.

Optimal offline algorithm

- OptimalBounded (Optb):
  Item permutations allowed
  In Figure 5.11, set $N := n$, $N^o := 0$, $f := 0$ and s according to all items seen. Solve the resulting IP formulation in Figure 5.12. Pack the items according to the obtained assignment bin after bin.

$$
\begin{array}{llcll}
\min & \displaystyle\sum_{j\in J} y_j + |J^o| & & & (5.12)\\
\text{s.t.} & \displaystyle\sum_{j\in J}\sum_{t\in T} x_{ijt} + \sum_{j^o\in J^o}\sum_{t\in T} x^o_{ij^ot} & = & 1 & i\in I \quad (5.13)\\
& \displaystyle\sum_{i\in I} s_i x_{ij} & \le & y_j & j\in J \quad (5.14)\\
& \displaystyle\sum_{i\in I} s_i x^o_{ij^o} & \le & 1-f_{j^o} & j^o\in J^o \quad (5.15)\\
& x_{ijt} & \le & x_{ij} & i\in I,\ j\in J,\ t\in T \quad (5.16)\\
& x^o_{ij^ot} & \le & x^o_{ij^o} & i\in I,\ j^o\in J^o,\ t\in T \quad (5.17)\\
& x_{ijt} & \le & y_{jt} & i\in I,\ j\in J,\ t\in T \quad (5.18)\\
& x^o_{ij^ot} & \le & y^o_{j^ot} & i\in I,\ j^o\in J^o,\ t\in T \quad (5.19)\\
& y_{jt} & \le & y_j & j\in J,\ t\in T \quad (5.20)\\
& \displaystyle\sum_{j\in J} y_{jt} + \sum_{j^o\in J^o} y^o_{j^ot} & \le & k & t\in T \quad (5.21)\\
& \displaystyle\sum_{j\in J} z^{open}_{jt} & \le & 1 & t\in T \quad (5.22)\\
& \displaystyle\sum_{t\in T} z^{open}_{jt} & \le & y_j & j\in J \quad (5.23)\\
& \displaystyle\sum_{t\in T} z^{close}_{jt} & \le & y_j & j\in J \quad (5.24)\\
& \displaystyle\sum_{t\in T} z^{o,close}_{j^ot} & \le & 1 & j^o\in J^o \quad (5.25)\\
& \displaystyle\sum_{t'=1}^{t}(z^{open}_{jt'} - z^{close}_{jt'}) & \le & y_{jt} & j\in J,\ t\in T \quad (5.26)\\
& \displaystyle 1-\sum_{t'=1}^{t} z^{o,close}_{jt'} & \le & y^o_{j^ot} & j^o\in J^o,\ t\in T \quad (5.27)\\
& \displaystyle\sum_{t'=1}^{t}(z^{open}_{jt'} - z^{close}_{jt'}) & \ge & x_{ijt} & i\in I,\ j\in J,\ t\in T \quad (5.28)\\
& \displaystyle 1-\sum_{t'=1}^{t} z^{o,close}_{jt'} & \ge & x^o_{ij^ot} & i\in I,\ j^o\in J,\ t\in T \quad (5.29)\\
& x_{ij}, x^o_{ij^o}, y_j & \in & \{0,1\} & i\in I,\ j\in J,\ j^o\in J^o \quad (5.30)
\end{array}
$$

Figure 5.20: IP formulation of the bounded-space bin packing problem when item permutations are forbidden.

Item permutations forbidden
In Figures 5.11 and 5.19, set $N := n$, $N^o := 0$, $f := 0$, k and s according to all items seen. Solve the resulting IP formulation in Figure 5.20. Pack the items according to the obtained assignment item after item.

In the IP formulation for forbidden item permutations in Figure 5.20, we have to account for the time dimension to make sure that an item is put into a bin that is open at packing time: In addition to $x_{ij}, x^o_{ij^o}$ and $y_j, y^o_{j^o}$, we introduce decision variables $x_{ijt}, x^o_{ij^ot}$ and $y_{jt}, y^o_{j^ot}$ where t refers to the time instant. Constraints 5.13 to 5.21 ensure that each item is packed in a bin, that each bin capacity is not exceeded and that at each time at most k bins are open. To facilitate the operations of openening and closing a bin at time t, we introduce variables z^{open}_{jt}, z^{close}_{jt} and $z^{o,close}_{j^ot}$. Constraints 5.22 to 5.25 take care that at each time and for each bin only one respective operation is carried out. Constraints 5.26 to 5.29 guarantee that opening and closing operations are consistent with bin statuses and packing operations.

5.3.2.1 Average Results

In Figures 5.21 and 5.22, the average algorithm behavior in the cases of allowed and forbidden permutations shows that although we are restricted to keep at most $k = 3$ bins open at a time there is no significant surplus of lookahead as opposed to classical bin packing. Our first intuition was that being forced to close a bin upon opening a new one would lead to a higher value of information since for late items not all but only the currently open bins are ready to accept an item such that careful decision making should pay off. While this is probable to be the reason for the slight boost in the lookahead effect under allowed item permutations, computational results show that the magnitude of this effect is rather small.

In classical bin packing, maximum improvements between online and offline situations were between 3.6 % and 4.7 % irrespective of item permutations. Now, in bounded-space bin packing, the improvement increases to levels between 4.6 % and 5.6 % under allowed permutations; under forbidden permutations, the benefit of lookahead drops to at most 3.7 % due to the additional restriction on the number of open bins. For $l = n = 25$, the optimal number of bins coincides with that of the classical problem for allowed item permutations. In case of forbidden item permutations, this value augments to 14.41, while in the unbounded case it was 14.12. The average number of bins also slightly increases under forbidden item permutations for small lookahead levels $l < n = 25$. Hence, we judge the delimiting influence of the bounded-space feature in bin packing as rather modest. Confidence intervals, coefficients of variation and objective deterioration under additional lookahead have analogous dimensions as in classical bin packing (cf. Tables A.25 and A.26 of Appendix A.2.3).

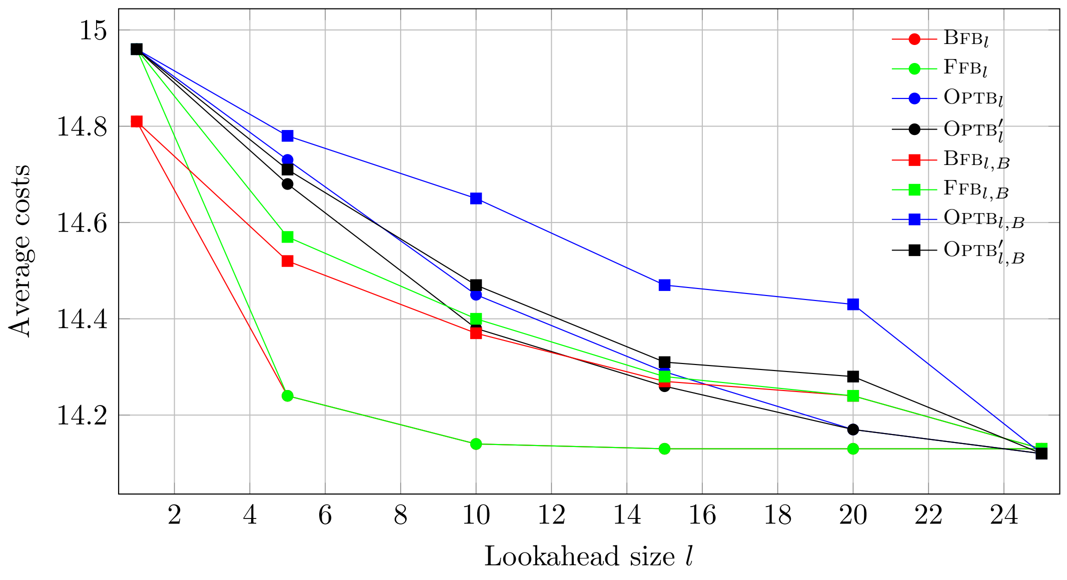


Figure 5.21: Average costs for different lookahead sizes and $n = 25$ in the bounded-space bin packing problem when item permutations are allowed.

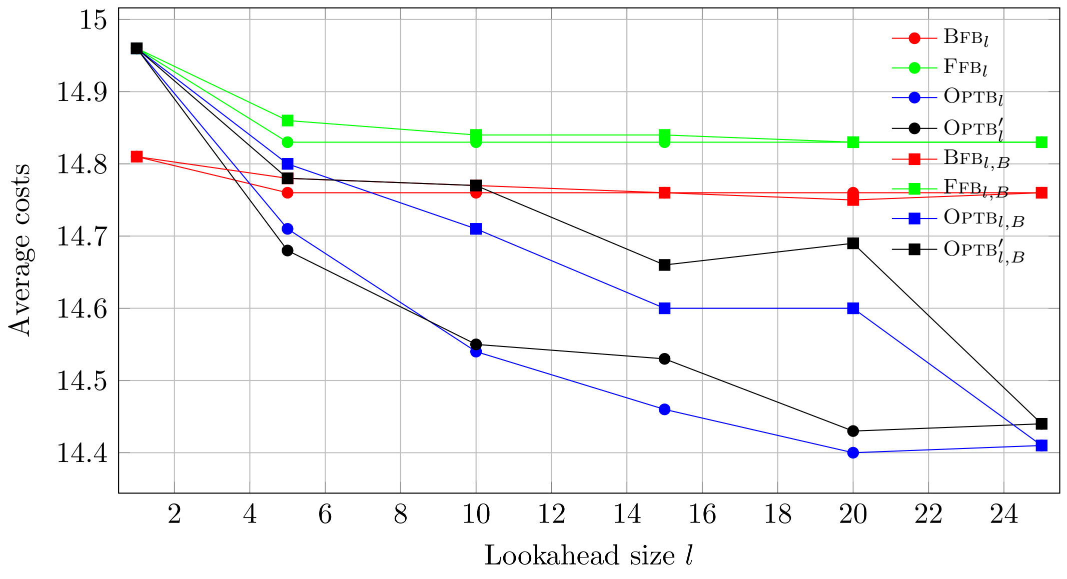


Figure 5.22: Average costs for different lookahead sizes and $n = 25$ in the bounded-space bin packing problem when item permutations are forbidden.

We recognize that because of closing restrictions the degrees of freedom of an algorithm while packing an item are pruned to a large extent such that additional information is hard to process for achieving guaranteed improvement due to lookahead. The improvement cutoff of the rule-based algorithms in case of forbidden item permutations is due to the restriction to $\min\{k, l\}$ lookahead units in order to ensure feasibility. The deviation between OPTB_{25} and OPTB'_{25} as well as between $\text{OPTB}'_{25,B}$ and $\text{OPTB}'_{25,B}$ stems from some few of the $m = 1000$

instances whose IP formulation in Figure 5.20 could not be solved to optimality within the prescribed time limit of 120 seconds.

We note that for item sequences of length $n = 100$, the improvement as a result of full lookahead amounts to values between 7.9 % and 9.1 % depending on the algorithm used when item permutations are allowed. Because we only consider the rule-based algorithms which are restricted to $\min\{k, l\}$ lookahead units for $n = 100$, improvement due to lookahead vanishes in the case of forbidden item permutations; for details, see Appendix A.2.3. We will restrict our discussion only to allowed item permutations from now on.

5.3.2.2 Distributional Results

The gap between the red and green curve in the plots of algorithm families BFB_l and FFB_l in Figure 5.23 implies that for the rule-based algorithms the first lookahead units are most valuable in terms of a reduction of the number of bins used. From Figure 5.21, we already knew that especially under these lookahead sizes the rule-based algorithms outperform the exact reoptimization strategies. Hence, we come to the overall recommendation to relinquish exact reoptimization methods in favor of stability-oriented rule-based algorithms in bounded-space bin packing. All plots in Figure 5.23 underline the small positive effect to be expected through additional lookahead by the ordering of the empirical counting distribution functions depending on the lookahead size l for each algorithm family. Under batched lookahead, the positive effect is acquired more evenly among additional lookahead units as opposed to the regular type of lookahead where the effect is mainly attributable to the first units. Confidence intervals of width smaller than 0.3 for all algorithms and lookahead levels in Tables A.25 and A.26 of Appendix A.2.3 provide sufficient statistical evidence for the validity of the observed algorithm performance on random input sequences. As seen from the percentage of item sequences with a deterioration between successive lookahead levels, exact reoptimization is more susceptible to misinterpretation of additional lookahead than rule-based heuristics are.

Figures 5.24 and 5.25 show the empirical counting distributions of the performance ratio relative to OPT and the online variants of the algorithms, respectively. Both from a qualitative and quantitative point of view, we draw the same conclusions as in classical bin packing:

- A large fraction of all item sequences already leads to an optimal number of bins even if no lookahead or few lookahead units are given.
- The maximum deviation from optimality is below 20 % for all algorithms.

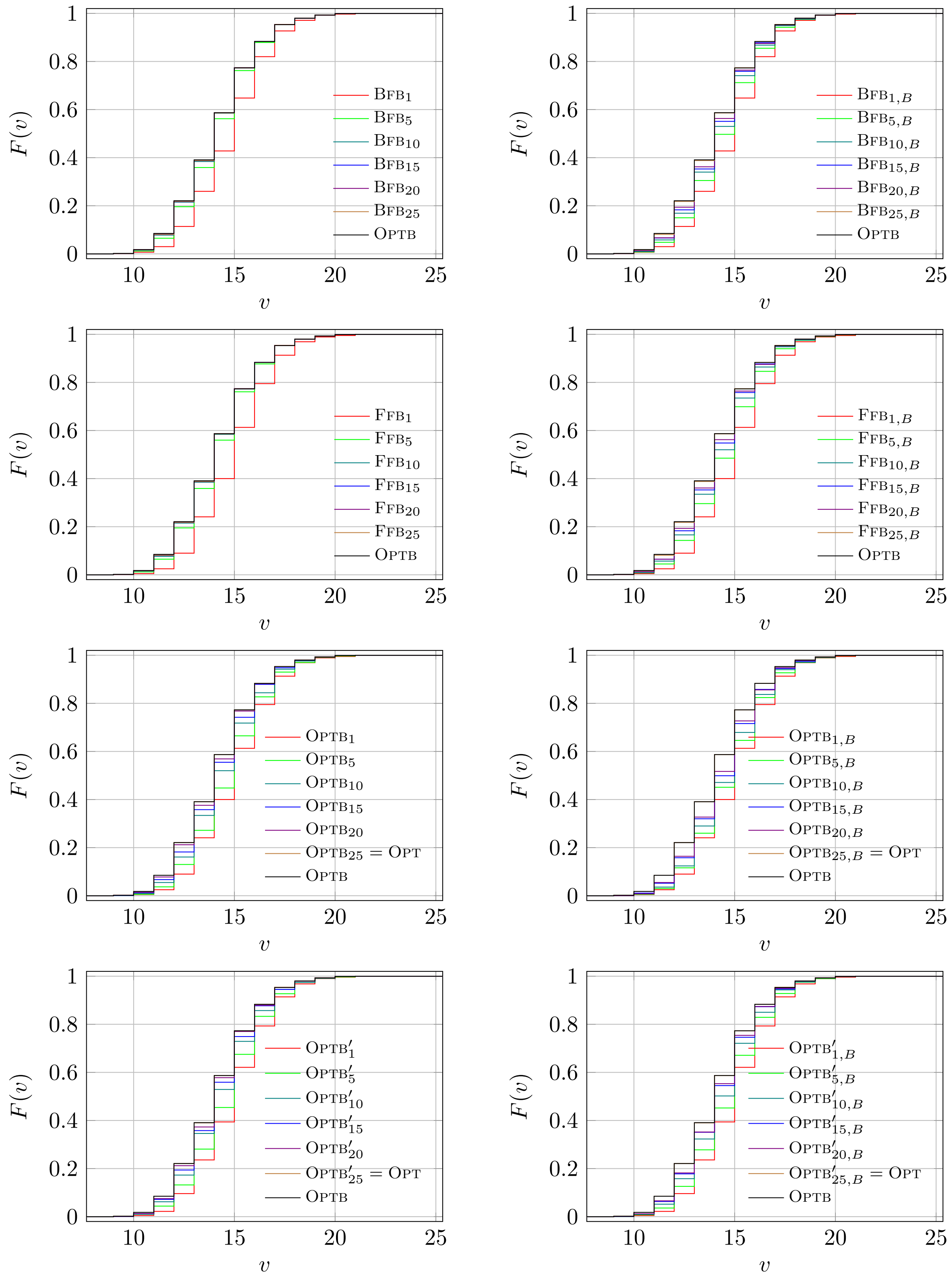


Figure 5.23: Empirical counting distribution functions of costs for $n = 25$ in the bounded-space bin packing problem when item permutations are allowed.

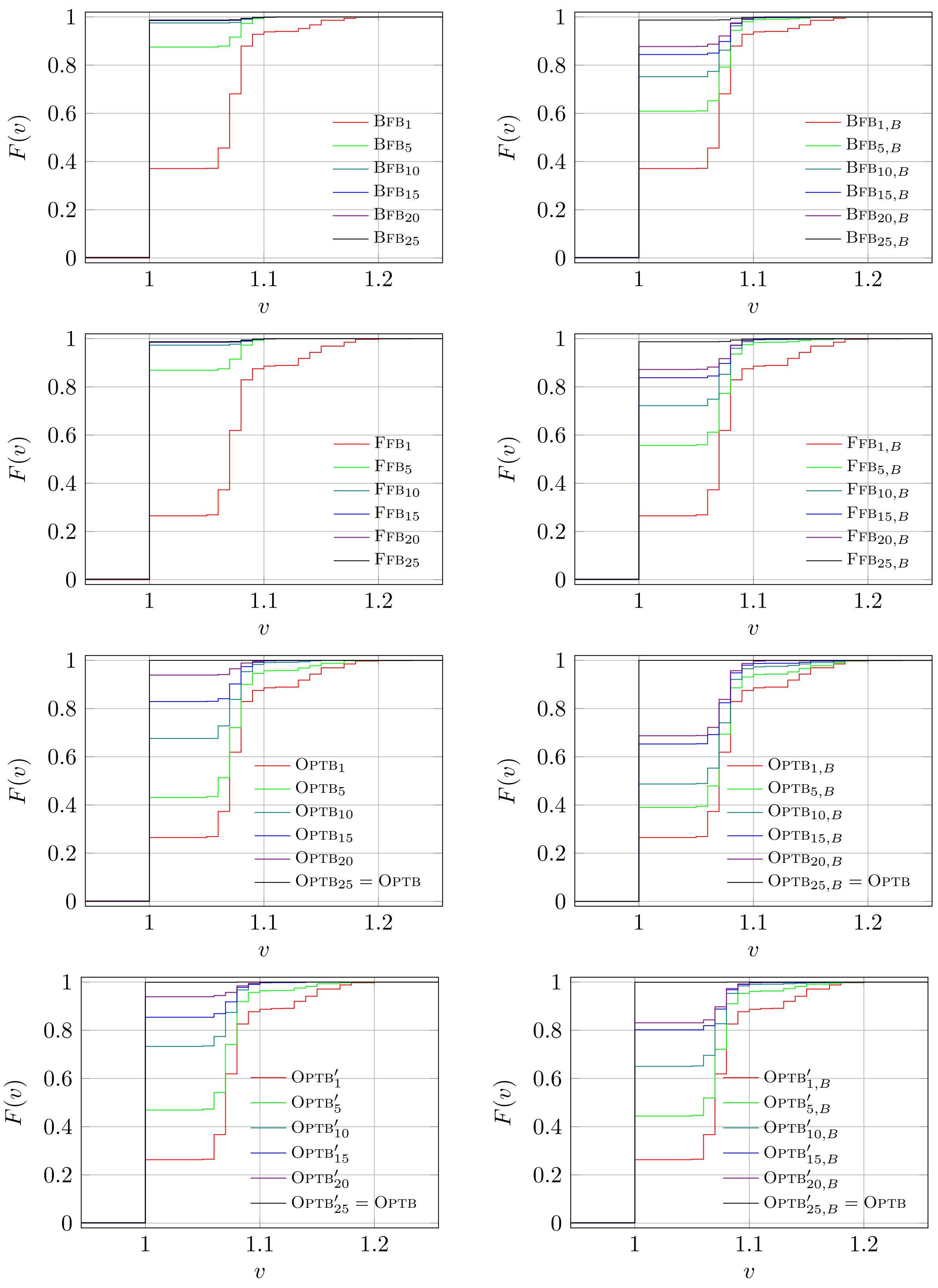


Figure 5.24: Empirical counting distribution functions of performance ratio of costs relative to OPT for $n = 25$ in the bounded-space bin packing problem when item permutations are allowed.

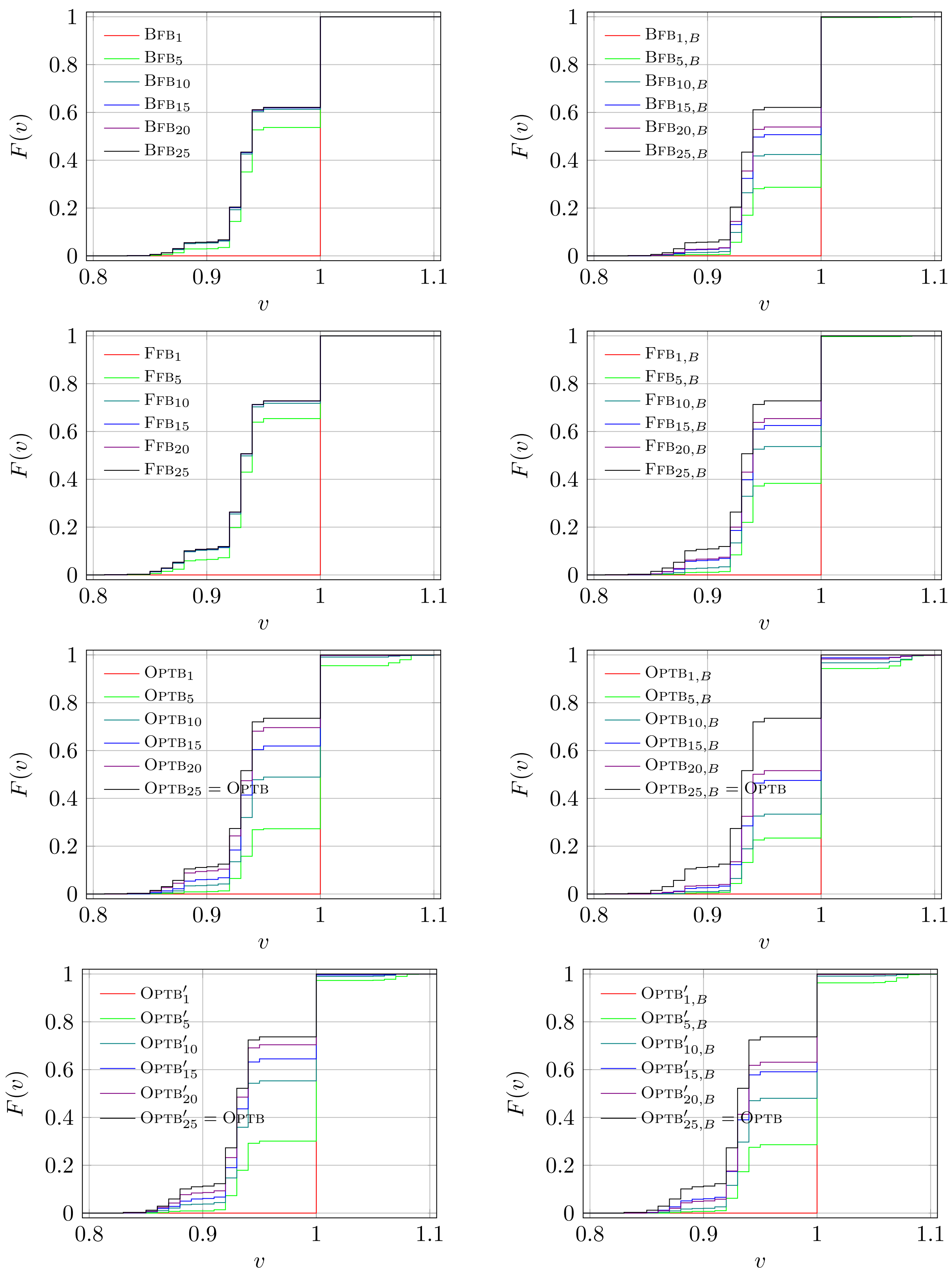


Figure 5.25: Empirical counting distribution functions of performance ratio of costs relative to the online version for $n = 25$ in the bounded-space bin packing problem when item permutations are allowed.

- Algorithms may interpret lookahead disadvantageously; however, this happens only very rarely.
- Most item sequences experience a lookahead effect of at most 10 %.

For a detailed statistical summary, see Tables A.27 to A.30 of Appendix A.2.3.

5.3.2.3 Markov Chain Results

Consider the online bounded-space bin packing problem with lookahead l and a maximum number k of open bins at a time under allowed item permutations. The elements of the state space $\mathcal{S}$ subsume the fill levels $f = (f_1, \dots, f_k)$ of the k open bins, the sizes $s = (s_1, \dots, s_l)$ of the lookahead items which are ordered non-increasingly and the number of bins v used so far, i.e.,

$$\mathcal{S} = \big\{(f, s, v) \mid f \in [0,1]^k, s \in (0,1]^l, v \in \mathbb{N}_0\big\}.$$

Upon arrival of a new item, each algorithm $\textsc{Alg}$ assigns the largest item in the current lookahead to one of the k open bins or to a new bin. In the latter case, an open bin is closed first and the objective value increases by one unit. Hence, $\textsc{Alg}$ is a function

$$\textsc{Alg} : \mathcal{S} \times (0,1] \to \{1, 2, \dots, k+1\}$$

which gives the bin into which the largest item will be put where $\textsc{Alg}(s, s_{new}) = k+1$ means that a new bin needs to be opened. The successor state of $s = (f, s, v)$ upon arrival of an item with size s_{new} is given by $s' = (f', s', v')$ where f' arises from f by adding s_1 to $f_{\textsc{Alg}(s,s_{new})}$ if $\textsc{Alg}(s, s_{new}) \neq k+1$ or replacing the largest component of f with s_1 if $\textsc{Alg}(s, s_{new}) = k+1$, s' arises from s by removing s_1, appending s_{new} and ordering non-increasingly, and

$$v' = \begin{cases} v & \text{if } \textsc{Alg}(s, s_{new}) \in \{1, \dots, k\}, \\ v+1 & \text{otherwise.} \end{cases}$$

In order to obtain a finite state space representation, we have to discretize item sizes: To this end, denote the number of possible item sizes by n_s and the resulting number of item size combinations with total size not larger than 1 by n_c, then the state space size amounts to $|\mathcal{S}| = n_c^k \cdot \binom{n_s+l-1}{l} \cdot (n+1)$[14].

[14] The number of combinations with repetition where n is the number of elements to choose from and r elements have to be chosen is $\binom{n+r-1}{r}$ ([137]).

For $\text{ALG} \in \{\text{BFB}_l, \text{FFB}_l, \text{OPTB}_l\}$ and selected lookahead level l, we determine the successor for each state and each new item to obtain the one-step frequency matrix and the n-step frequency matrix of the Markov chain.

In our numerical experiment, we use the parametrization $n = 25$, $k = 3$, $l \in \{1, 2, \ldots, 5\}$ and restrict item sizes to $\{0.4, 0.5, 0.6\}$. Hence, we have $n_s = 3$ and $n_c = |\{0.4, 0.5, 0.6, 0.4 + 0.4, 0.4 + 0.5, 0.4 + 0.6\}| = 6$. For this small exemplary setting, the state space size amounts to $6^3 \cdot \binom{3+5-1}{5} \cdot 26 = 117\,936$ elements for $l = 5$.

The resulting exact distribution functions in Figure 5.26 are affirmative to the main result found for bin packing, namely that lookahead has a benefit, albeit a rather small one.

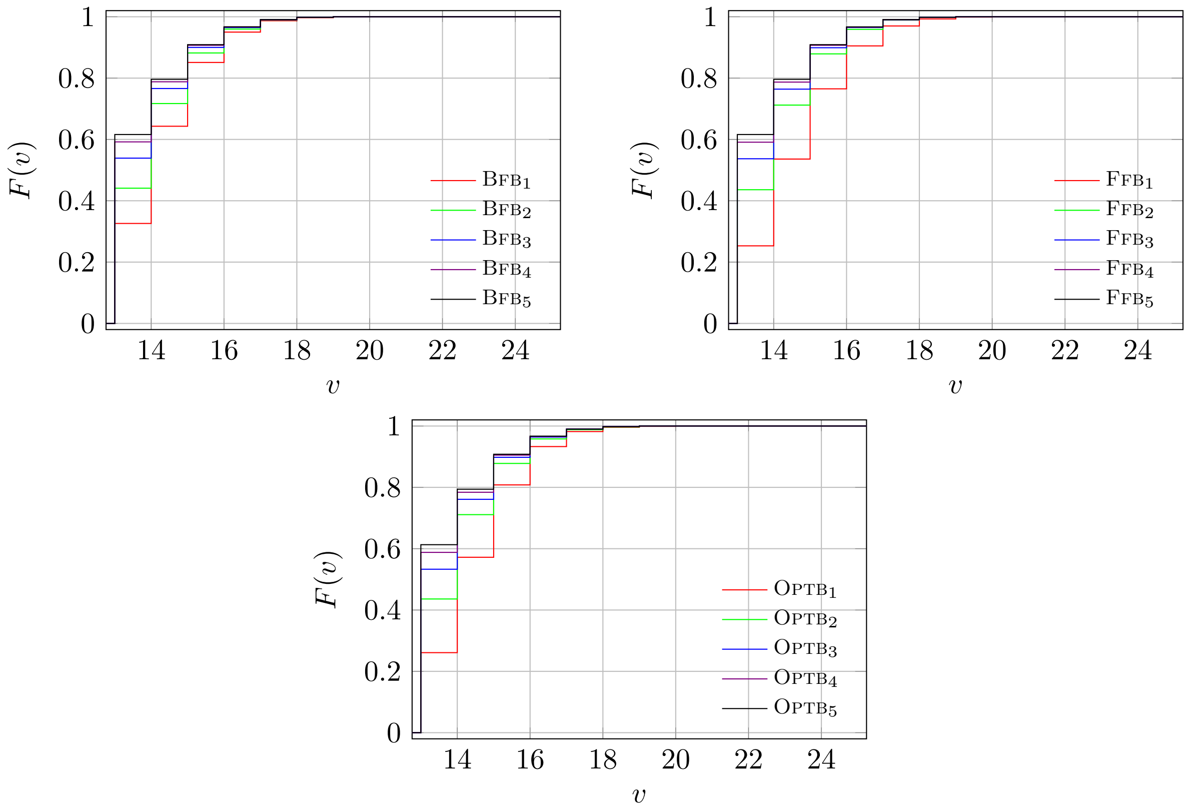


Figure 5.26: Exact distribution functions of costs for $n = 25$ in the bounded-space bin packing problem.

Moreover, we see that already in the case of no or few lookahead units the number of item sequences which lead to the smallest objective value of 13 is consistently larger than 25 % of the number of all item sequences. We conclude that online bin packing algorithms have quite a good chance of obtaining an optimal number of bins and that maximum deviations from optimality as suggested by (asymptotic) competitive ratios in both types of bin packing problems of 70 % ([55]) are quite untypical.

We conclude this section by pointing out that the results of the computational experiments on the bin packing problem are in line with the minor positive effect of lookahead as determined by exact analysis in Chapter 4.2 for a basic setting with two item sizes and $l = 2$. Hence, significant savings in the number of bins used can also not be expected for arbitrary item sizes. Nonetheless, the impact of lookahead is slightly stronger as a result of the increased number of item combinations which lead to a saving of a bin.

5.4 Online Traveling Salesman Problem with Lookahead

A specification of the online traveling salesman problem (TSP) with lookahead and algorithms FirstComeFirstServed and NearestNeighbor$_l$ is given in Chapter 4.3. Further algorithms for the online version with lookahead will be introduced in the sequel. For fixed $n \in \mathbb{N}$ and metric space $(\mathcal{M}, d)$, the set of all input sequences of length n is given by

$$\Sigma_n = \big\{(\sigma_1, \sigma_2, \ldots, \sigma_n) \,|\, \sigma_i := x_i \in \mathcal{M}, i = 1, \ldots, n\big\}$$

and comprises all request sequences of length n where σ_i is identified with point $x_i \in \mathcal{M}$ to be visited. Because in an optimal solution to an instance of the offline problem, all points will be approached exactly once, all σ_i are assumed pairwise different in the offline TSP. In the dynamic setting, of course, we have to allow multiple requests on the same location. In the online version, only σ_i with $i = 1, \ldots, n$ is known when σ_i has to be visited. In the online version with lookahead of size l, when for the ith time it has to be decided which location to be visited, the l unvisited locations from $\sigma_1, \sigma_2, \ldots, \sigma_{i+l-1}$ with $i = 1, \ldots, n$ are known if $i + l - 1 \leq n$, otherwise the $n - i + 1$ unvisited locations from $\sigma_1, \sigma_2, \ldots, \sigma_n$ are known[15]. In the offline version, all locations are known at the beginning.

Computational experiments feature two problem settings each of which underlies $m = 1000$ independently drawn request sequences. In the first setting, each sequence consists of $n = 25$ requests, while the second setting has $n = 100$ requests per sequence. Requests are located in the planar unit square $\mathcal{M} = [0, 1] \times [0, 1] \subset \mathbb{R}^2$ and distance is measured by the Euclidean metric. A set of requests establishes a complete graph consisting of vertices labeled with the requests as well as additional vertices for the current server location and the origin; edge weights correspond to distances between respective locations. For $n = 25$, lookahead sizes are $l \in \{1, 5, 10, \ldots, 25\}$; for $n = 100$, we have $l \in \{1, 5, 10, 20, 40, \ldots, 100\}$. We discuss results for $n = 25$ and refer the reader to Appendix A.2.4 for $n = 100$.

[15] In Chapter 6, we take into account time lookahead for the pickup and delivery problem with time windows – a problem closely related to the TSP.

A decision by an algorithm is required when the server starts initially or a request is reached; it consists of selecting the request to be visited next from the set of known requests. Note that any online algorithm without lookahead is trivial since it only sees the current request and has to visit it, i.e., it has no degrees of freedom at all.

Online algorithm

- FirstComeFirstServed (Fcfs): Choose the only unvisited known request next.

Online algorithms with lookahead of size l

- NearestNeighbor$_l$ (Nn$_l$): From the requests in the lookahead, choose a request closest to the server's current location next (see also [120]).
- Insertion$_l$ (Ins$_l$): Construct a Hamiltonian path H visiting each of the requests in the lookahead starting in the server's current location and ending in the origin as follows:
    1. At the beginning, H consists of the invariant starting and ending point only.
    2. Insert a request whose distance to the starting point is largest possible into H.
    3. Successively insert requests by choosing in each iteration a request whose smallest distance to a request in H is largest and insert it at a best possible position in terms of a smallest tour length increase.

  Choose the first request from H following the starting point next (see also [120]).
- 2Opt$_l$: Construct a Hamiltonian path H visiting each of the requests in the lookahead starting in the server's current location and ending in the origin as follows:
    1. Obtain H initially by applying Nn$_l$ or Ins$_l$.
    2. Choose two requests from H and reverse the order of requests between them to obtain H'.
    3. If H' is shorter than H, set $H := H'$.
    4. Return to step 2 until no further improvement is possible.

  Choose the first request from H following the starting point next (see also [120]).
- 3Opt$_l$: Construct a Hamiltonian path H visiting each of the requests in the lookahead starting in the server's current location and ending in the origin as follows:
    1. Obtain H initially by applying Nn$_l$ or Ins$_l$.

2. Choose three edges from H such that neither of them is incident to the starting or ending point and reorganize H as follows:
   - Form three new edges using the requests incident to the three edges.
   - Choose an order for the three new edges.
   - Adjust the order of the requests between the edges such that a feasible Hamiltonian path H' starting in the server's current location and ending in the origin is obtained if possible at all.
   - If H' is feasible and shorter than H, set $H := H'$.
3. Return to step 2 until no further improvement is possible.

Choose the first request from H following the starting point next (see also [120]).

- SimulatedAnnealing$_l$ (SA$_l$): Construct a Hamiltonian path H visiting each of the requests in the lookahead starting in the server's current location and ending in the origin as follows:
  1. Obtain H initially by applying NN$_l$ or Ins$_l$.
  2. Select initial temperature $T_0 \in \mathbb{R}$, minimum temperature $T_{min} \in \mathbb{R}$ with $T_{min} < T_0$, maximum number of iterations $L_{max} \in \mathbb{N}$ with unchanged temperature and set $T := T_0$, $L := 0$.
  3. If $L = L_{max}$, set $T := 0.9 \cdot T$ and $L := 1$; else set $L := L + 1$.
  4. Choose two requests from H and reverse the order of requests between them to obtain H'.
  5. If H' is shorter than H, set $H := H'$; else set $H := H'$ with probability $\exp(-\frac{\Delta}{T})$ where Δ is the difference between the length of H' and the length of H.
  6. Return to step 3 until $T < T_{min}$.

  Choose the first request following the starting point from the shortest Hamiltonian path obtained throughout the procedure next (see also [120]).

- TabuSearch$_l$ (Ts$_l$): Construct a Hamiltonian path H visiting each of the lookahead points starting in the server's current location and ending in the origin as follows:
  1. Obtain H initially by applying NN$_l$ or Ins$_l$.
  2. Select tabu time $T \in \mathbb{N}$, maximum number D_{max} of diversifications, number of swaps s per diversification and set $D := 0$.

3. Choose two requests from H and swap them to obtain H'.
4. If H' is shorter than H and the swapped request pair is not included in the tabu list, set $H := H'$ and add the swapped request pair to the tabu list with remaining tabu time T; else if H' is shorter than the best Hamiltonian path obtained so far and the swapped request pair is tabu, set $H := H'$ (aspiration).
5. For all tabu list entries except the new one, decrease the remaining tabu time by one iteration; return to step 3 until all pairs of points have been examined.
6. Perform a swap of two random points in H' for s times (diversification), set $H := H'$, $D := D + 1$.
7. Return to step 3 until $D \geq D_{max}$.

Choose the first request following the starting point from the shortest Hamiltonian path obtained throughout the procedure next (see also [79]).

Sets

J	set of requests with $J = \{1, \dots, N\}$

Parameters

N	number of requests
c_{ij}	distance between requests $i, j \in J$
c_j^{start}	distance of request $j \in J$ to current server position
c_j^{end}	distance of request $j \in J$ to origin
M	sufficiently large constant (big M)

Variables

$$x_{ij} = \begin{cases} 1 & \text{if request } i \in J \text{ immediately precedes request } j \in J, \\ 0 & \text{else} \end{cases}$$

$$x_j^{start} = \begin{cases} 1 & \text{if request } j \in J \text{ is visited first,} \\ 0 & \text{else} \end{cases}$$

$$x_j^{end} = \begin{cases} 1 & \text{if request } j \in J \text{ is visited last,} \\ 0 & \text{else} \end{cases}$$

$T_j \geq 0$	start time of request $j \in J$

Figure 5.27: Sets, parameters and variables in the MIP formulation of the Hamiltonian path problem with fixed starting and ending point.

- Optimal$_l$ (Opt$_l$): In Figure 5.27, set N, c, c^{start} and c^{end} according to the distances between current lookahead requests to each other and to the current server location. Solve the MIP formulation in Figure 5.28. Choose the first request following the starting point in the obtained solution next.

$$\min \quad \sum_{i\in J}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} c_j^{start}x_j^{start} + \sum_{j\in J} c_j^{end}x_j^{end} \tag{5.31}$$

$$\text{s.t.} \quad \sum_{\substack{j\in J,\\ j\neq i}} x_{ij} + x_i^{end} = 1 \qquad i \in J \tag{5.32}$$

$$\sum_{\substack{i\in J,\\ i\neq j}} x_{ij} + x_j^{start} = 1 \qquad j \in J \tag{5.33}$$

$$\sum_{j\in J} x_j^{start} = 1 \tag{5.34}$$

$$\sum_{j\in J} x_j^{end} = 1 \tag{5.35}$$

$$T_i + 1 \leq T_j + \mathrm{M}\cdot(1 - x_{ij}) \qquad i,j \in J \tag{5.36}$$

$$x_{ij}, x_j^{start}, x_j^{end} \in \{0,1\} \qquad i,j \in J \tag{5.37}$$

$$T_j \geq 0 \qquad j \in J \tag{5.38}$$

Figure 5.28: MIP formulation of the Hamiltonian path problem with fixed starting and ending point.

Optimal offline algorithm

- Optimal (Opt): In Figure 5.27, set N, c, c^{start} and c^{end} according to the distances between all requests to each other and to the initial server location. Solve the MIP formulation in Figure 5.28. Visit the requests according to the obtained solution.

Note that all algorithms with lookahead collapse into Fcfs for $l = 1$. There are plenty of IP and MIP formulations for the TSP (see, e.g., [120]); in the MIP formulation in Figure 5.28, we decide to get rid of the subtour elimination constraints by establishing precedence relations in Constraint 5.36 between requests based on decision variables for time instants at which requests are served ([132]).

Similar to the previous problems, we check whether regular request lookahead of size l leads to significantly better algorithm behavior as opposed to batched provision of lookahead units. If algorithms operate under batched lookahead, we indicate them with an added suffix B in the algorithm name.

5.4.1 Average Results

As Figure 5.29 shows for four requests with origin in request 1, providing an algorithm with only one additional unit of lookahead already leads to an order readjustment which avoids frequent detours. We recognize that – in contrast to bin packing – the objective value is immediately and heavily affected by the selected order of input element processing. We conclude that there should be a substantial effect of lookahead which results from visiting requests in an order different from their release order, i.e., from a rule set substitution.

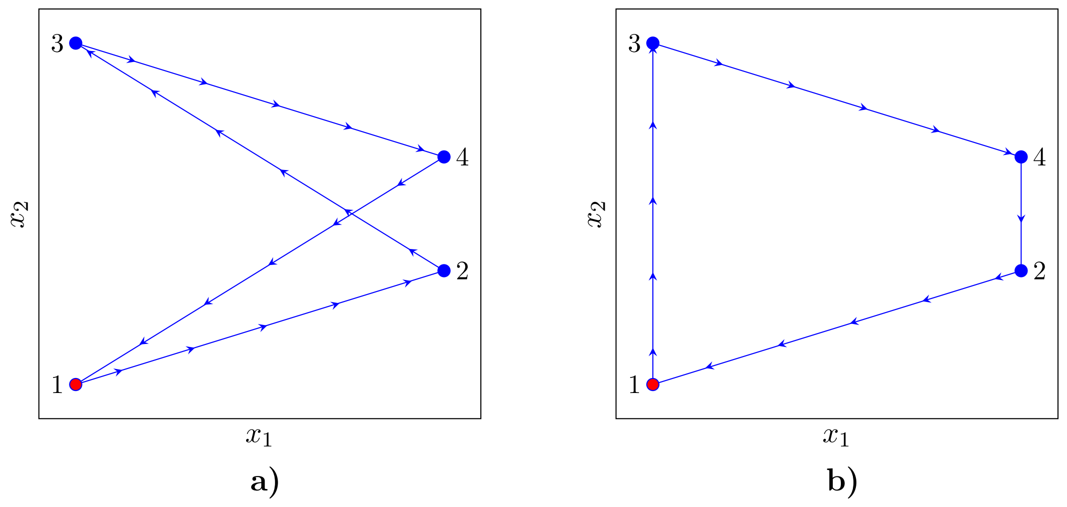


Figure 5.29: Comparison of routes obtained in the TSP. **a)** Without lookahead ($l = 1$). **b)** With lookahead of only one additional request ($l = 2$).

Figure 5.30 confirms the huge benefit attainable by providing the server with lookahead in form of a preview of future requests and allowing it to visit known requests in arbitrary order. Comparing the online with the offline case, reductions in the overall tour length of 60.5 % to 67.8 % are achieved depending on the algorithm used. Lookahead positively affects all algorithms in the same order of magnitude. Confidence interval widths smaller than 0.2 for all algorithms and lookahead levels as well as comparatively small coefficients of variation in Table A.37 of Appendix A.2.4 support the statistical validity of the results on lookahead improvements found over the randomly chosen input instances. The marginal benefit of an additional lookahead unit is strictly decreasing for all algorithms. In this sense, already provisioning an algorithm with small lookahead leads to considerable improvement. For instance, already for $l = 5$ overall tour lengths are reduced to between 57.5 % and 74.1 % of the online tour length depending on the algorithm used. Finally, we observe improved behavior of algorithms with regular request lookahead over those which have to adhere to batched lookahead as a result of the potential for tour length reduction that any additional lookahead unit has.

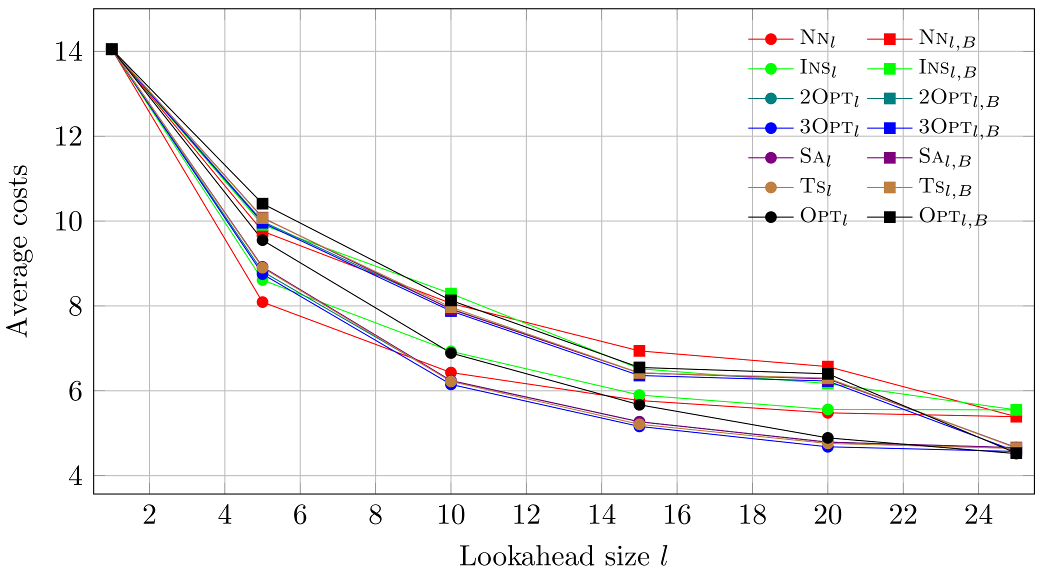


Figure 5.30: Average costs for different lookahead sizes and $n = 25$ in the TSP.

In a more detailed view, it is seen that for information regimes corresponding to small lookahead the simple algorithm NN$_l$ fares best; only for large lookahead sizes the more sophisticated algorithms exhibit superior performance. We conclude that a surplus of using refined algorithms rather than simple algorithms can only be realized when the overseen time horizon is as large as to guarantee that no severe deviations between previously calculated routes and recalculated routes upon additional request arrivals will occur. Clearly, for small lookahead this requirement is likely to be violated due to newly arriving requests rendering once calculated "locally optimal" plans obsolete.

We illustrate this result by contrasting NN$_l$ and OPT$_l$: NN$_l$ proves superior for small and medium lookahead sizes l because of its consistent behavior which results from always choosing the closest known request next, whereas OPT$_l$ may choose a request from a distant region next if suggested by an optimal solution of the snapshot problem. Hence, the danger of a zigzag route is increased for OPT$_l$. Consider Figure 5.31 with $n = 10$ requests where requests are labeled as $(1, 2, \ldots, 10)$ and also released in that order, current server position and origin in request 1, and lookahead $l = 2$. After arrival in request 2, the lookahead comprises requests 3 and 4 for both algorithms. OPT$_2$ (in Figure 5.31 b)) next chooses request 3 because route $(2, 3, 4, 1)$ is shorter than $(2, 4, 3, 1)$. Later, when request 6 is reached, the lookahead consists of requests 4 and 7 such that a distant region has to be revisited although the server had initially been there. Contrarily, NN$_2$ (in Figure 5.31 a)) avoids revisiting this region by choosing request 4 as successor of request 2. Hence, the probability for zigzagging is reduced under NN$_l$ compared to OPT$_l$, albeit returning to previously seen regions cannot be excluded in general because of future requests.

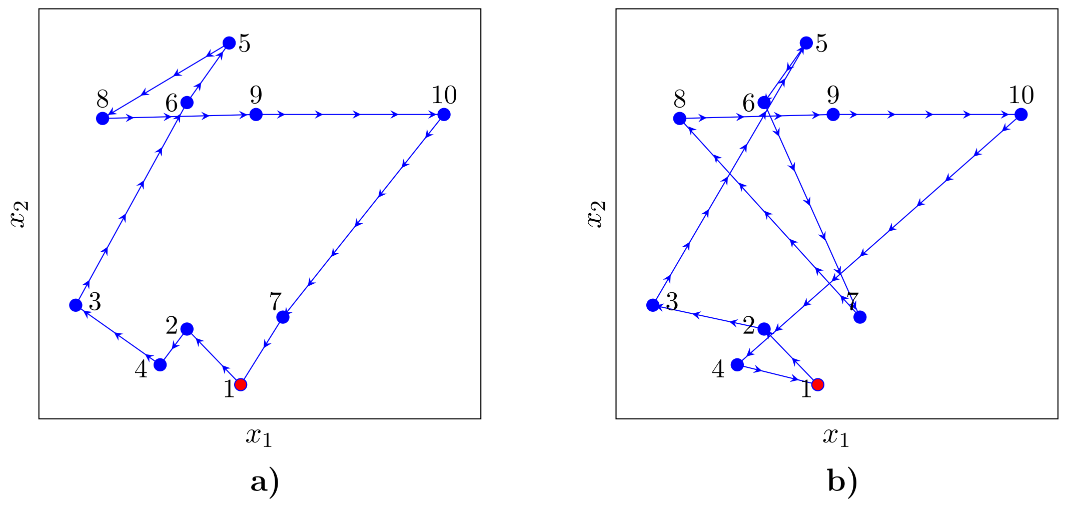


Figure 5.31: Threat of bad decisions in the TSP. **a)** Low risk for stability-oriented algorithm Nn_2. **b)** High risk for exact reoptimization algorithm Opt_2.

For $n = 100$ items per sequence, the same qualitative effects are observed: The simple algorithm Nn_l excels the sophisticated algorithms for small lookahead; for medium to large lookahead it is recommended to use the elaborated algorithms. The marginal benefit of additional lookahead is strictly decreasing. Comparing the online and offline situation, improvements are even higher – between 81.3 % and 84.6 % depending on the algorithm – than for $n = 25$; for details, see Appendix A.2.4.

5.4.2 Distributional Results

Empirical counting distribution functions of objective values and performance ratios as shown in Figures 5.32 to 5.34 appear clearly segregated from each other for successive lookahead levels of small to medium size with decreasing gap for increasing lookahead level. Moreover, confidence intervals both of objective values and performance ratios as listed in Tables A.37 to A.39 of Appendix A.2.4 are predominantly disjunct and have small width. Conjointly, these findings are affirmative both to the huge magnitude of lookahead and the decreasing marginal value of information.

From Figure 5.32, it is seen that the absolute value of tour lengths reduces drastically under lookahead. The reduction is as substantial as to cause the supports of the density functions related to the given plots not to come to an overlap for $l = 1$ and $l = 10$. Moreover, we recommend to exclude algorithm family Ins_l from further consideration by cause of poor behavior over all lookahead levels l as compared to Nn_l on small to medium lookahead levels and compared to the sophisticated algorithms on medium to large lookahead levels.

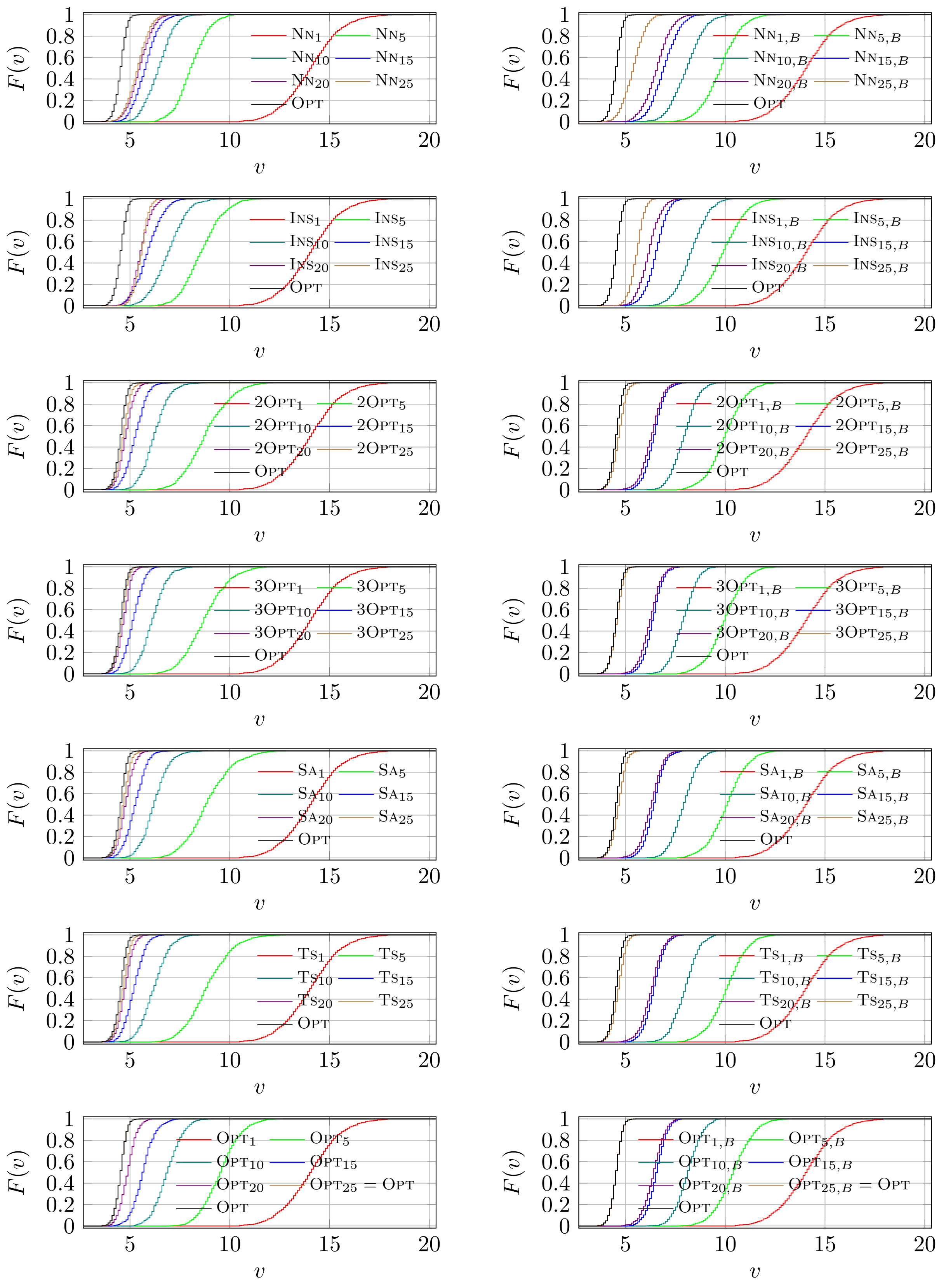


Figure 5.32: Empirical counting distribution functions of costs for $n = 25$ in the TSP.

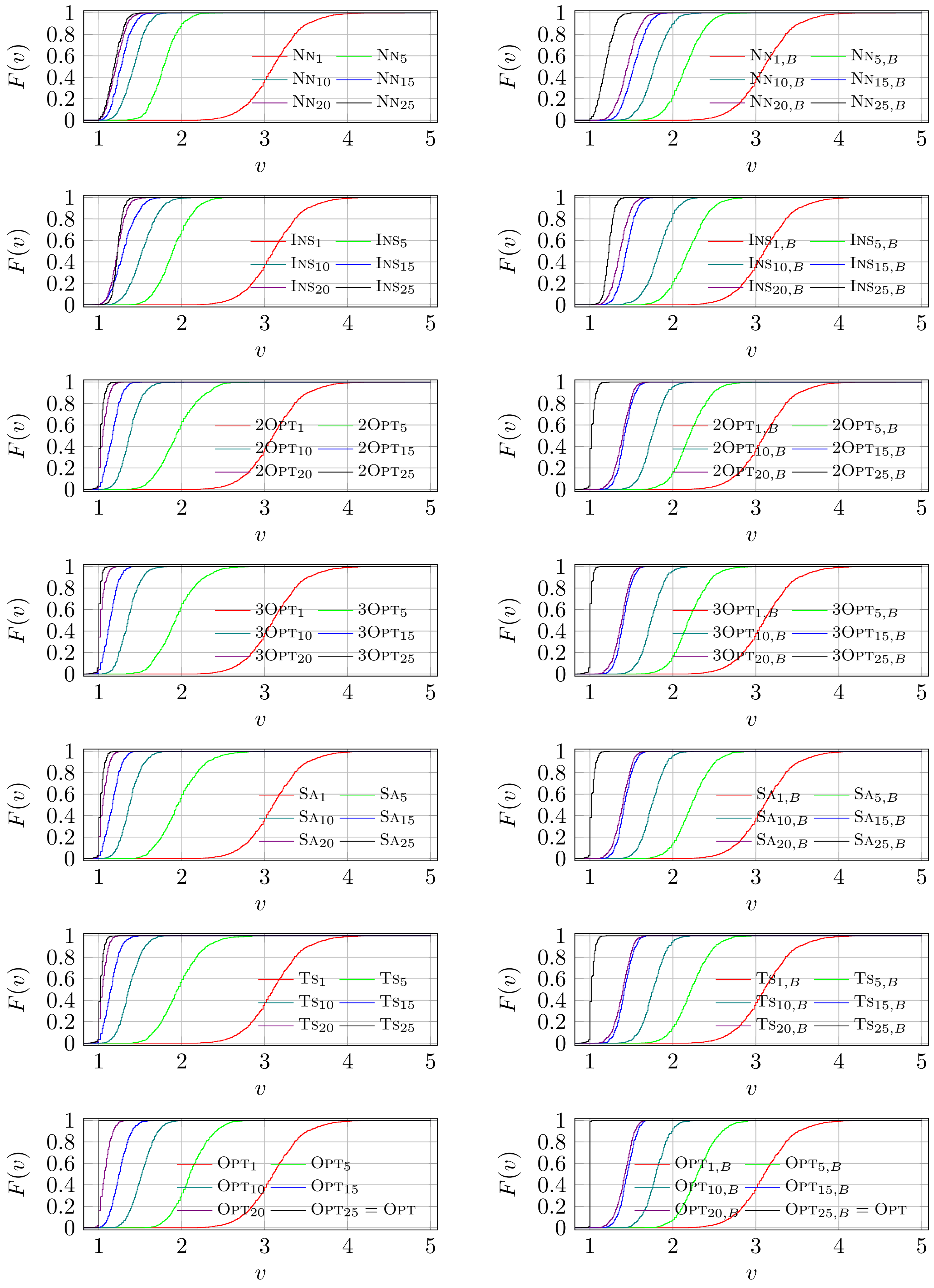


Figure 5.33: Empirical counting distribution functions of performance ratio of costs relative to OPT for $n = 25$ in the TSP.

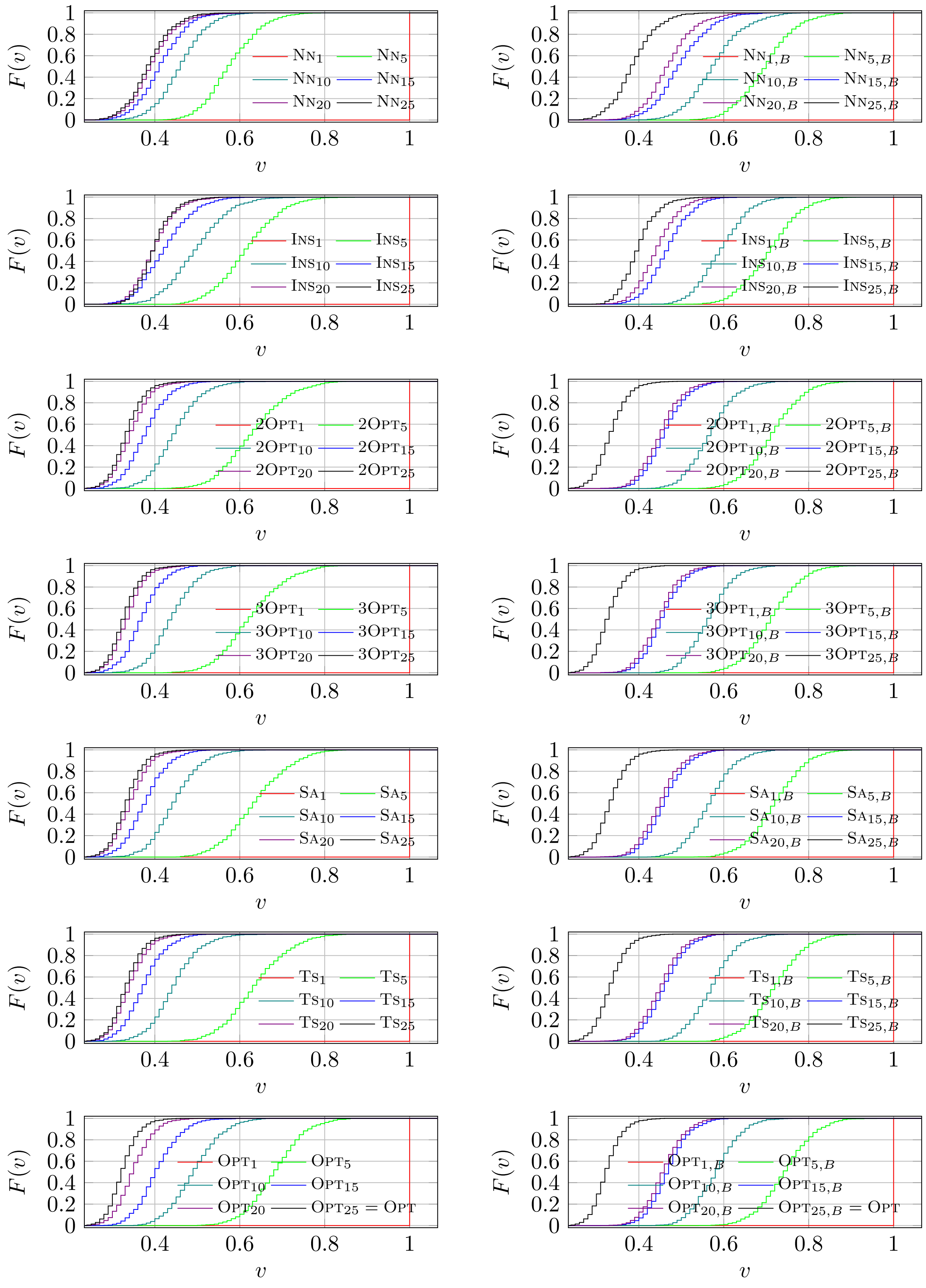


Figure 5.34: Empirical counting distribution functions of performance ratio of costs relative to the online version for $n = 25$ in the TSP.

Furthermore, exact reoptimization approach $\textsc{Opt}_l$ is outperformed by $2\textsc{Opt}_l$, $3\textsc{Opt}_l$, $\textsc{Sa}_l$ and $\textsc{Ts}_l$ for all lookahead levels l except for the state of full informational knowledge ($l = 25$). The latter algorithms are considered to have equal performance. However, from the percentage of request sequences with a deterioration between two successive lookahead levels in the last column of Table A.37 of Appendix A.2.4, we see that there are sporadic instances where additional lookahead was misleading for an algorithm.

While performance ratios relative to $\textsc{Opt}$ were below 1.25 for all instances of the bin packing problems, the picture remarkably changes in the TSP as seen in Figure 5.33. Here, performance ratios larger than 4.5 are encountered. However, already for $l = 10$ none of the algorithms endowed with regular request lookahead except for $\textsc{Ins}_l$ incurred a performance ratio larger than 2 on any instance which points towards the huge impact of the first lookahead requests and the decreasing marginal benefit of additional lookahead units. The proportion of instances which lead to a performance ratio of 1 is rather modest for all lookahead levels smaller than $l = n = 25$. Thus, each additional lookahead request tends to unfold its potential for tour length reduction over the course of requests revealed in a sequence. As can be seen from the minimum performance ratios and the percentage with performance ratio smaller than 1 in Table A.38 of Appendix A.2.4, not all of the $m = 1000$ instances could be solved to optimality within 120 seconds. Hence, in some few cases $\textsc{Opt}$ could be outperformed by some of the other algorithms.

The empirical counting distribution functions relative to the respective online versions of the algorithms in Figure 5.34 lead to the same conclusions that could be drawn from the previous figures. Note that no instance exists for which the provision of lookahead leads to a deterioration in the resulting tour length of any algorithm when compared to the online case without lookahead. Confidence intervals shrink to a single point when the bounds are rounded to two decimal places (cf. also Table A.39 of Appendix A.2.4).

Observe that for large lookahead all counting distribution functions are relatively steep in a characteristic interval which illustrates the homogenous positive effect of additional information on all kinds of request sequences.

5.4.3 Markov Chain Results

Assume all requests in $\mathcal{M}$ are given a name which allows to establish a lexicographical order among them. The elements of the state space $\mathcal{S}$ subsume the current server location x_s, the requests $x = (x_1, \dots, x_l)$ in the lookahead in lexicographical order of their name and the distance v traveled by the server so far, i.e.,

$$\mathcal{S} = \big\{(x_s, x, v) \,|\, x_s \in \mathcal{M}, x \in \mathcal{M}^l, v \in \mathbb{R}^{\geq 0}\big\}.$$

Upon arrival of a new request to be visited, each algorithm Alg decides which one of the requests in the lookahead is visited next. Hence, Alg is a function

$$\text{Alg} : \mathcal{S} \times \mathcal{M} \to \{1, \dots, l\}$$

which gives the position of the request to be visited next in the lookahead. The successor state of $s = (x_s, x, v)$ upon arrival of request x_{new} is given by $s' = (x_s', x', v')$ where $x_s' = x_{\text{Alg}(s,x_{new})}$, x' arises from x by removing element $x_{\text{Alg}(s,x_{new})}$, appending element x_{new} and ordering lexicographically, and $v' = v + d(x_s, x_{\text{Alg}(s,x_{new})})$ with distance function d on $\mathcal{M}$.

In order to obtain a finite state space representation, we have to ensure that $\mathcal{M}$ is a finite metric space and that the possible values for v form a finite set: To this end, denote the diameter of $\mathcal{M}$ by d_{max}, then for integer distances the state space size amounts to

$$|\mathcal{S}| = |\mathcal{M}| \cdot \binom{|\mathcal{M}| + l - 1}{l} \cdot \big((n+1) \cdot d_{max}\big).$$

For Alg $\in \{\text{Nn}_l, \text{Ins}_l, 2\text{Opt}_l, \text{Opt}_l\}$ and selected lookahead level l, we determine the successor for each state and each new request to obtain the one-step frequency matrix and the n-step frequency matrix of the Markov chain.

In our numerical experiment, we use the parametrization $n = 25$, $|\mathcal{M}| = 4$, $d_{max} = 3$ and $l \in \{1, 2, \dots, 5\}$. For this small exemplary setting, the state space size amounts to $4 \cdot \binom{4+5-1}{5} \cdot (26 \cdot 3) = 17\,472$ elements for $l = 5$.

Figure 5.35 displays the exact distribution functions of the objective values obtained for the four algorithm families. The overall picture shows that each additional lookahead unit corresponds to a decrease in the objective value for the vast majority of all request sequences.

Algorithms $\text{Nn}_l, \text{Ins}_l$ and 2Opt_l exhibit exclusive improvement due to lookahead which is seen by the perfect ordering of the plots corresponding to different lookahead levels. For exact reoptimization algorithms in Opt_l, we realize that the provided lookahead levels $l \in \{1, 2, \dots, 5\}$ are too small to be exploited more advantageously than by the other algorithms. Moreover, we find that the distribution functions for Opt_4 and Opt_5 intersect such that no exclusive benefit of lookahead can be attested. Opt_2 if found to score poorly compared to Opt_1 considering that it is provided an additional lookahead request.

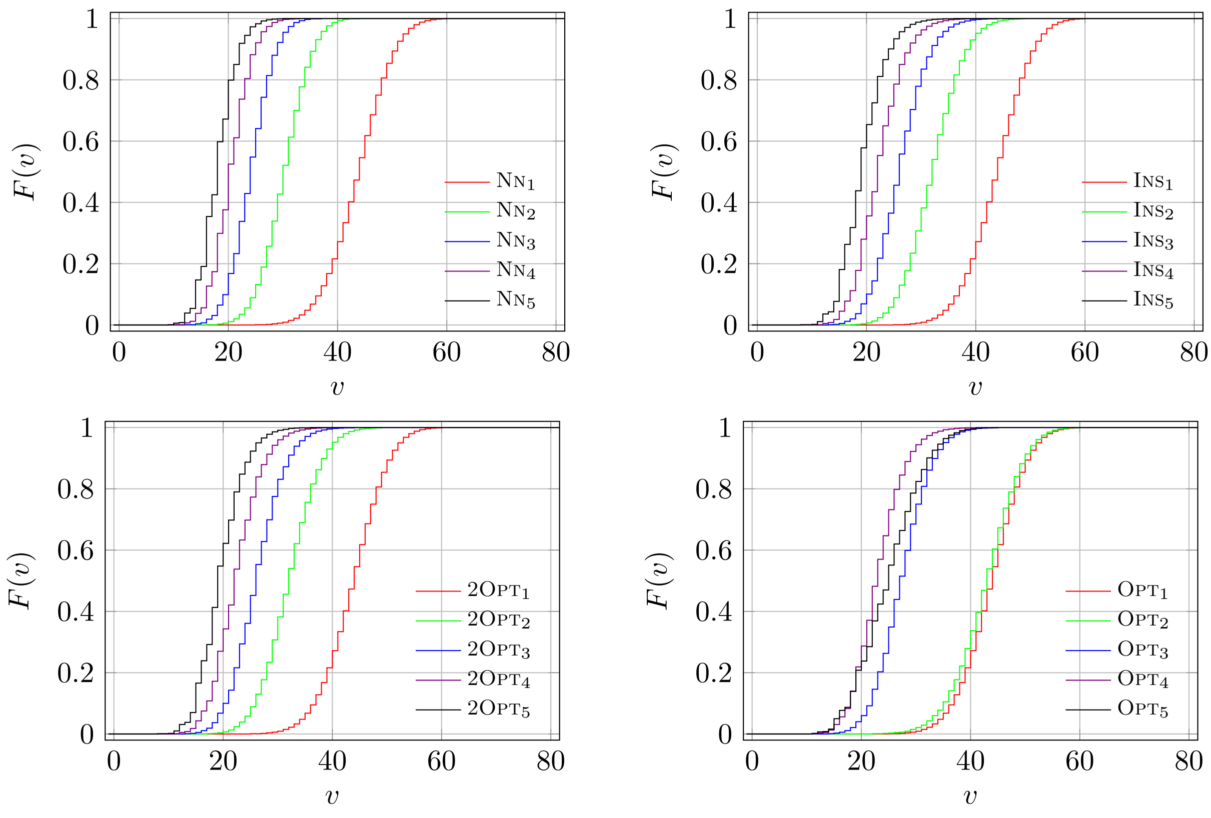


Figure 5.35: Exact distribution functions of costs for $n = 25$ in the TSP.

We conclude this section by pointing out that the computational experiments on the TSP confirm the large positive effect of lookahead on the objective value as determined by exact analysis for a basic setting with two request locations and $l = 2$ in Chapter 4.3. Hence, significant savings in the total distance can also be obtained in $\mathbb{R}^2$, although the impact of lookahead – in comparison to the case of $\mathcal{M} = \{0, 1\}$ from Chapter 4.3 – is mitigated to a certain extent by requests now being scattered over $\mathbb{R}^2$ rather than over $\{0, 1\}$.

5.5 Online Scheduling with Lookahead

Scheduling deals with the allocation of tasks to a set of resources over a given time horizon in order to meet some objective ([30], [138]). In the sequel, resources are called machines and tasks are referred to as jobs. Problem types differ in the machine environment, the processing restrictions and the objective(s) to be optimized. Frequent machine environments are a single machine, parallel machines, flow shops or job shops; processing restrictions may include precedence constraints, preemptions, release times or due dates; and common objectives are to minimize the maximum completion time (makespan), the total completion time or the maximum lateness. A problem can be classified using the three position scheme $\alpha \,|\, \beta \,|\, \gamma$ of Graham ([81]) with α indicating the machine environment, β the processing restrictions and γ the objective.

Since our analysis encompasses online scheduling with lookahead, we face online counterparts of offline scheduling problems; we indicate this with an additional entry *on* in β. Moreover, we distinguish whether a job is allowed to be processed immediately when it becomes known due to lookahead or whether it has to wait until its time of notification in the case without lookahead. The latter time is called the release time r_j of a job j. For job set J, the required processing time of job $j \in J$ is given as p_j; the completion time of j is denoted by C_j. The total completion time of all jobs in J is denoted symbolically by $\sum C_j := \sum_{j \in J} C_j$ and the makespan by $C_{max} := \max\{C_j \,|\, j \in J\}$. We study problems for a single machine ($\alpha = 1$) and two identical parallel machines ($\alpha = P2$). Overall, we obtain four problem settings:

- Online single machine with total completion time objective: $1 \,|\, on \,|\, \sum C_j$
- Online single machine with release times and total completion time objective: $1 \,|\, on, r_j \,|\, \sum C_j$
- Online parallel machines with makespan objective: $P2 \,|\, on \,|\, C_{max}$
- Online parallel machines with release times and makespan objective: $P2 \,|\, on, r_j \,|\, C_{max}$

Under total completion time objective, a job sequence has to be found which schedules short jobs as early as possible; under makespan objective, machines have to cooperate so as to find balanced workloads over the machines ([138]).

For fixed $n \in \mathbb{N}$, the set of all input sequences of length n is given by

$$\Sigma_n = \big\{(\sigma_1, \sigma_2, \ldots, \sigma_n) \,|\, \sigma_i := p_i \in \mathbb{R}^{>0}, i = 1, \ldots, n\big\}$$

and comprises all job sequences of length n where σ_i is identified with the processing time p_i of the ith job. In the online version, σ_i with $i = 1, \ldots, n$ is known when the current time reaches release time r_i. In the online version with lookahead of duration D, σ_i with $i = 1, \ldots, n$ is known when the current time reaches its forwarded release time $\max\{0, r_i - D\}$. In the offline version, all jobs are known at the beginning.

Our numerical experiments examine two problem settings each of which features $m = 1000$ independently drawn job sequences. In the first setting, each sequence consists of $n = 25$ jobs, while the second setting has $n = 100$ jobs per sequence. Jobs are assumed to have a regular release time in time interval $[0, 100]$ and a processing time from $(0, 4]$. Lookahead levels correspond to lookahead durations of $D \in \{0, 5, 10, 25, 50, 100\}$ time units. We discuss results for $n = 25$ and refer the reader to Appendix A.2.5 for $n = 100$.

A decision by an algorithm is required in two situations: First, a machine is freed and processable jobs are available; second, the machine is idle and a job becomes processable. The decision amounts to selecting a known processable job to be started on a free machine.

The earliest start time of a job is the earliest time where it is allowed to go on a machine: In the case of allowed immediate processing, the earliest start time of a job coincides with its notification time; in the case of forbidden immediate processing, the earliest start time of a job is its (regular) release time from the case without lookahead.

5.5.1 Online Single Machine Scheduling

For the offline problem without release times ($1 \mid \mid \sum C_j$), it is optimal to schedule jobs in order of non-decreasing processing times as can be shown easily by contradiction ([138]). The following algorithms mimic this behavior on the known jobs in the lookahead set.

Online algorithm

- ShortestProcessingTime (Spt): If a new job arrives at time r_j and the machine is idle, start the shortest job with release time r_j; if the machine finishes a job and unprocessed jobs are known, start a job with shortest processing time among these jobs ([138]).

Online algorithms with lookahead of duration D

- ShortestProcessingTime$_D$ (Spt$_D$):
  If the earliest start time of a job is reached and the machine is idle, start the shortest job whose earliest start time is reached; if the machine finishes a job and there are unprocessed jobs whose earliest start time has already been reached, start a job with shortest processing time among these jobs.

- Optimal$_D$ (Opt$_D$): If the earliest start time of a job is reached and the machine is idle or if the machine finishes a job and unprocessed jobs are known whose earliest start time has already been reached, then in Figure 5.36, set $M := 1$ and N, e, p according to the current jobs in the lookahead set. Solve the MIP formulation in Figure 5.37. Start the first processable job as suggested by the obtained schedule.

Optimal offline algorithm

- Optimal (Opt): In Figure 5.36, set $N := n, M := 1$ and e, p according to the known jobs. Solve the MIP formulation in Figure 5.37. Execute the obtained schedule.

Sets

J	set of jobs with $J = \{1, \dots, N\}$
K	set of machines with $K = \{1, \dots, M\}$

Parameters

N	number of jobs
M	number of machines
e_j	earliest start time of job $j \in J$
p_j	processing time of job $j \in J$
c_k	next completion time of machine $k \in K$
M	sufficiently large constant (big M)

Variables

$$x_{ij} = \begin{cases} 1 & \text{if job } i \in J \text{ precedes job } j \in J, \\ 0 & \text{else} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if job } i \in J \text{ precedes job } j \in J \text{ on machine } k \in K, \\ 0 & \text{else} \end{cases}$$

$$z_{jk} = \begin{cases} 1 & \text{if job } j \in J \text{ is scheduled on machine } k \in K, \\ 0 & \text{else} \end{cases}$$

$s_j \geq 0$	start time of job $j \in J$
$s_{jk} \geq 0$	start time of job $j \in J$ on machine $k \in K$

Figure 5.36: Sets, parameters and variables in the MIP formulations of the scheduling problems.

$$\begin{aligned}
\min \quad & \sum_{j \in J} (s_j + p_j) && && (5.39)\\
\text{s.t.} \quad & x_{ij} + x_{ji} = 1 && i, j \in J,\, i \neq j && (5.40)\\
& x_{jj} = 0 && j \in J && (5.41)\\
& s_j \geq e_j && j \in J && (5.42)\\
& s_i + p_i \leq s_j + \mathrm{M} \cdot (1 - x_{ij}) && i, j \in J,\, i \neq j && (5.43)\\
& x_{ij} \in \{0, 1\} && i, j \in J && (5.44)\\
& s_j \geq 0 && j \in J && (5.45)
\end{aligned}$$

Figure 5.37: MIP formulation of the scheduling problem $1 \,|\, e_j \,|\, \sum C_j$.

In case of allowed immediate processing, we expect $\textsc{Spt}_D$ and $\textsc{Opt}_D$ to coincide because in a reoptimization step it should never be advantageous for $\sum C_j$ to schedule a long job before a short one when both may be scheduled. In case of forbidden immediate processing, we do not expect much: $\textsc{Spt}_D$ collapses into $\textsc{Spt}$ for arbitrary D because it has to wait until the regular release time of a job to start processing; $\textsc{Opt}_D$ may only profit from lookahead when jobs accumulate while the machine is idle and it is "locally" advantageous to delay a job although it could be scheduled because a shorter job reaches its release date soon. This situation is displayed in Figure 5.38 where it is better not to schedule job 1 at time 0 but to wait for job 2 to start it at time 1; observe also that for all successive jobs the initial position as incurred by $\textsc{Opt}_D$ is worse than that incurred by $\textsc{Spt}_D$.

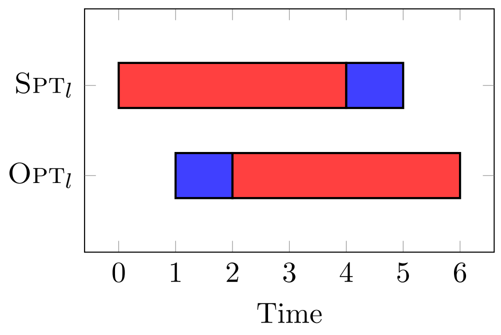


Figure 5.38: Local improvement of $\textsc{Opt}_D$ over $\textsc{Spt}_D$. For two jobs with $(r_1, p_1) = (0, 4)$ and $(r_2, p_2) = (1, 1)$, we have that $C_1(\textsc{Spt}_D) + C_2(\textsc{Spt}_D) = 9 > C_1(\textsc{Opt}_D) + C_2(\textsc{Opt}_D) = 8$.

5.5.1.1 Average Results

From the average total completion times in Figures 5.39 and 5.40, we conclude that for the algorithms under investigation mentionable improvements in the objective value can only be realized when immediate processing is allowed. In this case, $\textsc{Spt}_D$ and $\textsc{Opt}_D$ yield identical schedules as expected and there is no necessity of applying exact reoptimization. Comparing the extreme cases of the online and offline setting, improvements of up to 60.8 % are possible if all jobs were known at the beginning. The marginal benefit of an additional time unit of lookahead appears nearly constant over the first lookahead time units; for higher lookahead levels the marginal benefit decreases more and more. Considering the small coefficients of variation and the relatively small width of confidence intervals in Table A.43 of Appendix A.2.5, we regard the results as representative for randomly drawn job sequences. Also note that no instances can occur where additional lookahead leads to a larger total completion time.

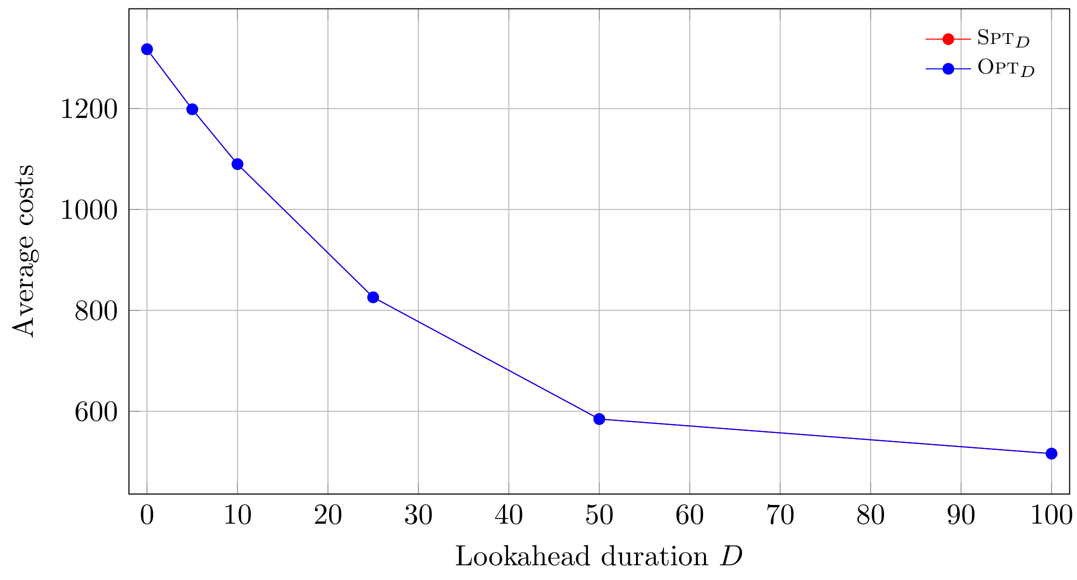


Figure 5.39: Average costs for different lookahead durations and $n = 25$ in the single machine scheduling problem when immediate processing is allowed.

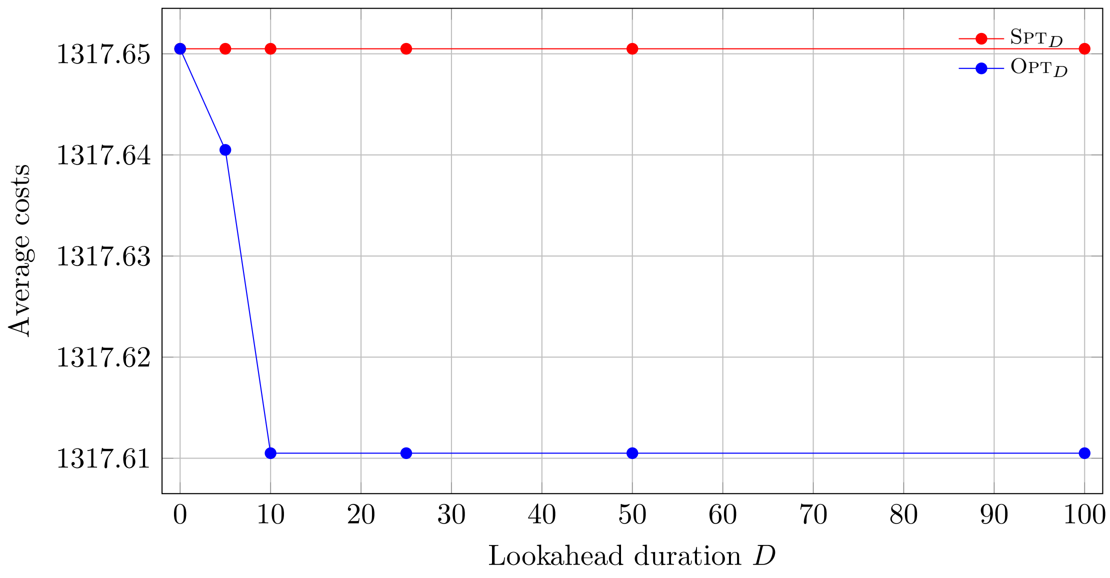


Figure 5.40: Average costs for different lookahead durations and $n = 25$ in the single machine scheduling problem when immediate processing is forbidden.

For algorithm family SPT$_D$ it was already clear from the algorithm description that additional lookahead cannot be exploited at all when immediate processing is prohibited, whereas for algorithm family OPT$_D$ there was hope that deciding to delay a job although it was available to be processed could pay off when another shorter job jumps in quickly after that decision (cf. Figure 5.38). Regrettably, this approach leads for the given parameters only to an average improvement of 0.04 time units, i.e., 0.003 %, when OPT$_{100}$ is used instead of OPT$_0$.

We conclude that the value of lookahead is attributable solely to the change of the rule set in form of allowed immediate processing which comes along with the provision of lookahead. We focus on the case of allowed immediate processing in the sequel. Note that the potential for improvement due to lookahead is strongly affected by the utilization of the machine: In the case of $n = 100$ jobs per sequence, improvements were much smaller – on average up to 8.7 % when the offline situation is compared to the online situation – since the 100 jobs are released over the same interval of 100 time units such that at any time there are more than enough short jobs as desired by the total completion time objective; for details, see Appendix A.2.5.

5.5.1.2 Distributional Results

Since SPT_D has been found to coincide with OPT_D in the case of allowed immediate processing, Figure 5.41 displays the objective value distribution of ALG_D for $\text{ALG} \in \{\text{SPT}, \text{OPT}\}$. The gap between the plots of successive lookahead levels is affirmative to the first additional lookahead time units' value being of comparable magnitude, whereas for medium to larger lookahead durations the benefit of additional lookahead time decreases. Non-overlapping confidence intervals and no degradations under additional lookahead as listed in Table A.43 of Appendix A.2.5 support the result of exclusively positive effects. The development of the distance between the bounds for confidence intervals of two successive lookahead levels confirms the decreasing marginal benefit of additional lookahead time units.

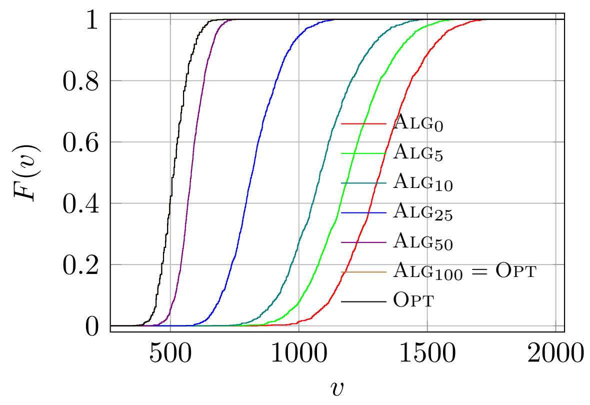


Figure 5.41: Empirical counting distribution functions of costs for $n = 25$ in the single machine scheduling problem when immediate processing is allowed.

From the empirical counting distribution functions of the performance ratios relative to OPT and relative to ALG_0 in Figures 5.42 and 5.43, we see the exclusively positive impact of lookahead which is also confirmed by the last columns in Tables A.45 and A.47 of Appendix

A.2.5 showing that no degradation of the total completion time occurs whenever additional lookahead is supplied as compared to the optimal offline algorithm and the algorithm's online version, respectively.

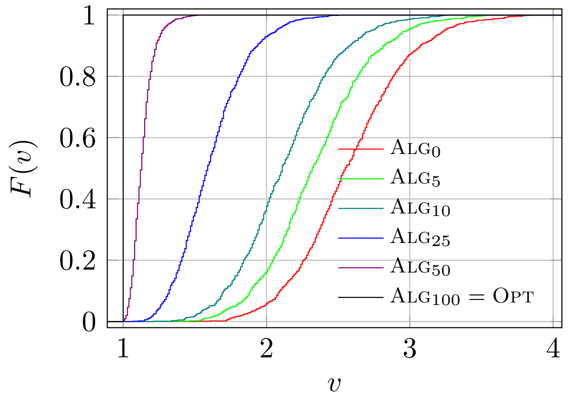


Figure 5.42: Empirical counting distribution functions of performance ratio of costs relative to OPT for $n = 25$ in the single machine scheduling problem when immediate processing is allowed.

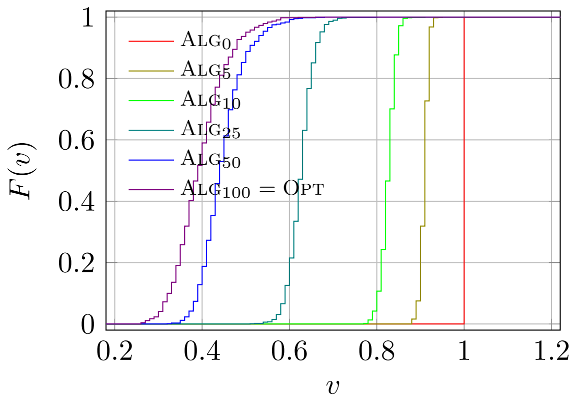


Figure 5.43: Empirical counting distribution functions of performance ratio of costs relative to the online version for $n = 25$ in the single machine scheduling problem when immediate processing is allowed.

Observe that for large lookahead all counting distribution functions are relatively steep in a characteristic interval which illustrates the homogenous positive effect of additional information on all kinds of request sequences.

5.5.1.3 Markov Chain Results

In the Markov chain analysis, we consider the case of allowed immediate processing. In order to facilitate an analysis in reasonable computing time, we abandon time lookahead and switch to request lookahead of size l. The elements of the state space $\mathcal{S}$ subsume the last completion time c of a job scheduled on the machine, the processing times $p = (p_1, \dots, p_l)$ of the lookahead jobs in non-decreasing order and the current objective value v of the total completion time, i.e.,

$$\mathcal{S} = \big\{(c, p, v) \,|\, c \in \mathbb{R}^{\geq 0}, p \in (\mathbb{R}^{>0})^l, v \in \mathbb{R}^{\geq 0}\big\}.$$

Note that completion of a job and arrival of a new job coincide under request lookahead. Upon arrival of a new job, each algorithm $\textsc{Alg}$ chooses a job from the lookahead to be scheduled next on the machine. Hence, $\textsc{Alg}$ is a function

$$\textsc{Alg} : \mathcal{S} \times \mathbb{R}^{>0} \to \{1, 2, \dots, l\}$$

which gives the position of the selected job in the lookahead. The successor state of $s = (c, p, v)$ upon arrival of a job with processing time p_{new} is given by $s' = (c', p', v')$ where $c' = c + p_{\textsc{Alg}(s, p_{new})}$, p' arises from p by removing element $p_{\textsc{Alg}(s, p_{new})}$, appending p_{new} and ordering non-decreasingly, and $v' = v + c'$.

In order to obtain a finite state space representation, we have to discretize job processing times: To this end, denote the number of possible processing times by n_p and the maximum possible processing time by p_{max}, then the state space size amounts to

$$|\mathcal{S}| = (np_{max} + 1) \cdot \binom{n_p + l - 1}{l} \cdot \Big(\frac{n(n+1)}{2} p_{max} + 1\Big).$$

When immediate processing is allowed, deviating from the shortest processing time first rule would lead to a deterioration in the objective value as shown easily by a job interchange argument. Hence, $\textsc{Spt}_l$ and $\textsc{Opt}_l$ coincide. For $\textsc{Alg}_l$ with $\textsc{Alg} \in \{\textsc{Spt}, \textsc{Opt}\}$ and selected lookahead level l, we determine the successor for each state and each new job to obtain the one-step frequency matrix and the n-step frequency matrix of the Markov chain.

In our numerical experiment, we use the parameterization $n = 25$, $n_p = 2$, $p_{max} = 2$ and $l \in \{1, 2, \dots, 5\}$. For this small exemplary setting, the state space size amounts to $(25 \cdot 2 + 1) \cdot \binom{2+5-1}{5} \cdot (\frac{25 \cdot 26}{2} \cdot 2 + 1) = 199\,206$ elements for $l = 5$.

The exclusively beneficial effect of lookahead for the resulting objective value distributions is presented in Figure 5.44. The horizontal distance between the plots of two successive lookahead levels suggests that the marginal improvement attainable by an additional lookahead unit remains the same for increasing lookahead size.

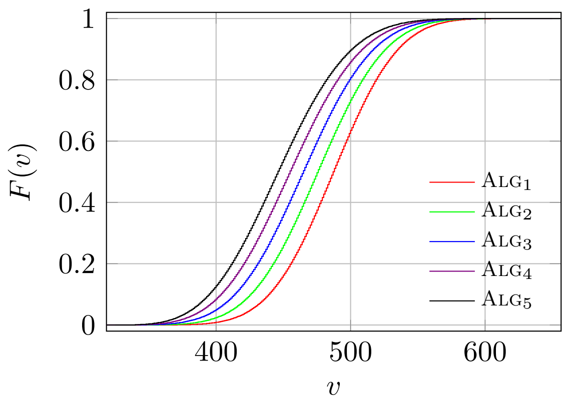


Figure 5.44: Exact distribution functions of costs for $n = 25$ in the single machine scheduling problem when immediate processing is allowed.

5.5.2 Online Parallel Machines Scheduling

Similar to ordering items by non-increasing size in bin packing, ordering jobs by non-increasing processing times is a reasonable strategy to balance workload among parallel machines. For the offline problem without release times ($Pm \,|\,|\, C_{max}$), this strategy is a $(\frac{4}{3} - \frac{1}{3m})$-approximation algorithm as shown easily by contradiction ([138]). For $m = 2$, we get a $\frac{7}{6}$-approximation algorithm. The following algorithms mimic this behavior on the known jobs in the lookahead set. Note that the problem is $\mathcal{NP}$-hard for $m \geq 2$ by reduction from $\mathcal{NP}$-complete problem `Partition`.

Online algorithm

- LongestProcessingTime (Lpt): If a new job j arrives at time r_j and a machine is idle, start the longest job with release time r_j on this machine; if a machine finishes a job and unprocessed jobs are known, start a job with longest processing time among these jobs on this machine ([138]).

Online algorithms with lookahead of duration D

- LongestProcessingTime$_D$ (Lpt$_D$):
  If the earliest start time of a job is reached and the machine is idle, start the longest job whose earliest start time is reached; if a machine finishes a job and there are unprocessed jobs whose earliest start time has already been reached, start a job with longest processing time among these jobs on this machine.

- Optimal$_D$ (Opt$_D$): If the earliest start time of a job is reached and a machine is idle or if a machine finishes a job and unprocessed jobs are known whose earliest start time has already been reached, then in Figure 5.36, set $M := 2$ and N, e, p, c according to the current jobs in the lookahead set and the current machine statuses. Solve the MIP formulation in Figure 5.45. Start the first processable job on the free machine as suggested by the obtained schedule.

$$\begin{array}{llrcllr}
\min & & C_{\max} & & & & (5.46) \\
\text{s.t.} & & x_{ijk} + x_{jik} & \le & z_{ik} & i,j \in J,\ i \neq j,\ k \in K & (5.47) \\
& & x_{ijk} + x_{jik} & \ge & z_{ik} + z_{jk} - 1 & i,j \in J,\ i \neq j,\ k \in K & (5.48) \\
& & x_{jjk} & = & 0 & j \in J,\ k \in K & (5.49) \\
& & \sum_{k \in K} z_{jk} & = & 1 & j \in J & (5.50) \\
& & s_{jk} & \ge & \max\{e_j, c_k\} z_{jk} & j \in J,\ k \in K & (5.51) \\
& & s_{ik} + p_i & \le & s_{jk} + \mathrm{M} \cdot (1 - x_{ijk}) & i,j \in J,\ i \neq j,\ k \in K & (5.52) \\
& & C_{\max} & \ge & s_{jk} + p_j & j \in J,\ k \in K & (5.53) \\
& & x_{ijk}, z_{jk} & \in & \{0,1\} & i,j \in J,\ k \in K & (5.54) \\
& & s_{jk} & \ge & 0 & j \in J,\ k \in K & (5.55)
\end{array}$$

Figure 5.45: MIP formulation of the scheduling problem $Pm \,|\, e_j \,|\, C_{\max}$.

Optimal offline algorithm

- Optimal (Opt): In Figure 5.36, set $N := n, M := 2, c := 0$ and e, p according to the known jobs. Solve the MIP formulation in Figure 5.45. Execute the obtained schedule.

In the MIP formulation in Figure 5.45, we use additional decision variables z_{ik} in order to express on which machine k job i is processed. C_{max} is used as an auxiliary variable to indicate the end of the schedule. All other variables are also extended with an additional index referring to the machine. In particular, Constraint 5.48 ensures that there is a precedence relation between each pair of jobs that goes on the same machine.

In case of forbidden immediate processing, we do not expect lookahead to be exploited much: Lpt$_D$ cannot take advantage of lookahead at all because it collapses into Lpt; for Opt$_D$ in order to excel Lpt$_D$, job processing times have to be such that arranging them "locally" optimal on the two machines leads to a deviation from Lpt$_D$ as shown in Figure 5.46. Clearly, the probability for such a pathologic construction is not too high. Note that this effect also occurs under allowed immediate processing. However, the lookahead effect there is mainly attributed to the rule set substitution, whereas under forbidden immediate processing could only be attributed to Opt$_D$'s farsightedness.

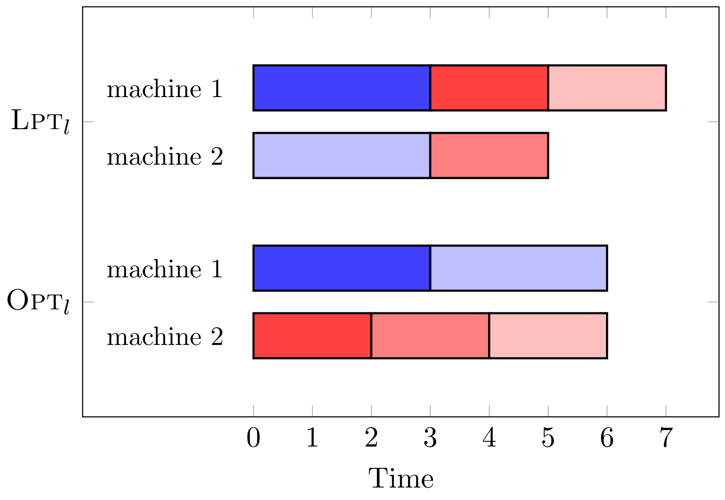


Figure 5.46: Local improvement of OPT$_D$ over LPT$_D$. For five jobs with $(r_1, p_1) = (r_2, p_2) = (0, 3)$, $(r_3, p_3) = (r_4, p_4) = (r_5, p_5) = (0, 2)$, we have that $C_{max}(\text{LPT}_D) = 7 > C_{max}(\text{OPT}_D) = 6$.

5.5.2.1 Average Results

We find massive makespan reductions when immediate processing is allowed and virtually no effect when immediate processing is forbidden as depicted in Figures 5.47 and 5.48. In case of allowed immediate processing, the makespan can be drastically reduced as a result of avoided idle times on the machines and the privilege of not having to wait until regular release times. Between online and offline situations, algorithm performance can be enhanced such that makespan reductions of up to 74.5 % are incurred. Additionally, the potential of sophisticated exact reoptimization in order to capitalize from situations as displayed in Figure 5.46 is considered insignificant. Small coefficients of variation and tight confidence intervals indicate statistical validity of the obtained results (cf. also Table A.55 of Appendix A.2.5). As in the single machine problem, no instance encountered exhibits a deterioration in the objective value when algorithms are provided additional lookahead time.

Contrarily, under forbidden immediate processing the makespan is by definition crucially determined by the release times of the last jobs to finish processing on the machines; this holds especially true in case of low to moderate machine utilization as given for $n = 25$ with an overall horizon of 100 time units: Looking at the scale in Figure 5.48, we declare the improvement under forbidden immediate processing as negligible. For LPT$_D$ it was already clear from the algorithm description that no lookahead value exists at all; whereas for OPT$_D$ we found that there are too few situations that would allow to determine an elaborate partial schedule by applying exact reoptimization techniques similar to the one shown in Figure 5.46 so as to influence the overall makespan attained at the end of the time horizon.

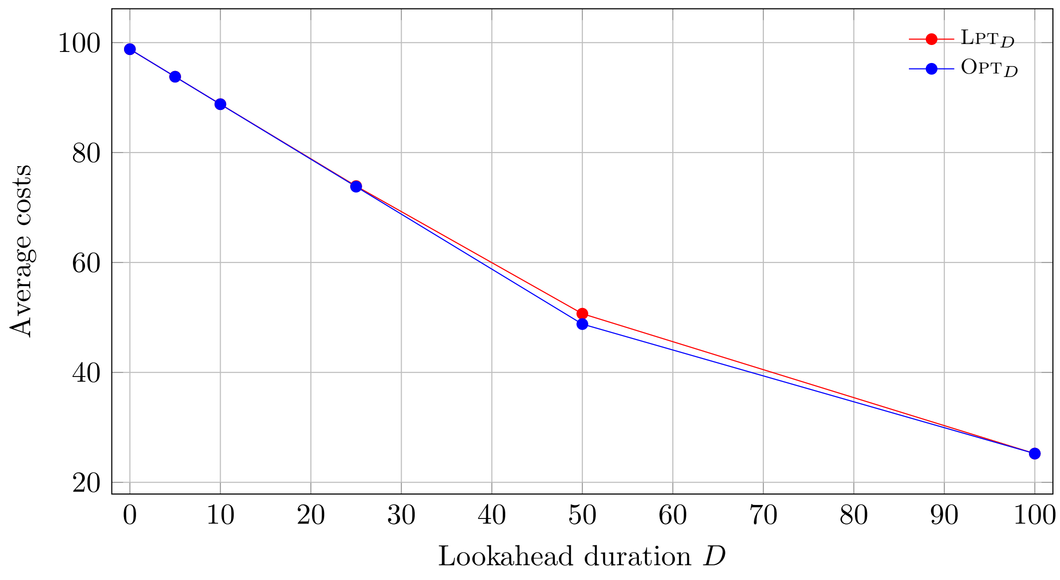


Figure 5.47: Average costs for different lookahead durations and $n = 25$ in the parallel machines scheduling problem when immediate processing is allowed.

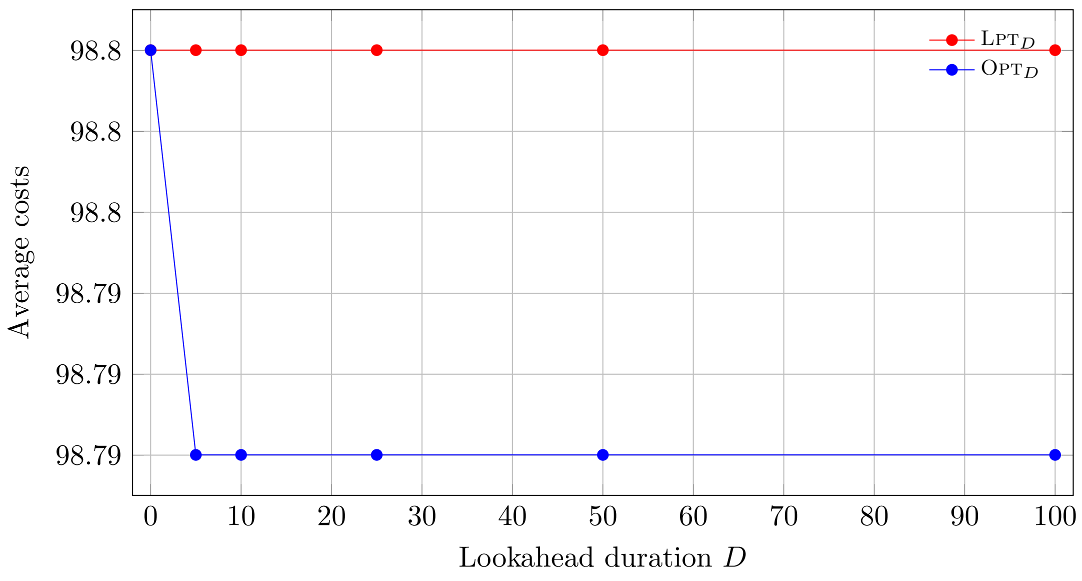


Figure 5.48: Average costs for different lookahead durations and $n = 25$ in the parallel machines scheduling problem when immediate processing is forbidden.

In summary, makespan reductions are achieved only as a result of a changed problem setting in favor of the decision maker by allowing immediate processing. Hence, we concentrate on this setting subsequently. As seen in Appendix A.2.5, also for $n = 100$ jobs per sequence makespan reductions are achieved. However, as in the single machine problem, the reduction is sharply constricted by the high machine utilization which makes sure that most of the times when a machine is freed there are plenty of long jobs to go on that machine.

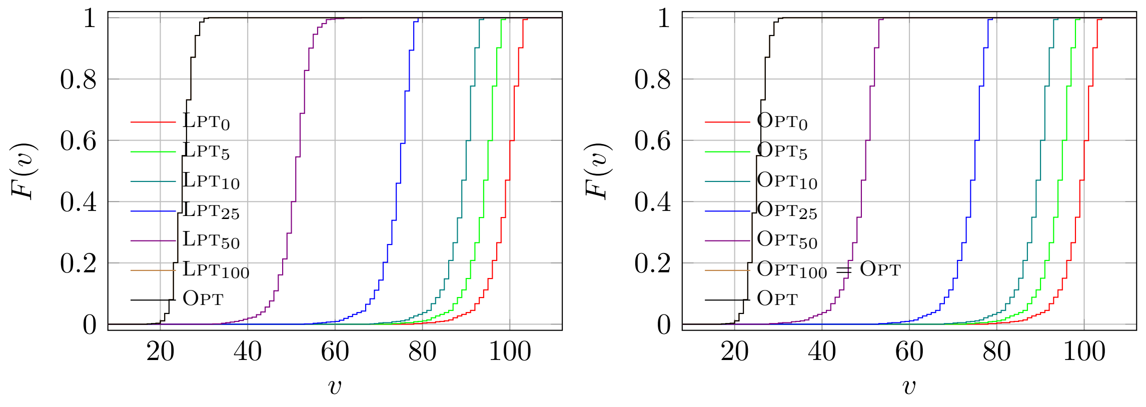


Figure 5.49: Empirical counting distribution functions of costs for $n = 25$ in the parallel machines scheduling problem when immediate processing is allowed.

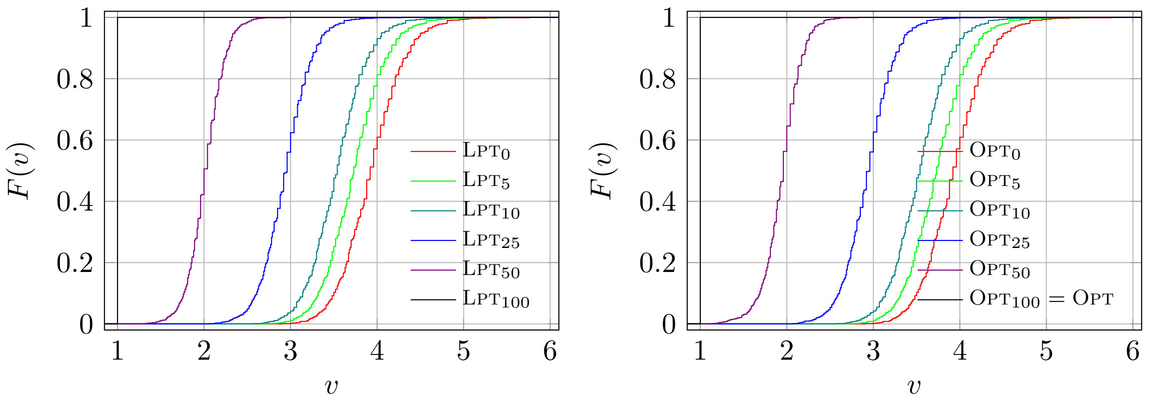


Figure 5.50: Empirical counting distribution functions of performance ratio of costs relative to OPT for $n = 25$ in the parallel machines scheduling problem when immediate processing is allowed.

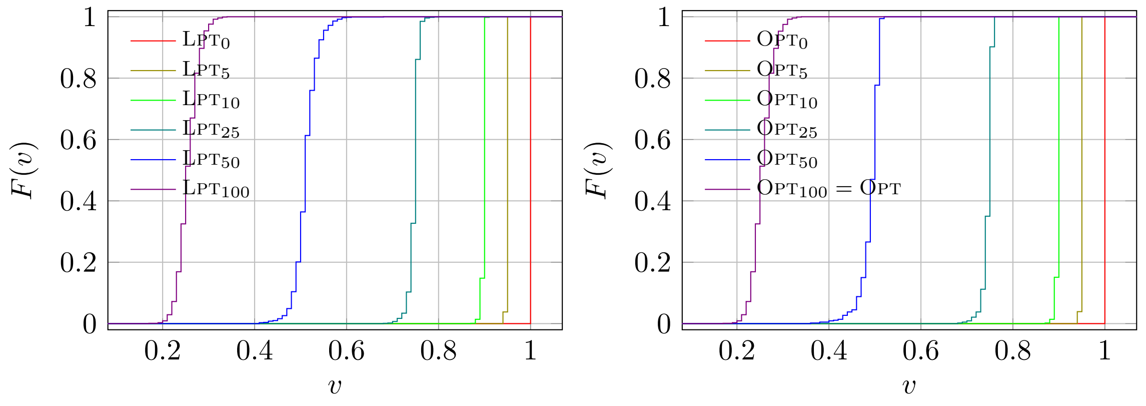


Figure 5.51: Empirical counting distribution functions of performance ratio of costs relative to the online version for $n = 25$ in the parallel machines scheduling problem when immediate processing is allowed.

5.5.2.2 Distributional Results

Figure 5.49 illustrates the huge benefit of lookahead when immediate processing is allowed in terms of a left shift of the plots of the empirical counting distribution functions of the objective value when additional time units of lookahead are supplied. Algorithm families $\textsc{Lpt}_D$ and $\textsc{Opt}_D$ lead to schedules of the same quality such that it is not necessary to apply exact reoptimization methods. From the large slope value of the empirical counting distribution functions in a characteristic interval for each combination of algorithm and lookahead level as well as from the width of all confidence intervals being smaller than one time unit (cf. Table A.55 of Appendix A.2.5), we infer that the sampled results give a profound impression of the performance of the respective algorithms. From the distance of the curves to each other, we see that for small to medium lookahead levels the marginal benefit of an additional time unit of lookahead stays constant and that it gradually decreases for larger lookahead durations.

Figures 5.50 and 5.51 display the empirical counting distribution functions of the performance ratios relative to $\textsc{Opt}$ and the online versions of the algorithms, respectively. Confidence interval widths collectively smaller than 0.04 in the case relative to $\textsc{Opt}$ and even coinciding at the second decimal place in the case relative to the online version imply the characteristic improvement for each lookahead level (cf. Tables A.57 and A.59 of Appendix A.2.5). No algorithm exhibits worsened performance on any instance when lookahead is supplied.

We conclude this section by pointing out the similarities between the computational results for the scheduling problem and those for the TSP. In both cases, being allowed to process an input element as soon as its existence emerges leads to the major impact of lookahead.

5.6 Concluding Discussion

The computational experiments from this chapter covered different problem classes:

- In the ski rental problem, an instance of a one-shot rent-or-buy problem was solved.
- In paging, instances of replacement problems were solved repetitively.
- In bin packing, instances of packing problems were solved repetitively.
- In the TSP, instances of sequencing problems were solved repetitively.
- In scheduling, instances of (combined) packing, sequencing and timing problems were solved repetitively.

Numerical results in each problem setting suggest that there *is* an overall positive effect on the objective value that can be obtained if algorithms are designed and allowed to take advantage of lookahead. However, the magnitude of the effect is strongly problem-specific: In the TSP and in scheduling with allowed immediate processing, exploiting additional lookahead directly paid off upon changing the processing order of input elements because of the *direct* impact on the objective value. In bin packing, the observed effect was much smaller as a result of the *indirect* impact of item assignments on the objective value: An item occupies the same bin capacity no matter in which bin it lies and at which time it is packed. Moreover, we found that improvements in packing problems could only be attained by a clever arrangement of the small objects that have to be put in the large objects. Unfortunately, constellations where such improvements can be made are encountered rarely in typical item or job sequences. Lookahead in the paging and ski rental problem was also found to have a major positive impact on the costs incurred by respective algorithms. The explanation lies in the risk-free exploitation of additional lookahead based on a larger part of the time horizon that is overseen. The additional knowledge allows an algorithm to make a decision that is guaranteed to have no future ramifications as compared to a decision made under fewer available lookahead information. Hence, in these types of problems there is no threat of bad decisions.

In some problem settings, it was possible to devise algorithms such that additional lookahead proved exclusively beneficial, whereas in others there were instances with degraded algorithm performance for any of the devised algorithms since lookahead was leading them towards a wrong direction as discovered only later in the processing of the input sequence. Yet, such instances were encountered only sporadically in all problem settings.

Concerning the usage of exact reoptimization algorithms for the subproblems, we found that the role of exact reoptimization methods to unveil improvement by lookahead is smaller than previously expected: Although in some problems these methods led to a slight performance enhancement especially for large lookahead sizes, they also tended to be of nearly no benefit for small lookahead sizes when compared to heuristic reoptimization strategies: Optimality of a partial solution often does not migrate to the overall solution because substructures in partial solutions with a positive influence on the objective value are likely to be relinquished during the future solution process. In the case of the TSP, we also discovered that applying sophisticated reoptimization methods could be harmful since stability as imposed by simple algorithms may get lost; for small to medium lookahead sizes, exact reoptimization was frequently beaten by much simpler heuristic approaches. It is up to the decision maker to figure out whether the computationally intense usage of exact reoptimization methods is worth the effort or not.

Table 5.1 subsumes the findings from this chapter on a granular level: The column for the total lookahead effect ($\Delta f^{r,r',P,P'}_{\textsc{Alg},\textsc{Alg}'}$) admits that observed lookahead effects strongly relied on the problem settings themselves. The two columns for the partial lookahead effects due to instance revelation rule substitution ($\Delta f^{r,r'}_{\textsc{Alg}}$) and due to rule set substitution ($\Delta f^{P,P'}_{\textsc{Alg},\textsc{Alg}'}$) show that in the ski rental, paging and bin packing problem merely the additional information was responsible for improvements, whereas in the TSP and the scheduling problem the change of the rule set lead to improved objective values in the first place. Column $\textsc{Alg}^*$ tells us whether there was one algorithm (or several comparably good algorithms) that could be considered the champion over all lookahead levels. Although this happened only rarely, we often observed that heuristics did especially well for small lookahead sizes, whereas exact reoptimization approaches outperformed heuristics by a very slight margin for large lookahead sizes, i.e., when the problem approaches its offline version. Column $\textsc{Opt}$ indicates whether exact reoptimization lead to significant improvements. As shown before, only minor improvements were observed such that exact reoptimization is not deemed a must-have in online optimization. There was no homogeneous picture about whether additional lookahead can also lead to objective value deterioration or not (column *Deterioration*).

We remark that the computational results from this chapter are affirmative to the findings of the exact analysis in the previous chapter concerning the magnitude of the lookahead impact as well as its primarily responsible factors.

Our approach of evaluating algorithm performance in the context of a potential provision of lookahead encompassed two stages: In the first stage, an average-case analysis allowed us to find the most promising algorithm candidates for a given lookahead level in terms of expected algorithm behavior. In the second stage, distributional analysis lead to a fine-grained assessment of each candidate's individual risk profile with respect to attainable objective values and performance ratios.

Problem	Type	Attribute	Rule	$\Delta f^{r,r',P,P'}_{\text{ALG},\text{ALG}'}$	$\Delta f^{r,r'}_{\text{ALG}}$	$\Delta f^{P,P'}_{\text{ALG},\text{ALG}'}$	ALG*	OPT	Deterioration
Ski rental	request lookahead	–	–	large	large	zero	yes	–	no
Paging	request lookahead	equal probabilities	–	large	large	zero	no	–	no/yes*
		access graph	–	medium	medium	zero	yes	–	no/yes*
		page frequencies	–	large	large	zero	yes	–	no/yes*
Bin packing	request lookahead	classical	permutations	small	small	zero	no	no	yes
			no permutations	small	small	zero	no	no	yes
		bounded-space	permutations	small	small	negligible	yes	no	yes
			no permutations	small	small	negligible	no	no	yes
TSP	request lookahead	–	–	large	zero	large	no	no	yes
Scheduling	time lookahead	single machine	immediate processing	large	negligible	large	no	no	no
			no immediate processing	negligible	negligible	zero	no	no	yes
		parallel machines	immediate processing	large	negligible	large	no	no	no
			no immediate processing	negligible	negligible	zero	no	no	no

*Instances with deteriorated objective value in the paging problem were only observed for batching algorithms.

Table 5.1: Qualitative summary of the experimental results from Chapter 5.

6 Simulation of Real World Applications

In the previous two chapters, we studied the influence of lookahead on solution quality in standard online optimization problems. This chapter extends our analysis to more complex dynamic systems. We consider two real world applications which reveal their online character not only in form of an input element disclosure over time but also in form of additional unforeseeable events:

- Online Order Picking with Lookahead
- Online Pickup and Delivery with Lookahead

Examples for additional random events are breakdowns, no-shows or processing time variations. Since future implications of these event types are uncertain due to our non-clairvoyance, it is now not even possible to provide an exact formulation of the snapshot problem at a given time. Moreover, since applications originate from practice, typically not only one but a set of objectives is relevant to the decision maker. Accordingly, we find ourselves in the intrinsically more difficult setting of multicriteria optimization. We are concerned with the question whether endowing algorithms with lookahead proves beneficial to their outcome given that we have to regard several optimization goals and additional random influences.

Due to the high complexity incurred by a large number of dependent random variables and the existence of several optimization goals, we use simulation models to derive statements about algorithm performance. Optimization algorithms are integrated into the simulation environment such that whenever the logic of the simulation requires a decision, a corresponding algorithm is invoked to deliver that decision (see also Chapter 2.4.4 and [68]).

Computational experiments were performed on the same hardware as specified at the beginning of Chapter 5. All simulation models were developed in AnyLogic 6.9.0 as discrete event models. Algorithms were implemented in the native Java environment of AnyLogic and arising instances of IP and MIP formulations were solved using IBM ILOG CPLEX 12.5 with a prescribed time limit of 120 seconds. A detailed summary of statistical key figures for each experiment is given in Appendix A.3.

6.1 Online Order Picking with Lookahead

In a manual order picking system, pickers move through the aisles of a warehouse to retrieve items packed in boxes as demanded by the orders from external customers. When a picker has collected all boxes assigned to him, he returns to a depot to unload the boxes and wait for the next assignment of boxes to be picked ([89], [92]).

Efficiently managing an order picking system is a crucial task both in manufacturing and distribution logistics which is underlined by the fact that more than the half of all warehouse operating costs can be attributed to order picking ([57]).

Typically, customer orders arrive throughout the day and the objective is to make the pickers collect all boxes of the customer orders in a way that meets the decision maker's goal system best. Figure 6.1 displays the warehouse under consideration along with charts for some performance indicators of interest.

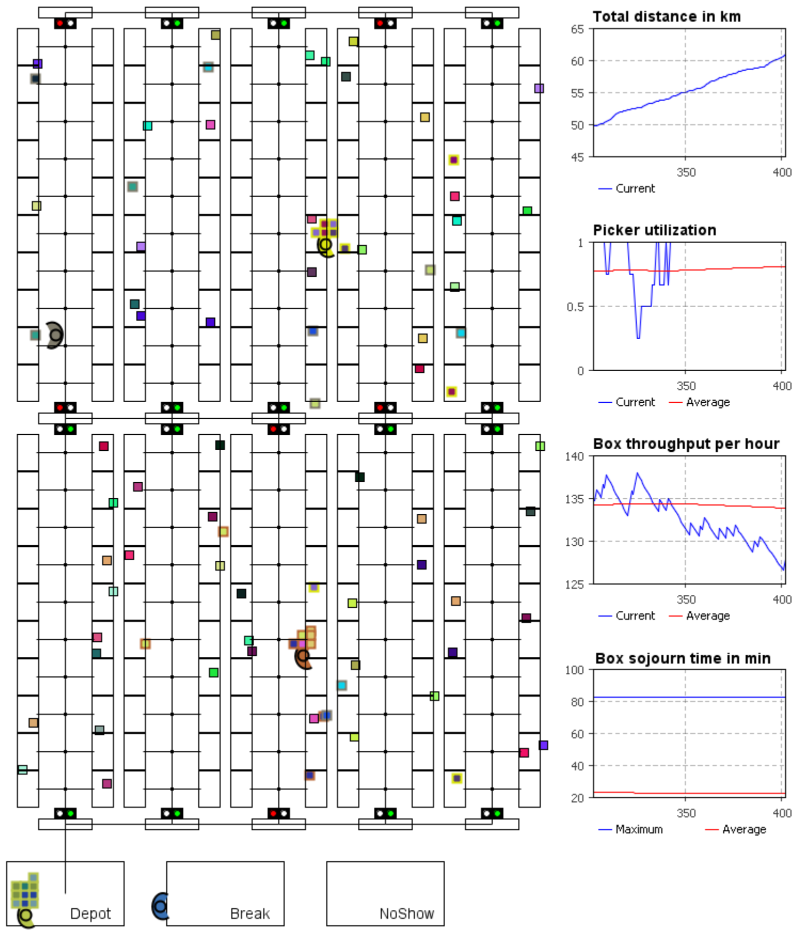


Figure 6.1: Animation of the simulation model for an order picking system in AnyLogic.

The warehouse layout consists of ten aisles arranged in a lower and an upper block of five aisles each. Each aisle has 20 storage locations, ten to the left and right, respectively. Depot and break location are positioned in the lower left corner of the warehouse. The length of each aisle amounts to 30 meters, the horizontal (vertical) distance between two aisles is 8 (2.5) meters. On average, pickers move at a speed of 1 meter per second. When a picker is not working and absent without leave, he is displayed in the no-show area. Five pickers have to commission $n = 625$ orders over a work day of 600 minutes plus potential overtime. An order may consist of up to three boxes and the picker capacity amounts to ten boxes.

Aisle traversal is subject to blocking effects ([92]), e.g., because of security or space considerations: Only one picker is allowed inside an aisle at a time. For this reason, each aisle has a traffic light at its front and rear entry (cf. Figure 6.1) to signal that the aisle is accessible (green light) or inaccessible (red light) at the moment. Pickers that need to enter an inaccessible aisle have to enqueue at an aisle entry point and wait until the aisle is freed.

Order arrival and data are random, i.e., release time, number of boxes, pick time, drop time and location of an order are realizations of random variables that are unknown to the algorithms which have to determine pick lists and routes. In addition, order picking operations underlie the following random influences:

- Picker velocity profile
- Picker break start and end time
- Picker no-show occurrence
- If applicable, picker no-show start and end time

Because of blocking effects and the large number of random processes, it is out of scope to give an exact formulation of an optimization problem that takes into account all of these features. Hence, simulation is deemed a suitable method of analysis.

At the end of a run of an independent simulation replication, the decision maker is supplied with the following quality indicators for the pick lists and routes which are used to judge on the quality of the responsible algorithm:

- Makespan, i.e., the time when the last box is unloaded at the depot
- Total distance covered by all pickers
- Picker utilization, i.e., the mean percentage of working pickers over the time horizon
- Box throughput, i.e., the box delivery rate at the depot

Lookahead appears as time lookahead of duration $D \in \{0, 60, 120, \ldots, 600\}$ minutes. We assume that the warehouse initially contains all items in a quantity sufficient to satisfy all customer orders arriving over the day such that no additional storage operations are required in the course of a day. Once an order arrives, its contents are ready to be picked by a picker. Hence, lookahead forwards the earliest start time of an order, and waiting until the regular release time from the pure online case is not necessary. In this way, lookahead implies a change of the rule set in favor of the decision maker.

In our computational experiments, we draw $m = 50$ independent simulation replications by initializing the random number generator with a different seed and ensuring independence of all stochastic processes in each replication.

An algorithm is required to determine the pick lists and routes for all pickers available at that time so as to fulfill the known and yet unfulfilled customer orders in a way that best matches the decision maker's preferences with respect to the quality criteria specified above. We agree upon algorithm execution whenever the current situation changes as a result of a customer order arrival, a picker's re-entry in the system after a break or no-show, or a picker becoming available upon finished unloading at the depot.

Commonly, all boxes of an order are picked by the same picker in a single route such that no sorting device needs to be installed. Hence, managing an order picking system consists of repetitively solving two interrelated subproblems for yet unserved orders:

1. Assignment of orders to pickers such that the total number of boxes of assigned orders does not exceed the picker capacity (batching).
2. Route calculation[16] for each picker such that all boxes of assigned orders are visited and the route starts and ends at the depot (routing).

Algorithms solve these tasks either sequentially or simultaneously. Sequential methods cope with the complexity of the problem by decoupling the subproblems from each other similar to cluster-first route-second approaches for vehicle routing ([101]); the batching algorithm first assigns orders to pickers; afterwards the routing algorithm determines picker traversal paths through the aisles including aisle entry and exit points. Note that some batching algorithms take potential routes as determined by a routing algorithm into account. Due to the multitude of different batching and routing algorithms, our computational experiments check which combination of a batching and routing policy is most promising. Simultaneous methods solve the batching and routing problem at the same time. Only simultaneous

[16] Because a route in an order picking system has to start and end in the depot, the decision version of the "problem" contains `Hamiltonian Circuit` which is known to be an $\mathcal{NP}$-complete problem.

methods provide exact solutions to snapshot problem instances. Unfortunately, when many customer orders are known, e.g., due to lookahead of large duration, problem instances become huge and solving them almost surely exceeds the prescribed time limit of 120 seconds. In this case, a substitute solution is retrieved by applying a sequential method. Except for the exact reoptimization approaches, all of the following algorithms represent well-known and commonly accepted algorithms for order picking systems (see the survey in [89] and the included references).

Batching Algorithms

- PriorityBatching (Prio): Sort orders according to a criterion, e.g., non-increasingly according to their number of boxes. Assign orders successively to batches in a first fit manner ([89]).
- SeedBatching (Seed): Batches are built sequentially: Initialize each batch with a seed order, e.g., by selecting an order with the largest number of boxes; fill the batch with additional orders according to an order congruency rule, e.g., select an order with the smallest number of additional aisles ([89]).
- SavingsBatching (Svgs): Batches are built simultaneously: Initialize the batch building process with each order forming a separate batch. In the improvement phase, combine orders of two batches into one batch if the total distance is reduced according to the routing algorithm applied until no further improvement is possible ([89]).
- LocalSearchBatching (Ls): Let the neighborhood of a batch set be given by all batch sets which are obtained by a swap or shift move. A swap move exchanges one selected order per batch between two batches; a shift move transfers a selected order from one batch to another. A perturbation of a batch set consists of transferring a random number of orders from one batch to another if the receiving batch remains feasible. Execute the following two steps until the total distance of all batches cannot be reduced in either step: Search within the neighborhood of the current batch set for a batch set with smaller total distance. Perturb the current batch set for a fixed number of times to find a batch set with smaller total distance ([89]).
- TabuSearchBatching (Ts): Apply Ls with the modifications that a swap or shift move that has just been carried out is forbidden along with its inverse move for a prescribed number of iterations and that always the best solution within the neighborhood of the current batch set is selected even if it leads to a longer total distance ([89]).

Routing Algorithms

- ReturnRouting (Ret): Lower aisles with boxes are visited first from left to right; upper aisles with boxes are visited afterwards from right to left. Each aisle except for the last lower aisle with a box is entered and exited at its front entry; the last lower aisle with a box is traversed entirely ([89]).
- S-ShapedRouting (S): First, the two leftmost aisles with boxes of both blocks are traversed upwards entirely. Second, upper aisles with boxes are visited from left to right where each aisle except for the rightmost is traversed entirely in the direction opposite to that of the previous aisle; the rightmost aisle is traversed entirely if it is entered at its rear entry, otherwise it is entered and exited at its front entry. Third, lower aisles with boxes are visited from right to left where each aisle except for the rightmost is traversed entirely in the direction opposite to that of the previous aisle; the rightmost aisle is traversed downwards entirely ([89]).
- LargestGapRouting (Gap): The largest gap of an aisle is its largest segment that contains no box, i.e., either the segment between two adjacent boxes, between front entry and lowermost box, or between rear entry and uppermost box. The largest gap of an aisle separates two parts of the aisle from each other: The lower (upper) part starts at the front (rear) entry and finishes at the lowermost (uppermost) point of the largest gap. First, lower parts of lower aisles with boxes are visited from left to right. Second, upper parts of lower aisles with boxes are visited from right to left. Third, lower parts of upper aisles with boxes are visited from left to right. Fourth, upper parts of upper aisles with boxes are visited from right to left. The rightmost aisle of each block is traversed entirely; in all other aisles, lower (upper) parts are entered and exited at the front (rear) entry ([89]).
- OptimalRouting (Opt): In Figure 6.2, set N according to the number of boxes in the batch assigned to the picker and c according to the distances between the boxes to each other and to the depot. Solve the MIP formulation in Figure 6.3. Visit the boxes in the order suggested by the obtained solution.

Simultaneous Batching and Routing Algorithm

- Optimal,Optimal (Opt,Opt): In Figure 6.4, set K, M, N, s, κ, u according to the available pickers and orders in the lookahead and c according to the distances between the boxes to each other and to the depot. Solve the MIP formulation in Figure 6.5. Assign orders to pickers and apply for each picker's boxes the visiting order as suggested by the obtained solution.

Sets

V	set of boxes and box locations with $V = \{1, \dots, N\}$

Parameters

N	number of boxes
c_{ij}	travel costs between boxes $i \in V$ and $j \in V$
c_{0j}	travel costs between depot and box $j \in V$
c_{j0}	travel costs between box $j \in V$ and depot
M	sufficiently large constant (big M)

Variables

$$x_{ij} = \begin{cases} 1 & \text{if box } i \in V \text{ is picked immediately before box } j \in V, \\ 0 & \text{else} \end{cases}$$

$$x_{0j} = \begin{cases} 1 & \text{if box } j \in V \text{ is the first box picked,} \\ 0 & \text{else} \end{cases}$$

$$x_{j0} = \begin{cases} 1 & \text{if box } j \in V \text{ is the last box picked,} \\ 0 & \text{else} \end{cases}$$

$T_j \geq 0$	picking start time of box $j \in V$

Figure 6.2: Sets, parameters and variables in the MIP formulation of the order routing problem.

$$\min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{j \in V} c_{0j} x_{0j} + \sum_{j \in V} c_{j0} x_{j0} \tag{6.1}$$

$$\text{s.t.} \quad \sum_{\substack{j \in V, \\ j \neq i}} x_{ij} + x_{i0} = 1 \qquad i \in V \tag{6.2}$$

$$\sum_{\substack{i \in V, \\ i \neq j}} x_{ij} + x_{0j} = 1 \qquad j \in V \tag{6.3}$$

$$\sum_{j \in V} x_{0j} = 1 \tag{6.4}$$

$$\sum_{j \in V} x_{j0} = 1 \tag{6.5}$$

$$T_i + 1 \leq T_j + \mathrm{M} \cdot (1 - x_{ij}) \qquad i, j \in V \tag{6.6}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V \tag{6.7}$$

$$x_{0j}, x_{j0} \in \{0, 1\} \qquad j \in V \tag{6.8}$$

$$T_j \geq 0 \qquad j \in V \tag{6.9}$$

Figure 6.3: MIP formulation of the order routing problem.

Sets

P	set of pickers including dummy picker with $P = \{1, \dots, K+1\}$
V	set of boxes and box locations with $V = \{1, \dots, N\}$
O	set of orders with $O = \{1, \dots, M\}$

Parameters

K	number of pickers
M	number of orders
N	number of boxes
s_i	number of boxes in order $i \in O$
κ_k	capacity of picker $k \in P$
$c_{j_1 j_2 k}$	travel costs of picker $k \in P$ between boxes $j_1 \in V$ and $j_2 \in V$
c_{0jk}	travel costs of picker $k \in P$ between depot and box $j \in V$
c_{j0k}	travel costs of picker $k \in P$ between box and depot $j \in V$
M	sufficiently large constant (big M)

$$u_{ij} = \begin{cases} 1 & \text{if box } j \text{ belongs to order } i, \\ 0 & \text{else} \end{cases}$$

Variables

$$x_{j_1 j_2 k} = \begin{cases} 1 & \text{if box } j_1 \in V \text{ is picked immediately before box } j_2 \in V \\ & \text{by picker } k \in P, \\ 0 & \text{else} \end{cases}$$

$$x_{0jk} = \begin{cases} 1 & \text{if box } j \in V \text{ is the first box picked by picker } k \in P, \\ 0 & \text{else} \end{cases}$$

$$x_{j0k} = \begin{cases} 1 & \text{if box } j \in V \text{ is the last box picked by picker } k \in P, \\ 0 & \text{else} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{if order } i \in V \text{ is served by picker } k \in P, \\ 0 & \text{else} \end{cases}$$

$$z_{jk} = \begin{cases} 1 & \text{if box } j \in V \text{ is picked by picker } k \in P, \\ 0 & \text{else} \end{cases}$$

$T_{jk} \geq 0$	picking start time of box $j \in V$ for picker $k \in P$

Figure 6.4: Sets, parameters and variables in the MIP formulation of the order batching and routing problem.

$$
\begin{aligned}
\min \quad & \sum_{k\in P}\sum_{j_1\in V}\sum_{j_2\in V} c_{j_1j_2k}x_{j_1j_2k} + \sum_{k\in P}\sum_{j\in V} c_{0jk}x_{0jk} + \sum_{k\in P}\sum_{j\in V} c_{j0k}x_{j0k} && && (6.10)\\
\text{s.t.} \quad & \sum_{k\in K_1}\sum_{\substack{j_2\in V,\\ j_2\neq j_1}} x_{j_1j_2k} + \sum_{k\in P} x_{j_10k} = 1 && j_1\in V && (6.11)\\
& \sum_{k\in P}\sum_{\substack{j_1\in V,\\ j_1\neq j_2}} x_{j_1j_2k} + \sum_{k\in P} x_{0j_2k} = 1 && j_2\in V && (6.12)\\
& \sum_{j_1\in V} x_{j_1jk} + x_{0jk} = \sum_{j_2\in V} x_{jj_2k} + x_{j0k} && j\in V,\ k\in P && (6.13)\\
& \sum_{j\in V} x_{0jk} \leq 1 && k\in P && (6.14)\\
& \sum_{j\in V} x_{j0k} \leq 1 && k\in P && (6.15)\\
& \sum_{j\in V} x_{0jk} = y_{ik} && i\in O,\ k\in P && (6.16)\\
& \sum_{j\in V} x_{j0k} = y_{ik} && i\in O,\ k\in P && (6.17)\\
& \sum_{i\in O} s_iy_{ik} \leq \kappa_k && k\in P && (6.18)\\
& \sum_{j_2\in V} x_{j_1j_2k} \leq z_{j_1k} && j_1\in V,\ k\in P && (6.19)\\
& \sum_{j_1\in V} x_{j_1j_2k} \leq z_{j_2k} && j_2\in V,\ k\in P && (6.20)\\
& \sum_{j\in V} u_{ij}z_{jk} = s_iy_{ik} && i\in O,\ k\in P && (6.21)\\
& \sum_{k\in P} y_{ik} = 1 && i\in O && (6.22)\\
& T_{j_1k}+1 \leq T_{j_2k} + \mathrm{M}\cdot(1-x_{j_1j_2k}) && j_1,j_2\in V,\ k\in P && (6.23)\\
& x_{j_1j_2k} \in \{0,1\} && j_1,j_2\in V,\ k\in P && (6.24)\\
& x_{0jk}, x_{j0k}, z_{jk} \in \{0,1\} && j\in V,\ k\in P && (6.25)\\
& y_{ik} \in \{0,1\} && i\in O,\ k\in P && (6.26)\\
& T_{jk} \geq 0 && j\in V,\ k\in P && (6.27)
\end{aligned}
$$

Figure 6.5: MIP formulation of the order batching and routing problem.

In the MIP formulations in Figures 6.3 and 6.5, we decide to get rid of the subtour elimination constraints by establishing precedence relations between pick operations based on decision variables for time instants at which boxes are picked in Constraints 6.6 and 6.23, respectively ([132]). In the order batching and routing problem (cf. Figure 6.5), we have to introduce additional variables y_{ik} and z_{jk} to express the (exclusive) assignment of order i (i.e., all of its associated boxes j) to picker k. In particular, Constraints 6.18 to 6.22 ensure that picker capacities are respected and that all boxes of an order are picked by the same picker.

6.1.1 Average Results

All combinations of batching and routing algorithms as well as the simultaneous algorithm were tested. We preface the discussion by noting that rule-based routing policies like Ret, S or Gap are commonly accepted in practice because of their simplicity, whilst optimal routes as calculated by Opt or the simultaneous approach in general yield shapelessness routes which my be hard to follow for a picker and thereby represent an additional source of failure. A summary of the simulation results obtained for all considered performance criteria is given in Appendix A.3.1; we restrict attention to two selected performance criteria:

- Total travel distance in kilometers
- Box throughput in boxes per hour

Figure 6.6 shows that – apart from algorithms which rely on a batching determined by Prio – a substantial reduction in the total distance covered by all pickers is achieved as a result of providing time lookahead. For lookahead durations $D \geq 300$, the optimization potential appears satiated and the marginal benefit of an additional minute of lookahead becomes approximately zero.

The positive effect is explained by the same change of the rule set as experienced already in the TSP in Chapters 4.3 and 5.4: Since a box may be picked up as soon as it becomes known, increasing the lookahead duration leads to increased probabilities for encountering spatially proximate boxes which then can be consolidated in a tour. Hence, distance savings as observed in the TSP carry over to this realistic problem setting which takes random disturbances into account. However, there are slight differences: Because of the bounded picker capacity, lookahead cannot be exploited as excessively as in the unrestricted case. Pickers must return to the depot after ten boxes have been picked; if picker capacities were unlimited, they could pick all boxes assigned to them in a single tour if full a-priori knowledge was provided.

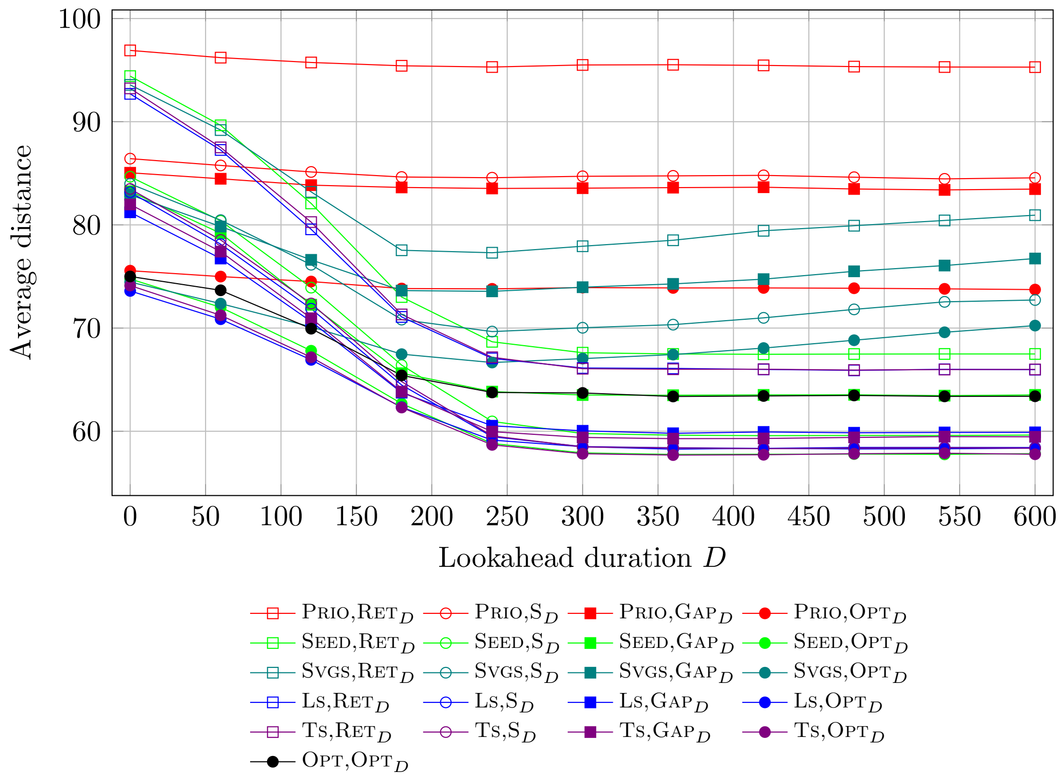


Figure 6.6: Average distances for different lookahead sizes and $n = 625$ in the order picking system.

Batching by PRIO suffers poor overall performance regardless of the lookahead level and is eliminated from further consideration; SVGS is also considered inferior and repelled as it consistently loses to all remaining batching policies; these two batching algorithms are also outperformed on the other objectives (see Appendix A.3.1). We recognize that the right batching decision is an essential first step to capitalize from the potential of lookahead.

Batching policies SEED, LS and TS exhibit comparable but not identical behavior. We draw a more precise picture based on the selected routing strategy: Routing policy RET fails in comparison to the other routing strategies by cause of its naive approach. Concerning routing policy OPT, all three batching algorithms are considered equal. Under GAP routing and small lookahead, LS batching has slight advantages over the two other batching rules, whereas in case of medium to large information preview TS excels LS and SEED. Under routing policy S, all algorithms exhibit the same quality; however, performance is degraded as compared to OPT's routing for small lookahead. Note that there are no deadheads after all boxes in an aisle have been collected in S's routing as opposed to GAP's routing.

Simultaneously solving the batching and routing problem as targeted by OPT,OPT is found computationally impracticable to obtain solutions quickly: Whenever more than ten orders, i.e., up to 30 boxes, are open, we had to forfeit almost always the exact reoptimization approach in favor of applying SEED,GAP as a substitute to keep computational efforts low. For this reason, the plots of OPT,OPT and SEED,GAP coincide for $D \geq 180$. This means that under a three-hour lookahead too many orders and boxes are known as to guarantee that OPT,OPT could terminate within less than 120 seconds. For the order routing problem alone, computing time is uncritical because of picker capacities delimiting problem sizes.

Based on the total distance objective function only, we come to the interim conclusion that TS, LS and SEED are deemed the most promising batching policies along with either S or OPT as recommendable routing strategies. Exact reoptimization is found to be computationally too hard so as to elicit its full potential.

Before we proceed with the assessment of the results on the box throughput, we first note that total travel distance and box throughput are no competing optimization goals by definition. Thus, we do not expect throughput to be negatively affected by the implicit rationale of all batching and routing policies which lies in finding routes as short as possible. Note that throughput is to be maximized and used as a frequent key performance indicator for chief operating officers in warehouses to estimate the overall efficiency of the system consisting of pickers, orders and the warehouse configuration.

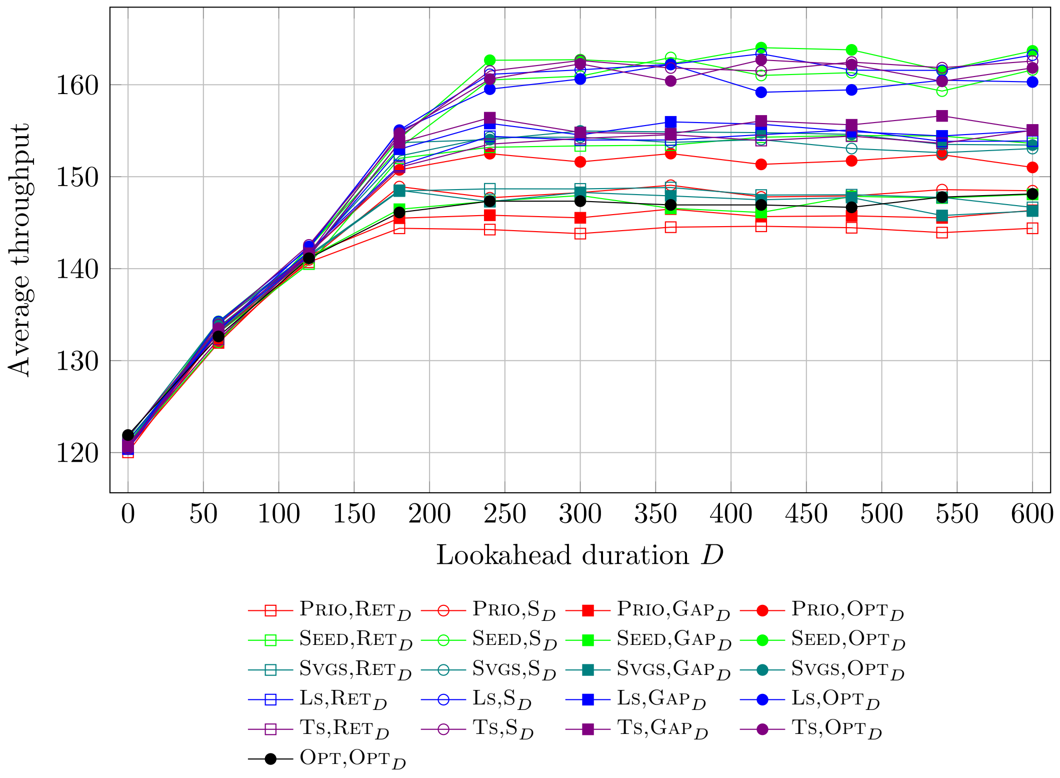


Figure 6.7: Average throughput for different lookahead sizes and $n = 625$ in the order picking system.

The box throughput attained by the pickers as imputed by the decisions of respective algorithms is entirely affirmative to the previously found ranking of batching and routing algorithms based on the total distance objective as can be seen in Figure 6.7.

While differences in throughput as accomplished by the different algorithms are virtually non-existent for small lookahead durations, batching algorithms Ts, Ls and Seed unfold their potential and consistently outperform all other batching policies as well as the exact reoptimization method for medium to large lookahead window. Routings determined by S and Opt clearly outperform the other routing strategies.

6.1.2 Distributional Results

Since routing policy Opt intrinsically leads to the shortest route for a given batch, we restrict the discussion to this routing strategy. Analogous results can be derived for all other routing policies by consulting Appendix A.3.1. Because batching algorithm Ts has been identified as one of the top candidates, we select the combined batching and routing policy Ts,Opt_{600} to be the reference for performance ratios relative to the "best" offline algorithm.

Figure 6.8 confirms the positive effect of lookahead on the total distance. Concerning batching algorithm candidates Ts, Ls and Seed, a perfect ordering of the empirical counting distribution functions is observed for successive lookahead durations when the larger one of them is at most $D = 240$. Hence, the results point to an exclusive lookahead effect for these information regimes. For lookahead durations $D \geq 300$, we find that empirical counting distribution functions intransparently cross each other for countless times; hence, no exclusive benefit can be attested for these lookahead durations and the marginal benefit of an additional time unit of lookahead is approximately zero. Each of the plots has a large steepness in a characteristic interval of total distance values and nearly no steepness elsewhere indicating that each algorithm corresponds to a specific range of total distances. From the small confidence intervals with widths collectively smaller than 1 kilometer for each combination of algorithm and lookahead level as well as from the tiny coefficients of variation collectively not larger than 0.03 in Table A.70 of Appendix A.3.1, we infer representativeness of the obtained total distances for the given warehouse and picker configuration. However, we also discover in the table that for larger lookahead durations, instances where additional lookahead leads to a degradation in the objective value are encountered every now and then. Yet, since the marginal benefit of lookahead in this information regime is negligible anyway, this effect is considered unimportant. In fact, the numerous intersection points of the plots corresponding to these lookahead durations are explained by this observation.

The empirical counting distribution functions of the performance ratio for the total distance relative to Ts,Opt$_{600}$ and relative to the online version of the algorithms are displayed in Figures 6.9 and 6.10, respectively. Experimental competitive ratios are not larger than 1.33 for batching algorithms Ts, Ls and Seed as also seen in Table A.71 of Appendix A.3.1. The distance of the plots for two successive lookahead levels to each other admits the same conclusions as already drawn from the total distance distributions. For lookahead durations $D \geq 300$, performance ratios relative to Ts,Opt$_{600}$ appear centered around the value 1, i.e., deterioration – albeit only at a small magnitude – in consequence of additional lookahead is encountered on a regular basis. Relative to the pure online version of an algorithm, additional lookahead leads to performance ratios smaller than 1 for batching algorithm candidates Ts, Ls and Seed on the largest part of input sequences (cf. Table A.71 of Appendix A.3.1).

Figures 6.11 to 6.13 illustrate the distributional results for the box throughput. For the empirical counting distribution functions of the throughput it is desirable to have larger parts of the distribution on larger values, i.e., by lookahead we intend to shift the plots to the right. We observe a qualitative difference compared to the distributional results of the total distance: The variability of the throughput as well as of performance ratios relative to the online versions of the algorithms increases considerably for increasing lookahead durations as confirmed by the coefficients of variation and confidence intervals in Tables A.76 and A.78 of Appendix A.3.1; likewise, the variability of performance ratios relative to Ts,Opt$_{600}$ decreases considerably for increasing lookahead durations (see Table A.77 of Appendix A.3.1): In the pure online case, orders arrive over the whole time horizon of 600 minutes and the throughput is considerably influenced by the arrival process, particularly by the last orders; under full lookahead the throughput is a sole consequence of the pickers' efficiency on coping with the input sequence that was revealed at the outset. Hence, throughput is inherently throttled and regulated in the online case leading to a much more invariant behavior as when the system is allowed to unfold its throughput freely. This explains the high throughput variability in the case of large lookahead as opposed to small informational preview.

We conclude this section by pointing out that in order picking where boxes can be picked up once they are known, massive improvements in all goals could be observed as a result of an enlarged planning basis for pick list and route generation when additional information is provided. Since objectives are not conflicting, all goals can be improved and no trade-offs between different goals have to be taken into account by the algorithms. From a managerial point of view, we recommend to install technical devices which allow for the retrieval of lookahead information and to check whether operating strategies currently implemented in the warehouse are conform with potential warehouse efficiency as extracted by simulation under batching policies Ts, Ls and Seed combined with routing policies S and Opt.

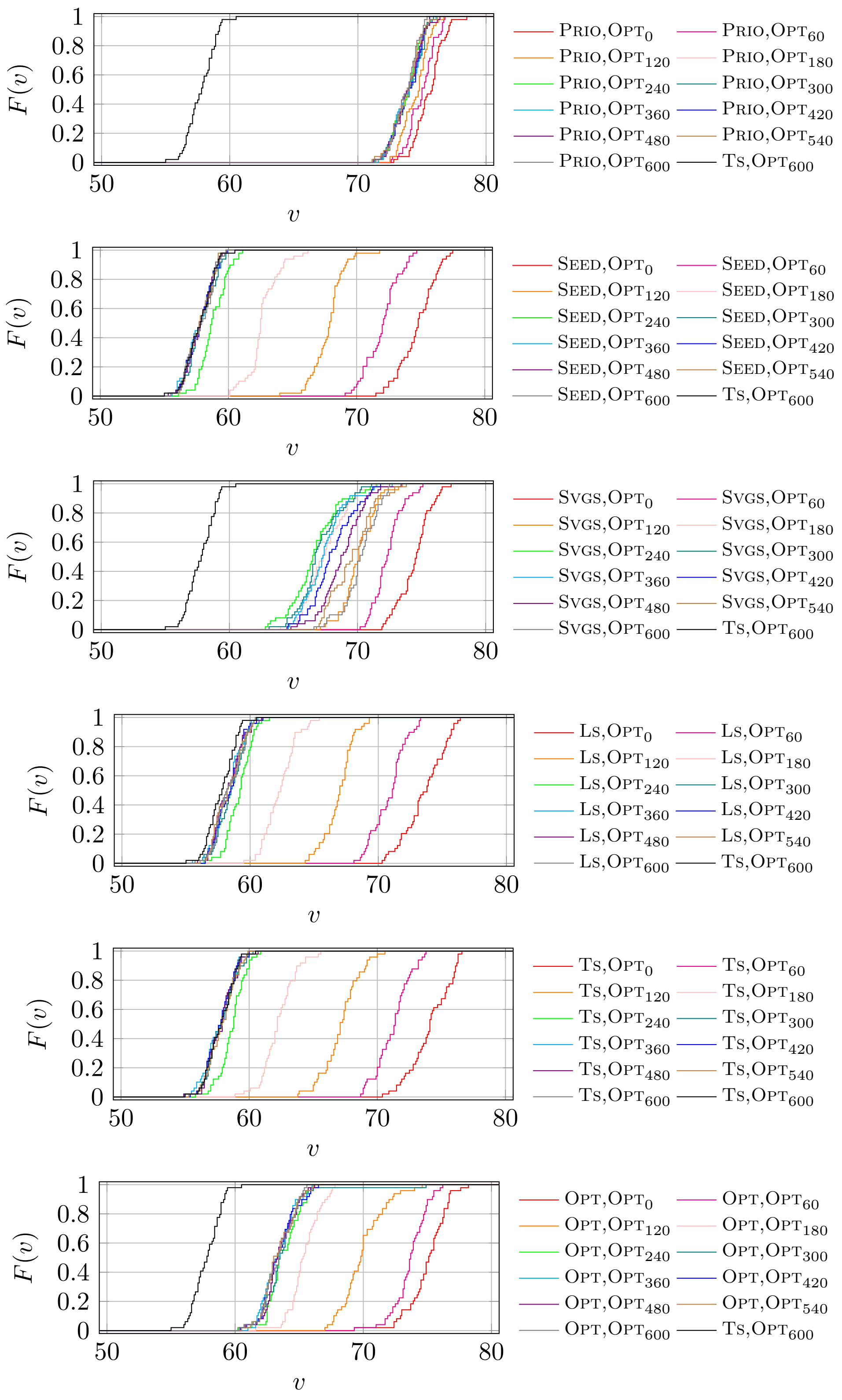


Figure 6.8: Empirical counting distribution functions of distance for $n = 625$ in the order picking system.

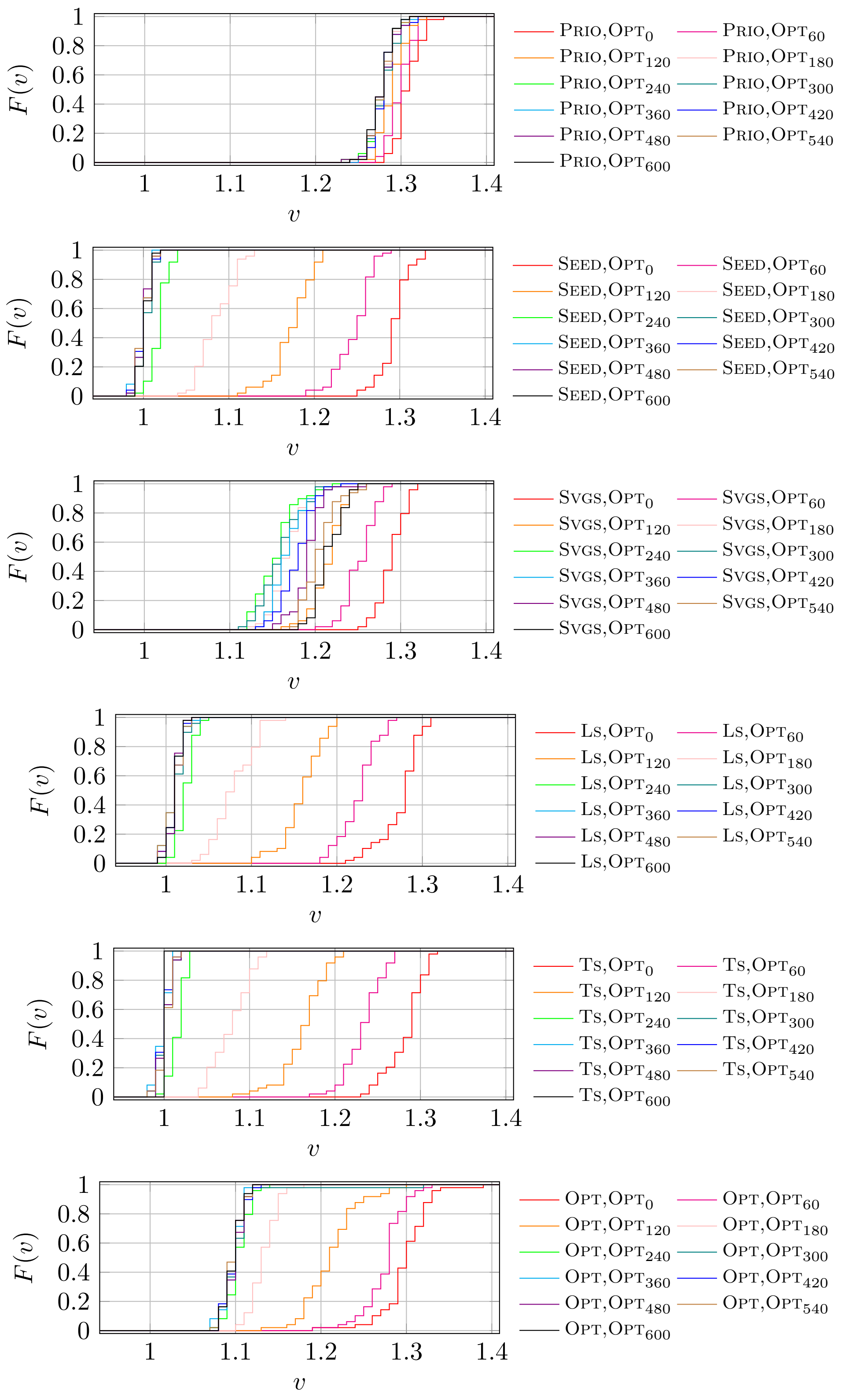


Figure 6.9: Empirical counting distribution functions of performance ratio of distance relative to TS,OPT$_{600}$ for $n = 625$ in the order picking system.

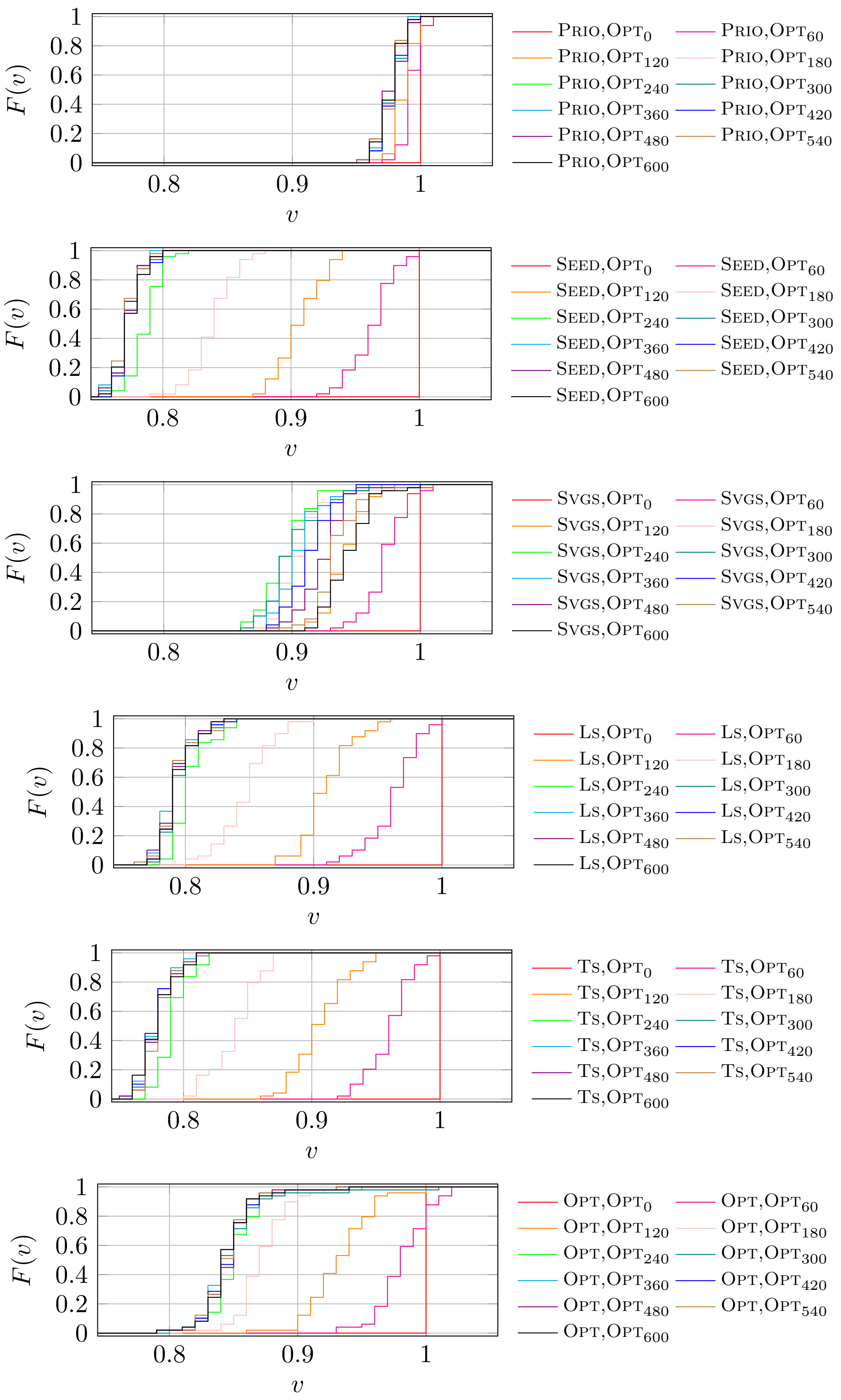


Figure 6.10: Empirical counting distribution functions of performance ratio of throughput relative to the online version for $n = 625$ in the order picking system.

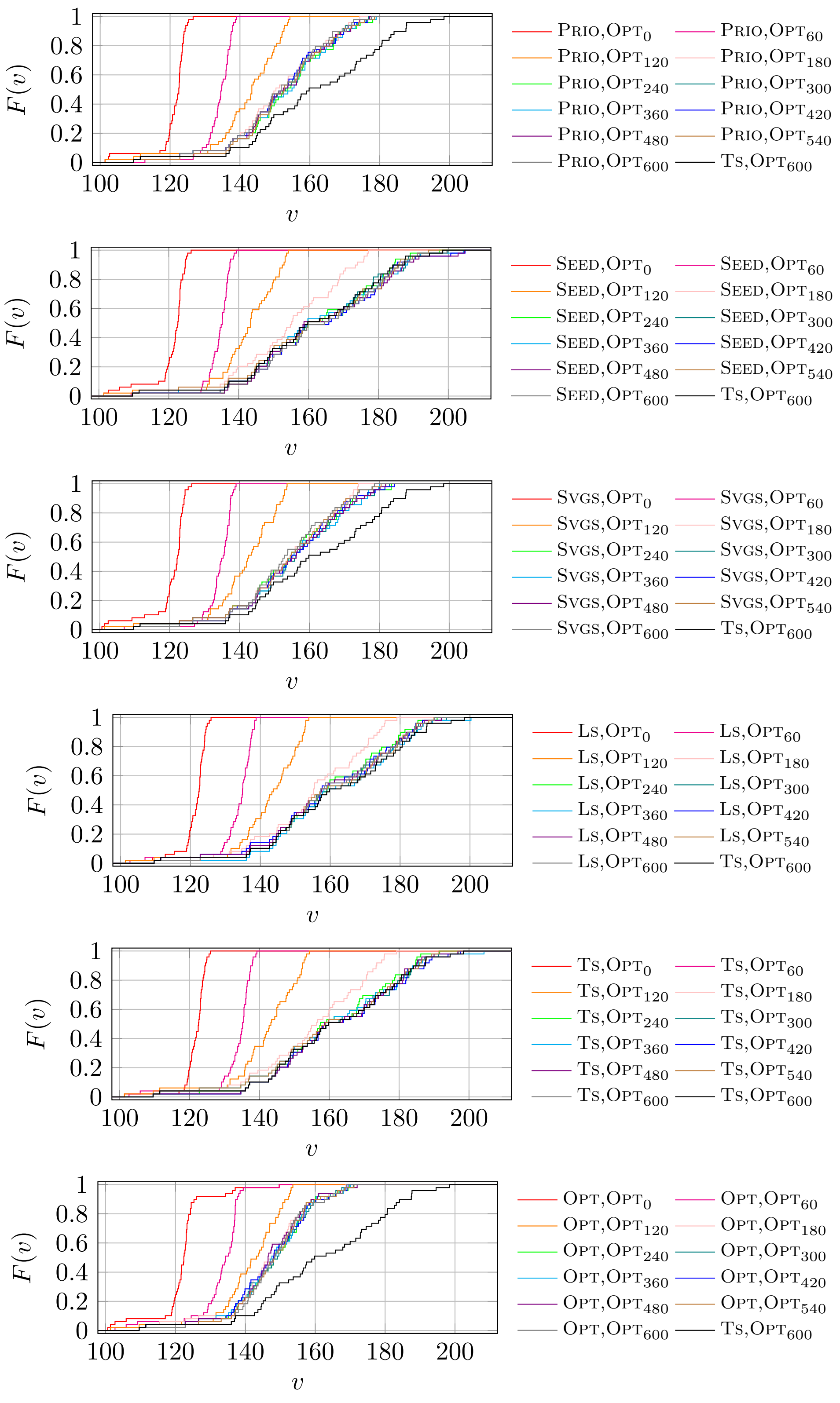


Figure 6.11: Empirical counting distribution functions of throughput for $n = 625$ in the order picking system.

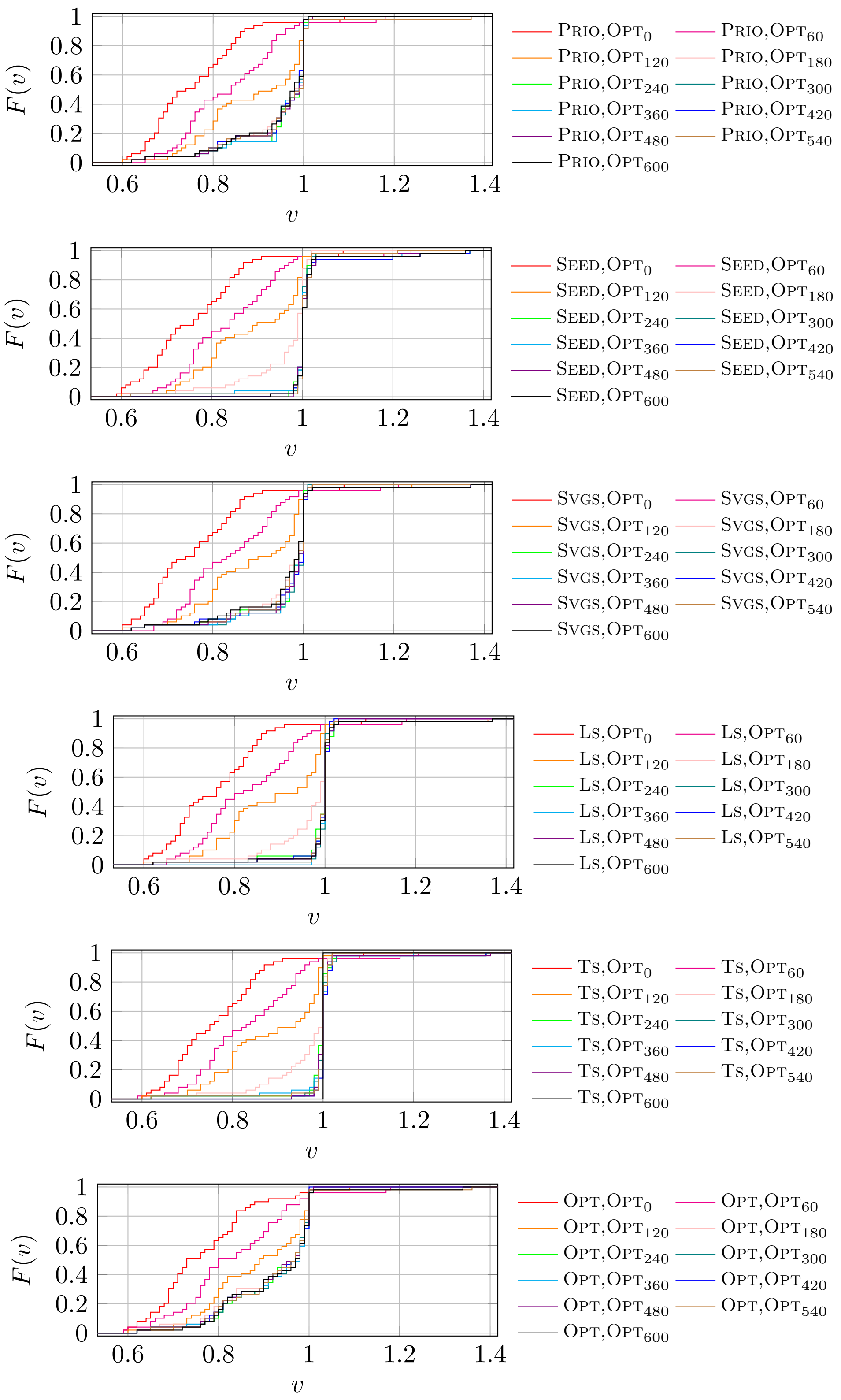


Figure 6.12: Empirical counting distribution functions of performance ratio of throughput relative to TS,OPT$_{600}$ for $n = 625$ in the order picking system.

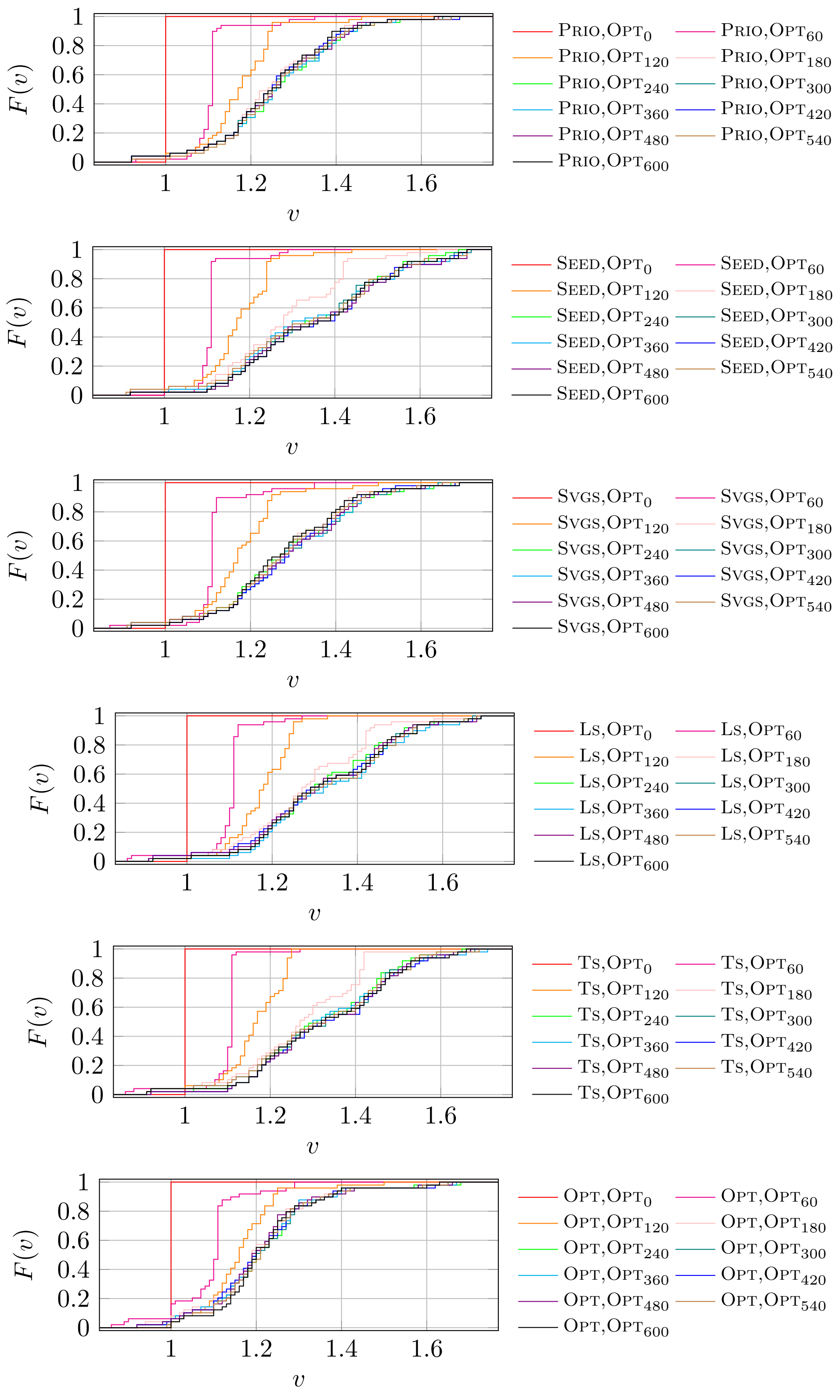


Figure 6.13: Empirical counting distribution functions of performance ratio of throughput relative to the online version for $n = 625$ in the order picking system.

6.2 Online Pickup and Delivery with Lookahead

In a pickup and delivery service, customers specify transportation orders between individual origins and destinations in a road network. In addition, customers provide preferred time windows for their pickup and delivery time. Transportation orders are served by a fleet of vehicles and each vehicle has to start and end its routes at an individual depot ([27], [53]).

Dial-a-ride problems are special types of pickup and delivery problems where apart from travel costs also user convenience in the sense of service quality matters ([53]). While this may not be relevant to the transportation of goods, it definitely is for the transportation of persons or perishable freight, e.g., in patient ambulance, taxi services, school bus routing, or food provision ([158]). Real world pickup and delivery services have to take into account multiple criteria as intended by dial-a-ride problems due to customer requirements.

Typically, transportation orders arrive throughout the day and the objective is to make the vehicles pick up and deliver all transportation orders in a way that meets the decision maker's goal system best. Figure 6.14 shows the road network under consideration along with charts for some performance indicators of interest.

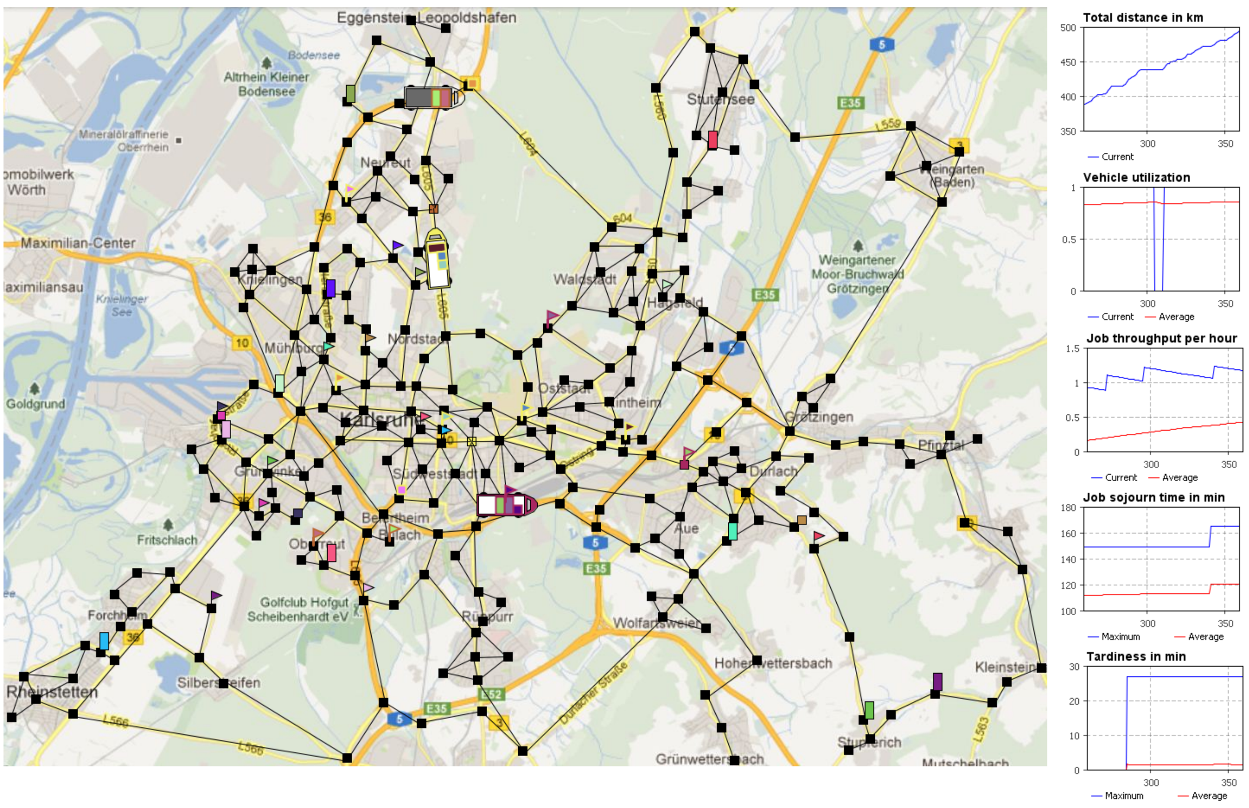


Figure 6.14: Animation of the simulation model for a pickup and delivery service in AnyLogic.

The network represents the urban and suburban region of Karlsruhe and consists of 269 central points and 449 major roads between them. The diameter of the network is 28.5 kilometers; each road is prescribed a specific maximum allowed speed limit. Three vehicles have to serve $n = 50$ customer orders arriving over a work day of 600 minutes plus potential overtime. When a vehicle is not working because of a break or absence without leave (no-show), its loading space is displayed in gray (cf. Figure 6.14). An order may consist of up to two units to be transported and the vehicle capacity amounts to five units.

Vehicles are subject to the traffic scene encountered during their service rides, i.e., they have to adapt to reduced speed limits whenever a traffic jam forbids traveling at regular speed on a road. Hence, traffic jams represent additional random events in our simulation model.

Order arrival and data are random, i.e., release time, number of units to be transported, pickup time, delivery time, pickup time window, delivery time window, pickup location and delivery location of an order are realizations of random variables that are unknown to the algorithms which have to determine the routes of the vehicles. Pickup and delivery time windows refer to the preferred service start times of pickups and deliveries, respectively. In addition, pickup and delivery operations underlie the following random influences:

- Vehicle break start and end time
- Vehicle no-show occurrence
- If applicable, vehicle no-show start and end time

Because of traffic jams and the large number of random processes, it is out of scope to give an exact formulation of an optimization problem that takes into account all of these features. Hence, simulation is deemed a suitable method of analysis.

At the end of a run of an independent simulation replication, the decision maker is supplied with the following quality indicators for the routes which are used to judge on the quality of the responsible algorithm:

- Makespan, i.e., the time when the last vehicle returns to its depot
- Total distance covered by all vehicles
- Tardiness, i.e., the mean tardiness over all pickup and delivery orders
- Maximum tardiness, i.e., the maximum tardiness over all pickup and delivery orders
- Vehicle utilization, i.e., the mean percentage of working vehicles over the time horizon
- Order throughput, i.e., order fulfillment rate

Lookahead appears as time lookahead of duration $D \in \{0, 60, 120, \ldots, 600\}$ minutes. Once a transportation request arrives, it has to be waited until the corresponding time window starts before the order can be picked up or delivered, respectively: It is not allowed to incur an earliness both in the pickup and delivery of an order. Hence, earliest start times of transportation orders under lookahead are preserved from the pure online case. Lookahead does not imply a change of the rule set in favor of the decision maker and only serves as a means of forwarded information release.

In our computational experiments, we draw $m = 100$ independent simulation replications by initializing the random number generator with a different seed and ensuring independence of all stochastic processes in each replication.

An algorithm is required to determine the routes for all vehicles available at that time so as to fulfill the known and yet unfulfilled transportation orders in a way that best matches the decision maker's preferences with respect to the quality criteria specified above. We agree upon algorithm execution whenever the current situation changes as a result of a transportation order arrival or a vehicle's re-entry in the system after a break or no-show.

Despite obvious similarities to the order picking system, some features make the pickup and delivery service substantially different and demand for other algorithmic approaches:

- Temporal restrictions have to be regarded.
- Earliest start times for pickups and deliveries are not forwarded but retained from the pure online setting.
- Assignment decisions may be revoked as long as an order has not yet been picked up, i.e., a route assigned to a vehicle is not fixed when the vehicle leaves the depot.

Managing a pickup and delivery service comprises solving the two interrelated subproblems of assigning each customer order to a vehicle (batching) and calculating scheduled routes[17] for each vehicle according to the assigned orders (routing). Because of the factor time, not only spatial proximity of locations with respect to a vehicle's current position but also temporal proximity of time windows with respect to the current time has to be taken into account by an algorithm both in its batching and routing decisions. Moreover, each pickup of an order has to be seen in logical conjunction with the corresponding delivery operation. Therefore, the variety of solution methods is more limited than for similar problems without time windows and there is no a-priori subdivision into batching and routing algorithms as outlined in the order picking system (cf. Chapter 6.1).

[17] Because a route in a pickup and delivery service has to start and end in the vehicle's depot, the decision version of the "problem" contains `Hamiltonian Circuit` which is known to be an $\mathcal{NP}$-complete problem.

Note that earliest pickup and delivery times represent hard constraints which may not be violated. As a result, incurring earliness is not possible and the scheduling problem of finding the timings for a given sequence of pickup and delivery operations becomes trivial since we schedule each operation as early as possible. In the following, we tacitly assume rescheduling in this form to be carried out upon each change of a vehicle's route.

Some algorithms resort to a performance measure for the quality of a route during their computations. To this end, the quality of a route is evaluated by aggregating total travel distance, average tardiness and maximum tardiness into an auxiliary objective function in form of a linear combination of these three performance indicators (scalarization). Low objective values are deemed to correlate with high quality routes. We remark that each of the following algorithms takes into account that a vehicle may currently have already picked up but not yet delivered units on board, e.g., by inserting a time-uncritical dummy pickup of load zero at the delivery location. Except for the tabu search heuristic and the exact reoptimization approach, all of the following algorithms represent modified versions of the algorithms provided by Kallrath ([101]) for vehicle routing problems arising on hospital campuses. The neighborhood structure in the tabu search algorithm is taken from the setting of dial-a-ride problems discussed by Cordeau and Laporte ([53]).

Sequential algorithms

- SequencingReassignmentHeuristic (Srh):
    1. *Assignment of orders to vehicles*: Sort unassigned orders by non-decreasing earliest pickup times and assign orders within a time slice of prescribed length (e.g., 100 minutes) to vehicles by a modified first fit rule which ensures that orders with close earliest pickup time (e.g., less than 25 minutes) are not assigned to the same vehicle. Assign previously unassigned orders by the common first fit rule.
    2. *Route construction*: Create a route for each vehicle by successively inserting its assigned pickup and delivery locations at best possible points in terms of a minimum objective value increase.
    3. *Route improvement by resequencing*: Remove in each vehicle's route an order (both pickup and delivery location) with maximum tardiness, if any, and reinsert its pickup and delivery locations at best possible points in terms of a maximum objective value decrease until no further improvement is possible.
    4. *Route improvement by reassignment*: Remove an order with maximum lateness in each vehicle's route, if any, and reinsert its pickup and delivery locations in another vehicle's route at best possible points in terms of a maximum total objective value

decrease until no further improvement is possible. In order to choose the vehicle which receives the order, check all vehicles and select one that leads to smallest total objective value.

5. *Route improvement by resequencing*: See step 3 ([101]).

- 2Opt (2Opt): Obtain initial routes for each available vehicle by applying Srh. Apply to each vehicle's route algorihm 2Opt$_l$ as outlined in Chapter 5.4 for the TSP without the first step where l equals the number of pickup and delivery locations; use route quality as the objective and ensure that feasible sequences of pickup and delivery locations are obtained (see also [101], [120]).

- SimulatedAnnealing (Sa): A swap move in a route consists of exchanging the positions of two locations if a feasible sequence of pickup and delivery locations is obtained; a shift move in a route consists of shifting a number of successive locations to another position in the route if a feasible sequence of pickup and delivery locations is obtained. Obtain initial routes for each available vehicle by applying Srh. Apply to each vehicle's route algorihm Sa$_l$ as outlined in Chapter 5.4 for the TSP without the first step and step 4 replaced by

  4. Obtain H' by performing a swap or shift move on H at random.

  where l equals the number of pickup and delivery locations; use route quality as the objective ([101])

- TabuSearch (Ts): Let the neighborhood of a route set consist of all route sets which emanate from removing the pickup and delivery locations of an order from a first route and inserting them at the best possible points of a second route in terms of a minimum objective value increase. In Ts, the auxiliary objective function for route quality is modified by adding a penalty term proportional to the number of times that the move resulting in the neighboring route set has been applied previously. A route is called tabu if it results from reinserting an order which has been removed from it within a prescribed maximum number of immediately preceding iterations. Obtain initial routes for each available vehicle by applying Srh and set the current route set to this solution. Repeatedly set the current route set to a route set with minimum total objective value among all route sets in the neighborhood of the current route set such that each of the contained routes is non-tabu until no further improvement is made over a prescribed number of iterations ([53]).

Simultaneous algorithm

- OPTIMAL (OPT): In Figure 6.15, set K, N, s, l, a, b, κ, l^{init} according to the available vehicles and orders in the lookahead and c, t according to the distances and travel times between the locations to each other, to the depots and to the current vehicle locations. Solve the MIP formulation in Figure 6.16. Assign orders to vehicles and route them as suggested by the obtained solution.

Sets

C	set of vehicles and current vehicle locations with $C = \{1, \ldots, K\}$
P	set of pickup locations with $P = \{K+1, \ldots, K+N\}$
C'	set of vehicle depot locations with $C' = \{K+N+1, \ldots, 2K+N\}$
D	set of delivery locations with $D = \{2K+N+1, \ldots, 2K+2N\}$
V	set of all locations with $V = C \cup C' \cup P \cup D$

Parameters

K	number of vehicles
N	number of transportation orders
c_{ijk}	travel costs (e.g., distance) between locations $i \in V$ and $j \in V$
t_{ijk}	travel time between locations $i \in V$ and $j \in V$
s_i	service time at location $i \in V$
l_i	load to be picked up / delivered at location $i \in V$; delivery loads are the negative value of corresponding pickup loads
a_i	earliest possible service start time at location $i \in V$
b_i	latest possible service start time at location $i \in V$
κ_k	capacity of vehicle $k \in C$
l_k^{init}	initial load of vehicle $k \in C$
$\alpha_1, \alpha_2, \alpha_3$	weights for distance (α_1), tardiness (α_2), maximum tardiness (α_3)
M	sufficiently large constant (big M)

Variables

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \in C \text{ visits location } j \in V \text{ immediately after location } i \in V, \\ 0 & \text{else} \end{cases}$$

$T_{ik} \geq 0$	service start time of vehicle $k \in C$ at location $i \in V$
$T_{ik}^{tardy} \geq 0$	tardiness of service start time of vehicle $k \in C$ at location $i \in V$
$d_{max} \geq 0$	maximum tardiness of service start times
$L_{ik} \geq 0$	load of vehicle $k \in C$ after service at location $i \in V$

Figure 6.15: Sets, parameters and variables in the MIP formulation of the pickup and delivery problem.

$$
\begin{aligned}
\min \quad & \alpha_1 \sum_{k\in C}\sum_{i\in V}\sum_{j\in V} c_{ijk}x_{ijk} + \alpha_2 \sum_{k\in C}\sum_{i\in V} T_{ik}^{tardy} + \alpha_3 d_{max} && && (6.28)\\
\text{s.t.} \quad & \sum_{k\in C}\sum_{j\in P\cup D} x_{ijk} + x_{i,K+N+k,k} = 1 && i\in P\cup D && (6.29)\\
& \sum_{j\in P\cup D} x_{ijk} = \sum_{j\in P\cup D} x_{j,K+N+i,k} && i\in P,\ k\in C && (6.30)\\
& \sum_{i\in P\cup D} x_{ijk} + x_{kjk} = \sum_{i\in P\cup D} x_{jik} + x_{j,K+N+k,k} && j\in P\cup D,\ k\in C && (6.31)\\
& \sum_{j\in P} x_{kjk} + x_{k,K+N+k,k} = 1 && k\in C && (6.32)\\
& \sum_{i\in D} x_{i,K+N+k,k} + x_{k,K+N+k,k} = 1 && k\in C && (6.33)\\
& a_i \le T_{ik} \le b_i + T_{ik}^{tardy} && i\in P\cup D,\ k\in C && (6.34)\\
& T_{ik}^{tardy} \le d_{max} && i\in V,\ k\in C && (6.35)\\
& T_{ik} + s_i + t_{ijk} \le T_{jk} + \mathrm{M}\cdot(1-x_{ijk}) && i,j\in V,\ k\in C && (6.36)\\
& T_{ik} + t_{i,K+N+i,k} \le T_{K+N+i,k} && i\in P,\ k\in C && (6.37)\\
& L_{ik} + l_j \le L_{jk} + \mathrm{M}\cdot(1-x_{ijk}) && i,j\in V,\ k\in C && (6.38)\\
& L_{jk} - l_j \le L_{ik} + \mathrm{M}\cdot(1-x_{ijk}) && i,j\in V,\ k\in C && (6.39)\\
& l_i \le L_{ik} \le \kappa_k && i\in P\cup D,\ k\in C && (6.40)\\
& L_{kk} = l_k^{init} && k\in C && (6.41)\\
& x_{ijk} \in \{0,1\} && i,j\in V,\ k\in C && (6.42)\\
& T_{ik}, T_{ik}^{tardy}, L_{ik}, d_{max} \ge 0 && i\in V,\ k\in C && (6.43)
\end{aligned}
$$

Figure 6.16: MIP formulation of the pickup and delivery problem.

In the MIP formulation in Figure 6.16, the Objective Function 6.28 minimizes the auxiliary objective function which takes into account travel costs and tardinesses. Constraints 6.29 to 6.33 make sure that each pickup and delivery location is serviced, that a pickup by vehicle k is also associated with a delivery by vehicle k, that each approached location is left again, and that each vehicle starts in its current location and ends in its depot. Constraints 6.34 to 6.37 govern the temporal course of events such that time windows are met as good as possible, travel and service times are respected, and pickups are scheduled before deliveries. Constraints 6.38 to 6.41 take care of the loads of the vehicles at each time.

6.2.1 Average Results

All sequential algorithms as well as the simultaneous algorithm were tested. A summary of the simulation results obtained for all considered performance criteria is given in Appendix A.3.2; we restrict attention to three selected performance criteria which already illustrate the trade-offs between competing goals:

- Total travel distance in kilometers
- Tardiness of pickup and delivery operations in minutes
- Order throughput in orders per hour

Figure 6.17 shows the average total distance covered by all vehicles for different lookahead durations. Apart from OPT, all algorithms possess the potential to acquire reductions in the total travel distance by providing additional lookahead time. However, because of time windows, improvement through lookahead in the total travel distance appears neither as drastic nor as reliable as in all previously considered problem settings that were based on the TSP with allowed immediate service of requests. We attribute the major degree of unpredictability concerning the travel distance to the algorithms' rationale which also opts at minimizing average and maximum tardiness by means of the auxiliary objective function.

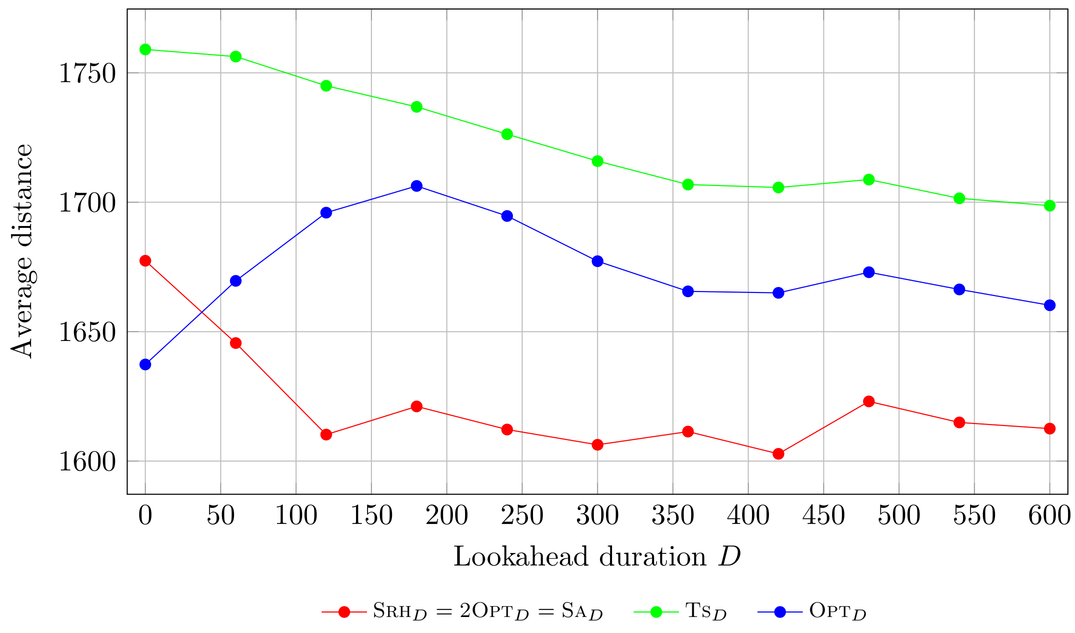


Figure 6.17: Average distances for different lookahead sizes and $n = 50$ in the pickup and delivery service.

Srh, 2Opt and Sa are found to fare best. These algorithms exhibit identical behavior which means that applying edge exchanging moves as well as swap and shift moves on the route set determined by Srh did not result in a single route improvement. Hence, algorithms 2Opt and Sa which are focused on intra-route improvement ([101]) offer no additional benefit. This is explained by the already elaborate route construction of Srh based on a best insertion policy and the problem-related difficulty of obtaining feasible routes upon route modifications due to time windows in conjunction with logical precedence restrictions between pickup and delivery operations. Although Ts also resorts to routes initially determined by Srh, the added possibility of order reassignments from one route to another – which was not taken into account by 2Opt and Sa – leads to structurally different routes that are obviously worse for the total distance criterion. Yet, for other objective functions we will see that the inter-route improvement ([101]) approach of Ts is profitable. Concerning the behavior of exact reoptimization by Opt, we observe that lookahead seems to lead to unstable routings as already figured out for the TSP in Chapter 5.4: Partial solutions which may be advantageous for the given snapshot situation may turn out disastrous whenever the current situation changes as new transportation orders pop up; as a result, subsequent partial solutions may exhibit a substantially different character.

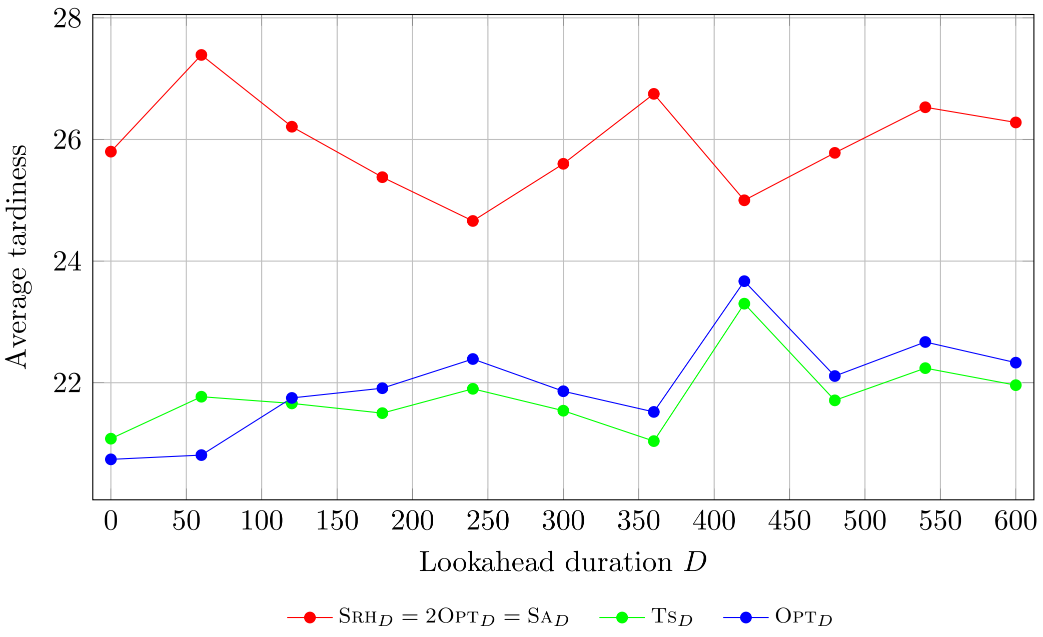


Figure 6.18: Average tardinesses for different lookahead sizes and $n = 50$ in the pickup and delivery service.

The drawn picture of algorithm quality overly changes when the average tardiness of the vehicles over all pickup and delivery operations is deemed the predominant performance yardstick: Figure 6.18 suggests algorithms Ts and Opt as the most promising algorithm candidates in order to keep tardiness low which is in sharp contrast to superiority of SRH-based algorithms with respect to the total distance criterion. At this point, we recall that route quality is assessed by an auxiliary objective function that takes on the form of a linear combination of total distance, mean tardiness and maximum tardiness. Hence, algorithms tend to shift their focus on whatever optimization goal can be addressed best by their rationale. In this sense, algorithm TS fares best with respect to tardiness-related objectives by trading an increase in the total distance for a decrease in the average (and maximum) tardiness. We note that with respect to our selected objective function, TS yielded routes with lower aggregated objective value; however, we refrain from drawing general conclusions upon the objective value dimension due to the drawbacks of scalarization in multicriteria optimization (see also [70]).

Unfortunately, none of the algorithms is found to benefit from lookahead with respect to tardiness-related goals. Quite to the contrary, even sophisticated algorithms like TS and OPT have to struggle with the instability of "locally" good solutions whose advantages are likely to be relinquished in the upcoming part of the request sequence. Instead, ad-hoc planning without taking into account any additionally provided future information is advisable because large deviations between previously calculated routes and actual travel routes are likely to occur anyway as a result of changed circumstances once new orders arrive. Because already in the pure online setting, at each time there are enough unfulfilled orders to induce high vehicle utilization (cf. Table A.91 of Appendix A.3.2), it suffices to consider the transportation orders known in this case. The parallel behavior of TS and OPT is due to our stipulation to use TS as a substitute for OPT in case of more than ten orders so as to guarantee reasonable computational effort. After a closer look at the average distance in Figure 6.17, we also recognize the parallel behavior of TS and OPT in this objective for $D \geq 180$. Hence, we draw the same conclusion as in the order picking system: When the lookahead duration exceeds three hours, problem sizes become too large as to guarantee that instances of OPT can be solved within the prescribed computing time limit of 120 seconds. Observe that the parallel but shifted behavior also illustrates the different trade-offs of the algorithms.

After a glance at the first two objectives, we come to the interim conclusion that the general dilemma of multicriteria optimization is preserved even if large lookahead capabilities are provided and that it is not as easy to design algorithms compliant with all objectives in a system of conflicting goals as in a system of complementing or independent goals like in the order picking system in Chapter 6.1. Since creating short distance routes may only come along with the price of large violations in the time window constraints, we recognize that in the application under consideration the decision maker has to be aware of his own trade-off relations for different goals in order to reach a final decision on algorithm quality.

The average throughput of transportation orders as achieved by the different algorithms is shown in Figure 6.19.

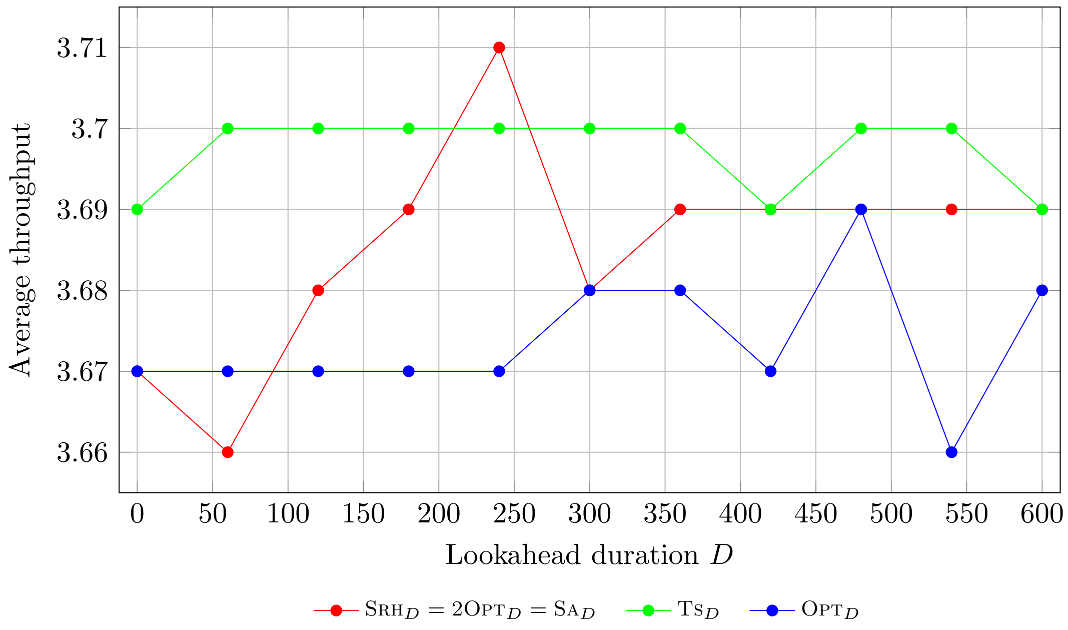


Figure 6.19: Average throughput for different lookahead sizes and $n = 50$ in the pickup and delivery service.

Looking at the scale of the diagram, we find that differences between the algorithms are only of minor magnitude. This observation is explained by the hard restriction on the earliest possible start time of pickup and delivery operations at the lower bound of coresponding time window intervals that any algorithm has to respect. Thus, we refrain from deriving further judgments on algorithm quality based upon the throughput attained by the vehicles.

We come to the overall conclusion that in the problem setting under consideration with time windows and multiple types of unpredictable events, there is no essential benefit from lookahead in terms of major improvements in the overall route plan. Immediate planning upon arrival of transportation orders that does not account for too much future information proves to be a sufficient methodology to determine feasible routes of fair quality. However, this recommendation only holds true for the investigated setting with an order dispatching of $n = 50$ orders which leads to almost full vehicle utilization (cf. also Table A.91 of Appendix A.3.2). The decision maker is left over with the task of choosing the algorithm candidate most consonant with his individual preferences: If goals are equally important, OPT leads to balanced results concerning several objectives; if travel distance (tardiness and maximum tardiness) minimization is considered the most important goal, SRH (TS) is the most promising candidate.

6.2.2 Distributional Results

The empirical counting distribution functions incurred by the algorithms with respect to the total travel distance and tardiness are displayed in Figures 6.20 to 6.22 and in Figures 6.23 to 6.25, respectively. A discussion of the distributional results for the order throughput is omitted because of the criterion's irrelevance to the decision making process concerning the selection of a most appropriate algorithm in the given setting.

In Figure 6.20, the empirical counting distribution functions of the total distance are found to lie close to each other for different lookahead durations and there are countless intersections of the plots contradicting an exclusively positive benefit from lookahead. Table A.82 of Appendix A.3.2 also shows that instances with deteriorated total distance value are encountered every now and then, even if more lookahead was provided. Nevertheless, the slight positive influence of lookahead can clearly be seen by the relative position of the plots of successive lookahead levels to each other. Medium widths of confidence intervals and coefficients of variation admit that the distance to be expected cannot be preestimated as accurately as in the simulation of the order picking system in Chapter 6.1. Yet, results are representative because of a fair average deviation of 70 meters per kilometer of the average total distance.

Performance ratios of the total distance incurred by the online algorithms under lookahead relative to $\text{T}\textsc{s},\text{O}\textsc{pt}_{600}$ in Figure 6.21 appear centered around the value of 1. This means that – albeit their informational state is worse – online algorithms under lookahead lead to shorter routes on a considerable proportion of instances as compared to the routes determined under complete information. The plots of the empirical counting distribution functions exhibit numerous intersection points with each other, yet allow to establish an approximate order by their relative positions to each other for different lookahead durations. Table A.83 of Appendix A.3.2 confirms that a significant fraction of input instances has experimental competitive ratio smaller than 1. Analogous statements are derived for the performance ratio of the total distance relative to the online version of an algorithm (cf. Figure 6.22 and Table A.84 of Appendix A.3.2).

Concerning the distributional results with respect to the tardiness over all pickup and delivery operations as incurred by the different algorithms, Figures 6.23 to 6.25 in combination with Tables A.85 to A.87 of Appendix A.3.2 are affirmative to the ineffectiveness of additional time units of lookahead: Plots of all different lookahead durations intersect with each other in a disordered fashion for countless times such that no order relation between any of the empirical counting distribution functions from different information regimes is recognizable at all.

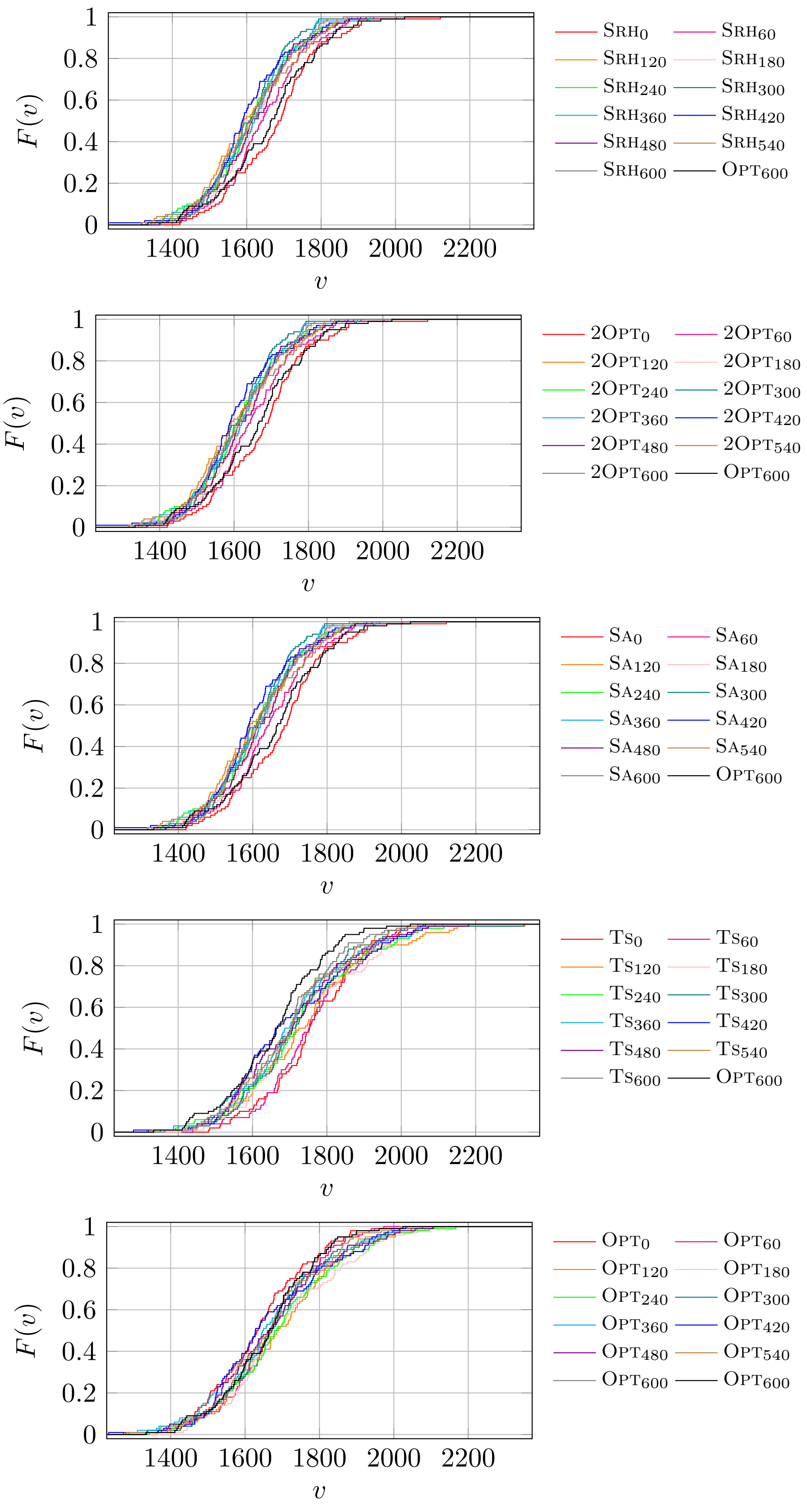


Figure 6.20: Empirical counting distribution functions of distance for $n = 50$ in the pickup and delivery service.

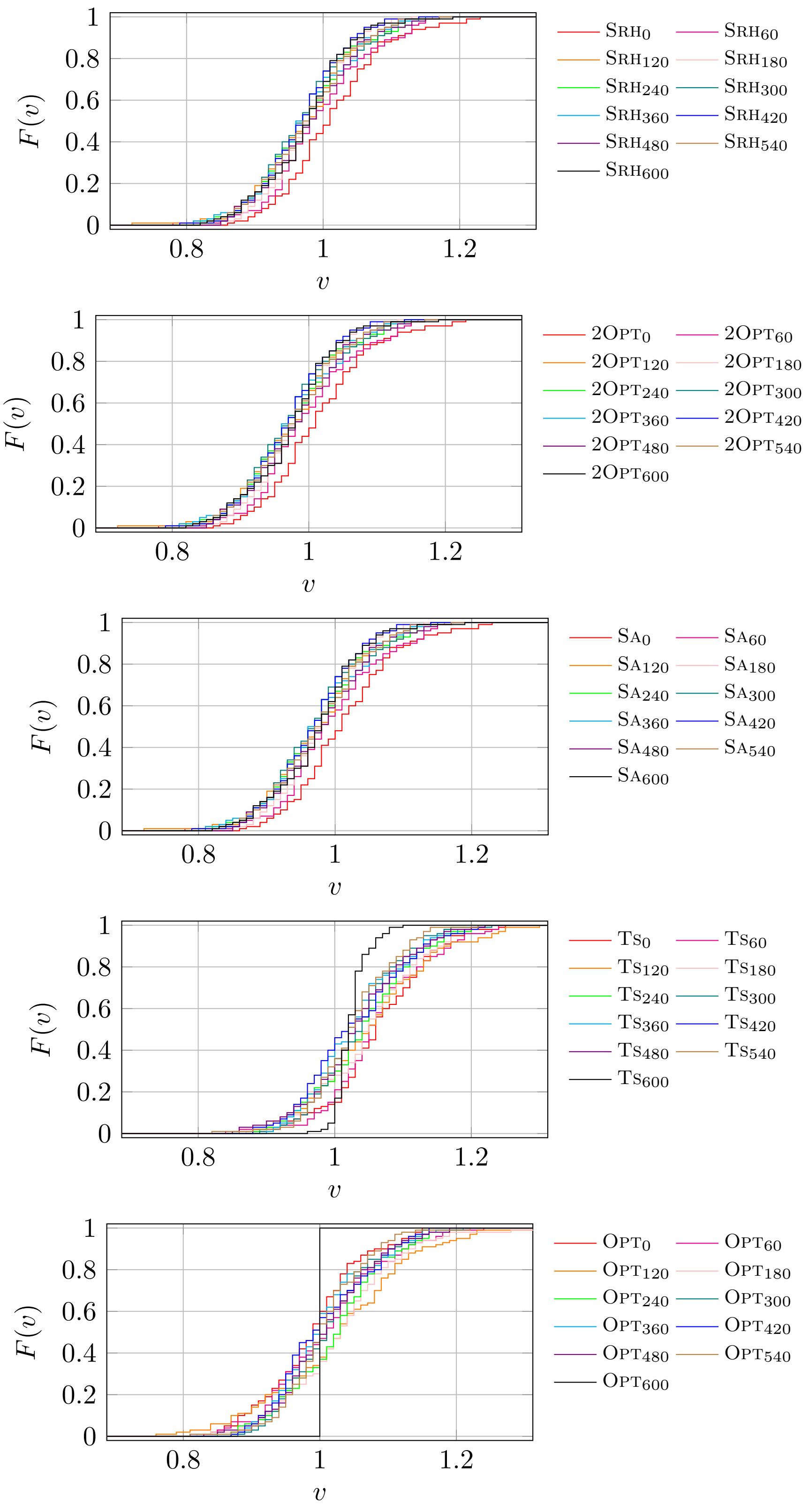


Figure 6.21: Empirical counting distribution functions of performance ratio of distance relative to TS,OPT$_{600}$ for $n = 50$ in the pickup and delivery service.

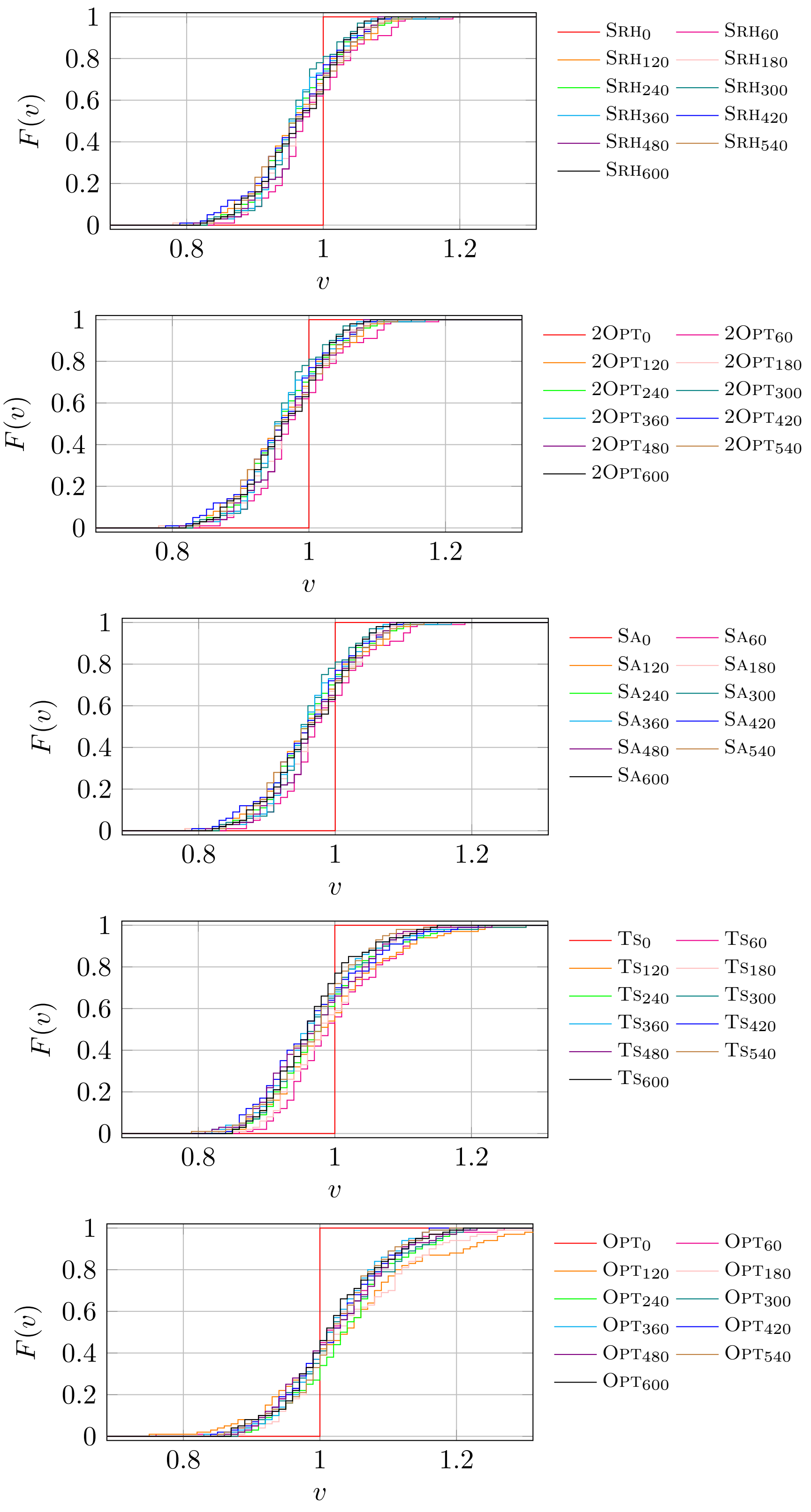
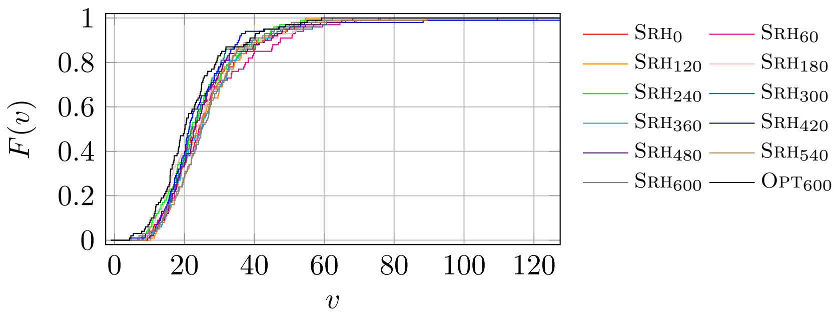
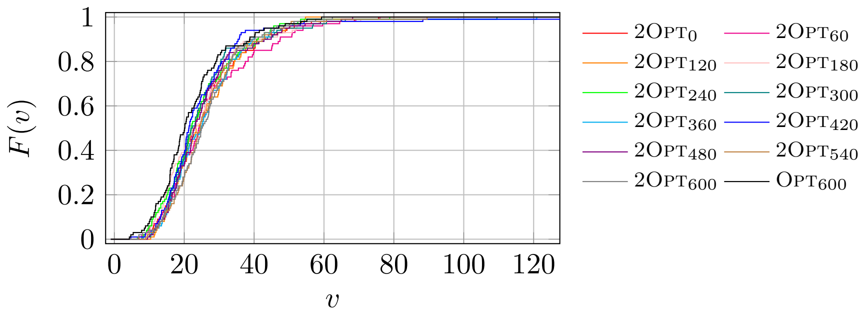
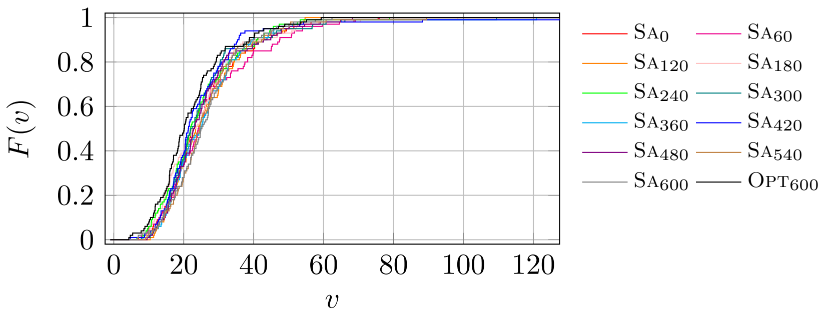
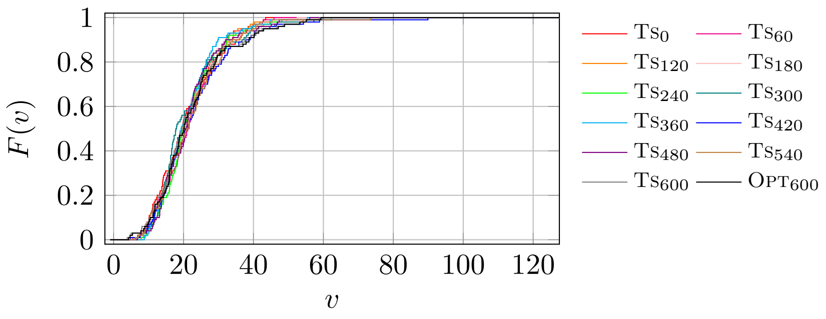


Figure 6.22: Empirical counting distribution functions of performance ratio of distance relative to the online version for $n = 50$ in the pickup and delivery service.









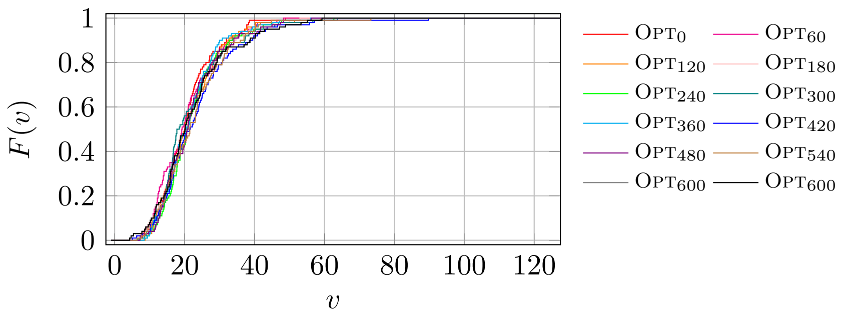
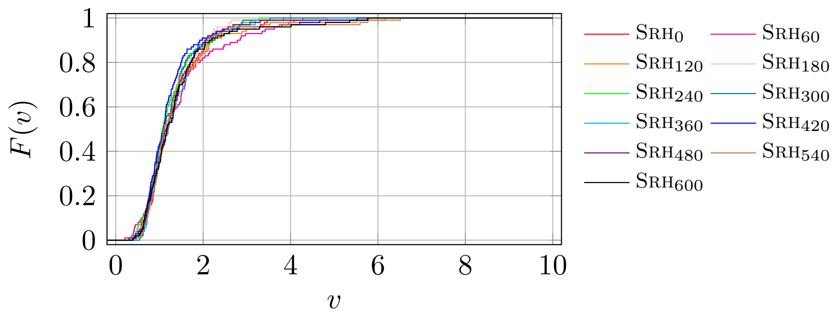
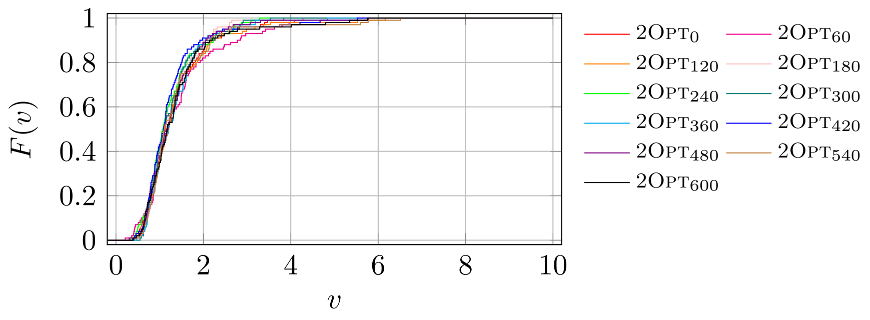
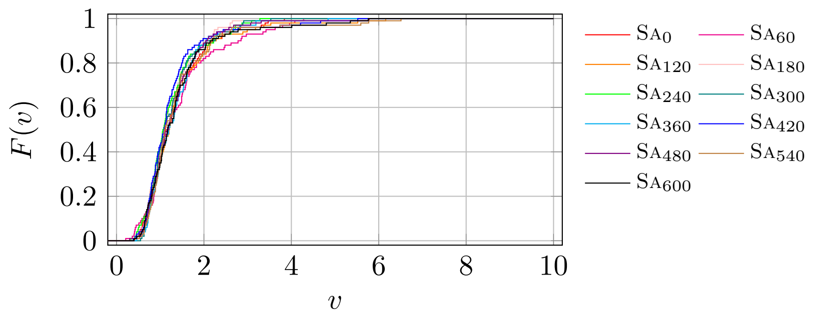
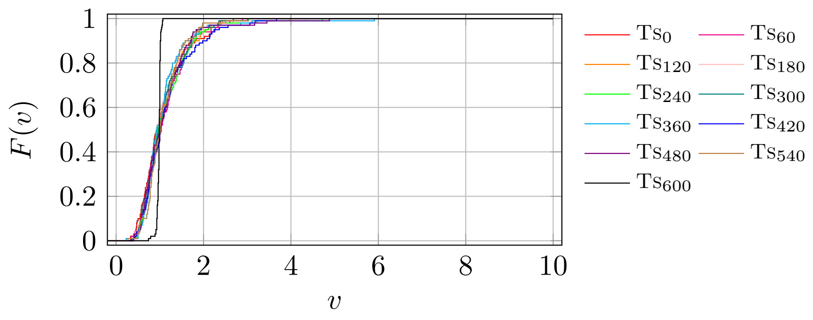


Figure 6.23: Empirical counting distribution functions of tardiness for $n = 50$ in the pickup and delivery service.









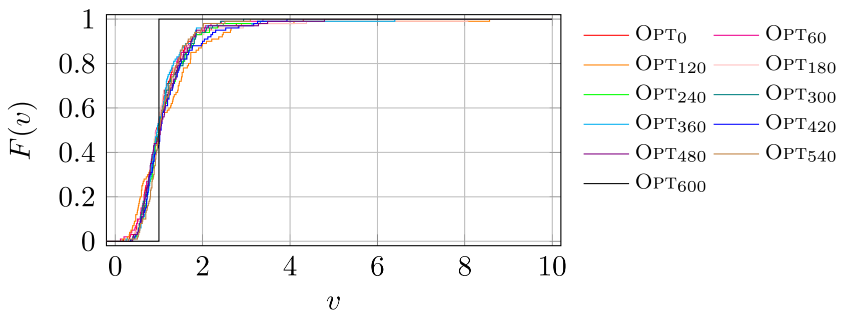


Figure 6.24: Empirical counting distribution functions of performance ratio of tardiness relative to TS,OPT$_{600}$ for $n = 50$ in the pickup and delivery service.

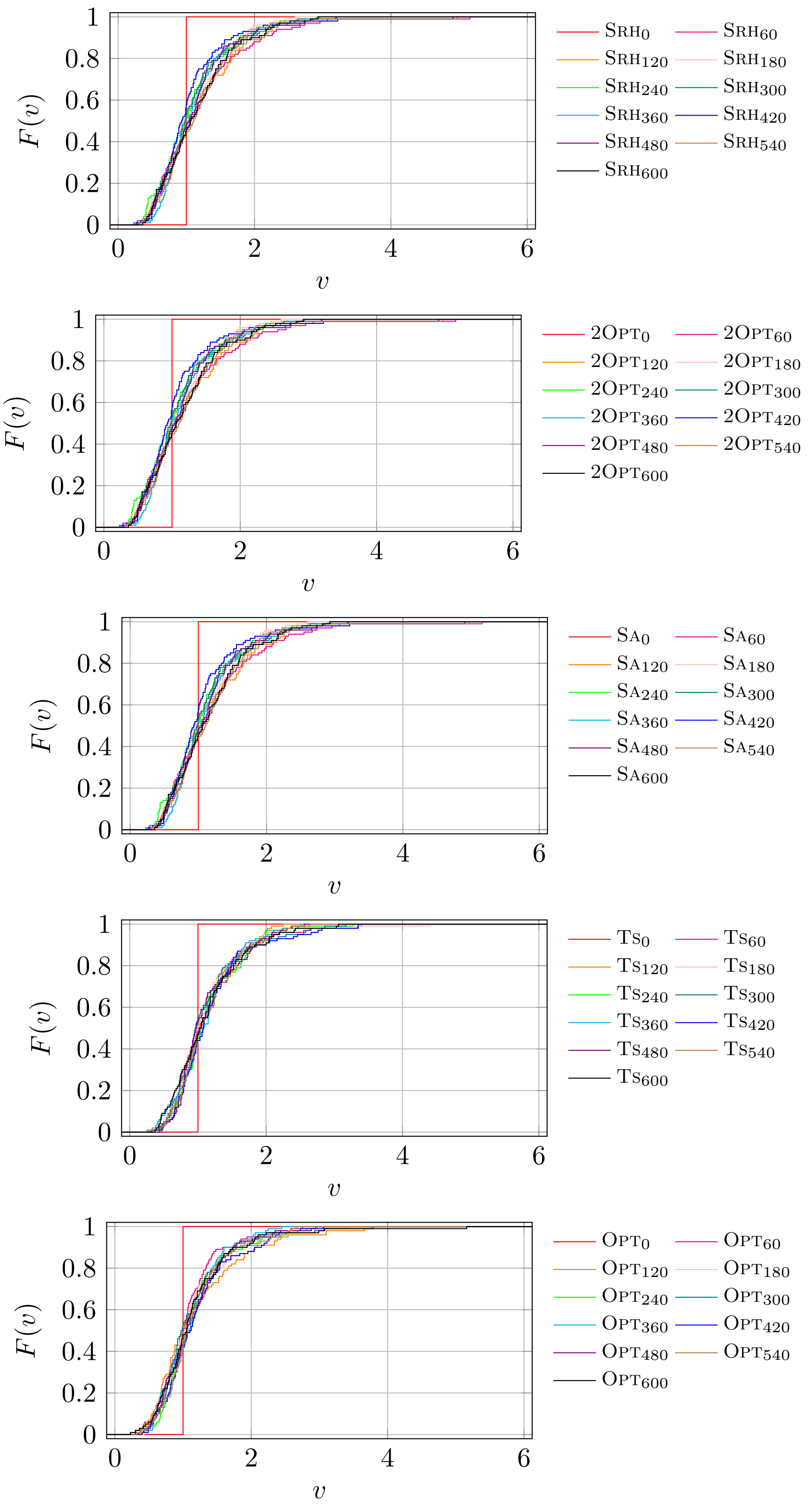


Figure 6.25: Empirical counting distribution functions of performance ratio of tardiness relative to the online version for $n = 50$ in the pickup and delivery service.

We conclude this section by pointing out that hard constraints, such as retained earliest service times, and competing objectives, such as distance and tardiness, make it considerably harder for algorithms to elicit any improvement out of additional information preview at all. In particular, straight-forward extensions of pure online algorithms to the lookahead case are of no use. From a managerial point of view, we recommend not to blindly install technical devices for the retrieval of lookahead information and not to hope for improvement without having an algorithm at hand which could reliably make use of lookahead in the face of competing objectives. Instead of that, it is advisable to spend more money on research for algorithms that align their rationale with lookahead and multiple goals.

6.3 Concluding Discussion

We analyzed the effect of lookahead on the quality of solutions obtained by several algorithm candidates in two real world applications by means of simulation studies. In contrast to the standard online optimization problems examined in Chapters 4 and 5, the settings are characterized by a higher degree of complexity as a result of additional random events, realistic restrictions upon system operations and relevance of multiple performance criteria.

We found that lookahead can be exploited to a fair extent similar as in the underlying standard problems whenever goals are complementary to each other. Under conflicting objectives, however, we discovered that the positive effects of lookahead found in corresponding standard problems with only one objective could not be adapted to the multicriteria case. The future challenge in this field mainly lies in designing algorithms which are able to cope with the relations of several objectives to each other and also capitalize from lookahead.

Concerning the transferability of statements about the lookahead effect from standard problems to complex dynamic settings, we recognize that a one-to-one mapping is not possible. Yet, the standard problems still deliver the vast majority of explanations for the effects that are observed also in the more realistic settings. For instance, the large difference in the impact of lookahead in the two applications from this chapter is mainly attributable to the rule set exchange in form of allowed immediate processing that was invoked in the order picking system but not in the pickup and delivery service because of time windows.

Added practical features may counteract the effects from the standard problems: The set of constraints in an application determines the degrees of freedom that an algorithm can take advantage of to calculate its solution proposal. As a result, a problem's constraint set already implies the optimization potential that may be exploited by appropriate algorithms. In this context, specified time windows in the pickup and delivery service which are retained under lookahead appeared as the major spoiler for performance improvement upon provision of additional information by pruning the problem's optimization potential already in advance. Blocking effects in the order picking system had no influence on the total distance objective, but lead to minor degradations in the throughput and makespan objectives.

Algorithms based on exact reoptimization showed no additional benefit compared to heuristic methods. In particular, the computing time limit of 120 seconds per snapshot problem as prescribed by real-time requirements in both applications was exceeded almost surely for lookahead duration $D \geq 180$ minutes due to the size of the resulting IP and MIP models such that substitute solutions had to be determined by heuristic approaches anyway.

Table 6.1 displays the results of the simulation studies in a condensed form: For the order picking system, we found considerable improvement in all goals through additional lookahead (column $\Delta f^{r,r',P,P'}_{\text{ALG},\text{ALG}'}$). The benefit is accrued through allowed immediate processing of boxes and not having to wait for their arrivals (column $\Delta f^{P,P'}_{\text{ALG},\text{ALG}'}$). As a consequence, the mere benefit of information is deemed negligible (column $\Delta f^{r,r'}_{\text{ALG}}$). Quite to the contrary, in the pickup and delivery service hardly any improvement occurred because of the algorithms' incapability to deal with competing objectives in general. Since time windows were retained, the minor lookahead effects are attributed to their informational component. The column ALG* indicates whether there was some algorithm or a group of comparably good algorithms that outperformed the remaining algorithms. In the order picking system, routing strategies OPT and S along with batching algorithms LS, TS and SEED excelled remaining strategy combinations for three of the four objectives; in the remaining objective there were no remarkable differences. In the pickup and delivery service, each algorithm exhibited a trade-off between competing objectives. While SRH, 2OPT and SA lead to shorter total distances than TS and OPT, the latter group did considerably better with respect to the objectives related to tardiness. In both applications, exact reoptimization did not result in significant advantages compared to heuristics (column OPT) and deterioration in the objective value could occur although additional lookahead was provided (column *Deterioration*).

While in standard problems viable algorithms often perform at comparable quality levels, the simulation studies revealed that algorithms may exhibit fundamentally different performance in realistic settings. Hence, the first step towards efficient logistics operations consists of selecting the right algorithm which matches the decision maker's preferences best. We conclude that simulation is a suitable method to elicit the information (e.g., key performance indicators or counting distributions of algorithm performance for multiple goals) needed by managers of real world systems to successfully deploy their decision making processes.

Application	Rules	Type	Objective	$\Delta f^{r,r',P,P'}_{\textsc{Alg},\textsc{Alg}'}$	$\Delta f^{r,r'}_{\textsc{Alg}}$	$\Delta f^{P,P'}_{\textsc{Alg},\textsc{Alg}'}$	$\textsc{Alg}^*$	$\textsc{Opt}$	Deterioration
Order picking	permutations, blocking	time lookahead	makespan	large	negligible	large	no	no	yes
			distance	large	negligible	large	yes	no	yes
			utilization	large	negligible	large	yes	yes	yes
			throughput	large	negligible	large	yes	no	yes
Pickup and delivery	time windows, traffic scene	time lookahead	makespan	negligible	negligible	zero	yes	no	yes
			distance	small	small	zero	yes	no	yes
			tardiness	zero	zero	zero	yes	no	yes
			maximum tardiness	zero	zero	zero	yes	no	yes
			utilization	medium	medium	zero	no	no	yes
			throughput	negligible	negligible	zero	yes	no	yes

Table 6.1: Qualitative summary of the simulation results from Chapter 6.

7 Conclusion and Outlook

In this thesis, the topic of online optimization with lookahead was addressed from a multi-faceted point of view with the intention to lay out generically applicable methodologies for the analysis of algorithms and to demonstrate their applicability in different problem settings. We close this thesis by summarizing the main contributions, relating them to the research questions introduced in Chapter 1 and pointing out to directions for future research.

7.1 Conclusion

Catalyzed by dozens of applications featuring online optimization problems with lookahead on the one hand, but given the poor set of comprehensive methods for algorithm analysis in this optimization paradigm on the other hand, we started our research in the fundamentals of the concepts of online optimization and lookahead information. Since both terms are understood and used in a versatile fashion, we first established clear, but yet flexible definitions for these terms which facilitate a modeling of lookahead in different online optimization problems using the same taxonomy and notation. Additionally, we subdivided the mechanism of lookahead into an informational and a processual component. Referring to Chapter 2, this leads us to the answer of the first research question:

RQ1 What do we understand by lookahead?

Lookahead is a mechanism of information release that specifies the difference in the process of information disclosure as compared to a reference online optimization problem (instance revelation rule substitution) and that might impose a set of constraints differing from the set of constraints in the reference online optimization problem upon the processing of the input elements (rule set substitution).

In conjunction with the answer of this question, we emphasize that lookahead – both the way it leads to forwarded information release and also the way it affects processing of input elements – is defined in existing literature only in problem-specific contexts of publications which prohibits a structurally-oriented view on the effects caused by it. Using the approach outlined in this thesis, we are able to attribute a reason to lookahead effects observed in problems and to transfer that knowledge also to other settings.

We realized that in order to design algorithms for online optimization problems with lookahead, we first need to understand how they should act within the solution process of an instance. Driven by the sequentiality in the information element release process, we found a large number of analogies to the control of discrete event dynamic systems (cf. Chapter 2). As a result, we adapted the concepts from this domain and coined them towards the needs of the solution process in an instance of an online optimization problem with lookahead. This allows us to address research question *RQ2* based on the general modeling framework for online optimization with lookahead developed in Chapter 3:

RQ2 Which formalism can be used to model the solution process in an online optimization problem with lookahead in a generally applicable framework?

The solution process is abstractly emulated as the state trajectory of our state-based discrete event modeling framework. The evolution of the state trajectory over time is induced by the decisions of an algorithm for the snapshot problems of the online optimization problem under consideration. Algorithm execution occurs whenever the state is stimulated to advance, e.g., upon an external trigger in form of an event such as the release of a new input element.

A particular emphasis in the modeling framework is put on lookahead-related issues such as processing modes, orders and accessibilities. Hence, devising algorithms for online optimization with lookahead is no more seen as a problem-specific task independent of a general optimization paradigm, but closely intertwined with the abstract concept of lookahead as introduced in Chapter 2 and the framework from Chapter 3. Moreover, solution concepts can now be described independent of domains in an abstract way using a unified taxonomy.

The commonly used standard performance yardstick for online algorithms is competitive analysis although there is undisputed agreement that this worst-case analysis lacks displaying the typical behavior of an algorithm over all input sequences and has numerous other drawbacks ([65], [73]). We are lead to the conclusion that other, more comprehensive analysis methods are required in practice: According to the very nature of online optimization, no stochastic information is given, i.e., one has to hedge against all possible scenarios. As a consequence, we impute the uniform distribution as the distribution of maximum entropy (or analogously of minimum prior information) on the release of input elements ([97], [98]).

What we obtain are counting results yielding frequency information about the occurrences of specific objective function or performance ratio values. We called the corresponding distribution functions the counting distribution functions of the objective value and of the performance ratio, respectively (see Chapter 2); they form the basis of our approach to performance measurement in online optimization with lookahead as addressed in *RQ3* and conducted in Chapters 4 to 6:

RQ3 Which performance measurement approach is best suitable to analyze the performance of algorithms in online optimization problems with lookahead and to relate the quality of algorithms to each other?

Algorithm performance is not measured based on a single performance indicator. Algorithm performance is evaluated by the decision maker with his individual preferences based on the counting distribution functions of the objective value and of the performance ratio for candidate algorithms relative to each other. The objective value distribution yields global information concerning the absolute performance achieved by individual algorithms, whereas performance ratio distributions indicate the relative performance of algorithms to each other on the same input instance, i.e., a type of local information. Different lookahead regimes are accounted for by relating algorithm performance to some reference information regime which yields the baseline for counting distributions both of the objective value and of the performance ratio.

Presenting a decision maker with the counting distribution functions instead of single performance indicators puts the burden of defining trade-offs between worst case, best case and average case off our shoulders. The decision maker is equipped with all possible information except for the direct mapping of objective values and performance ratios to input instances. For holistic decision making, we cannot think of a more comprehensive way to supply a decision maker with information about algorithm behavior. In case of a high number of algorithm alternatives, one can conduct an average-case analysis in the first step to filter out algorithms which shall be investigated by distributional analysis in the second step.

Knowing the magnitude of the lookahead impact on solution quality for different algorithms in a given application is important to practitioners when they have to decide whether it would be beneficial to invest in new machinery with lookahead devices, and if so, what level of lookahead capability should be selected. From an academic point of view, it is of interest which kinds of problems are amenable to lookahead in terms of improved algorithm performance and which are not. We studied the impact of lookahead in theory (cf. Chapter 4), in basic practical settings (cf. Chapter 5), and in real world applications (cf. Chapter 6) which brings us in a position to answer the last research question:

RQ4 What is the value of different degrees of lookahead in specific online optimization problems with lookahead?

The extent of the lookahead effect clearly depends on the problem setting under investigation. Overall, we found that there *is* a positive effect on objective values and on performance ratios when algorithms are supplied with lookahead. However, the magnitude of the effect depends on several key characteristics of the problem such as:

- Allowance of algorithms to take advantage of lookahead by rule set substitution
- Admissible degrees of freedom for algorithms as imposed by problem constraints
- Possibility of bad decisions and risk of deterioration upon lookahead provision
- Performance quality gap between offline algorithms and pure online algorithms without lookahead

Details on the magnitude of the lookahead effects for different levels of information preview as determined for a large number of problem classes can be looked up in Chapters 4 to 6 along with corresponding explanations. We remark that problems where solution quality is strongly affected by permuting the input element release order during processing and where algorithms are allowed to change that order have a high potential for significant benefits from lookahead as long as the feasible set of the instance admits the necessary degrees of freedom. Moreover, there are problem types where more information never leads to a degradation in performance, but also problem types where "wrong" decisions may be made although more information was provided.

To the best of our knowledge, our exact analyses in Chapter 4 are the first to give an exact image reproducing algorithm behavior over all input sequences in the respective problems. Despite the small size of the basic settings, as a byproduct, the proofs already provided explanations for the lookahead effects which were also encountered in the more realistic settings. The computational results of Chapter 5 may serve to future works in the field of online optimization as an information pool concerning the impact of lookahead in several standard problems. In Chapter 6, we already took advantage of this information pool by easily delivering explanations for the effects of lookahead in two real world applications. Nonetheless, in this context it became clear that results can never be transferred in a one-to-one fashion due to additional constraints in practical problems. We recognized that in real world applications there is an additional dimension of onlineness in form of random occurrences that are out of control in operations and hardly calculable at all. Due to the complexity of the settings, we used simulation models to study the influence of lookahead with respect to the input element disclosure when systems are additionally subject to unforeseeable random events. We recommend to make use of simulation studies in order to determine the lookahead effect prior to operating a system whenever an algorithm needs to be selected but the number of dependent random variables – whose realizations taken together lead to the realization of the random variable for the attained objective value – is too large.

Algorithms in online optimization are required to respond to the arrival of input data within few seconds, i.e., they have to be real-time compliant. Throughout our computational experiments, we could not observe an advantage of exact reoptimization over heuristic methods in solving the snapshot problems: "Locally" optimal solutions lost their efficacy once the situation changed upon arrival of new input elements. Quite to the contrary, computational effort is out of scale compared to heuristics such that no need is seen for exact reoptimization.

Overall, this thesis

- provided a clear and versatile definition of the optimization paradigm of online optimization with lookahead,
- related it to the established paradigms of online and offline optimization,
- characterized components of the lookahead effect,
- presented a holistic approach to performance assessment of candidate algorithms which may resort to different information regimes,
- developed a generic modeling framework for online optimization with lookahead, and
- determined the value of information using exact analysis in basic problem settings, extensive sample-based analysis in standard problem settings and simulation studies in real world applications.

7.2 Outlook

Future research directions in the field of online optimization with lookahead emerge from limitations of the presented approaches on the one hand and from related topics that were not addressed in this thesis on the other hand.

The derivation of exact expressions for the counting distribution functions in Chapter 4 showed that already in basic settings it is hard to gain access to the combinatorial structures that govern the behavior of algorithms during input element processing. We recognize that for more complex settings an analysis of this type is likely to be out of scope such that the approach has to be transferred to sample-based methods (cf. Chapters 5 and 6) or reduced to order relations between objective value distributions of algorithms (cf. also [9], [10], [90]). With respect to the first direction, we identify the need for a thorough examination of how to propagate methods of distributional analysis to experimental algorithm analysis in general and especially in settings which involve multicriteria online optimization with lookahead; concerning the second direction, it has to be checked which types of mathematical statements are realistic to be elicited in further exact analysis.

Because the provision of lookahead may intrinsically yield an altered set of constraints as a consequence of a rule set substitution, we argue that defining the snapshot problem to be solved in each reoptimization step in a way different from just adapting the overall problem to the current lookahead set could contribute to enhanced algorithm performance. The rationale behind this idea is to bring the algorithm in a best possible position for future steps at the end of each reoptimization step. As an example, we suggest in the snapshot problems of the TSP not to compute a Hamiltonian path that returns to the origin but to a point that minimizes the (expectation of the) total distance of all locations to that point (median); likewise, one could avoid zig-zagging in routes by additionally incorporating enforced stability constraints which forbid frequent moves to distant regions. For the bin packing problem, we carried out this approach by modifying the objective function such that not only the total number of bins is minimized but also that bins as full and empty as possible are generated and not bins at medium capacity (see also [72]). The stream of publications on online stochastic combinatorial optimization by Bent and Van Hentenryck ([22], [23], [24], [25]) represents a first step in this direction by taking into account several alternative objective functions. Future research could also consider variable forms of the snapshot problem as input to algorithms already in their design phase, and the goal would be to determine the snapshot problem type which yields the best algorithm performance.

In the same line of argumentation, we point out that stability (in the sense of consistency over time) of the partial solutions with regard to some problem-related criterion could be a major contributor to solid decision making: For instance, in the TSP, "nearness"-oriented algorithm NearestNeighbor lead to routes that were more robust under future scenarios than those determined by exact reoptimization. Similar results are also known for (reactive) project scheduling ([58]) where it is recommended to insert time buffers in schedules to make them robust for future deviations. Hence, we conclude that the inclusion of stability-enforcing constraints into subproblem formulations should be addressed in future works.

Throughout this thesis, we were only concerned with the value of lookahead but not with the costs that are related to it. Realizing lookahead in practice amounts to the installation of costly technical devices such as barcode or radio-frequency identification (RFID) tag scanners which facilitate transmitting information about input elements at an earlier point in time. Installation and operation of such machinery induces a fair amount of costs, and it needs to be checked whether the benefits of lookahead exceed the costs of lookahead device installation and operations. To this end, economic models and methods that translate the value of lookahead into a monetary equivalent are still needed.

Real world applications typically feature multiple, oftentimes conflicting optimization goals. As figured out in Chapter 6, the task of designing algorithms which are able to exploit lookahead in more than one goal at the same time is not trivial and requires more sophisticated approaches than mere adaptation of pure online algorithms to the lookahead case. Future research in this direction would comprise devising algorithms for practical settings specifically tailored towards lookahead utilization that take several objectives as well as their trade-offs into account in their rationale.

With the paging and ski rental problem, we identified two problem classes where algorithms could not go wrong by using additional information. Contrarily, in all other problems we encountered instances with objective value degradation although lookahead capabilities were enlarged. It is of general interest to find characterizations for problem settings where it can be guaranteed that additional information never leads to worse decisions (similar to matroid structures ensuring optimality of greedy choices in offline optimization ([82])). We conjecture that properties akin to those required for employing dynamic programming upon a given problem, namely optimality of subsolutions and independence of substructures ([54]), are required in order to assure no threat by providing additional lookahead information. We leave a detailed examination of this question to future research.

Another interesting research question encompasses the relation between resource augmentation and lookahead: We might ask whether physical resources and informational resources are interchangeable. In the framework of competitive analysis, it has been shown for two single machine scheduling problems that increasing processor speed is more valuable than allowing an algorithm to foresee the future ([28], [102]). The effect of resource augmentation on competitive ratios has also been investigated in paging (by increased cache size ([149])) and bin packing (by increased bin sizes ([56], [71])). In these problems, the connection of resource augmentation to lookahead – not only in competitive analysis but also in distributional analysis – remains an open question.

Although tremendous research effort has been spent on online optimization over the past two decades, it is still widely believed that the state of the art is yet far from reaching maturity ([84], [85]). In particular, there is no agreed groundwork of methods and tools for comprehensive algorithm analysis in online optimization, not to mention in online optimization with lookahead. This thesis aimed at contributing towards the elimination of this deficiency by providing a commonly agreeable basis for the modeling and the analysis of algorithms in this optimization paradigm in order to establish a harmonized understanding of the mechanism of information preview and to foster the use of holistic algorithm assessment methods across application domains. We recognize that there is still a long way to go in this direction, but we hope for future research to revisit, foster or extend some of the ideas and concepts brought up in this thesis.

A Appendix

A.1 Additional Proofs from Chapter 4

A.1.1 Proof of Lemma 4.4

a) See [131], chapter 7, pages 115 and 116.

b) The proof is a modification of [127]: The number of recurring unit-sloped paths of length $2i$ with $s_k \geq -1$ for $k = 0, 1, 2, \dots, 2i$ is the number of all recurring unit-sloped paths of length $2i$ minus the number of all recurring unit-sloped paths of length $2i$ which hit the number -2 at least once.

   For each of these paths hitting -2 at least once, define T as the first time -2 is hit by the path (see Figure A.1). According to the reflection principle, we can start a mirror path at T with respect to the horizontal axis with ordinate -2 that necessarily ends at height -4 at time $2k$ (the mirror position of 0 with respect to -2).

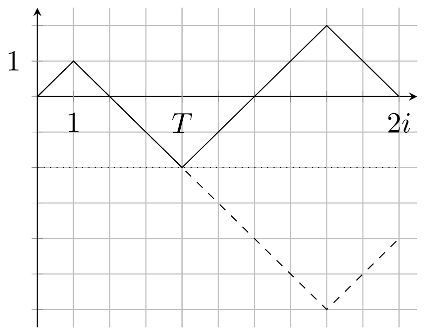


Figure A.1: Reflection principle.

   Thus, counting the paths from $(0, 0)$ to $(2k, 0)$ hitting -2 at least once is the same as counting the paths from $(0, 0)$ to $(2k, -4)$ hitting -2 at least once. But any such path must hit -2 at some point, i.e., we are computing the total number of paths from $(0, 0)$ to $(2k, -4)$.

In total, we obtain that the number of recurring unit-sloped paths of length $2i$ with $s_k \geq -1$ for $k = 0, 1, 2, \ldots, 2i$ is equal to the total number of paths from $(0, 0)$ to $(2k, 0)$ minus the total number of paths from $(0, 0)$ to $(2k, -4)$ which is equal to

$$\begin{aligned}
\binom{2k}{k} - \binom{2k}{k+2} &= \frac{(2k)!}{(k!)^2} - \frac{(2k)!}{(k+2)!(k-2)!} \\
&= \frac{(2k)!}{(k!)^2}\Big(1 - \frac{k(k-1)}{(k+1)(k+2)}\Big) \\
&= \frac{(2k)!}{(k!)^2} \frac{(k+1)(k+2) - k(k-1)}{(k+1)(k+2)} \\
&= \frac{(2k)!}{(k!)^2} \frac{4k+2}{(k+1)(k+2)} \\
&= \frac{1}{k+2} \frac{1}{(k+1)!} \frac{(2k)!(4k+2)}{k!} \\
&= \frac{1}{k+2} \frac{1}{(k+1)!} \frac{(2k)!(4k+2)(k+1)}{(k+1)!} \\
&= \frac{1}{k+2} \frac{1}{(k+1)!} \frac{(2k)!(4k^2+6k+2)}{(k+1)!} \\
&= \frac{1}{k+2} \frac{1}{(k+1)!} \frac{(2k)!(2k+1)(2k+2)}{(k+1)!} \\
&= \frac{1}{k+2} \binom{2k+2}{k+1} = C_{k+1}.
\end{aligned}$$

A.1.2 Proof of Lemma 4.9

We show that

$$\sum_{i\geq 1} C_i \binom{2n-2i}{m-i} - \sum_{i\geq 1} C_{i-1} \binom{2n-2i+1}{m-i+1} = 0.$$

Notice that from the definition of the binomial coefficient, i ranges in $\{1, 2, \ldots, 2n - m\}$ in both terms.

Hence,

$$\sum_{i\geq 1} C_i \binom{2n-2i}{m-i} - \sum_{i\geq 1} C_{i-1}\binom{2n-2i+1}{m-i+1}$$

$$\begin{aligned}
&= C_1\binom{2n-2}{m-1} + C_2\binom{2n-4}{m-2} + C_3\binom{2n-6}{m-3} + \ldots + C_{2n-m}\binom{2m-n}{2m-n} \\
&\quad - \left(C_0\binom{2n-1}{m} + C_1\binom{2n-3}{m-1} + C_2\binom{2n-5}{m-2} + \ldots + C_{2n-m-1}\binom{2m-n+1}{2m-n+1}\right) \\
&= C_1\binom{2n-3}{m-2} + C_2\binom{2n-5}{m-3} + \ldots + C_{2n-m-1}\binom{2m-n+1}{2m-2n} + C_{2n-m} - \binom{2n-1}{m} \\
&= \sum_{i=1}^{2n-m-1} C_i\binom{2n-2i-1}{m-i-1} + C_{2n-m} - \binom{2n-1}{m} \\
&= \sum_{i=0}^{2n-m-1} C_i\binom{2n-2i-1}{m-i-1} - \binom{2n-1}{m-1} + C_{2n-m} - \binom{2n-1}{m} \\
&= \sum_{i=0}^{2n-m} C_i\binom{2n-2i-1}{m-i-1} - C_{2n-m}\binom{2m-2n-1}{2m-2n-1} - \binom{2n-1}{m-1} + C_{2n-m} - \binom{2n-1}{m} \\
&= \sum_{i\geq 0} C_i\binom{2n-2i-1}{m-i-1} - C_{2n-m} - \binom{2n-1}{m-1} + C_{2n-m} - \binom{2n-1}{m} \\
&= \binom{2n}{m} - \binom{2n-1}{m-1} - \binom{2n-1}{m} = \binom{2n}{m} - \binom{2n}{m} = 0.
\end{aligned}$$

A.2 Numerical Results from Chapter 5

This section contains a detailed statistical summary of the numerical results gathered during the experimental analysis in Chapter 5. For each problem, we give three tables:

- The first table subsumes in one line per algorithm and lookahead level the key figures with respect to the objective values incurred over the set of sampled input instances.
- The second table subsumes in one line per algorithm and lookahead level the key figures with respect to the performance ratio relative to an optimal offline algorithm if available, or relative to the best offline algorithm applied.
- The third table subsumes in one line per algorithm and lookahead level the key figures with respect to the performance ratio relative to an online algorithm from the same class of algorithms.

The following key figures were calculated from the samples of random input instances:

$\boldsymbol{\mu}$ Average of objective value or performance ratio

$\boldsymbol{CV}$ Coefficient of variation of objective value or performance ratio

95 % CI 95 % confidence interval of objective value or performance ratio

min Minimum objective value or performance ratio

max Maximum objective value or performance ratio

$\boldsymbol{q_{0.01}}$ First percentile of objective value or performance ratio counting distribution

$\boldsymbol{q_{0.5}}$ Median of objective value or performance ratio counting distribution

$\boldsymbol{q_{0.99}}$ 99th percentile of objective value or performance ratio counting distribution

% det. Fraction of samples with deterioration in the objective value when compared to the same algorithm with the lookahead level preceding the algorithm's lookahead level

$\boldsymbol{F(1)}$ Fraction of samples with performance ratio smaller than 1 relative to the optimal offline algorithm if available, or relative to a best possible offline algorithm among those offline algorithms which terminated

$\boldsymbol{1 - F(1)}$ Fraction of samples with performance ratio larger than 1 relative to an online algorithm from the same class of algorithms

A.2.1 Online Ski Rental with Lookahead

Costs for $n_{max} = 100$									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Alg}_{1,1}$	50	0	[50, 50]	50	50	50	50	50	
$\textsc{Alg}_{1,5}$	48.1	0.19	[46.3, 49.9]	1	50	1.5	50	50	0
$\textsc{Alg}_{1,10}$	45.95	0.28	[43.42, 48.48]	1	50	1.5	50	50	0
$\textsc{Alg}_{1,20}$	42.4	0.37	[39.31, 45.49]	1	50	1.5	50	50	0
$\textsc{Alg}_{1,30}$	39.85	0.41	[36.63, 43.07]	1	50	1.5	50	50	0
$\textsc{Alg}_{1,40}$	38.3	0.42	[35.13, 41.47]	1	50	1.5	50	50	0
$\textsc{Alg}_{1,50\ldots100}$	37.75	0.42	[34.63, 40.87]	1	50	1.5	50	50	0
$\textsc{Alg}_{25,1}$	59.24	0.45	[53.99, 64.49]	1	74	1.5	74	74	
$\textsc{Alg}_{25,5}$	57.34	0.47	[52.03, 62.65]	1	74	1.5	74	74	0
$\textsc{Alg}_{25,10}$	55.19	0.5	[49.75, 60.63]	1	74	1.5	74	74	0
$\textsc{Alg}_{25,20}$	51.64	0.52	[46.35, 56.93]	1	74	1.5	74	74	0
$\textsc{Alg}_{25,30}$	49.09	0.53	[43.96, 54.22]	1	74	1.5	50.5	74	0
$\textsc{Alg}_{25,40}$	47.54	0.52	[42.67, 52.41]	1	74	1.5	50.5	74	0
$\textsc{Alg}_{25,50\ldots100}$	37.75	0.42	[34.63, 40.87]	1	50	1.5	50	50	0
$\textsc{Alg}_{50,1}$	62.74	0.61	[55.2, 70.28]	1	99	1.5	99	99	
$\textsc{Alg}_{50,5}$	60.84	0.62	[53.41, 68.27]	1	99	1.5	50.5	99	0
$\textsc{Alg}_{50,10}$	58.69	0.62	[51.52, 65.86]	1	99	1.5	50.5	99	0
$\textsc{Alg}_{50,20}$	55.14	0.62	[48.41, 61.87]	1	99	1.5	50.5	99	0
$\textsc{Alg}_{50,30}$	52.59	0.6	[46.37, 58.81]	1	99	1.5	50.5	99	0
$\textsc{Alg}_{50,40}$	51.04	0.58	[45.21, 56.87]	1	99	1.5	50.5	99	0
$\textsc{Alg}_{50,50\ldots100}$	37.75	0.42	[34.63, 40.87]	1	50	1.5	50	50	0
$\textsc{Alg}_{75,1}$	59.99	0.7	[51.72, 68.26]	1	124	1.5	50.5	124	
$\textsc{Alg}_{75,5}$	58.09	0.69	[50.19, 65.99]	1	124	1.5	50.5	124	0
$\textsc{Alg}_{75,10}$	55.94	0.67	[48.56, 63.32]	1	124	1.5	50.5	124	0
$\textsc{Alg}_{75,20}$	52.39	0.62	[45.99, 58.79]	1	124	1.5	50.5	124	0
$\textsc{Alg}_{75,30}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{75,40}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{75,50\ldots100}$	37.75	0.42	[34.63, 40.87]	1	50	1.5	50	50	0
$\textsc{Alg}_{100,1}$	50.99	0.59	[45.06, 56.92]	1	149	1.5	50.5	124	
$\textsc{Alg}_{100,5}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{100,10}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{100,20}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{100,30}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{100,40}$	50.5	0.57	[44.83, 56.17]	1	100	1.5	50.5	99.5	0
$\textsc{Alg}_{100,50\ldots100}$	37.75	0.42	[34.63, 40.87]	1	50	1.5	50	50	0

Table A.1: Costs in the ski rental problem.

Performance ratios of costs relative to OPT for $n_{max} = 100$									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
$\text{ALG}_{1,1}$	2.75	2.11	[1.61, 3.89]	1	50	1	1	37.5	0
$\text{ALG}_{1,5}$	1.75	0.91	[1.44, 2.06]	1	10	1	1	9.17	0
$\text{ALG}_{1,10}$	1.43	0.58	[1.27, 1.59]	1	5	1	1	4.77	0
$\text{ALG}_{1,20}$	1.17	0.29	[1.1, 1.24]	1	2.5	1	1	2.44	0
$\text{ALG}_{1,30}$	1.06	0.14	[1.03, 1.09]	1	1.67	1	1	1.64	0
$\text{ALG}_{1,40}$	1.01	0.04	[1, 1.02]	1	1.25	1	1	1.23	0
$\text{ALG}_{1,50...100}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{25,1}$	1.52	0.29	[1.43, 1.61]	1	2.96	1	1.48	2.9	0
$\text{ALG}_{25,5}$	1.44	0.25	[1.37, 1.51]	1	2.55	1	1.48	2.51	0
$\text{ALG}_{25,10}$	1.37	0.22	[1.31, 1.43]	1	2.18	1	1.48	2.15	0
$\text{ALG}_{25,20}$	1.28	0.19	[1.23, 1.33]	1	1.68	1	1.48	1.66	0
$\text{ALG}_{25,30}$	1.23	0.19	[1.18, 1.28]	1	1.48	1	1.01	1.48	0
$\text{ALG}_{25,40}$	1.2	0.19	[1.16, 1.24]	1	1.48	1	1.01	1.48	0
$\text{ALG}_{25,50...100}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{50,1}$	1.5	0.33	[1.4, 1.6]	1	1.98	1	1.98	1.98	0
$\text{ALG}_{50,5}$	1.46	0.33	[1.37, 1.55]	1	1.98	1	1.01	1.98	0
$\text{ALG}_{50,10}$	1.42	0.34	[1.32, 1.52]	1	1.98	1	1.01	1.98	0
$\text{ALG}_{50,20}$	1.35	0.33	[1.26, 1.44]	1	1.98	1	1.01	1.98	0
$\text{ALG}_{50,30}$	1.3	0.3	[1.22, 1.38]	1	1.98	1	1.01	1.98	0
$\text{ALG}_{50,40}$	1.27	0.27	[1.2, 1.34]	1	1.98	1	1.01	1.98	0
$\text{ALG}_{50,50...100}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{75,1}$	1.44	0.43	[1.32, 1.56]	1	2.48	1	1.01	2.48	0
$\text{ALG}_{75,5}$	1.41	0.42	[1.29, 1.53]	1	2.48	1	1.01	2.48	0
$\text{ALG}_{75,10}$	1.36	0.4	[1.25, 1.47]	1	2.48	1	1.01	2.48	0
$\text{ALG}_{75,20}$	1.29	0.33	[1.21, 1.37]	1	2.48	1	1.01	2.48	0
$\text{ALG}_{75,30}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{75,40}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{75,50...100}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{100,1}$	1.26	0.29	[1.19, 1.33]	1	2.98	1	1.01	2.48	0
$\text{ALG}_{100,5}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{100,10}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{100,20}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{100,30}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{100,40}$	1.25	0.26	[1.19, 1.31]	1	2	1	1.01	1.99	0
$\text{ALG}_{100,50...100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.2: Performance ratios of costs relative to OPT in the ski rental problem.

Performance ratios of costs relative to online version for $n_{max} = 100$									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
$\text{ALG}_{1,1}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{1,5}$	0.96	0.19	[0.92, 1]	0.02	1	0.03	1	1	0
$\text{ALG}_{1,10}$	0.92	0.28	[0.87, 0.97]	0.02	1	0.03	1	1	0
$\text{ALG}_{1,20}$	0.85	0.37	[0.79, 0.91]	0.02	1	0.03	1	1	0
$\text{ALG}_{1,30}$	0.8	0.41	[0.74, 0.86]	0.02	1	0.03	1	1	0
$\text{ALG}_{1,40}$	0.77	0.42	[0.71, 0.83]	0.02	1	0.03	1	1	0
$\text{ALG}_{1,50...100}$	0.76	0.42	[0.7, 0.82]	0.02	1	0.03	1	1	0
$\text{ALG}_{25,1}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{25,5}$	0.97	0.13	[0.95, 0.99]	0.34	1	0.34	1	1	0
$\text{ALG}_{25,10}$	0.95	0.18	[0.92, 0.98]	0.34	1	0.34	1	1	0
$\text{ALG}_{25,20}$	0.9	0.24	[0.86, 0.94]	0.34	1	0.34	1	1	0
$\text{ALG}_{25,30}$	0.86	0.26	[0.82, 0.9]	0.34	1	0.34	1	1	0
$\text{ALG}_{25,40}$	0.84	0.26	[0.8, 0.88]	0.34	1	0.34	1	1	0
$\text{ALG}_{25,50...100}$	0.71	0.26	[0.67, 0.75]	0.34	1	0.34	0.68	1	0
$\text{ALG}_{50,1}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{50,5}$	0.98	0.1	[0.96, 1]	0.51	1	0.51	1	1	0
$\text{ALG}_{50,10}$	0.96	0.14	[0.93, 0.99]	0.51	1	0.51	1	1	0
$\text{ALG}_{50,20}$	0.92	0.17	[0.89, 0.95]	0.51	1	0.51	1	1	0
$\text{ALG}_{50,30}$	0.9	0.19	[0.87, 0.93]	0.51	1	0.51	1	1	0
$\text{ALG}_{50,40}$	0.88	0.19	[0.85, 0.91]	0.51	1	0.51	1	1	0
$\text{ALG}_{50,50...100}$	0.75	0.33	[0.7, 0.8]	0.51	1	0.51	0.51	1	0
$\text{ALG}_{75,1}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{75,5}$	0.98	0.08	[0.96, 1]	0.6	1	0.61	1	1	0
$\text{ALG}_{75,10}$	0.97	0.11	[0.95, 0.99]	0.6	1	0.61	1	1	0
$\text{ALG}_{75,20}$	0.94	0.14	[0.91, 0.97]	0.6	1	0.61	1	1	0
$\text{ALG}_{75,30}$	0.92	0.14	[0.89, 0.95]	0.6	1	0.61	1	1	0
$\text{ALG}_{75,40}$	0.92	0.14	[0.89, 0.95]	0.6	1	0.61	1	1	0
$\text{ALG}_{75,50...100}$	0.8	0.31	[0.75, 0.85]	0.4	1	0.4	0.99	1	0
$\text{ALG}_{100,1}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{ALG}_{100,5}$	1	0.03	[0.99, 1.01]	0.67	1	0.84	1	1	0
$\text{ALG}_{100,10}$	1	0.03	[0.99, 1.01]	0.67	1	0.84	1	1	0
$\text{ALG}_{100,20}$	1	0.03	[0.99, 1.01]	0.67	1	0.84	1	1	0
$\text{ALG}_{100,30}$	1	0.03	[0.99, 1.01]	0.67	1	0.84	1	1	0
$\text{ALG}_{100,40}$	1	0.03	[0.99, 1.01]	0.67	1	0.84	1	1	0
$\text{ALG}_{100,50...100}$	0.84	0.22	[0.8, 0.88]	0.34	1	0.42	0.99	1	0

Table A.3: Performance ratios of costs relative to the online version of an algorithm in the ski rental problem.

A.2.2 Online Paging with Lookahead

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Fifo}_{1}$	63.71	0.07	[63.43, 63.99]	46	77	52	64	74	
$\textsc{Fifo}_{5}$	56.49	0.07	[56.24, 56.74]	44	72	46.5	56.5	66	0
$\textsc{Fifo}_{10}$	50.34	0.07	[50.12, 50.56]	40	66	42	50	58.5	0
$\textsc{Fifo}_{20}$	42.69	0.06	[42.53, 42.85]	35	52	37	43	49	0
$\textsc{Fifo}_{40}$	40.31	0.06	[40.16, 40.46]	33	48	34	40	47	0
$\textsc{Fifo}_{60}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Fifo}_{80}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Fifo}_{100}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Fifo}_{1,B}$	63.71	0.07	[63.43, 63.99]	46	77	52	64	74	
$\textsc{Fifo}_{5,B}$	60.29	0.07	[60.03, 60.55]	46	76	49.5	60	70	0.01
$\textsc{Fifo}_{10,B}$	56.11	0.07	[55.87, 56.35]	44	74	46	56	66	0.04
$\textsc{Fifo}_{20,B}$	49.97	0.07	[49.75, 50.19]	40	65	42	50	58	0.01
$\textsc{Fifo}_{40,B}$	44.89	0.07	[44.7, 45.08]	35	55	38	45	52	0.01
$\textsc{Fifo}_{60,B}$	42.57	0.07	[42.39, 42.75]	34	53	36	42	49	0.1
$\textsc{Fifo}_{80,B}$	42.76	0.07	[42.57, 42.95]	35	52	36	43	50	0.43
$\textsc{Fifo}_{100,B}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lifo}_{1}$	63.97	0.07	[63.69, 64.25]	52	78	53	64	75	
$\textsc{Lifo}_{5}$	56.82	0.07	[56.57, 57.07]	45	69	47	57	67	0
$\textsc{Lifo}_{10}$	50.48	0.07	[50.26, 50.7]	40	61	43	50	59	0
$\textsc{Lifo}_{20}$	42.77	0.07	[42.58, 42.96]	35	51	36.5	43	49.5	0
$\textsc{Lifo}_{40}$	40.31	0.06	[40.16, 40.46]	33	48	34	40	47	0
$\textsc{Lifo}_{60}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lifo}_{80}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lifo}_{100}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lifo}_{1,B}$	63.97	0.07	[63.69, 64.25]	52	78	53	64	75	
$\textsc{Lifo}_{5,B}$	60.51	0.07	[60.25, 60.77]	49	73	51	60	71	0.04
$\textsc{Lifo}_{10,B}$	56.39	0.07	[56.15, 56.63]	44	69	47	56	65.5	0.04
$\textsc{Lifo}_{20,B}$	49.98	0.07	[49.76, 50.2]	40	61	42	50	58	0.01
$\textsc{Lifo}_{40,B}$	44.96	0.07	[44.76, 45.16]	36	55	38	45	52	0.01
$\textsc{Lifo}_{60,B}$	42.58	0.07	[42.4, 42.76]	33	52	36	42.5	49	0.08
$\textsc{Lifo}_{80,B}$	42.79	0.07	[42.6, 42.98]	34	53	36	43	49	0.45
$\textsc{Lifo}_{100,B}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lfu}_{1}$	63.89	0.07	[63.61, 64.17]	50	77	53	64	74	
$\textsc{Lfu}_{5}$	56.86	0.07	[56.61, 57.11]	45	69	47	57	67	0
$\textsc{Lfu}_{10}$	50.53	0.07	[50.31, 50.75]	40	63	42.5	50	59.5	0
$\textsc{Lfu}_{20}$	42.75	0.07	[42.56, 42.94]	34	51	36	43	49.5	0
$\textsc{Lfu}_{40}$	40.31	0.06	[40.16, 40.46]	33	48	34	40	47	0
$\textsc{Lfu}_{60}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lfu}_{80}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lfu}_{100}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lfu}_{1,B}$	63.89	0.07	[63.61, 64.17]	50	77	53	64	74	
$\textsc{Lfu}_{5,B}$	60.59	0.07	[60.33, 60.85]	47	73	50	60	71	0
$\textsc{Lfu}_{10,B}$	56.4	0.07	[56.16, 56.64]	43	69	47	56	66	0.03
$\textsc{Lfu}_{20,B}$	49.97	0.07	[49.75, 50.19]	41	62	42	50	59	0
$\textsc{Lfu}_{40,B}$	44.94	0.07	[44.74, 45.14]	37	55	39	45	52	0
$\textsc{Lfu}_{60,B}$	42.5	0.07	[42.32, 42.68]	33	52	36	42	49	0.08
$\textsc{Lfu}_{80,B}$	42.77	0.07	[42.58, 42.96]	34	51	36.5	43	50	0.44
$\textsc{Lfu}_{100,B}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lru}_{1}$	63.63	0.07	[63.35, 63.91]	46	78	52	64	74	
$\textsc{Lru}_{5}$	56.68	0.07	[56.43, 56.93]	44	72	48	57	66.5	0
$\textsc{Lru}_{10}$	50.41	0.07	[50.19, 50.63]	38	64	42	50	58	0
$\textsc{Lru}_{20}$	42.71	0.07	[42.52, 42.9]	35	51	36	43	49	0
$\textsc{Lru}_{40}$	40.31	0.06	[40.16, 40.46]	33	48	34	40	47	0
$\textsc{Lru}_{60}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lru}_{80}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lru}_{100}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0
$\textsc{Lru}_{1,B}$	63.63	0.07	[63.35, 63.91]	46	78	52	64	74	
$\textsc{Lru}_{5,B}$	60.39	0.07	[60.13, 60.65]	46	76	50	60	70	0
$\textsc{Lru}_{10,B}$	56.15	0.07	[55.91, 56.39]	44	72	46.5	56	66	0.01
$\textsc{Lru}_{20,B}$	49.86	0.07	[49.64, 50.08]	39	64	42	50	58	0
$\textsc{Lru}_{40,B}$	44.94	0.07	[44.74, 45.14]	37	55	38	45	52	0
$\textsc{Lru}_{60,B}$	42.52	0.07	[42.34, 42.7]	33	54	36	42	49	0.04
$\textsc{Lru}_{80,B}$	42.81	0.07	[42.62, 43]	35	51	36	43	49.5	0.45
$\textsc{Lru}_{100,B}$	40.3	0.06	[40.15, 40.45]	33	48	34	40	47	0

Table A.4: Costs in the paging problem when each page is equally probable.

Performance ratios of costs relative to OPT for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
FIFO$_1$	1.58	0.05	[1.58, 1.58]	1.29	1.86	1.39	1.58	1.79	0
FIFO$_5$	1.4	0.05	[1.4, 1.4]	1.18	1.6	1.24	1.4	1.55	0
FIFO$_{10}$	1.25	0.04	[1.25, 1.25]	1.07	1.42	1.12	1.25	1.38	0
FIFO$_{20}$	1.06	0.03	[1.06, 1.06]	1	1.21	1	1.05	1.16	0
FIFO$_{40}$	1	0	[1, 1]	1	1.03	1	1	1	0
FIFO$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
FIFO$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
FIFO$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
FIFO$_{1,B}$	1.58	0.05	[1.58, 1.58]	1.29	1.86	1.39	1.58	1.79	0
FIFO$_{5,B}$	1.5	0.05	[1.5, 1.5]	1.28	1.73	1.32	1.5	1.66	0
FIFO$_{10,B}$	1.39	0.05	[1.39, 1.39]	1.16	1.61	1.23	1.39	1.55	0
FIFO$_{20,B}$	1.24	0.05	[1.24, 1.24]	1.07	1.41	1.11	1.24	1.38	0
FIFO$_{40,B}$	1.11	0.04	[1.11, 1.11]	1	1.32	1.02	1.11	1.23	0
FIFO$_{60,B}$	1.06	0.03	[1.06, 1.06]	1	1.17	1	1.05	1.13	0
FIFO$_{80,B}$	1.06	0.03	[1.06, 1.06]	1	1.18	1	1.05	1.14	0
FIFO$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_1$	1.59	0.08	[1.58, 1.6]	1.2	2.14	1.31	1.59	1.95	0
LIFO$_5$	1.41	0.07	[1.4, 1.42]	1.14	1.74	1.21	1.41	1.69	0
LIFO$_{10}$	1.25	0.06	[1.25, 1.25]	1.07	1.51	1.11	1.25	1.44	0
LIFO$_{20}$	1.06	0.03	[1.06, 1.06]	1	1.19	1	1.05	1.15	0
LIFO$_{40}$	1	0	[1, 1]	1	1.03	1	1	1	0
LIFO$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{1,B}$	1.59	0.08	[1.58, 1.6]	1.2	2.14	1.31	1.59	1.95	0
LIFO$_{5,B}$	1.5	0.08	[1.49, 1.51]	1.2	1.89	1.26	1.5	1.81	0
LIFO$_{10,B}$	1.4	0.07	[1.39, 1.41]	1.16	1.8	1.2	1.4	1.67	0
LIFO$_{20,B}$	1.24	0.05	[1.24, 1.24]	1.04	1.49	1.1	1.24	1.41	0
LIFO$_{40,B}$	1.12	0.04	[1.12, 1.12]	1	1.29	1.02	1.11	1.23	0
LIFO$_{60,B}$	1.06	0.03	[1.06, 1.06]	1	1.19	1	1.05	1.14	0
LIFO$_{80,B}$	1.06	0.03	[1.06, 1.06]	1	1.19	1	1.06	1.14	0
LIFO$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_1$	1.59	0.07	[1.58, 1.6]	1.3	1.89	1.36	1.59	1.85	0
LFU$_5$	1.41	0.06	[1.4, 1.42]	1.15	1.72	1.22	1.41	1.62	0
LFU$_{10}$	1.25	0.05	[1.25, 1.25]	1.07	1.49	1.11	1.25	1.42	0
LFU$_{20}$	1.06	0.03	[1.06, 1.06]	1	1.19	1	1.05	1.15	0
LFU$_{40}$	1	0	[1, 1]	1	1.03	1	1	1	0
LFU$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_{1,B}$	1.59	0.07	[1.58, 1.6]	1.3	1.89	1.36	1.59	1.85	0
LFU$_{5,B}$	1.51	0.06	[1.5, 1.52]	1.24	1.79	1.29	1.51	1.72	0
LFU$_{10,B}$	1.4	0.06	[1.39, 1.41]	1.13	1.69	1.22	1.4	1.61	0
LFU$_{20,B}$	1.24	0.05	[1.24, 1.24]	1.07	1.51	1.11	1.24	1.41	0
LFU$_{40,B}$	1.12	0.04	[1.12, 1.12]	1	1.26	1.02	1.11	1.23	0
LFU$_{60,B}$	1.05	0.03	[1.05, 1.05]	1	1.19	1	1.05	1.14	0
LFU$_{80,B}$	1.06	0.03	[1.06, 1.06]	1	1.18	1	1.05	1.15	0
LFU$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_1$	1.58	0.05	[1.58, 1.58]	1.3	1.82	1.41	1.58	1.77	0
LRU$_5$	1.41	0.04	[1.41, 1.41]	1.21	1.61	1.26	1.41	1.54	0
LRU$_{10}$	1.25	0.04	[1.25, 1.25]	1.08	1.43	1.14	1.25	1.37	0
LRU$_{20}$	1.06	0.03	[1.06, 1.06]	1	1.17	1	1.05	1.15	0
LRU$_{40}$	1	0	[1, 1]	1	1.03	1	1	1	0
LRU$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_{1,B}$	1.58	0.05	[1.58, 1.58]	1.3	1.82	1.41	1.58	1.77	0
LRU$_{5,B}$	1.5	0.05	[1.5, 1.5]	1.23	1.7	1.34	1.5	1.65	0
LRU$_{10,B}$	1.39	0.04	[1.39, 1.39]	1.21	1.57	1.24	1.39	1.53	0
LRU$_{20,B}$	1.24	0.04	[1.24, 1.24]	1.1	1.39	1.12	1.24	1.35	0
LRU$_{40,B}$	1.12	0.03	[1.12, 1.12]	1.02	1.24	1.03	1.12	1.22	0
LRU$_{60,B}$	1.06	0.03	[1.06, 1.06]	1	1.15	1	1.05	1.12	0
LRU$_{80,B}$	1.06	0.03	[1.06, 1.06]	1	1.15	1	1.07	1.14	0
LRU$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.5: Performance ratios of costs relative to OPT in the paging problem when each page is equally probable.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1-F(1)$
$\textsc{Fifo}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Fifo}_{5}$	0.89	0.05	[0.89, 0.89]	0.73	1	0.77	0.89	0.98	0
$\textsc{Fifo}_{10}$	0.79	0.06	[0.79, 0.79]	0.66	0.93	0.69	0.79	0.9	0
$\textsc{Fifo}_{20}$	0.67	0.06	[0.67, 0.67]	0.56	0.81	0.59	0.67	0.77	0
$\textsc{Fifo}_{40}$	0.63	0.05	[0.63, 0.63]	0.54	0.78	0.56	0.63	0.72	0
$\textsc{Fifo}_{60}$	0.63	0.05	[0.63, 0.63]	0.54	0.78	0.56	0.63	0.72	0
$\textsc{Fifo}_{80}$	0.63	0.05	[0.63, 0.63]	0.54	0.78	0.56	0.63	0.72	0
$\textsc{Fifo}_{100}$	0.63	0.05	[0.63, 0.63]	0.54	0.78	0.56	0.63	0.72	0
$\textsc{Fifo}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Fifo}_{5,B}$	0.95	0.04	[0.95, 0.95]	0.76	1.04	0.83	0.95	1	0.01
$\textsc{Fifo}_{10,B}$	0.88	0.06	[0.88, 0.88]	0.7	1	0.77	0.88	0.98	0
$\textsc{Fifo}_{20,B}$	0.79	0.06	[0.79, 0.79]	0.65	0.96	0.68	0.78	0.9	0
$\textsc{Fifo}_{40,B}$	0.71	0.06	[0.71, 0.71]	0.59	0.87	0.61	0.7	0.81	0
$\textsc{Fifo}_{60,B}$	0.67	0.06	[0.67, 0.67]	0.56	0.8	0.59	0.67	0.76	0
$\textsc{Fifo}_{80,B}$	0.67	0.06	[0.67, 0.67]	0.54	0.8	0.59	0.67	0.77	0
$\textsc{Fifo}_{100,B}$	0.63	0.05	[0.63, 0.63]	0.54	0.78	0.56	0.63	0.72	0
$\textsc{Lifo}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lifo}_{5}$	0.89	0.05	[0.89, 0.89]	0.74	1.02	0.77	0.89	1	0
$\textsc{Lifo}_{10}$	0.79	0.07	[0.79, 0.79]	0.62	1	0.67	0.79	0.92	0
$\textsc{Lifo}_{20}$	0.67	0.08	[0.67, 0.67]	0.52	0.85	0.56	0.67	0.8	0
$\textsc{Lifo}_{40}$	0.63	0.08	[0.63, 0.63]	0.47	0.83	0.51	0.63	0.76	0
$\textsc{Lifo}_{60}$	0.63	0.08	[0.63, 0.63]	0.47	0.83	0.51	0.63	0.76	0
$\textsc{Lifo}_{80}$	0.63	0.08	[0.63, 0.63]	0.47	0.83	0.51	0.63	0.76	0
$\textsc{Lifo}_{100}$	0.63	0.08	[0.63, 0.63]	0.47	0.83	0.51	0.63	0.76	0
$\textsc{Lifo}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lifo}_{5,B}$	0.95	0.04	[0.95, 0.95]	0.8	1.05	0.85	0.95	1.03	0.04
$\textsc{Lifo}_{10,B}$	0.88	0.06	[0.88, 0.88]	0.68	1.03	0.76	0.88	1	0
$\textsc{Lifo}_{20,B}$	0.78	0.07	[0.78, 0.78]	0.63	0.96	0.65	0.78	0.9	0
$\textsc{Lifo}_{40,B}$	0.71	0.08	[0.71, 0.71]	0.55	0.89	0.58	0.7	0.85	0
$\textsc{Lifo}_{60,B}$	0.67	0.08	[0.67, 0.67]	0.5	0.87	0.54	0.67	0.8	0
$\textsc{Lifo}_{80,B}$	0.67	0.09	[0.67, 0.67]	0.51	0.89	0.55	0.67	0.81	0
$\textsc{Lifo}_{100,B}$	0.63	0.08	[0.63, 0.63]	0.47	0.83	0.51	0.63	0.76	0
$\textsc{Lfu}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lfu}_{5}$	0.89	0.04	[0.89, 0.89]	0.76	1	0.79	0.89	0.97	0
$\textsc{Lfu}_{10}$	0.79	0.05	[0.79, 0.79]	0.65	0.95	0.7	0.79	0.9	0
$\textsc{Lfu}_{20}$	0.67	0.06	[0.67, 0.67]	0.54	0.8	0.58	0.67	0.78	0
$\textsc{Lfu}_{40}$	0.63	0.07	[0.63, 0.63]	0.53	0.77	0.54	0.63	0.73	0
$\textsc{Lfu}_{60}$	0.63	0.07	[0.63, 0.63]	0.53	0.77	0.54	0.63	0.73	0
$\textsc{Lfu}_{80}$	0.63	0.07	[0.63, 0.63]	0.53	0.77	0.54	0.63	0.73	0
$\textsc{Lfu}_{100}$	0.63	0.07	[0.63, 0.63]	0.53	0.77	0.54	0.63	0.73	0
$\textsc{Lfu}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lfu}_{5,B}$	0.95	0.03	[0.95, 0.95]	0.82	1.03	0.87	0.95	1	0
$\textsc{Lfu}_{10,B}$	0.88	0.04	[0.88, 0.88]	0.75	1	0.79	0.89	0.97	0
$\textsc{Lfu}_{20,B}$	0.78	0.06	[0.78, 0.78]	0.65	0.93	0.68	0.78	0.89	0
$\textsc{Lfu}_{40,B}$	0.71	0.06	[0.71, 0.71]	0.57	0.85	0.6	0.7	0.81	0
$\textsc{Lfu}_{60,B}$	0.67	0.07	[0.67, 0.67]	0.53	0.8	0.57	0.67	0.77	0
$\textsc{Lfu}_{80,B}$	0.67	0.07	[0.67, 0.67]	0.55	0.82	0.58	0.67	0.78	0
$\textsc{Lfu}_{100,B}$	0.63	0.07	[0.63, 0.63]	0.53	0.77	0.54	0.63	0.73	0
$\textsc{Lru}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lru}_{5}$	0.89	0.04	[0.89, 0.89]	0.78	0.98	0.81	0.89	0.97	0
$\textsc{Lru}_{10}$	0.79	0.05	[0.79, 0.79]	0.66	0.92	0.7	0.79	0.9	0
$\textsc{Lru}_{20}$	0.67	0.06	[0.67, 0.67]	0.57	0.83	0.59	0.67	0.77	0
$\textsc{Lru}_{40}$	0.63	0.05	[0.63, 0.63]	0.55	0.77	0.57	0.63	0.71	0
$\textsc{Lru}_{60}$	0.63	0.05	[0.63, 0.63]	0.55	0.77	0.57	0.63	0.71	0
$\textsc{Lru}_{80}$	0.63	0.05	[0.63, 0.63]	0.55	0.77	0.57	0.63	0.71	0
$\textsc{Lru}_{100}$	0.63	0.05	[0.63, 0.63]	0.55	0.77	0.57	0.63	0.71	0
$\textsc{Lru}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lru}_{5,B}$	0.95	0.03	[0.95, 0.95]	0.86	1	0.88	0.95	1	0
$\textsc{Lru}_{10,B}$	0.88	0.04	[0.88, 0.88]	0.77	0.98	0.79	0.88	0.97	0
$\textsc{Lru}_{20,B}$	0.78	0.05	[0.78, 0.78]	0.66	0.9	0.69	0.78	0.89	0
$\textsc{Lru}_{40,B}$	0.71	0.05	[0.71, 0.71]	0.59	0.83	0.63	0.71	0.8	0
$\textsc{Lru}_{60,B}$	0.67	0.05	[0.67, 0.67]	0.58	0.81	0.6	0.67	0.75	0
$\textsc{Lru}_{80,B}$	0.67	0.05	[0.67, 0.67]	0.57	0.79	0.6	0.68	0.76	0
$\textsc{Lru}_{100,B}$	0.63	0.05	[0.63, 0.63]	0.55	0.77	0.57	0.63	0.71	0

Table A.6: Performance ratios of costs relative to the online version of an algorithm in the paging problem when each page is equally probable.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
FIFO$_{1}$	22.77	0.26	[22.4, 23.14]	6	42	10	23	37	
FIFO$_{5}$	21.78	0.25	[21.44, 22.12]	6	40	10	22	35	0
FIFO$_{10}$	21.05	0.25	[20.72, 21.38]	6	35	10	21	33.5	0
FIFO$_{20}$	19.91	0.23	[19.63, 20.19]	6	35	10	20	30	0
FIFO$_{40}$	18.88	0.21	[18.63, 19.13]	6	31	10	19	28	0
FIFO$_{60}$	18.64	0.2	[18.41, 18.87]	6	29	10	19	27	0
FIFO$_{80}$	18.6	0.2	[18.37, 18.83]	6	29	10	19	27	0
FIFO$_{100}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
FIFO$_{1,B}$	22.77	0.26	[22.4, 23.14]	6	42	10	23	37	
FIFO$_{5,B}$	22.28	0.26	[21.92, 22.64]	6	42	10	22	36	0
FIFO$_{10,B}$	21.84	0.25	[21.5, 22.18]	6	40	10	22	34.5	0.05
FIFO$_{20,B}$	21.02	0.24	[20.71, 21.33]	6	35	10	21	33	0.03
FIFO$_{40,B}$	20.26	0.23	[19.97, 20.55]	6	35	10	20	31	0.05
FIFO$_{60,B}$	19.66	0.22	[19.39, 19.93]	6	32	10	20	29	0.15
FIFO$_{80,B}$	19.6	0.22	[19.33, 19.87]	6	33	10	20	29	0.27
FIFO$_{100,B}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LIFO$_{1}$	45.08	0.41	[43.93, 46.23]	6	90	10	46	86	
LIFO$_{5}$	23.92	0.27	[23.52, 24.32]	6	46	10	24	39	0
LIFO$_{10}$	21.67	0.25	[21.33, 22.01]	6	36	10	22	33	0
LIFO$_{20}$	19.97	0.22	[19.7, 20.24]	6	33	10	20	30	0
LIFO$_{40}$	18.88	0.21	[18.63, 19.13]	6	30	10	19	27.5	0
LIFO$_{60}$	18.63	0.2	[18.4, 18.86]	6	29	10	19	27	0
LIFO$_{80}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LIFO$_{100}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LIFO$_{1,B}$	45.08	0.41	[43.93, 46.23]	6	90	10	46	86	
LIFO$_{5,B}$	27.45	0.3	[26.94, 27.96]	6	52	10	28	46.5	0.01
LIFO$_{10,B}$	24.53	0.28	[24.1, 24.96]	6	49	10	25	40.5	0.1
LIFO$_{20,B}$	22.03	0.25	[21.69, 22.37]	6	37	10	22	35	0.08
LIFO$_{40,B}$	20.66	0.23	[20.37, 20.95]	6	34	10	21	32	0.08
LIFO$_{60,B}$	19.84	0.22	[19.57, 20.11]	6	32	10	20	29	0.19
LIFO$_{80,B}$	19.71	0.21	[19.45, 19.97]	6	31	10	20	29	0.28
LIFO$_{100,B}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LFU$_{1}$	30.55	0.32	[29.94, 31.16]	6	55	10	31	51	
LFU$_{5}$	22.6	0.26	[22.24, 22.96]	6	38	10	23	35	0
LFU$_{10}$	21.09	0.24	[20.78, 21.4]	6	36	10	21	32	0
LFU$_{20}$	19.79	0.22	[19.52, 20.06]	6	33	10	20	29	0
LFU$_{40}$	18.83	0.21	[18.58, 19.08]	6	29	10	19	27	0
LFU$_{60}$	18.63	0.2	[18.4, 18.86]	6	29	10	19	27	0
LFU$_{80}$	18.6	0.2	[18.37, 18.83]	6	29	10	19	27	0
LFU$_{100}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LFU$_{1,B}$	30.55	0.32	[29.94, 31.16]	6	55	10	31	51	
LFU$_{5,B}$	24.85	0.28	[24.42, 25.28]	6	44	10	25	39	0
LFU$_{10,B}$	23.16	0.26	[22.79, 23.53]	6	40	10	23	36.5	0.11
LFU$_{20,B}$	21.52	0.24	[21.2, 21.84]	6	36	10	22	33	0.08
LFU$_{40,B}$	20.45	0.23	[20.16, 20.74]	6	33	10	21	31	0.08
LFU$_{60,B}$	19.72	0.22	[19.45, 19.99]	6	31	10	20	29	0.19
LFU$_{80,B}$	19.67	0.22	[19.4, 19.94]	6	31	10	20	29	0.28
LFU$_{100,B}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LRU$_{1}$	21	0.24	[20.69, 21.31]	6	38	10	21	33	
LRU$_{5}$	20.61	0.24	[20.3, 20.92]	6	37	10	21	32	0
LRU$_{10}$	20.18	0.23	[19.89, 20.47]	6	34	10	20	31	0
LRU$_{20}$	19.47	0.22	[19.2, 19.74]	6	32	10	20	29	0
LRU$_{40}$	18.78	0.21	[18.54, 19.02]	6	29	10	19	27.5	0
LRU$_{60}$	18.61	0.2	[18.38, 18.84]	6	29	10	19	27	0
LRU$_{80}$	18.6	0.2	[18.37, 18.83]	6	29	10	19	27	0
LRU$_{100}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0
LRU$_{1,B}$	21	0.24	[20.69, 21.31]	6	38	10	21	33	
LRU$_{5,B}$	20.83	0.24	[20.52, 21.14]	6	37	10	21	33	0
LRU$_{10,B}$	20.56	0.24	[20.25, 20.87]	6	36	10	21	32	0.02
LRU$_{20,B}$	20.14	0.23	[19.85, 20.43]	6	36	10	20	31	0.04
LRU$_{40,B}$	19.65	0.22	[19.38, 19.92]	6	34	10	20	30	0.03
LRU$_{60,B}$	19.29	0.22	[19.03, 19.55]	6	31	10	19	28.5	0.15
LRU$_{80,B}$	19.17	0.21	[18.92, 19.42]	6	30	10	19	28.5	0.2
LRU$_{100,B}$	18.59	0.2	[18.36, 18.82]	6	29	10	19	27	0

Table A.7: Costs in the paging problem when page sequences are generated according to an access graph.

Performance ratios of costs relative to Opt for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Fifo$_1$	1.22	0.14	[1.21, 1.23]	1	1.94	1	1.21	1.69	0
Fifo$_5$	1.16	0.11	[1.15, 1.17]	1	1.65	1	1.16	1.49	0
Fifo$_{10}$	1.13	0.1	[1.12, 1.14]	1	1.5	1	1.11	1.4	0
Fifo$_{20}$	1.07	0.07	[1.07, 1.07]	1	1.33	1	1.05	1.28	0
Fifo$_{40}$	1.01	0.03	[1.01, 1.01]	1	1.19	1	1	1.15	0
Fifo$_{60}$	1	0.01	[1, 1]	1	1.16	1	1	1.06	0
Fifo$_{80}$	1	0	[1, 1]	1	1.06	1	1	1	0
Fifo$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Fifo$_{1,B}$	1.22	0.14	[1.21, 1.23]	1	1.94	1	1.21	1.69	0
Fifo$_{5,B}$	1.19	0.13	[1.18, 1.2]	1	1.76	1	1.18	1.58	0
Fifo$_{10,B}$	1.17	0.12	[1.16, 1.18]	1	1.58	1	1.16	1.5	0
Fifo$_{20,B}$	1.12	0.1	[1.11, 1.13]	1	1.53	1	1.11	1.39	0
Fifo$_{40,B}$	1.09	0.08	[1.08, 1.1]	1	1.39	1	1.06	1.31	0
Fifo$_{60,B}$	1.05	0.07	[1.05, 1.05]	1	1.33	1	1.04	1.27	0
Fifo$_{80,B}$	1.05	0.06	[1.05, 1.05]	1	1.35	1	1.04	1.24	0
Fifo$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
Lifo$_1$	2.38	0.35	[2.33, 2.43]	1	5.47	1	2.31	4.63	0
Lifo$_5$	1.28	0.16	[1.27, 1.29]	1	2.18	1	1.24	1.89	0
Lifo$_{10}$	1.16	0.12	[1.15, 1.17]	1	1.76	1	1.14	1.53	0
Lifo$_{20}$	1.07	0.07	[1.07, 1.07]	1	1.47	1	1.05	1.3	0
Lifo$_{40}$	1.01	0.03	[1.01, 1.01]	1	1.22	1	1	1.16	0
Lifo$_{60}$	1	0.01	[1, 1]	1	1.19	1	1	1.06	0
Lifo$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
Lifo$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Lifo$_{1,B}$	2.38	0.35	[2.33, 2.43]	1	5.47	1	2.31	4.63	0
Lifo$_{5,B}$	1.46	0.2	[1.44, 1.48]	1	2.6	1	1.44	2.3	0
Lifo$_{10,B}$	1.31	0.16	[1.3, 1.32]	1	2.33	1	1.28	1.89	0
Lifo$_{20,B}$	1.18	0.12	[1.17, 1.19]	1	1.76	1	1.15	1.58	0
Lifo$_{40,B}$	1.11	0.08	[1.1, 1.12]	1	1.59	1	1.1	1.38	0
Lifo$_{60,B}$	1.07	0.07	[1.07, 1.07]	1	1.41	1	1.05	1.26	0
Lifo$_{80,B}$	1.06	0.06	[1.06, 1.06]	1	1.36	1	1.05	1.25	0
Lifo$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
Lfu$_1$	1.61	0.19	[1.59, 1.63]	1	2.45	1	1.62	2.3	0
Lfu$_5$	1.2	0.11	[1.19, 1.21]	1	1.71	1	1.2	1.53	0
Lfu$_{10}$	1.13	0.08	[1.12, 1.14]	1	1.5	1	1.12	1.35	0
Lfu$_{20}$	1.06	0.06	[1.06, 1.06]	1	1.29	1	1.05	1.24	0
Lfu$_{40}$	1.01	0.03	[1.01, 1.01]	1	1.19	1	1	1.13	0
Lfu$_{60}$	1	0.01	[1, 1]	1	1.12	1	1	1.06	0
Lfu$_{80}$	1	0	[1, 1]	1	1.06	1	1	1	0
Lfu$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Lfu$_{1,B}$	1.61	0.19	[1.59, 1.63]	1	2.45	1	1.62	2.3	0
Lfu$_{5,B}$	1.32	0.14	[1.31, 1.33]	1	2.05	1	1.3	1.77	0
Lfu$_{10,B}$	1.23	0.11	[1.22, 1.24]	1	1.71	1	1.22	1.6	0
Lfu$_{20,B}$	1.15	0.09	[1.14, 1.16]	1	1.5	1	1.14	1.42	0
Lfu$_{40,B}$	1.1	0.07	[1.1, 1.1]	1	1.44	1	1.09	1.33	0
Lfu$_{60,B}$	1.06	0.06	[1.06, 1.06]	1	1.26	1	1.05	1.22	0
Lfu$_{80,B}$	1.06	0.06	[1.06, 1.06]	1	1.31	1	1.05	1.23	0
Lfu$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
Lru$_1$	1.12	0.1	[1.11, 1.13]	1	1.5	1	1.11	1.4	0
Lru$_5$	1.1	0.09	[1.09, 1.11]	1	1.43	1	1.08	1.35	0
Lru$_{10}$	1.08	0.08	[1.07, 1.09]	1	1.36	1	1.06	1.3	0
Lru$_{20}$	1.04	0.06	[1.04, 1.04]	1	1.32	1	1	1.22	0
Lru$_{40}$	1.01	0.03	[1.01, 1.01]	1	1.19	1	1	1.12	0
Lru$_{60}$	1	0.01	[1, 1]	1	1.12	1	1	1.05	0
Lru$_{80}$	1	0	[1, 1]	1	1.06	1	1	1	0
Lru$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Lru$_{1,B}$	1.12	0.1	[1.11, 1.13]	1	1.5	1	1.11	1.4	0
Lru$_{5,B}$	1.11	0.09	[1.1, 1.12]	1	1.5	1	1.1	1.39	0
Lru$_{10,B}$	1.1	0.09	[1.09, 1.11]	1	1.41	1	1.09	1.33	0
Lru$_{20,B}$	1.08	0.07	[1.08, 1.08]	1	1.36	1	1.06	1.3	0
Lru$_{40,B}$	1.05	0.06	[1.05, 1.05]	1	1.33	1	1.04	1.25	0
Lru$_{60,B}$	1.04	0.05	[1.04, 1.04]	1	1.33	1	1	1.21	0
Lru$_{80,B}$	1.03	0.05	[1.03, 1.03]	1	1.25	1	1	1.2	0
Lru$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.8: Performance ratios of costs relative to Opt in the paging problem when page sequences are generated according to an access graph.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
$\textsc{Fifo}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Fifo}_{5}$	0.96	0.06	[0.96, 0.96]	0.7	1.04	0.73	1	1	0
$\textsc{Fifo}_{10}$	0.93	0.08	[0.93, 0.93]	0.59	1.04	0.69	0.95	1	0
$\textsc{Fifo}_{20}$	0.89	0.11	[0.88, 0.9]	0.59	1	0.64	0.91	1	0
$\textsc{Fifo}_{40}$	0.85	0.14	[0.84, 0.86]	0.52	1	0.59	0.85	1	0
$\textsc{Fifo}_{60}$	0.84	0.14	[0.83, 0.85]	0.52	1	0.59	0.83	1	0
$\textsc{Fifo}_{80}$	0.84	0.14	[0.83, 0.85]	0.52	1	0.59	0.83	1	0
$\textsc{Fifo}_{100}$	0.84	0.14	[0.83, 0.85]	0.52	1	0.59	0.83	1	0
$\textsc{Fifo}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Fifo}_{5,B}$	0.98	0.04	[0.98, 0.98]	0.71	1.12	0.78	1	1	0
$\textsc{Fifo}_{10,B}$	0.96	0.06	[0.96, 0.96]	0.67	1.05	0.74	1	1	0
$\textsc{Fifo}_{20,B}$	0.93	0.09	[0.92, 0.94]	0.62	1.04	0.68	0.96	1	0
$\textsc{Fifo}_{40,B}$	0.9	0.11	[0.89, 0.91]	0.6	1	0.64	0.92	1	0
$\textsc{Fifo}_{60,B}$	0.88	0.12	[0.87, 0.89]	0.52	1.05	0.62	0.89	1	0
$\textsc{Fifo}_{80,B}$	0.88	0.12	[0.87, 0.89]	0.57	1	0.62	0.89	1	0
$\textsc{Fifo}_{100,B}$	0.84	0.14	[0.83, 0.85]	0.52	1	0.59	0.83	1	0
$\textsc{Lifo}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lifo}_{5}$	0.59	0.29	[0.58, 0.6]	0.24	1.06	0.29	0.56	1	0
$\textsc{Lifo}_{10}$	0.54	0.33	[0.53, 0.55]	0.22	1	0.26	0.51	1	0
$\textsc{Lifo}_{20}$	0.51	0.36	[0.5, 0.52]	0.2	1	0.24	0.47	1	0
$\textsc{Lifo}_{40}$	0.48	0.38	[0.47, 0.49]	0.18	1	0.22	0.44	1	0
$\textsc{Lifo}_{60}$	0.48	0.39	[0.47, 0.49]	0.18	1	0.22	0.43	1	0
$\textsc{Lifo}_{80}$	0.48	0.39	[0.47, 0.49]	0.18	1	0.22	0.43	1	0
$\textsc{Lifo}_{100}$	0.48	0.39	[0.47, 0.49]	0.18	1	0.22	0.43	1	0
$\textsc{Lifo}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lifo}_{5,B}$	0.66	0.25	[0.65, 0.67]	0.3	1.28	0.35	0.64	1	0.01
$\textsc{Lifo}_{10,B}$	0.6	0.29	[0.59, 0.61]	0.26	1.06	0.3	0.58	1	0
$\textsc{Lifo}_{20,B}$	0.55	0.32	[0.54, 0.56]	0.2	1	0.26	0.52	1	0
$\textsc{Lifo}_{40,B}$	0.52	0.35	[0.51, 0.53]	0.2	1	0.24	0.48	1	0
$\textsc{Lifo}_{60,B}$	0.51	0.36	[0.5, 0.52]	0.2	1	0.23	0.46	1	0
$\textsc{Lifo}_{80,B}$	0.5	0.37	[0.49, 0.51]	0.18	1	0.23	0.46	1	0
$\textsc{Lifo}_{100,B}$	0.48	0.39	[0.47, 0.49]	0.18	1	0.22	0.43	1	0
$\textsc{Lfu}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lfu}_{5}$	0.77	0.14	[0.76, 0.78]	0.51	1	0.55	0.75	1	0
$\textsc{Lfu}_{10}$	0.72	0.17	[0.71, 0.73]	0.46	1	0.51	0.7	1	0
$\textsc{Lfu}_{20}$	0.68	0.19	[0.67, 0.69]	0.43	1	0.48	0.66	1	0
$\textsc{Lfu}_{40}$	0.65	0.21	[0.64, 0.66]	0.41	1	0.44	0.63	1	0
$\textsc{Lfu}_{60}$	0.65	0.21	[0.64, 0.66]	0.41	1	0.43	0.62	1	0
$\textsc{Lfu}_{80}$	0.65	0.21	[0.64, 0.66]	0.41	1	0.43	0.62	1	0
$\textsc{Lfu}_{100}$	0.65	0.21	[0.64, 0.66]	0.41	1	0.43	0.62	1	0
$\textsc{Lfu}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lfu}_{5,B}$	0.83	0.12	[0.82, 0.84]	0.55	1	0.63	0.83	1	0
$\textsc{Lfu}_{10,B}$	0.78	0.14	[0.77, 0.79]	0.48	1	0.57	0.77	1	0
$\textsc{Lfu}_{20,B}$	0.74	0.16	[0.73, 0.75]	0.49	1	0.53	0.72	1	0
$\textsc{Lfu}_{40,B}$	0.7	0.18	[0.69, 0.71]	0.46	1	0.49	0.68	1	0
$\textsc{Lfu}_{60,B}$	0.68	0.19	[0.67, 0.69]	0.42	1	0.47	0.66	1	0
$\textsc{Lfu}_{80,B}$	0.68	0.19	[0.67, 0.69]	0.44	1	0.46	0.66	1	0
$\textsc{Lfu}_{100,B}$	0.65	0.21	[0.64, 0.66]	0.41	1	0.43	0.62	1	0
$\textsc{Lru}_{1}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lru}_{5}$	0.98	0.03	[0.98, 0.98]	0.79	1	0.86	1	1	0
$\textsc{Lru}_{10}$	0.97	0.05	[0.97, 0.97]	0.74	1	0.8	1	1	0
$\textsc{Lru}_{20}$	0.94	0.07	[0.94, 0.94]	0.71	1	0.75	0.95	1	0
$\textsc{Lru}_{40}$	0.91	0.09	[0.9, 0.92]	0.67	1	0.71	0.92	1	0
$\textsc{Lru}_{60}$	0.9	0.1	[0.89, 0.91]	0.67	1	0.71	0.9	1	0
$\textsc{Lru}_{80}$	0.9	0.1	[0.89, 0.91]	0.67	1	0.71	0.9	1	0
$\textsc{Lru}_{100}$	0.9	0.1	[0.89, 0.91]	0.67	1	0.71	0.9	1	0
$\textsc{Lru}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Lru}_{5,B}$	0.99	0.02	[0.99, 0.99]	0.83	1	0.89	1	1	0
$\textsc{Lru}_{10,B}$	0.98	0.04	[0.98, 0.98]	0.79	1	0.85	1	1	0
$\textsc{Lru}_{20,B}$	0.96	0.05	[0.96, 0.96]	0.71	1	0.8	1	1	0
$\textsc{Lru}_{40,B}$	0.94	0.07	[0.94, 0.94]	0.68	1	0.75	0.96	1	0
$\textsc{Lru}_{60,B}$	0.93	0.08	[0.93, 0.93]	0.67	1	0.72	0.95	1	0
$\textsc{Lru}_{80,B}$	0.92	0.09	[0.91, 0.93]	0.67	1	0.74	0.95	1	0
$\textsc{Lru}_{100,B}$	0.9	0.1	[0.89, 0.91]	0.67	1	0.71	0.9	1	0

Table A.9: Performance ratios of costs relative to the online version of an algorithm in the paging problem when page sequences are generated according to an access graph.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Fifo$_{1}$	44.55	0.12	[44.22, 44.88]	27	60	33	44.5	56.5	
Fifo$_{5}$	39.3	0.11	[39.03, 39.57]	26	52	29	39	49.5	0
Fifo$_{10}$	34.46	0.11	[34.22, 34.7]	23	46	26	34	43	0
Fifo$_{20}$	28.87	0.1	[28.69, 29.05]	18	37	22	29	36	0
Fifo$_{40}$	26.62	0.1	[26.45, 26.79]	18	35	21	27	33	0
Fifo$_{60}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Fifo$_{80}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Fifo$_{100}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Fifo$_{1,B}$	44.55	0.12	[44.22, 44.88]	27	60	33	44.5	56.5	
Fifo$_{5,B}$	42.16	0.11	[41.87, 42.45]	25	56	31	42	53	0.03
Fifo$_{10,B}$	39	0.11	[38.73, 39.27]	24	53	29	39	50	0.05
Fifo$_{20,B}$	34.16	0.11	[33.93, 34.39]	23	45	26	34	43	0.01
Fifo$_{40,B}$	30.69	0.11	[30.48, 30.9]	21	40	23	31	38	0.02
Fifo$_{60,B}$	28.55	0.11	[28.36, 28.74]	19	37	22	28	36	0.1
Fifo$_{80,B}$	28.88	0.11	[28.68, 29.08]	18	37	22	29	36	0.45
Fifo$_{100,B}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lifo$_{1}$	41.78	0.16	[41.37, 42.19]	25	67	28	41	59	
Lifo$_{5}$	36.28	0.13	[35.99, 36.57]	24	54	26	36	48	0
Lifo$_{10}$	32.4	0.12	[32.16, 32.64]	21	47	24	32	42	0
Lifo$_{20}$	28.19	0.1	[28.02, 28.36]	19	36	22	28	35	0
Lifo$_{40}$	26.61	0.1	[26.44, 26.78]	18	35	21	27	33	0
Lifo$_{60}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lifo$_{80}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lifo$_{100}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lifo$_{1,B}$	41.78	0.16	[41.37, 42.19]	25	67	28	41	59	
Lifo$_{5,B}$	38.94	0.14	[38.6, 39.28]	24	60	27	39	52	0.08
Lifo$_{10,B}$	35.9	0.13	[35.61, 36.19]	23	52	26	36	47.5	0.07
Lifo$_{20,B}$	31.89	0.11	[31.67, 32.11]	21	43	24	32	40	0.02
Lifo$_{40,B}$	29.27	0.1	[29.09, 29.45]	19	38	22.5	29	36	0.04
Lifo$_{60,B}$	27.69	0.1	[27.52, 27.86]	20	37	22	28	34	0.1
Lifo$_{80,B}$	27.92	0.1	[27.75, 28.09]	19	37	22	28	35	0.44
Lifo$_{100,B}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lfu$_{1}$	37.46	0.13	[37.16, 37.76]	23	54	27	37	50	
Lfu$_{5}$	34.18	0.12	[33.93, 34.43]	22	45	25	34	44	0
Lfu$_{10}$	31.27	0.11	[31.06, 31.48]	22	42	24	31	39.5	0
Lfu$_{20}$	27.88	0.1	[27.71, 28.05]	20	36	21.5	28	34.5	0
Lfu$_{40}$	26.62	0.1	[26.45, 26.79]	18	35	21	27	33	0
Lfu$_{60}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lfu$_{80}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lfu$_{100}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lfu$_{1,B}$	37.46	0.13	[37.16, 37.76]	23	54	27	37	50	
Lfu$_{5,B}$	35.93	0.13	[35.64, 36.22]	23	50	26.5	36	46.5	0
Lfu$_{10,B}$	33.99	0.12	[33.74, 34.24]	22	46	26	34	43.5	0.06
Lfu$_{20,B}$	31.15	0.11	[30.94, 31.36]	23	43	24	31	39.5	0.03
Lfu$_{40,B}$	28.96	0.1	[28.78, 29.14]	20	37	22.5	29	36	0.04
Lfu$_{60,B}$	27.58	0.1	[27.41, 27.75]	20	37	21.5	27	34	0.1
Lfu$_{80,B}$	27.76	0.1	[27.59, 27.93]	19	36	21	28	34	0.39
Lfu$_{100,B}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lru$_{1}$	41.49	0.13	[41.16, 41.82]	25	61	30	41	53	
Lru$_{5}$	36.65	0.12	[36.38, 36.92]	20	51	27	37	47	0
Lru$_{10}$	32.77	0.11	[32.55, 32.99]	20	44	25	33	41	0
Lru$_{20}$	28.37	0.1	[28.19, 28.55]	18	36	22	28	35	0
Lru$_{40}$	26.62	0.1	[26.45, 26.79]	18	35	21	27	33	0
Lru$_{60}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lru$_{80}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lru$_{100}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0
Lru$_{1,B}$	41.49	0.13	[41.16, 41.82]	25	61	30	41	53	
Lru$_{5,B}$	39.16	0.12	[38.87, 39.45]	23	56	29	39	49	0
Lru$_{10,B}$	36.47	0.12	[36.2, 36.74]	22	51	26.5	36	47	0.03
Lru$_{20,B}$	32.55	0.11	[32.33, 32.77]	21	44	25	32.5	41	0.01
Lru$_{40,B}$	29.73	0.11	[29.53, 29.93]	20	40	23	30	37	0.01
Lru$_{60,B}$	27.98	0.1	[27.81, 28.15]	18	37	22	28	34	0.08
Lru$_{80,B}$	28.21	0.1	[28.04, 28.38]	19	37	22	28	35	0.41
Lru$_{100,B}$	26.59	0.1	[26.43, 26.75]	18	35	21	27	33	0

Table A.10: Costs in the paging problem when pages are stochastically distributed.

Performance ratios of costs relative to OPT for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
FIFO$_{1}$	1.68	0.07	[1.67, 1.69]	1.32	2.04	1.42	1.68	1.96	0
FIFO$_{5}$	1.48	0.06	[1.47, 1.49]	1.12	1.78	1.27	1.48	1.7	0
FIFO$_{10}$	1.3	0.06	[1.3, 1.3]	1.08	1.56	1.13	1.3	1.46	0
FIFO$_{20}$	1.09	0.05	[1.09, 1.09]	1	1.29	1	1.08	1.22	0
FIFO$_{40}$	1	0.01	[1, 1]	1	1.09	1	1	1.04	0
FIFO$_{60}$	1	0	[1, 1]	1	1.04	1	1	1	0
FIFO$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
FIFO$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
FIFO$_{1,B}$	1.68	0.07	[1.67, 1.69]	1.32	2.04	1.42	1.68	1.96	0
FIFO$_{5,B}$	1.59	0.06	[1.58, 1.6]	1.28	1.88	1.36	1.58	1.83	0
FIFO$_{10,B}$	1.47	0.06	[1.46, 1.48]	1.19	1.74	1.28	1.46	1.69	0
FIFO$_{20,B}$	1.29	0.06	[1.29, 1.29]	1.05	1.52	1.11	1.29	1.46	0
FIFO$_{40,B}$	1.16	0.05	[1.16, 1.16]	1	1.41	1.03	1.15	1.32	0
FIFO$_{60,B}$	1.07	0.04	[1.07, 1.07]	1	1.25	1	1.07	1.2	0
FIFO$_{80,B}$	1.09	0.04	[1.09, 1.09]	1	1.24	1	1.08	1.21	0
FIFO$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{1}$	1.57	0.14	[1.56, 1.58]	1.1	2.79	1.16	1.55	2.21	0
LIFO$_{5}$	1.37	0.1	[1.36, 1.38]	1.07	1.83	1.11	1.36	1.72	0
LIFO$_{10}$	1.22	0.08	[1.21, 1.23]	1.03	1.62	1.04	1.21	1.45	0
LIFO$_{20}$	1.06	0.04	[1.06, 1.06]	1	1.27	1	1.05	1.2	0
LIFO$_{40}$	1	0.01	[1, 1]	1	1.1	1	1	1.04	0
LIFO$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
LIFO$_{1,B}$	1.57	0.14	[1.56, 1.58]	1.1	2.79	1.16	1.55	2.21	0
LIFO$_{5,B}$	1.47	0.12	[1.46, 1.48]	1.08	2.13	1.13	1.45	1.96	0
LIFO$_{10,B}$	1.35	0.1	[1.34, 1.36]	1.07	1.83	1.11	1.34	1.72	0
LIFO$_{20,B}$	1.2	0.07	[1.19, 1.21]	1	1.5	1.04	1.19	1.42	0
LIFO$_{40,B}$	1.1	0.05	[1.1, 1.1]	1	1.29	1	1.1	1.24	0
LIFO$_{60,B}$	1.04	0.03	[1.04, 1.04]	1	1.2	1	1.04	1.14	0
LIFO$_{80,B}$	1.05	0.04	[1.05, 1.05]	1	1.21	1	1.04	1.16	0
LIFO$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_{1}$	1.41	0.09	[1.4, 1.42]	1.07	1.84	1.13	1.41	1.77	0
LFU$_{5}$	1.29	0.08	[1.28, 1.3]	1	1.64	1.08	1.28	1.55	0
LFU$_{10}$	1.18	0.06	[1.18, 1.18]	1	1.5	1.03	1.17	1.36	0
LFU$_{20}$	1.05	0.04	[1.05, 1.05]	1	1.27	1	1.04	1.17	0
LFU$_{40}$	1	0.01	[1, 1]	1	1.05	1	1	1.04	0
LFU$_{60}$	1	0	[1, 1]	1	1.04	1	1	1	0
LFU$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
LFU$_{1,B}$	1.41	0.09	[1.4, 1.42]	1.07	1.84	1.13	1.41	1.77	0
LFU$_{5,B}$	1.35	0.09	[1.34, 1.36]	1.07	1.8	1.12	1.35	1.65	0
LFU$_{10,B}$	1.28	0.08	[1.27, 1.29]	1.03	1.6	1.08	1.27	1.54	0
LFU$_{20,B}$	1.17	0.06	[1.17, 1.17]	1	1.44	1.02	1.17	1.36	0
LFU$_{40,B}$	1.09	0.05	[1.09, 1.09]	1	1.29	1	1.08	1.21	0
LFU$_{60,B}$	1.04	0.03	[1.04, 1.04]	1	1.18	1	1.04	1.14	0
LFU$_{80,B}$	1.04	0.03	[1.04, 1.04]	1	1.21	1	1.04	1.13	0
LFU$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_{1}$	1.56	0.07	[1.55, 1.57]	1.25	2	1.32	1.56	1.81	0
LRU$_{5}$	1.38	0.06	[1.37, 1.39]	1.11	1.61	1.2	1.38	1.55	0
LRU$_{10}$	1.23	0.05	[1.23, 1.23]	1.08	1.38	1.11	1.23	1.37	0
LRU$_{20}$	1.07	0.04	[1.07, 1.07]	1	1.21	1	1.07	1.18	0
LRU$_{40}$	1	0.01	[1, 1]	1	1.1	1	1	1.04	0
LRU$_{60}$	1	0	[1, 1]	1	1.04	1	1	1	0
LRU$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
LRU$_{1,B}$	1.56	0.07	[1.55, 1.57]	1.25	2	1.32	1.56	1.81	0
LRU$_{5,B}$	1.47	0.06	[1.46, 1.48]	1.17	1.75	1.26	1.47	1.69	0
LRU$_{10,B}$	1.37	0.06	[1.36, 1.38]	1.15	1.67	1.19	1.37	1.56	0
LRU$_{20,B}$	1.22	0.05	[1.22, 1.22]	1.04	1.42	1.08	1.22	1.38	0
LRU$_{40,B}$	1.12	0.05	[1.12, 1.12]	1	1.32	1.02	1.12	1.25	0
LRU$_{60,B}$	1.05	0.03	[1.05, 1.05]	1	1.18	1	1.04	1.14	0
LRU$_{80,B}$	1.06	0.03	[1.06, 1.06]	1	1.2	1	1.06	1.15	0
LRU$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.11: Performance ratios of costs relative to OPT in the paging problem when pages are stochastically distributed.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Fifo$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Fifo$_{5}$	0.88	0.07	[0.88, 0.88]	0.67	1.03	0.74	0.89	1	0
Fifo$_{10}$	0.78	0.07	[0.78, 0.78]	0.6	0.94	0.64	0.78	0.9	0
Fifo$_{20}$	0.65	0.08	[0.65, 0.65]	0.49	0.86	0.55	0.65	0.78	0
Fifo$_{40}$	0.6	0.07	[0.6, 0.6]	0.49	0.76	0.51	0.6	0.71	0
Fifo$_{60}$	0.6	0.07	[0.6, 0.6]	0.49	0.76	0.51	0.6	0.71	0
Fifo$_{80}$	0.6	0.07	[0.6, 0.6]	0.49	0.76	0.51	0.6	0.71	0
Fifo$_{100}$	0.6	0.07	[0.6, 0.6]	0.49	0.76	0.51	0.6	0.71	0
Fifo$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Fifo$_{5,B}$	0.95	0.05	[0.95, 0.95]	0.76	1.09	0.8	0.96	1.03	0.03
Fifo$_{10,B}$	0.88	0.07	[0.88, 0.88]	0.65	1	0.73	0.88	1	0
Fifo$_{20,B}$	0.77	0.08	[0.77, 0.77]	0.59	0.97	0.64	0.77	0.91	0
Fifo$_{40,B}$	0.69	0.08	[0.69, 0.69]	0.54	0.89	0.57	0.69	0.83	0
Fifo$_{60,B}$	0.64	0.08	[0.64, 0.64]	0.5	0.85	0.54	0.64	0.77	0
Fifo$_{80,B}$	0.65	0.08	[0.65, 0.65]	0.51	0.81	0.54	0.65	0.78	0
Fifo$_{100,B}$	0.6	0.07	[0.6, 0.6]	0.49	0.76	0.51	0.6	0.71	0
Lifo$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Lifo$_{5}$	0.88	0.09	[0.88, 0.88]	0.6	1.07	0.66	0.88	1.03	0
Lifo$_{10}$	0.79	0.11	[0.78, 0.8]	0.46	1.03	0.57	0.79	0.97	0
Lifo$_{20}$	0.69	0.13	[0.68, 0.7]	0.39	0.97	0.48	0.68	0.89	0
Lifo$_{40}$	0.65	0.14	[0.64, 0.66]	0.36	0.93	0.45	0.65	0.86	0
Lifo$_{60}$	0.65	0.14	[0.64, 0.66]	0.36	0.91	0.45	0.65	0.86	0
Lifo$_{80}$	0.65	0.14	[0.64, 0.66]	0.36	0.91	0.45	0.65	0.86	0
Lifo$_{100}$	0.65	0.14	[0.64, 0.66]	0.36	0.91	0.45	0.65	0.86	0
Lifo$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Lifo$_{5,B}$	0.94	0.07	[0.94, 0.94]	0.63	1.13	0.75	0.95	1.07	0.08
Lifo$_{10,B}$	0.87	0.1	[0.86, 0.88]	0.55	1.07	0.67	0.87	1.03	0.02
Lifo$_{20,B}$	0.77	0.12	[0.76, 0.78]	0.48	1	0.55	0.77	0.97	0
Lifo$_{40,B}$	0.71	0.13	[0.7, 0.72]	0.44	1	0.48	0.71	0.92	0
Lifo$_{60,B}$	0.67	0.14	[0.66, 0.68]	0.37	0.94	0.46	0.67	0.89	0
Lifo$_{80,B}$	0.68	0.14	[0.67, 0.69]	0.4	0.97	0.47	0.68	0.91	0
Lifo$_{100,B}$	0.65	0.14	[0.64, 0.66]	0.36	0.91	0.45	0.65	0.86	0
Lfu$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Lfu$_{5}$	0.91	0.05	[0.91, 0.91]	0.72	1.04	0.8	0.92	1	0
Lfu$_{10}$	0.84	0.07	[0.84, 0.84]	0.67	1	0.71	0.83	0.97	0
Lfu$_{20}$	0.75	0.09	[0.75, 0.75]	0.57	0.96	0.6	0.75	0.91	0
Lfu$_{40}$	0.72	0.09	[0.72, 0.72]	0.54	0.94	0.57	0.71	0.89	0
Lfu$_{60}$	0.72	0.09	[0.72, 0.72]	0.54	0.94	0.57	0.71	0.89	0
Lfu$_{80}$	0.72	0.09	[0.72, 0.72]	0.54	0.94	0.57	0.71	0.89	0
Lfu$_{100}$	0.72	0.09	[0.72, 0.72]	0.54	0.94	0.57	0.71	0.89	0
Lfu$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Lfu$_{5,B}$	0.96	0.04	[0.96, 0.96]	0.81	1.1	0.86	0.97	1	0
Lfu$_{10,B}$	0.91	0.06	[0.91, 0.91]	0.72	1	0.78	0.91	1	0
Lfu$_{20,B}$	0.84	0.07	[0.84, 0.84]	0.62	1	0.69	0.84	0.97	0
Lfu$_{40,B}$	0.78	0.09	[0.78, 0.78]	0.58	1	0.63	0.78	0.94	0
Lfu$_{60,B}$	0.74	0.09	[0.74, 0.74]	0.55	0.96	0.6	0.74	0.89	0
Lfu$_{80,B}$	0.75	0.09	[0.75, 0.75]	0.56	0.97	0.59	0.74	0.91	0
Lfu$_{100,B}$	0.72	0.09	[0.72, 0.72]	0.54	0.94	0.57	0.71	0.89	0
Lru$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Lru$_{5}$	0.89	0.05	[0.89, 0.89]	0.74	1	0.77	0.89	0.98	0
Lru$_{10}$	0.79	0.07	[0.79, 0.79]	0.64	0.94	0.67	0.79	0.93	0
Lru$_{20}$	0.69	0.08	[0.69, 0.69]	0.54	0.87	0.58	0.68	0.83	0
Lru$_{40}$	0.65	0.07	[0.65, 0.65]	0.5	0.8	0.55	0.64	0.76	0
Lru$_{60}$	0.64	0.07	[0.64, 0.64]	0.5	0.8	0.55	0.64	0.76	0
Lru$_{80}$	0.64	0.07	[0.64, 0.64]	0.5	0.8	0.55	0.64	0.76	0
Lru$_{100}$	0.64	0.07	[0.64, 0.64]	0.5	0.8	0.55	0.64	0.76	0
Lru$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Lru$_{5,B}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.95	1	0
Lru$_{10,B}$	0.88	0.05	[0.88, 0.88]	0.7	1	0.77	0.88	0.97	0
Lru$_{20,B}$	0.79	0.07	[0.79, 0.79]	0.63	0.97	0.65	0.79	0.92	0
Lru$_{40,B}$	0.72	0.07	[0.72, 0.72]	0.58	0.9	0.6	0.72	0.85	0
Lru$_{60,B}$	0.68	0.07	[0.68, 0.68]	0.54	0.84	0.57	0.68	0.8	0
Lru$_{80,B}$	0.68	0.07	[0.68, 0.68]	0.52	0.87	0.58	0.68	0.8	0
Lru$_{100,B}$	0.64	0.07	[0.64, 0.64]	0.5	0.8	0.55	0.64	0.76	0

Table A.12: Performance ratios of costs relative to the online version of an algorithm in the paging problem when pages are stochastically distributed.

A.2.3 Online Bin Packing with Lookahead

A.2.3.1 Classical Problem

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
BF_{1}	14.66	0.13	[14.54, 14.78]	9	21	11	15	19	
BF_{5}	14.33	0.13	[14.21, 14.45]	9	21	10	14	19	0
BF_{10}	14.14	0.14	[14.02, 14.26]	9	21	10	14	19	0
BF_{15}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
BF_{20}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
BF_{25}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\text{BF}_{1,B}$	14.66	0.13	[14.54, 14.78]	9	21	11	15	19	
$\text{BF}_{5,B}$	14.48	0.13	[14.36, 14.6]	9	21	11	14	19	0.01
$\text{BF}_{10,B}$	14.35	0.13	[14.23, 14.47]	9	21	10.5	14	19	0
$\text{BF}_{15,B}$	14.26	0.13	[14.15, 14.37]	9	21	10	14	19	0.04
$\text{BF}_{20,B}$	14.23	0.14	[14.11, 14.35]	9	21	10	14	19	0.05
$\text{BF}_{25,B}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
FF_{1}	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
FF_{5}	14.39	0.13	[14.27, 14.51]	9	21	10.5	14	19	0
FF_{10}	14.15	0.14	[14.03, 14.27]	9	21	10	14	19	0
FF_{15}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
FF_{20}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
FF_{25}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\text{FF}_{1,B}$	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
$\text{FF}_{5,B}$	14.54	0.13	[14.42, 14.66]	9	21	11	15	19	0.01
$\text{FF}_{10,B}$	14.38	0.13	[14.26, 14.5]	9	21	11	14	19	0
$\text{FF}_{15,B}$	14.27	0.13	[14.15, 14.39]	9	21	10	14	19	0.04
$\text{FF}_{20,B}$	14.24	0.13	[14.13, 14.35]	9	21	10	14	19	0.05
$\text{FF}_{25,B}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
OPT_{1}	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
OPT_{5}	14.63	0.12	[14.52, 14.74]	10	21	11	15	19	0.11
OPT_{10}	14.21	0.13	[14.1, 14.32]	9	21	10	14	19	0
OPT_{15}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
OPT_{20}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
OPT_{25}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\text{OPT}_{1,B}$	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
$\text{OPT}_{5,B}$	14.73	0.13	[14.61, 14.85]	9	21	11	15	19.5	0.09
$\text{OPT}_{10,B}$	14.63	0.13	[14.51, 14.75]	10	21	11	15	19	0.07
$\text{OPT}_{15,B}$	14.46	0.13	[14.34, 14.58]	9	21	10	14	19	0.06
$\text{OPT}_{20,B}$	14.42	0.13	[14.3, 14.54]	10	21	11	14	19	0.11
$\text{OPT}_{25,B}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
OPT'_{1}	14.84	0.13	[14.72, 14.96]	10	21	11	15	19	
OPT'_{5}	14.54	0.13	[14.42, 14.66]	9	21	11	15	19	0.04
OPT'_{10}	14.19	0.13	[14.08, 14.3]	9	21	10	14	19	0
OPT'_{15}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
OPT'_{20}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
OPT'_{25}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\text{OPT}'_{1,B}$	14.84	0.13	[14.72, 14.96]	10	21	11	15	19	
$\text{OPT}'_{5,B}$	14.64	0.13	[14.52, 14.76]	10	21	11	15	19	0.04
$\text{OPT}'_{10,B}$	14.42	0.13	[14.3, 14.54]	9	21	10	14	19	0.02
$\text{OPT}'_{15,B}$	14.29	0.13	[14.17, 14.41]	9	21	10	14	19	0.04
$\text{OPT}'_{20,B}$	14.27	0.13	[14.15, 14.39]	9	21	10	14	19	0.07
$\text{OPT}'_{25,B}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0

Table A.13: Costs in the classical bin packing problem when item permutations are allowed.

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
BF_{1}	14.66	0.13	[14.54, 14.78]	9	21	11	15	19	
BF_{5}	14.34	0.13	[14.22, 14.46]	9	21	10.5	14	19	0
BF_{10}	14.2	0.14	[14.08, 14.32]	9	21	10	14	19	0.01
BF_{15}	14.16	0.14	[14.04, 14.28]	9	21	10	14	19	0.01
BF_{20}	14.15	0.14	[14.03, 14.27]	9	21	10	14	19	0
BF_{25}	14.15	0.14	[14.03, 14.27]	9	21	10	14	19	0
$\text{BF}_{1,B}$	14.66	0.13	[14.54, 14.78]	9	21	11	15	19	
$\text{BF}_{5,B}$	14.48	0.13	[14.36, 14.6]	9	21	11	14	19	0.01
$\text{BF}_{10,B}$	14.37	0.13	[14.25, 14.49]	9	21	11	14	19	0.01
$\text{BF}_{15,B}$	14.28	0.13	[14.16, 14.4]	9	21	10	14	19	0.04
$\text{BF}_{20,B}$	14.25	0.13	[14.14, 14.36]	9	21	10	14	19	0.06
$\text{BF}_{25,B}$	14.15	0.14	[14.03, 14.27]	9	21	10	14	19	0.01
FF_{1}	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
FF_{5}	14.42	0.13	[14.3, 14.54]	9	21	10	14	19	0.01
FF_{10}	14.24	0.14	[14.12, 14.36]	9	21	10	14	19	0.01
FF_{15}	14.17	0.14	[14.05, 14.29]	9	21	10	14	19	0.01
FF_{20}	14.14	0.14	[14.02, 14.26]	9	21	10	14	19	0.01
FF_{25}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\text{FF}_{1,B}$	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
$\text{FF}_{5,B}$	14.54	0.13	[14.42, 14.66]	9	21	11	15	19	0.01
$\text{FF}_{10,B}$	14.38	0.13	[14.26, 14.5]	9	21	11	14	19	0
$\text{FF}_{15,B}$	14.27	0.13	[14.15, 14.39]	9	21	10	14	19	0.04
$\text{FF}_{20,B}$	14.24	0.13	[14.13, 14.35]	9	21	10	14	19	0.05
$\text{FF}_{25,B}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
OPT_{1}	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
OPT_{5}	14.63	0.13	[14.51, 14.75]	10	21	11	15	19	0.07
OPT_{10}	14.36	0.13	[14.24, 14.48]	9	21	10	14	19	0.02
OPT_{15}	14.2	0.14	[14.08, 14.32]	9	21	10	14	19	0.02
OPT_{20}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0.01
OPT_{25}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\text{OPT}_{1,B}$	14.76	0.13	[14.64, 14.88]	9	21	11	15	19	
$\text{OPT}_{5,B}$	14.73	0.13	[14.61, 14.85]	9	21	11	15	19.5	0.09
$\text{OPT}_{10,B}$	14.63	0.13	[14.51, 14.75]	10	21	11	15	19	0.07
$\text{OPT}_{15,B}$	14.46	0.13	[14.34, 14.58]	9	21	10	14	19	0.06
$\text{OPT}_{20,B}$	14.42	0.13	[14.3, 14.54]	10	21	11	14	19	0.11
$\text{OPT}_{25,B}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
OPT'_{1}	14.84	0.13	[14.72, 14.96]	10	21	11	15	19	
OPT'_{5}	14.52	0.13	[14.4, 14.64]	10	21	11	15	19	0.02
OPT'_{10}	14.29	0.13	[14.17, 14.41]	9	21	10.5	14	19	0.02
OPT'_{15}	14.18	0.14	[14.06, 14.3]	9	21	10	14	19	0.02
OPT'_{20}	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0.01
OPT'_{25}	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\text{OPT}'_{1,B}$	14.84	0.13	[14.72, 14.96]	10	21	11	15	19	
$\text{OPT}'_{5,B}$	14.64	0.13	[14.52, 14.76]	10	21	11	15	19	0.04
$\text{OPT}'_{10,B}$	14.42	0.13	[14.3, 14.54]	9	21	10	14	19	0.02
$\text{OPT}'_{15,B}$	14.29	0.13	[14.17, 14.41]	9	21	10	14	19	0.04
$\text{OPT}'_{20,B}$	14.27	0.13	[14.15, 14.39]	9	21	10	14	19	0.07
$\text{OPT}'_{25,B}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0

Table A.14: Costs in the classical bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to $\textsc{Opt}$ for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
$\textsc{Bf}_{1}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
$\textsc{Bf}_{5}$	1.02	0.03	[1.02, 1.02]	0.94	1.17	1	1	1.1	0
$\textsc{Bf}_{10}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Bf}_{15}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Bf}_{20}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Bf}_{25}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Bf}_{1,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
$\textsc{Bf}_{5,B}$	1.03	0.04	[1.03, 1.03]	1	1.2	1	1	1.12	0
$\textsc{Bf}_{10,B}$	1.02	0.03	[1.02, 1.02]	1	1.2	1	1	1.1	0
$\textsc{Bf}_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Bf}_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
$\textsc{Bf}_{25,B}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Ff}_{1}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Ff}_{5}$	1.02	0.04	[1.02, 1.02]	0.94	1.2	1	1	1.1	0
$\textsc{Ff}_{10}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Ff}_{15}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Ff}_{20}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Ff}_{25}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Ff}_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Ff}_{5,B}$	1.03	0.04	[1.03, 1.03]	1	1.2	1	1	1.14	0
$\textsc{Ff}_{10,B}$	1.02	0.03	[1.02, 1.02]	1	1.2	1	1	1.1	0
$\textsc{Ff}_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Ff}_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
$\textsc{Ff}_{25,B}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Opt}_{1}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Opt}_{5}$	1.04	0.04	[1.04, 1.04]	0.94	1.2	1	1	1.18	0
$\textsc{Opt}_{10}$	1.01	0.02	[1.01, 1.01]	0.94	1.11	1	1	1.09	0
$\textsc{Opt}_{15}$	1	0	[1, 1]	0.94	1.08	1	1	1	0
$\textsc{Opt}_{20}$	1	0	[1, 1]	0.94	1.08	1	1	1	0
$\textsc{Opt}_{25}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Opt}_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Opt}_{5,B}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.06	1.17	0
$\textsc{Opt}_{10,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1	1.15	0
$\textsc{Opt}_{15,B}$	1.03	0.04	[1.03, 1.03]	1	1.18	1	1	1.14	0
$\textsc{Opt}_{20,B}$	1.02	0.04	[1.02, 1.02]	1	1.17	1	1	1.1	0
$\textsc{Opt}_{25,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Opt}'_{1}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
$\textsc{Opt}'_{5}$	1.03	0.04	[1.03, 1.03]	0.94	1.18	1	1	1.15	0
$\textsc{Opt}'_{10}$	1.01	0.02	[1.01, 1.01]	0.94	1.11	1	1	1.09	0
$\textsc{Opt}'_{15}$	1	0.01	[1, 1]	0.94	1.11	1	1	1	0
$\textsc{Opt}'_{20}$	1	0	[1, 1]	0.94	1.08	1	1	1	0
$\textsc{Opt}'_{25}$	1	0	[1, 1]	0.94	1.08	1	1	1	0
$\textsc{Opt}'_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
$\textsc{Opt}'_{5,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1	1.15	0
$\textsc{Opt}'_{10,B}$	1.02	0.04	[1.02, 1.02]	0.94	1.2	1	1	1.1	0
$\textsc{Opt}'_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Opt}'_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
$\textsc{Opt}'_{25,B}$	1	0	[1, 1]	0.94	1.08	1	1	1	0

Table A.15: Performance ratios of costs relative to $\textsc{Opt}$ in the classical bin packing problem when item permutations are allowed.

Performance ratios of costs relative to $\textsc{Opt}$ for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
$\textsc{Bf}_{1}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
$\textsc{Bf}_{5}$	1.02	0.03	[1.02, 1.02]	0.94	1.15	1	1	1.1	0
$\textsc{Bf}_{10}$	1.01	0.02	[1.01, 1.01]	0.94	1.11	1	1	1.09	0
$\textsc{Bf}_{15}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Bf}_{20}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Bf}_{25}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Bf}_{1,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
$\textsc{Bf}_{5,B}$	1.03	0.04	[1.03, 1.03]	1	1.2	1	1	1.12	0
$\textsc{Bf}_{10,B}$	1.02	0.03	[1.02, 1.02]	1	1.2	1	1	1.1	0
$\textsc{Bf}_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.14	1	1	1.09	0
$\textsc{Bf}_{20,B}$	1.01	0.03	[1.01, 1.01]	1	1.17	1	1	1.09	0
$\textsc{Bf}_{25,B}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Ff}_{1}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Ff}_{5}$	1.02	0.04	[1.02, 1.02]	0.94	1.15	1	1	1.1	0
$\textsc{Ff}_{10}$	1.01	0.03	[1.01, 1.01]	0.94	1.1	1	1	1.09	0
$\textsc{Ff}_{15}$	1	0.02	[1, 1]	0.94	1.1	1	1	1.09	0
$\textsc{Ff}_{20}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.09	0
$\textsc{Ff}_{25}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Ff}_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Ff}_{5,B}$	1.03	0.04	[1.03, 1.03]	1	1.2	1	1	1.14	0
$\textsc{Ff}_{10,B}$	1.02	0.03	[1.02, 1.02]	1	1.2	1	1	1.1	0
$\textsc{Ff}_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Ff}_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
$\textsc{Ff}_{25,B}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
$\textsc{Opt}_{1}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Opt}_{5}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1	1.16	0
$\textsc{Opt}_{10}$	1.02	0.03	[1.02, 1.02]	0.94	1.15	1	1	1.09	0
$\textsc{Opt}_{15}$	1.01	0.02	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Opt}_{20}$	1	0.01	[1, 1]	0.94	1.1	1	1	1.07	0
$\textsc{Opt}_{25}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Opt}_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.23	1	1.07	1.17	0
$\textsc{Opt}_{5,B}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.06	1.17	0
$\textsc{Opt}_{10,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1	1.15	0
$\textsc{Opt}_{15,B}$	1.03	0.04	[1.03, 1.03]	1	1.18	1	1	1.14	0
$\textsc{Opt}_{20,B}$	1.02	0.04	[1.02, 1.02]	1	1.17	1	1	1.1	0
$\textsc{Opt}_{25,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Opt}'_{1}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
$\textsc{Opt}'_{5}$	1.03	0.04	[1.03, 1.03]	0.94	1.18	1	1	1.11	0
$\textsc{Opt}'_{10}$	1.01	0.03	[1.01, 1.01]	0.94	1.18	1	1	1.1	0
$\textsc{Opt}'_{15}$	1.01	0.02	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Opt}'_{20}$	1	0.01	[1, 1]	0.94	1.09	1	1	1.07	0
$\textsc{Opt}'_{25}$	1	0	[1, 1]	0.94	1.08	1	1	1	0
$\textsc{Opt}'_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
$\textsc{Opt}'_{5,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1	1.15	0
$\textsc{Opt}'_{10,B}$	1.02	0.04	[1.02, 1.02]	0.94	1.2	1	1	1.1	0
$\textsc{Opt}'_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
$\textsc{Opt}'_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
$\textsc{Opt}'_{25,B}$	1	0	[1, 1]	0.94	1.08	1	1	1	0

Table A.16: Performance ratios of costs relative to $\textsc{Opt}$ in the classical bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Bf$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Bf$_{5}$	0.98	0.03	[0.98, 0.98]	0.88	1.1	0.92	1	1	0
Bf$_{10}$	0.97	0.04	[0.97, 0.97]	0.83	1.1	0.87	1	1	0
Bf$_{15}$	0.96	0.04	[0.96, 0.96]	0.83	1	0.87	1	1	0
Bf$_{20}$	0.96	0.04	[0.96, 0.96]	0.83	1	0.87	1	1	0
Bf$_{25}$	0.96	0.04	[0.96, 0.96]	0.83	1	0.87	1	1	0
Bf$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Bf$_{5,B}$	0.99	0.03	[0.99, 0.99]	0.91	1.1	0.92	1	1	0.01
Bf$_{10,B}$	0.98	0.03	[0.98, 0.98]	0.87	1.1	0.92	1	1	0
Bf$_{15,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.1	0.9	1	1	0
Bf$_{20,B}$	0.97	0.04	[0.97, 0.97]	0.83	1.09	0.88	1	1	0
Bf$_{25,B}$	0.96	0.04	[0.96, 0.96]	0.83	1	0.87	1	1	0
Ff$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Ff$_{5}$	0.98	0.03	[0.98, 0.98]	0.88	1.09	0.92	1	1	0
Ff$_{10}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.87	0.94	1	0
Ff$_{15}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Ff$_{20}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Ff$_{25}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Ff$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Ff$_{5,B}$	0.99	0.03	[0.99, 0.99]	0.85	1.08	0.92	1	1	0.01
Ff$_{10,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.91	1	1	0
Ff$_{15,B}$	0.97	0.04	[0.97, 0.97]	0.83	1.08	0.88	1	1	0
Ff$_{20,B}$	0.96	0.04	[0.96, 0.96]	0.83	1.08	0.87	1	1	0
Ff$_{25,B}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Opt$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{5}$	0.99	0.04	[0.99, 0.99]	0.88	1.11	0.92	1	1.09	0.11
Opt$_{10}$	0.96	0.04	[0.96, 0.96]	0.81	1.1	0.87	0.94	1	0
Opt$_{15}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Opt$_{20}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Opt$_{25}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Opt$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{5,B}$	1	0.03	[1, 1]	0.89	1.1	0.92	1	1.08	0.09
Opt$_{10,B}$	0.99	0.04	[0.99, 0.99]	0.88	1.11	0.92	1	1.08	0.07
Opt$_{15,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.09	0.91	1	1.07	0.02
Opt$_{20,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.11	0.89	1	1.08	0.03
Opt$_{25,B}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
Opt$'_{1}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$'_{5}$	0.98	0.04	[0.98, 0.98]	0.87	1.1	0.91	1	1.08	0.04
Opt$'_{10}$	0.96	0.04	[0.96, 0.96]	0.85	1	0.86	0.94	1	0
Opt$'_{15}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
Opt$'_{20}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
Opt$'_{25}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
Opt$'_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$'_{5,B}$	0.99	0.03	[0.99, 0.99]	0.88	1.09	0.92	1	1.08	0.04
Opt$'_{10,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.89	1	1	0.01
Opt$'_{15,B}$	0.96	0.04	[0.96, 0.96]	0.85	1.07	0.87	0.94	1	0
Opt$'_{20,B}$	0.96	0.04	[0.96, 0.96]	0.85	1.07	0.88	0.94	1	0
Opt$'_{25,B}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0

Table A.17: Performance ratios of costs relative to the online version of an algorithm in the classical bin packing problem when item permutations are allowed.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
BF_1	1	0	[1, 1]	1	1	1	1	1	0
BF_5	0.98	0.03	[0.98, 0.98]	0.85	1.1	0.92	1	1	0
BF_{10}	0.97	0.04	[0.97, 0.97]	0.83	1.1	0.87	1	1	0.01
BF_{15}	0.97	0.04	[0.97, 0.97]	0.85	1.09	0.87	1	1	0
BF_{20}	0.97	0.04	[0.97, 0.97]	0.83	1.09	0.87	1	1	0
BF_{25}	0.97	0.04	[0.97, 0.97]	0.83	1.09	0.87	1	1	0
$\text{BF}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{BF}_{5,B}$	0.99	0.03	[0.99, 0.99]	0.91	1.1	0.92	1	1	0.01
$\text{BF}_{10,B}$	0.98	0.03	[0.98, 0.98]	0.87	1.1	0.92	1	1	0
$\text{BF}_{15,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.1	0.89	1	1	0.01
$\text{BF}_{20,B}$	0.97	0.04	[0.97, 0.97]	0.83	1.09	0.9	1	1	0
$\text{BF}_{25,B}$	0.97	0.04	[0.97, 0.97]	0.83	1.09	0.87	1	1	0
FF_1	1	0	[1, 1]	1	1	1	1	1	0
FF_5	0.98	0.04	[0.98, 0.98]	0.85	1.09	0.91	1	1	0.01
FF_{10}	0.97	0.04	[0.97, 0.97]	0.83	1.08	0.87	1	1	0
FF_{15}	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
FF_{20}	0.96	0.04	[0.96, 0.96]	0.81	1.08	0.86	0.94	1	0
FF_{25}	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
$\text{FF}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{FF}_{5,B}$	0.99	0.03	[0.99, 0.99]	0.85	1.08	0.92	1	1	0.01
$\text{FF}_{10,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.91	1	1	0
$\text{FF}_{15,B}$	0.97	0.04	[0.97, 0.97]	0.83	1.08	0.88	1	1	0
$\text{FF}_{20,B}$	0.96	0.04	[0.96, 0.96]	0.83	1.08	0.87	1	1	0
$\text{FF}_{25,B}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
OPT_1	1	0	[1, 1]	1	1	1	1	1	0
OPT_5	0.99	0.04	[0.99, 0.99]	0.87	1.11	0.92	1	1.08	0.07
OPT_{10}	0.97	0.04	[0.97, 0.97]	0.83	1.08	0.88	1	1.03	0.01
OPT_{15}	0.96	0.04	[0.96, 0.96]	0.83	1.07	0.87	0.94	1	0
OPT_{20}	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
OPT_{25}	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
$\text{OPT}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{OPT}_{5,B}$	1	0.03	[1, 1]	0.89	1.1	0.92	1	1.08	0.09
$\text{OPT}_{10,B}$	0.99	0.04	[0.99, 0.99]	0.88	1.11	0.92	1	1.08	0.07
$\text{OPT}_{15,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.09	0.91	1	1.07	0.02
$\text{OPT}_{20,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.11	0.89	1	1.08	0.03
$\text{OPT}_{25,B}$	0.96	0.04	[0.96, 0.96]	0.81	1	0.86	0.94	1	0
OPT'_1	1	0	[1, 1]	1	1	1	1	1	0
OPT'_5	0.98	0.04	[0.98, 0.98]	0.85	1.09	0.91	1	1.07	0.02
OPT'_{10}	0.96	0.04	[0.96, 0.96]	0.85	1.08	0.87	0.94	1	0
OPT'_{15}	0.96	0.04	[0.96, 0.96]	0.83	1.08	0.86	0.94	1	0
OPT'_{20}	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
OPT'_{25}	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
$\text{OPT}'_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{OPT}'_{5,B}$	0.99	0.03	[0.99, 0.99]	0.88	1.09	0.92	1	1.08	0.04
$\text{OPT}'_{10,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.89	1	1	0.01
$\text{OPT}'_{15,B}$	0.96	0.04	[0.96, 0.96]	0.85	1.07	0.87	0.94	1	0
$\text{OPT}'_{20,B}$	0.96	0.04	[0.96, 0.96]	0.85	1.07	0.88	0.94	1	0
$\text{OPT}'_{25,B}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0

Table A.18: Performance ratios of costs relative to the online version of an algorithm in the classical bin packing problem when item permutations are forbidden.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Bf}_{1}$	54.94	0.07	[54.7, 55.18]	44	68	47	55	64.5	
$\textsc{Bf}_{5}$	54.73	0.07	[54.49, 54.97]	44	67	47	55	64.5	0.02
$\textsc{Bf}_{10}$	54.37	0.07	[54.13, 54.61]	44	67	46	54	64	0.01
$\textsc{Bf}_{20}$	53.65	0.07	[53.42, 53.88]	43	66	46	54	63	0
$\textsc{Bf}_{40}$	52.88	0.07	[52.65, 53.11]	43	66	45	53	63	0
$\textsc{Bf}_{60}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Bf}_{80}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Bf}_{100}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Bf}_{1,B}$	54.94	0.07	[54.7, 55.18]	44	68	47	55	64.5	
$\textsc{Bf}_{5,B}$	54.68	0.07	[54.44, 54.92]	44	67	46	55	64.5	0.02
$\textsc{Bf}_{10,B}$	54.42	0.07	[54.18, 54.66]	44	67	46	54	64	0.02
$\textsc{Bf}_{20,B}$	54.03	0.07	[53.8, 54.26]	44	67	46	54	64	0
$\textsc{Bf}_{40,B}$	53.62	0.07	[53.39, 53.85]	43	67	46	54	63	0
$\textsc{Bf}_{60,B}$	53.38	0.07	[53.15, 53.61]	43	66	46	53	63	0.09
$\textsc{Bf}_{80,B}$	53.22	0.07	[52.99, 53.45]	43	66	45	53	63	0.1
$\textsc{Bf}_{100,B}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ff}_{1}$	55.61	0.07	[55.37, 55.85]	45	68	48	56	65	
$\textsc{Ff}_{5}$	55.34	0.07	[55.1, 55.58]	45	67	47	55	65	0.01
$\textsc{Ff}_{10}$	54.89	0.07	[54.65, 55.13]	45	67	47	55	64	0
$\textsc{Ff}_{20}$	53.89	0.07	[53.66, 54.12]	44	66	46	54	63	0
$\textsc{Ff}_{40}$	52.9	0.07	[52.67, 53.13]	43	66	45	53	63	0
$\textsc{Ff}_{60}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ff}_{80}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ff}_{100}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ff}_{1,B}$	55.61	0.07	[55.37, 55.85]	45	68	48	56	65	
$\textsc{Ff}_{5,B}$	55.14	0.07	[54.9, 55.38]	44	68	47	55	65	0.01
$\textsc{Ff}_{10,B}$	54.69	0.07	[54.45, 54.93]	44	68	47	55	64	0.01
$\textsc{Ff}_{20,B}$	54.14	0.07	[53.9, 54.38]	44	67	46	54	64	0
$\textsc{Ff}_{40,B}$	53.64	0.07	[53.41, 53.87]	43	67	46	54	63	0
$\textsc{Ff}_{60,B}$	53.39	0.07	[53.16, 53.62]	43	66	46	53	63	0.08
$\textsc{Ff}_{80,B}$	53.23	0.07	[53, 53.46]	43	66	45.5	53	63	0.1
$\textsc{Ff}_{100,B}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0

Table A.19: Costs in the classical bin packing problem when item permutations are allowed.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Bf}_{1}$	54.94	0.07	[54.7, 55.18]	44	68	47	55	64.5	
$\textsc{Bf}_{5}$	54.4	0.07	[54.16, 54.64]	44	67	46	54	64	0.01
$\textsc{Bf}_{10}$	53.93	0.07	[53.7, 54.16]	44	67	46	54	64	0.02
$\textsc{Bf}_{20}$	53.41	0.07	[53.18, 53.64]	43	66	46	53	63	0
$\textsc{Bf}_{40}$	53.05	0.07	[52.82, 53.28]	43	66	45	53	63	0
$\textsc{Bf}_{60}$	52.96	0.07	[52.73, 53.19]	43	66	45	53	63	0.01
$\textsc{Bf}_{80}$	52.93	0.07	[52.7, 53.16]	43	66	45	53	63	0.01
$\textsc{Bf}_{100}$	52.93	0.07	[52.7, 53.16]	43	66	45	53	63	0
$\textsc{Bf}_{1,B}$	54.94	0.07	[54.7, 55.18]	44	68	47	55	64.5	
$\textsc{Bf}_{5,B}$	54.67	0.07	[54.43, 54.91]	44	67	46	55	64.5	0.02
$\textsc{Bf}_{10,B}$	54.43	0.07	[54.19, 54.67]	44	67	46	54	64	0.02
$\textsc{Bf}_{20,B}$	54.09	0.07	[53.86, 54.32]	43	67	46	54	64	0.01
$\textsc{Bf}_{40,B}$	53.7	0.07	[53.47, 53.93]	43	67	46	54	63	0.01
$\textsc{Bf}_{60,B}$	53.46	0.07	[53.23, 53.69]	43	66	46	53	63	0.09
$\textsc{Bf}_{80,B}$	53.3	0.07	[53.07, 53.53]	43	66	46	53	63	0.1
$\textsc{Bf}_{100,B}$	52.93	0.07	[52.7, 53.16]	43	66	45	53	63	0.01
$\textsc{Ff}_{1}$	55.61	0.07	[55.37, 55.85]	45	68	48	56	65	
$\textsc{Ff}_{5}$	54.85	0.07	[54.61, 55.09]	44	68	47	55	64	0
$\textsc{Ff}_{10}$	54.23	0.07	[53.99, 54.47]	44	67	46	54	64	0.01
$\textsc{Ff}_{20}$	53.52	0.07	[53.29, 53.75]	43	67	46	53	63	0
$\textsc{Ff}_{40}$	53.07	0.07	[52.84, 53.3]	43	66	45.5	53	63	0.01
$\textsc{Ff}_{60}$	52.96	0.07	[52.73, 53.19]	43	66	45	53	63	0.02
$\textsc{Ff}_{80}$	52.9	0.07	[52.67, 53.13]	43	66	45	53	63	0.02
$\textsc{Ff}_{100}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ff}_{1,B}$	55.61	0.07	[55.37, 55.85]	45	68	48	56	65	
$\textsc{Ff}_{5,B}$	55.14	0.07	[54.9, 55.38]	44	68	47	55	65	0.01
$\textsc{Ff}_{10,B}$	54.69	0.07	[54.45, 54.93]	44	68	47	55	64	0.01
$\textsc{Ff}_{20,B}$	54.14	0.07	[53.9, 54.38]	44	67	46	54	64	0
$\textsc{Ff}_{40,B}$	53.64	0.07	[53.41, 53.87]	43	67	46	54	63	0
$\textsc{Ff}_{60,B}$	53.39	0.07	[53.16, 53.62]	43	66	46	53	63	0.08
$\textsc{Ff}_{80,B}$	53.23	0.07	[53, 53.46]	43	66	45.5	53	63	0.1
$\textsc{Ff}_{100,B}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0

Table A.20: Costs in the classical bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to BF$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
BF$_{1}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
BF$_{5}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
BF$_{10}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.02	1.09	0
BF$_{20}$	1.02	0.02	[1.02, 1.02]	1	1.08	1	1.02	1.06	0
BF$_{40}$	1	0	[1, 1]	1	1.02	1	1	1.02	0
BF$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{1,B}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
BF$_{5,B}$	1.04	0.02	[1.04, 1.04]	1	1.1	1	1.04	1.09	0
BF$_{10,B}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.02	1.08	0
BF$_{20,B}$	1.02	0.02	[1.02, 1.02]	1	1.1	1	1.02	1.06	0
BF$_{40,B}$	1.02	0.02	[1.02, 1.02]	1	1.07	1	1.02	1.06	0
BF$_{60,B}$	1.01	0.01	[1.01, 1.01]	1	1.08	1	1	1.05	0
BF$_{80,B}$	1.01	0.01	[1.01, 1.01]	1	1.06	1	1	1.04	0
BF$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
FF$_{1}$	1.05	0.02	[1.05, 1.05]	1	1.14	1.02	1.05	1.11	0
FF$_{5}$	1.05	0.02	[1.05, 1.05]	1	1.12	1	1.04	1.1	0
FF$_{10}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
FF$_{20}$	1.02	0.02	[1.02, 1.02]	1	1.08	1	1.02	1.07	0
FF$_{40}$	1	0	[1, 1]	1	1.02	1	1	1.02	0
FF$_{60}$	1	0	[1, 1]	0.98	1.02	1	1	1	0
FF$_{80}$	1	0	[1, 1]	1	1.02	1	1	1	0
FF$_{100}$	1	0	[1, 1]	1	1.02	1	1	1	0
FF$_{1,B}$	1.05	0.02	[1.05, 1.05]	1	1.14	1.02	1.05	1.11	0
FF$_{5,B}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
FF$_{10,B}$	1.04	0.02	[1.04, 1.04]	1	1.1	1	1.04	1.08	0
FF$_{20,B}$	1.02	0.02	[1.02, 1.02]	1	1.1	1	1.02	1.07	0
FF$_{40,B}$	1.02	0.02	[1.02, 1.02]	1	1.07	1	1.02	1.06	0
FF$_{60,B}$	1.01	0.01	[1.01, 1.01]	1	1.08	1	1	1.05	0
FF$_{80,B}$	1.01	0.01	[1.01, 1.01]	1	1.06	1	1	1.04	0
FF$_{100,B}$	1	0	[1, 1]	1	1.02	1	1	1	0

Table A.21: Performance ratios of costs relative to BF$_{100}$ in the classical bin packing problem when item permutations are allowed.

Performance ratios of costs relative to BF$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
BF$_{1}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
BF$_{5}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.02	1.08	0
BF$_{10}$	1.02	0.02	[1.02, 1.02]	0.98	1.08	1	1.02	1.06	0
BF$_{20}$	1.01	0.01	[1.01, 1.01]	0.98	1.08	1	1	1.04	0
BF$_{40}$	1	0.01	[1, 1]	0.98	1.04	1	1	1.02	0.01
BF$_{60}$	1	0	[1, 1]	0.98	1.02	1	1	1.02	0.01
BF$_{80}$	1	0	[1, 1]	0.98	1.02	1	1	1	0
BF$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{1,B}$	1.04	0.02	[1.04, 1.04]	1	1.12	1	1.04	1.1	0
BF$_{5,B}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.04	1.08	0
BF$_{10,B}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.02	1.08	0
BF$_{20,B}$	1.02	0.02	[1.02, 1.02]	0.98	1.1	1	1.02	1.07	0
BF$_{40,B}$	1.01	0.02	[1.01, 1.01]	0.98	1.07	1	1.02	1.06	0
BF$_{60,B}$	1.01	0.01	[1.01, 1.01]	0.98	1.08	1	1	1.06	0.01
BF$_{80,B}$	1.01	0.01	[1.01, 1.01]	0.98	1.06	1	1	1.04	0.01
BF$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
FF$_{1}$	1.05	0.02	[1.05, 1.05]	1	1.14	1.02	1.05	1.1	0
FF$_{5}$	1.04	0.02	[1.04, 1.04]	1	1.1	1	1.04	1.08	0
FF$_{10}$	1.02	0.02	[1.02, 1.02]	1	1.1	1	1.02	1.06	0
FF$_{20}$	1.01	0.01	[1.01, 1.01]	0.98	1.06	1	1.02	1.04	0
FF$_{40}$	1	0.01	[1, 1]	0.98	1.04	0.98	1	1.02	0.02
FF$_{60}$	1	0.01	[1, 1]	0.98	1.04	0.98	1	1.02	0.03
FF$_{80}$	1	0.01	[1, 1]	0.98	1.02	0.98	1	1.02	0.06
FF$_{100}$	1	0.01	[1, 1]	0.98	1.02	0.98	1	1	0.09
FF$_{1,B}$	1.05	0.02	[1.05, 1.05]	1	1.14	1.02	1.05	1.1	0
FF$_{5,B}$	1.04	0.02	[1.04, 1.04]	1	1.1	1	1.04	1.1	0
FF$_{10,B}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.04	1.08	0
FF$_{20,B}$	1.02	0.02	[1.02, 1.02]	0.98	1.1	1	1.02	1.07	0
FF$_{40,B}$	1.01	0.01	[1.01, 1.01]	0.98	1.07	1	1.02	1.06	0
FF$_{60,B}$	1.01	0.01	[1.01, 1.01]	0.98	1.08	0.98	1	1.05	0.02
FF$_{80,B}$	1.01	0.01	[1.01, 1.01]	0.98	1.06	0.98	1	1.04	0.02
FF$_{100,B}$	1	0.01	[1, 1]	0.98	1.02	0.98	1	1	0.09

Table A.22: Performance ratios of costs relative to BF$_{100}$ in the classical bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
BF$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{5}$	1	0.01	[1, 1]	0.98	1.02	0.98	1	1.02	0.02
BF$_{10}$	0.99	0.01	[0.99, 0.99]	0.95	1.02	0.96	0.98	1	0.01
BF$_{20}$	0.98	0.01	[0.98, 0.98]	0.93	1	0.94	0.98	1	0
BF$_{40}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{60}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{80}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{100}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{5,B}$	1	0.01	[1, 1]	0.97	1.02	0.98	1	1.02	0.02
BF$_{10,B}$	0.99	0.01	[0.99, 0.99]	0.96	1.02	0.96	0.99	1	0.01
BF$_{20,B}$	0.98	0.01	[0.98, 0.98]	0.94	1.02	0.95	0.98	1	0
BF$_{40,B}$	0.98	0.01	[0.98, 0.98]	0.93	1	0.94	0.98	1	0
BF$_{60,B}$	0.97	0.02	[0.97, 0.97]	0.91	1	0.93	0.97	1	0
BF$_{80,B}$	0.97	0.02	[0.97, 0.97]	0.91	1	0.93	0.97	1	0
BF$_{100,B}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
FF$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
FF$_{5}$	1	0.01	[1, 1]	0.98	1.02	0.98	1	1	0.01
FF$_{10}$	0.99	0.01	[0.99, 0.99]	0.95	1.02	0.96	0.98	1	0
FF$_{20}$	0.97	0.01	[0.97, 0.97]	0.93	1	0.94	0.97	1	0
FF$_{40}$	0.95	0.02	[0.95, 0.95]	0.89	1	0.91	0.95	0.98	0
FF$_{60}$	0.95	0.02	[0.95, 0.95]	0.88	1	0.9	0.95	0.98	0
FF$_{80}$	0.95	0.02	[0.95, 0.95]	0.88	1	0.9	0.95	0.98	0
FF$_{100}$	0.95	0.02	[0.95, 0.95]	0.88	1	0.9	0.95	0.98	0
FF$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
FF$_{5,B}$	0.99	0.01	[0.99, 0.99]	0.96	1.02	0.96	1	1	0.01
FF$_{10,B}$	0.98	0.01	[0.98, 0.98]	0.94	1	0.96	0.98	1	0
FF$_{20,B}$	0.97	0.01	[0.97, 0.97]	0.93	1	0.94	0.98	1	0
FF$_{40,B}$	0.96	0.02	[0.96, 0.96]	0.92	1	0.93	0.96	1	0
FF$_{60,B}$	0.96	0.02	[0.96, 0.96]	0.91	1	0.92	0.96	1	0
FF$_{80,B}$	0.96	0.02	[0.96, 0.96]	0.91	1	0.92	0.96	0.98	0
FF$_{100,B}$	0.95	0.02	[0.95, 0.95]	0.88	1	0.9	0.95	0.98	0

Table A.23: Performance ratios of costs relative to the online version of an algorithm in the classical bin packing problem when item permutations are allowed.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
BF$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{5}$	0.99	0.01	[0.99, 0.99]	0.96	1.02	0.96	0.98	1	0.01
BF$_{10}$	0.98	0.01	[0.98, 0.98]	0.94	1.02	0.95	0.98	1	0
BF$_{20}$	0.97	0.02	[0.97, 0.97]	0.92	1.02	0.93	0.98	1	0
BF$_{40}$	0.97	0.02	[0.97, 0.97]	0.89	1	0.92	0.96	1	0
BF$_{60}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{80}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{100}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
BF$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
BF$_{5,B}$	1	0.01	[1, 1]	0.96	1.02	0.98	1	1.02	0.02
BF$_{10,B}$	0.99	0.01	[0.99, 0.99]	0.96	1.02	0.96	1	1	0.01
BF$_{20,B}$	0.98	0.01	[0.98, 0.98]	0.94	1.02	0.96	0.98	1	0.01
BF$_{40,B}$	0.98	0.01	[0.98, 0.98]	0.93	1	0.94	0.98	1	0
BF$_{60,B}$	0.97	0.02	[0.97, 0.97]	0.91	1	0.93	0.98	1	0
BF$_{80,B}$	0.97	0.02	[0.97, 0.97]	0.91	1	0.93	0.97	1	0
BF$_{100,B}$	0.96	0.02	[0.96, 0.96]	0.89	1	0.91	0.96	1	0
FF$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
FF$_{5}$	0.99	0.01	[0.99, 0.99]	0.96	1.02	0.96	0.98	1	0
FF$_{10}$	0.98	0.01	[0.98, 0.98]	0.93	1	0.94	0.98	1	0
FF$_{20}$	0.96	0.02	[0.96, 0.96]	0.91	1	0.92	0.96	1	0
FF$_{40}$	0.95	0.02	[0.95, 0.95]	0.89	1	0.91	0.96	0.99	0
FF$_{60}$	0.95	0.02	[0.95, 0.95]	0.89	1	0.91	0.95	0.98	0
FF$_{80}$	0.95	0.02	[0.95, 0.95]	0.89	1	0.91	0.95	0.98	0
FF$_{100}$	0.95	0.02	[0.95, 0.95]	0.88	1	0.9	0.95	0.98	0
FF$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
FF$_{5,B}$	0.99	0.01	[0.99, 0.99]	0.96	1.02	0.96	1	1	0.01
FF$_{10,B}$	0.98	0.01	[0.98, 0.98]	0.94	1	0.96	0.98	1	0
FF$_{20,B}$	0.97	0.01	[0.97, 0.97]	0.93	1	0.94	0.98	1	0
FF$_{40,B}$	0.96	0.02	[0.96, 0.96]	0.92	1	0.93	0.96	1	0
FF$_{60,B}$	0.96	0.02	[0.96, 0.96]	0.91	1	0.92	0.96	1	0
FF$_{80,B}$	0.96	0.02	[0.96, 0.96]	0.91	1	0.92	0.96	0.98	0
FF$_{100,B}$	0.95	0.02	[0.95, 0.95]	0.88	1	0.9	0.95	0.98	0

Table A.24: Performance ratios of costs relative to the online version of an algorithm in the classical bin packing problem when item permutations are forbidden.

A.2.3.2 Bounded-Space Problem

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Bfb}_1$	14.81	0.13	[14.69, 14.93]	9	21	11	15	19	
$\textsc{Bfb}_5$	14.24	0.13	[14.13, 14.35]	9	21	10	14	19	0
$\textsc{Bfb}_{10}$	14.14	0.14	[14.02, 14.26]	9	21	10	14	19	0
$\textsc{Bfb}_{15}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Bfb}_{20}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Bfb}_{25}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Bfb}_{1,B}$	14.81	0.13	[14.69, 14.93]	9	21	11	15	19	
$\textsc{Bfb}_{5,B}$	14.52	0.13	[14.4, 14.64]	9	21	11	15	19	0
$\textsc{Bfb}_{10,B}$	14.37	0.13	[14.25, 14.49]	9	21	10.5	14	19	0
$\textsc{Bfb}_{15,B}$	14.27	0.13	[14.15, 14.39]	9	21	10	14	19	0.05
$\textsc{Bfb}_{20,B}$	14.24	0.14	[14.12, 14.36]	9	21	10	14	19	0.06
$\textsc{Bfb}_{25,B}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Ffb}_1$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
$\textsc{Ffb}_5$	14.24	0.13	[14.13, 14.35]	9	21	10	14	19	0
$\textsc{Ffb}_{10}$	14.14	0.14	[14.02, 14.26]	9	21	10	14	19	0
$\textsc{Ffb}_{15}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Ffb}_{20}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Ffb}_{25}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Ffb}_{1,B}$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
$\textsc{Ffb}_{5,B}$	14.57	0.13	[14.45, 14.69]	9	21	11	15	19	0
$\textsc{Ffb}_{10,B}$	14.4	0.13	[14.28, 14.52]	9	21	11	14	19	0
$\textsc{Ffb}_{15,B}$	14.28	0.13	[14.16, 14.4]	9	21	10	14	19	0.04
$\textsc{Ffb}_{20,B}$	14.24	0.13	[14.13, 14.35]	9	21	10	14	19	0.06
$\textsc{Ffb}_{25,B}$	14.13	0.14	[14.01, 14.25]	9	21	10	14	19	0
$\textsc{Optb}_1$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
$\textsc{Optb}_5$	14.73	0.13	[14.61, 14.85]	9	21	11	15	19	0.05
$\textsc{Optb}_{10}$	14.45	0.13	[14.33, 14.57]	9	21	10.5	14	19	0.03
$\textsc{Optb}_{15}$	14.29	0.14	[14.17, 14.41]	9	21	10	14	19	0.04
$\textsc{Optb}_{20}$	14.17	0.14	[14.05, 14.29]	9	21	10	14	19	0.03
$\textsc{Optb}_{25}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\textsc{Optb}_{1,B}$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
$\textsc{Optb}_{5,B}$	14.78	0.13	[14.66, 14.9]	10	21	11	15	19.5	0.06
$\textsc{Optb}_{10,B}$	14.65	0.13	[14.53, 14.77]	10	21	11	15	19	0.06
$\textsc{Optb}_{15,B}$	14.47	0.13	[14.35, 14.59]	9	21	10	15	19	0.06
$\textsc{Optb}_{20,B}$	14.43	0.13	[14.31, 14.55]	10	21	11	14	19	0.11
$\textsc{Optb}_{25,B}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\textsc{Optb}'_1$	14.96	0.13	[14.84, 15.08]	9	21	11	15	19.5	
$\textsc{Optb}'_5$	14.68	0.13	[14.56, 14.8]	9	21	10.5	15	19	0.03
$\textsc{Optb}'_{10}$	14.38	0.14	[14.26, 14.5]	9	21	10	14	19	0.03
$\textsc{Optb}'_{15}$	14.26	0.14	[14.14, 14.38]	9	21	10	14	19	0.04
$\textsc{Optb}'_{20}$	14.17	0.14	[14.05, 14.29]	9	21	10	14	19	0.02
$\textsc{Optb}'_{25}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0
$\textsc{Optb}'_{1,B}$	14.96	0.13	[14.84, 15.08]	9	21	11	15	19.5	
$\textsc{Optb}'_{5,B}$	14.71	0.13	[14.59, 14.83]	10	21	11	15	19.5	0.04
$\textsc{Optb}'_{10,B}$	14.47	0.13	[14.35, 14.59]	9	21	10	14	19	0.02
$\textsc{Optb}'_{15,B}$	14.31	0.13	[14.19, 14.43]	9	21	10	14	19	0.05
$\textsc{Optb}'_{20,B}$	14.28	0.13	[14.16, 14.4]	9	21	10	14	19	0.08
$\textsc{Optb}'_{25,B}$	14.12	0.14	[14, 14.24]	9	21	10	14	19	0

Table A.25: Costs in the bounded-space bin packing problem when item permutations are allowed.

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
BFB$_1$	14.81	0.13	[14.69, 14.93]	9	21	11	15	19	
BFB$_5$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0.12
BFB$_{10}$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0
BFB$_{15}$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0
BFB$_{20}$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0
BFB$_{25}$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0
BFB$_{1,B}$	14.81	0.13	[14.69, 14.93]	9	21	11	15	19	
BFB$_{5,B}$	14.78	0.13	[14.66, 14.9]	9	21	11	15	20	0.11
BFB$_{10,B}$	14.77	0.14	[14.64, 14.9]	9	21	11	15	20	0.03
BFB$_{15,B}$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0.05
BFB$_{20,B}$	14.75	0.14	[14.62, 14.88]	9	21	11	15	20	0.03
BFB$_{25,B}$	14.76	0.14	[14.63, 14.89]	9	21	10.5	15	20	0.02
FFB$_1$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
FFB$_5$	14.83	0.14	[14.7, 14.96]	9	21	11	15	20	0.09
FFB$_{10}$	14.83	0.14	[14.7, 14.96]	9	21	11	15	20	0
FFB$_{15}$	14.83	0.14	[14.7, 14.96]	9	21	11	15	20	0
FFB$_{20}$	14.83	0.14	[14.7, 14.96]	9	21	11	15	20	0
FFB$_{25}$	14.83	0.14	[14.7, 14.96]	9	21	11	15	20	0
FFB$_{1,B}$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
FFB$_{5,B}$	14.86	0.13	[14.74, 14.98]	9	21	11	15	20	0.08
FFB$_{10,B}$	14.84	0.13	[14.72, 14.96]	9	21	11	15	20	0.02
FFB$_{15,B}$	14.84	0.14	[14.71, 14.97]	9	21	11	15	20	0.05
FFB$_{20,B}$	14.83	0.13	[14.71, 14.95]	9	21	11	15	20	0.03
FFB$_{25,B}$	14.83	0.14	[14.7, 14.96]	9	21	11	15	20	0.02
OPTB$_1$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
OPTB$_5$	14.71	0.13	[14.59, 14.83]	10	21	11	15	19.5	0.02
OPTB$_{10}$	14.54	0.13	[14.42, 14.66]	9	21	11	15	19	0.02
OPTB$_{15}$	14.46	0.13	[14.34, 14.58]	9	21	11	14	19	0.02
OPTB$_{20}$	14.4	0.13	[14.28, 14.52]	9	21	10	14	19	0.01
OPTB$_{25}$	14.41	0.13	[14.29, 14.53]	9	21	10	14	19	0.05
OPTB$_{1,B}$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
OPTB$_{5,B}$	14.8	0.13	[14.68, 14.92]	10	21	11	15	19.5	0.03
OPTB$_{10,B}$	14.71	0.13	[14.59, 14.83]	10	21	11	15	19	0.06
OPTB$_{15,B}$	14.6	0.13	[14.48, 14.72]	9	21	10.5	15	19	0.07
OPTB$_{20,B}$	14.6	0.13	[14.48, 14.72]	10	21	11	15	19.5	0.12
OPTB$_{25,B}$	14.41	0.13	[14.29, 14.53]	9	21	10	14	19	0.03
OPTB$'_1$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
OPTB$'_5$	14.68	0.13	[14.56, 14.8]	10	21	11	15	19	0.01
OPTB$'_{10}$	14.55	0.13	[14.43, 14.67]	9	21	10.5	15	19	0.03
OPTB$'_{15}$	14.53	0.13	[14.41, 14.65]	9	21	10.5	15	19	0.08
OPTB$'_{20}$	14.43	0.13	[14.31, 14.55]	9	21	10	14	19.5	0.04
OPTB$'_{25}$	14.44	0.14	[14.31, 14.57]	9	21	10	14	19	0.06
OPTB$'_{1,B}$	14.96	0.13	[14.84, 15.08]	9	21	11	15	20	
OPTB$'_{5,B}$	14.78	0.13	[14.66, 14.9]	10	21	11	15	20	0.04
OPTB$'_{10,B}$	14.77	0.13	[14.65, 14.89]	10	21	11	15	20	0.12
OPTB$'_{15,B}$	14.66	0.13	[14.54, 14.78]	9	21	10	15	19	0.11
OPTB$'_{20,B}$	14.69	0.13	[14.57, 14.81]	10	21	11	15	19.5	0.18
OPTB$'_{25,B}$	14.44	0.14	[14.31, 14.57]	9	21	10	14	19	0.05

Table A.26: Costs in the bounded-space bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to OPT for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
BFB$_1$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
BFB$_5$	1.01	0.03	[1.01, 1.01]	0.94	1.11	1	1	1.09	0
BFB$_{10}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.09	0
BFB$_{15}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
BFB$_{20}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
BFB$_{25}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
BFB$_{1,B}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
BFB$_{5,B}$	1.03	0.04	[1.03, 1.03]	1	1.2	1	1	1.12	0
BFB$_{10,B}$	1.02	0.03	[1.02, 1.02]	1	1.2	1	1	1.1	0
BFB$_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
BFB$_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
BFB$_{25,B}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
FFB$_1$	1.06	0.04	[1.06, 1.06]	1	1.23	1	1.07	1.18	0
FFB$_5$	1.01	0.03	[1.01, 1.01]	0.94	1.11	1	1	1.09	0
FFB$_{10}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.09	0
FFB$_{15}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
FFB$_{20}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
FFB$_{25}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
FFB$_{1,B}$	1.06	0.04	[1.06, 1.06]	1	1.23	1	1.07	1.18	0
FFB$_{5,B}$	1.03	0.04	[1.03, 1.03]	1	1.2	1	1	1.14	0
FFB$_{10,B}$	1.02	0.03	[1.02, 1.02]	1	1.2	1	1	1.1	0
FFB$_{15,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.15	1	1	1.09	0
FFB$_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
FFB$_{25,B}$	1	0.01	[1, 1]	0.94	1.11	1	1	1.08	0
OPTB$_1$	1.06	0.04	[1.06, 1.06]	1	1.23	1	1.07	1.18	0
OPTB$_5$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.06	1.17	0
OPTB$_{10}$	1.02	0.04	[1.02, 1.02]	1	1.15	1	1	1.1	0
OPTB$_{15}$	1.01	0.03	[1.01, 1.01]	0.94	1.14	1	1	1.09	0
OPTB$_{20}$	1	0.02	[1, 1]	0.94	1.1	1	1	1.09	0
OPTB$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
OPTB$_{1,B}$	1.06	0.04	[1.06, 1.06]	1	1.23	1	1.07	1.18	0
OPTB$_{5,B}$	1.05	0.04	[1.05, 1.05]	1	1.2	1	1.07	1.17	0
OPTB$_{10,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
OPTB$_{15,B}$	1.03	0.04	[1.03, 1.03]	1	1.18	1	1	1.14	0
OPTB$_{20,B}$	1.02	0.04	[1.02, 1.02]	1	1.17	1	1	1.1	0
OPTB$_{25,B}$	1	0	[1, 1]	1	1	1	1	1	0
OPTB$'_1$	1.06	0.04	[1.06, 1.06]	1	1.2	1	1.07	1.18	0
OPTB$'_5$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
OPTB$'_{10}$	1.02	0.03	[1.02, 1.02]	0.94	1.14	1	1	1.1	0
OPTB$'_{15}$	1.01	0.03	[1.01, 1.01]	0.94	1.14	1	1	1.09	0
OPTB$'_{20}$	1	0.02	[1, 1]	0.94	1.11	1	1	1.09	0
OPTB$'_{25}$	1	0	[1, 1]	0.94	1.08	1	1	1	0
OPTB$'_{1,B}$	1.06	0.04	[1.06, 1.06]	1	1.2	1	1.07	1.18	0
OPTB$'_{5,B}$	1.04	0.04	[1.04, 1.04]	1	1.2	1	1.06	1.15	0
OPTB$'_{10,B}$	1.03	0.04	[1.03, 1.03]	0.94	1.2	1	1	1.11	0
OPTB$'_{15,B}$	1.02	0.03	[1.02, 1.02]	0.94	1.17	1	1	1.1	0
OPTB$'_{20,B}$	1.01	0.03	[1.01, 1.01]	0.94	1.17	1	1	1.09	0
OPTB$'_{25,B}$	1	0	[1, 1]	0.94	1.08	1	1	1	0

Table A.27: Performance ratios of costs relative to OPTB in the bounded-space bin packing problem when item permutations are allowed.

Performance ratios of costs relative to Opt for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Bfb$_{1}$	1.03	0.04	[1.03, 1.03]	0.89	1.18	0.94	1	1.1	0.01
Bfb$_{5}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Bfb$_{10}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Bfb$_{15}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Bfb$_{20}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Bfb$_{25}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Bfb$_{1,B}$	1.03	0.04	[1.03, 1.03]	0.89	1.18	0.94	1	1.1	0.01
Bfb$_{5,B}$	1.03	0.04	[1.03, 1.03]	0.88	1.15	0.94	1	1.13	0.01
Bfb$_{10,B}$	1.03	0.04	[1.03, 1.03]	0.88	1.15	0.94	1	1.13	0.02
Bfb$_{15,B}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Bfb$_{20,B}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.13	0.02
Bfb$_{25,B}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.94	1	1.12	0.02
Ffb$_{1}$	1.04	0.04	[1.04, 1.04]	0.89	1.2	0.94	1.06	1.14	0.01
Ffb$_{5}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{10}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{15}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{20}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{25}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{1,B}$	1.04	0.04	[1.04, 1.04]	0.89	1.2	0.94	1.06	1.14	0.01
Ffb$_{5,B}$	1.03	0.04	[1.03, 1.03]	0.92	1.2	0.94	1	1.13	0.01
Ffb$_{10,B}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{15,B}$	1.03	0.04	[1.03, 1.03]	0.92	1.2	0.94	1	1.13	0.01
Ffb$_{20,B}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Ffb$_{25,B}$	1.03	0.04	[1.03, 1.03]	0.93	1.2	0.94	1	1.13	0.01
Optb$_{1}$	1.04	0.04	[1.04, 1.04]	0.89	1.2	0.94	1.06	1.14	0.01
Optb$_{5}$	1.02	0.04	[1.02, 1.02]	0.89	1.17	0.94	1	1.09	0.02
Optb$_{10}$	1.01	0.03	[1.01, 1.01]	0.89	1.1	0.93	1	1.09	0.03
Optb$_{15}$	1	0.03	[1, 1]	0.88	1.1	0.93	1	1.09	0.04
Optb$_{20}$	1	0.02	[1, 1]	0.88	1.1	0.93	1	1.08	0.05
Optb$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
Optb$_{1,B}$	1.04	0.04	[1.04, 1.04]	0.89	1.2	0.94	1.06	1.14	0.01
Optb$_{5,B}$	1.03	0.04	[1.03, 1.03]	0.89	1.2	0.94	1	1.09	0.02
Optb$_{10,B}$	1.02	0.04	[1.02, 1.02]	0.88	1.2	0.93	1	1.09	0.03
Optb$_{15,B}$	1.01	0.03	[1.01, 1.01]	0.88	1.1	0.93	1	1.09	0.03
Optb$_{20,B}$	1.01	0.03	[1.01, 1.01]	0.88	1.11	0.93	1	1.09	0.03
Optb$_{25,B}$	1	0.01	[1, 1]	0.88	1.07	1	1	1	0
Optb$'_{1}$	1.04	0.04	[1.04, 1.04]	0.89	1.2	0.94	1.06	1.14	0.01
Optb$'_{5}$	1.02	0.03	[1.02, 1.02]	0.89	1.11	0.94	1	1.09	0.02
Optb$'_{10}$	1.01	0.03	[1.01, 1.01]	0.89	1.1	0.93	1	1.09	0.03
Optb$'_{15}$	1.01	0.03	[1.01, 1.01]	0.88	1.13	0.93	1	1.09	0.03
Optb$'_{20}$	1	0.03	[1, 1]	0.88	1.17	0.93	1	1.08	0.05
Optb$'_{25}$	1	0.01	[1, 1]	0.88	1.13	1	1	1.07	0.01
Optb$'_{1,B}$	1.04	0.04	[1.04, 1.04]	0.89	1.2	0.94	1.06	1.14	0.01
Optb$'_{5,B}$	1.03	0.04	[1.03, 1.03]	0.89	1.18	0.94	1	1.11	0.02
Optb$'_{10,B}$	1.03	0.04	[1.03, 1.03]	0.89	1.2	0.93	1	1.12	0.03
Optb$'_{15,B}$	1.02	0.04	[1.02, 1.02]	0.89	1.14	0.93	1	1.09	0.03
Optb$'_{20,B}$	1.02	0.04	[1.02, 1.02]	0.88	1.15	0.93	1	1.1	0.04
Optb$'_{25,B}$	1	0.01	[1, 1]	0.88	1.13	1	1	1.07	0.01

Table A.28: Performance ratios of costs relative to Optb in the bounded-space bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
BFB$_1$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_5$	0.96	0.04	[0.96, 0.96]	0.85	1	0.87	0.94	1	0
BFB$_{10}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
BFB$_{15}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
BFB$_{20}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
BFB$_{25}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
BFB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{5,B}$	0.98	0.03	[0.98, 0.98]	0.85	1.08	0.92	1	1	0
BFB$_{10,B}$	0.97	0.04	[0.97, 0.97]	0.85	1	0.88	1	1	0
BFB$_{15,B}$	0.96	0.04	[0.96, 0.96]	0.85	1	0.87	0.95	1	0
BFB$_{20,B}$	0.96	0.04	[0.96, 0.96]	0.83	1.05	0.87	0.94	1	0
BFB$_{25,B}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
FFB$_1$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_5$	0.95	0.04	[0.95, 0.95]	0.81	1	0.86	0.94	1	0
FFB$_{10}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.94	1	0
FFB$_{15}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.93	1	0
FFB$_{20}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.93	1	0
FFB$_{25}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.93	1	0
FFB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_{5,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.89	1	1	0
FFB$_{10,B}$	0.96	0.04	[0.96, 0.96]	0.85	1	0.87	0.94	1	0
FFB$_{15,B}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
FFB$_{20,B}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
FFB$_{25,B}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.93	1	0
OPTB$_1$	1	0	[1, 1]	1	1	1	1	1	0
OPTB$_5$	0.98	0.04	[0.98, 0.98]	0.85	1.1	0.91	1	1.08	0.05
OPTB$_{10}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.87	1	1	0.01
OPTB$_{15}$	0.95	0.04	[0.95, 0.95]	0.83	1.1	0.86	0.94	1	0
OPTB$_{20}$	0.95	0.04	[0.95, 0.95]	0.83	1.08	0.85	0.94	1	0
OPTB$_{25}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.93	1	0
OPTB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
OPTB$_{5,B}$	0.99	0.04	[0.99, 0.99]	0.86	1.11	0.92	1	1.08	0.06
OPTB$_{10,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.11	0.9	1	1.08	0.03
OPTB$_{15,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.09	0.87	1	1.06	0.01
OPTB$_{20,B}$	0.96	0.04	[0.96, 0.96]	0.85	1.11	0.87	0.94	1.07	0.02
OPTB$_{25,B}$	0.94	0.04	[0.94, 0.94]	0.81	1	0.85	0.93	1	0
OPTB$'_1$	1	0	[1, 1]	1	1	1	1	1	0
OPTB$'_5$	0.98	0.04	[0.98, 0.98]	0.86	1.09	0.91	1	1.07	0.03
OPTB$'_{10}$	0.96	0.04	[0.96, 0.96]	0.85	1.07	0.86	0.94	1.03	0.01
OPTB$'_{15}$	0.95	0.04	[0.95, 0.95]	0.83	1.07	0.86	0.94	1	0
OPTB$'_{20}$	0.95	0.04	[0.95, 0.95]	0.83	1	0.86	0.94	1	0
OPTB$'_{25}$	0.94	0.04	[0.94, 0.94]	0.83	1	0.85	0.93	1	0
OPTB$'_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
OPTB$'_{5,B}$	0.98	0.04	[0.98, 0.98]	0.87	1.11	0.91	1	1.08	0.04
OPTB$'_{10,B}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.88	1	1	0.01
OPTB$'_{15,B}$	0.96	0.04	[0.96, 0.96]	0.85	1	0.86	0.94	1	0
OPTB$'_{20,B}$	0.95	0.04	[0.95, 0.95]	0.85	1.07	0.86	0.94	1	0
OPTB$'_{25,B}$	0.94	0.04	[0.94, 0.94]	0.83	1	0.85	0.93	1	0

Table A.29: Performance ratios of costs relative to the online version of an algorithm in the bounded-space bin packing problem when item permutations are allowed.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Bfb$_1$	1	0	[1, 1]	1	1	1	1	1	0
Bfb$_5$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Bfb$_{10}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Bfb$_{15}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Bfb$_{20}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Bfb$_{25}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Bfb$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Bfb$_{5,B}$	1	0.03	[1, 1]	0.91	1.11	0.92	1	1.08	0.11
Bfb$_{10,B}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.11
Bfb$_{15,B}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Bfb$_{20,B}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.11
Bfb$_{25,B}$	1	0.04	[1, 1]	0.87	1.13	0.92	1	1.08	0.12
Ffb$_1$	1	0	[1, 1]	1	1	1	1	1	0
Ffb$_5$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Ffb$_{10}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Ffb$_{15}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Ffb$_{20}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Ffb$_{25}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Ffb$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Ffb$_{5,B}$	0.99	0.03	[0.99, 0.99]	0.89	1.08	0.92	1	1.07	0.08
Ffb$_{10,B}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.08
Ffb$_{15,B}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Ffb$_{20,B}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.08
Ffb$_{25,B}$	0.99	0.04	[0.99, 0.99]	0.87	1.13	0.92	1	1.08	0.09
Optb$_1$	1	0	[1, 1]	1	1	1	1	1	0
Optb$_5$	0.98	0.03	[0.98, 0.98]	0.88	1.11	0.92	1	1.08	0.02
Optb$_{10}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.89	1	1	0
Optb$_{15}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.88	1	1	0
Optb$_{20}$	0.96	0.04	[0.96, 0.96]	0.85	1.08	0.88	0.94	1	0
Optb$_{25}$	0.96	0.04	[0.96, 0.96]	0.83	1.13	0.88	0.94	1.06	0.01
Optb$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Optb$_{5,B}$	0.99	0.03	[0.99, 0.99]	0.88	1.11	0.92	1	1.08	0.03
Optb$_{10,B}$	0.98	0.04	[0.98, 0.98]	0.87	1.11	0.92	1	1.08	0.03
Optb$_{15,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.09	0.88	1	1.08	0.02
Optb$_{20,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.11	0.91	1	1.07	0.03
Optb$_{25,B}$	0.96	0.04	[0.96, 0.96]	0.83	1.13	0.88	0.94	1.06	0.01
Optb$'_1$	1	0	[1, 1]	1	1	1	1	1	0
Optb$'_5$	0.98	0.03	[0.98, 0.98]	0.86	1.11	0.91	1	1.03	0.01
Optb$'_{10}$	0.97	0.04	[0.97, 0.97]	0.85	1.08	0.88	1	1.03	0.01
Optb$'_{15}$	0.97	0.04	[0.97, 0.97]	0.85	1.09	0.89	1	1.07	0.02
Optb$'_{20}$	0.96	0.04	[0.96, 0.96]	0.85	1.08	0.88	0.94	1.06	0.01
Optb$'_{25}$	0.96	0.04	[0.96, 0.96]	0.83	1.13	0.88	0.94	1.07	0.02
Optb$'_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
Optb$'_{5,B}$	0.99	0.03	[0.99, 0.99]	0.88	1.11	0.92	1	1.08	0.04
Optb$'_{10,B}$	0.99	0.04	[0.99, 0.99]	0.87	1.11	0.92	1	1.08	0.07
Optb$'_{15,B}$	0.98	0.04	[0.98, 0.98]	0.86	1.09	0.88	1	1.08	0.04
Optb$'_{20,B}$	0.98	0.04	[0.98, 0.98]	0.85	1.13	0.88	1	1.08	0.06
Optb$'_{25,B}$	0.96	0.04	[0.96, 0.96]	0.83	1.12	0.88	0.94	1.07	0.02

Table A.30: Performance ratios of costs relative to the online version of an algorithm in the bounded-space bin packing problem when item permutations are forbidden.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Bfb}_{1}$	57.36	0.07	[57.11, 57.61]	47	70	49	57	67	
$\textsc{Bfb}_{5}$	54.48	0.07	[54.24, 54.72]	45	67	47	54	64	0
$\textsc{Bfb}_{10}$	53.62	0.07	[53.39, 53.85]	44	66	46	54	63	0
$\textsc{Bfb}_{20}$	53.19	0.07	[52.96, 53.42]	43	66	45.5	53	63	0
$\textsc{Bfb}_{40}$	52.92	0.07	[52.69, 53.15]	43	66	45	53	63	0
$\textsc{Bfb}_{60}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Bfb}_{80}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Bfb}_{100}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Bfb}_{1,B}$	57.36	0.07	[57.11, 57.61]	47	70	49	57	67	
$\textsc{Bfb}_{5,B}$	55.83	0.07	[55.59, 56.07]	45	68	47	56	65.5	0
$\textsc{Bfb}_{10,B}$	55.14	0.07	[54.9, 55.38]	45	68	47	55	65	0.01
$\textsc{Bfb}_{20,B}$	54.4	0.07	[54.16, 54.64]	44	67	46	54	64	0.01
$\textsc{Bfb}_{40,B}$	53.81	0.07	[53.58, 54.04]	43	67	46	54	63.5	0
$\textsc{Bfb}_{60,B}$	53.47	0.07	[53.24, 53.7]	43	66	46	53	63	0.09
$\textsc{Bfb}_{80,B}$	53.28	0.07	[53.05, 53.51]	43	66	45	53	63	0.12
$\textsc{Bfb}_{100,B}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ffb}_{1}$	58.15	0.06	[57.93, 58.37]	47	70	50	58	68	
$\textsc{Ffb}_{5}$	54.57	0.07	[54.33, 54.81]	45	66	47	54	64	0
$\textsc{Ffb}_{10}$	53.68	0.07	[53.45, 53.91]	44	66	46	54	63.5	0
$\textsc{Ffb}_{20}$	53.19	0.07	[52.96, 53.42]	43	66	45.5	53	63	0
$\textsc{Ffb}_{40}$	52.92	0.07	[52.69, 53.15]	43	66	45	53	63	0
$\textsc{Ffb}_{60}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ffb}_{80}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ffb}_{100}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0
$\textsc{Ffb}_{1,B}$	58.15	0.06	[57.93, 58.37]	47	70	50	58	68	
$\textsc{Ffb}_{5,B}$	56.35	0.07	[56.11, 56.59]	45	69	48	56	66	0
$\textsc{Ffb}_{10,B}$	55.4	0.07	[55.16, 55.64]	45	68	47	55	65	0
$\textsc{Ffb}_{20,B}$	54.47	0.07	[54.23, 54.71]	44	67	46	54	64.5	0
$\textsc{Ffb}_{40,B}$	53.82	0.07	[53.59, 54.05]	43	67	46	54	63.5	0
$\textsc{Ffb}_{60,B}$	53.48	0.07	[53.25, 53.71]	43	66	46	53	63	0.09
$\textsc{Ffb}_{80,B}$	53.3	0.07	[53.07, 53.53]	43	66	45.5	53	63	0.11
$\textsc{Ffb}_{100,B}$	52.85	0.07	[52.62, 53.08]	43	66	45	53	63	0

Table A.31: Costs in the bounded-space bin packing problem when item permutations are allowed.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Bfb}_{1}$	57.36	0.07	[57.11, 57.61]	47	70	49	57	67	
$\textsc{Bfb}_{5}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0.29
$\textsc{Bfb}_{10}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0
$\textsc{Bfb}_{20}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0
$\textsc{Bfb}_{40}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0
$\textsc{Bfb}_{60}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0
$\textsc{Bfb}_{80}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0
$\textsc{Bfb}_{100}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0
$\textsc{Bfb}_{1,B}$	57.36	0.07	[57.11, 57.61]	47	70	49	57	67	
$\textsc{Bfb}_{5,B}$	57.46	0.07	[57.21, 57.71]	47	71	49	57	67	0.31
$\textsc{Bfb}_{10,B}$	57.44	0.07	[57.19, 57.69]	46	71	49	57	67	0.14
$\textsc{Bfb}_{20,B}$	57.43	0.07	[57.18, 57.68]	46	71	49	57	67	0.1
$\textsc{Bfb}_{40,B}$	57.43	0.07	[57.18, 57.68]	46	71	49	57	67	0.04
$\textsc{Bfb}_{60,B}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0.05
$\textsc{Bfb}_{80,B}$	57.43	0.07	[57.18, 57.68]	46	71	49	57	67	0.05
$\textsc{Bfb}_{100,B}$	57.42	0.07	[57.17, 57.67]	46	71	49	57	67	0.02
$\textsc{Ffb}_{1}$	58.15	0.06	[57.93, 58.37]	47	70	50	58	68	
$\textsc{Ffb}_{5}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0.2
$\textsc{Ffb}_{10}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0
$\textsc{Ffb}_{20}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0
$\textsc{Ffb}_{40}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0
$\textsc{Ffb}_{60}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0
$\textsc{Ffb}_{80}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0
$\textsc{Ffb}_{100}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0
$\textsc{Ffb}_{1,B}$	58.15	0.06	[57.93, 58.37]	47	70	50	58	68	
$\textsc{Ffb}_{5,B}$	57.95	0.07	[57.7, 58.2]	48	71	49.5	58	68	0.19
$\textsc{Ffb}_{10,B}$	57.91	0.07	[57.66, 58.16]	47	71	49	58	68	0.13
$\textsc{Ffb}_{20,B}$	57.88	0.07	[57.63, 58.13]	47	71	49	58	68	0.08
$\textsc{Ffb}_{40,B}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0.03
$\textsc{Ffb}_{60,B}$	57.88	0.07	[57.63, 58.13]	47	71	49	58	68	0.05
$\textsc{Ffb}_{80,B}$	57.88	0.07	[57.63, 58.13]	46	71	49	58	68	0.04
$\textsc{Ffb}_{100,B}$	57.87	0.07	[57.62, 58.12]	46	71	49	58	68	0.01

Table A.32: Costs in the bounded-space bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to BFB$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
BFB$_1$	1.09	0.02	[1.09, 1.09]	1.02	1.15	1.03	1.09	1.14	0
BFB$_5$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.04	1.07	0
BFB$_{10}$	1.01	0.01	[1.01, 1.01]	1	1.08	1	1.02	1.04	0
BFB$_{20}$	1.01	0.01	[1.01, 1.01]	1	1.04	1	1	1.04	0
BFB$_{40}$	1	0.01	[1, 1]	1	1.02	1	1	1.02	0
BFB$_{60}$	1	0	[1, 1]	1	1.02	1	1	1	0
BFB$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{1,B}$	1.09	0.02	[1.09, 1.09]	1.02	1.15	1.03	1.09	1.14	0
BFB$_{5,B}$	1.06	0.02	[1.06, 1.06]	1	1.13	1.02	1.06	1.1	0
BFB$_{10,B}$	1.04	0.02	[1.04, 1.04]	1	1.1	1	1.04	1.1	0
BFB$_{20,B}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.02	1.08	0
BFB$_{40,B}$	1.02	0.01	[1.02, 1.02]	1	1.08	1	1.02	1.06	0
BFB$_{60,B}$	1.01	0.01	[1.01, 1.01]	1	1.08	1	1.02	1.05	0
BFB$_{80,B}$	1.01	0.01	[1.01, 1.01]	1	1.06	1	1	1.04	0
BFB$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_1$	1.1	0.02	[1.1, 1.1]	1.03	1.19	1.05	1.1	1.16	0
FFB$_5$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.04	1.08	0
FFB$_{10}$	1.02	0.01	[1.02, 1.02]	1	1.06	1	1.02	1.04	0
FFB$_{20}$	1.01	0.01	[1.01, 1.01]	1	1.04	1	1	1.04	0
FFB$_{40}$	1	0.01	[1, 1]	1	1.02	1	1	1.02	0
FFB$_{60}$	1	0	[1, 1]	1	1.02	1	1	1	0
FFB$_{80}$	1	0	[1, 1]	1	1.02	1	1	1	0
FFB$_{100}$	1	0	[1, 1]	1	1.02	1	1	1	0
FFB$_{1,B}$	1.1	0.02	[1.1, 1.1]	1.03	1.19	1.05	1.1	1.16	0
FFB$_{5,B}$	1.07	0.02	[1.07, 1.07]	1	1.14	1.02	1.07	1.12	0
FFB$_{10,B}$	1.05	0.02	[1.05, 1.05]	1	1.12	1.02	1.05	1.1	0
FFB$_{20,B}$	1.03	0.02	[1.03, 1.03]	1	1.1	1	1.03	1.08	0
FFB$_{40,B}$	1.02	0.01	[1.02, 1.02]	1	1.08	1	1.02	1.06	0
FFB$_{60,B}$	1.01	0.01	[1.01, 1.01]	1	1.08	1	1.02	1.05	0
FFB$_{80,B}$	1.01	0.01	[1.01, 1.01]	1	1.06	1	1	1.04	0
FFB$_{100,B}$	1	0	[1, 1]	1	1.02	1	1	1	0

Table A.33: Performance ratios of costs relative to BFB$_{100}$ in the bounded-space bin packing problem when item permutations are allowed.

Performance ratios of costs relative to BFB$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
BFB$_1$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_5$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{20}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{40}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{60}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{80}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{1,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{5,B}$	1	0.01	[1, 1]	0.96	1.05	0.97	1	1.04	0.2
BFB$_{10,B}$	1	0.01	[1, 1]	0.96	1.04	0.98	1	1.02	0.14
BFB$_{20,B}$	1	0.01	[1, 1]	0.96	1.04	0.98	1	1.02	0.07
BFB$_{40,B}$	1	0	[1, 1]	0.98	1.02	0.98	1	1.02	0.03
BFB$_{60,B}$	1	0	[1, 1]	0.97	1.02	0.98	1	1.02	0.02
BFB$_{80,B}$	1	0	[1, 1]	0.98	1.02	0.98	1	1.02	0.02
BFB$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_1$	1.01	0.02	[1.01, 1.01]	0.96	1.08	0.97	1.02	1.06	0.1
FFB$_5$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{10}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{20}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{40}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{60}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{80}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{100}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03
FFB$_{1,B}$	1.01	0.02	[1.01, 1.01]	0.96	1.08	0.97	1.02	1.06	0.1
FFB$_{5,B}$	1.01	0.01	[1.01, 1.01]	0.97	1.06	0.98	1	1.04	0.08
FFB$_{10,B}$	1.01	0.01	[1.01, 1.01]	0.97	1.06	0.98	1	1.04	0.06
FFB$_{20,B}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.05
FFB$_{40,B}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.04
FFB$_{60,B}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.04
FFB$_{80,B}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.04
FFB$_{100,B}$	1.01	0.01	[1.01, 1.01]	0.96	1.06	0.98	1	1.04	0.03

Table A.34: Performance ratios of costs relative to BFB$_{100}$ in the bounded-space bin packing problem when item permutations are forbidden.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
BFB$_1$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_5$	0.95	0.02	[0.95, 0.95]	0.9	1	0.91	0.95	0.98	0
BFB$_{10}$	0.93	0.02	[0.93, 0.93]	0.88	0.98	0.89	0.93	0.97	0
BFB$_{20}$	0.93	0.02	[0.93, 0.93]	0.87	0.98	0.88	0.93	0.97	0
BFB$_{40}$	0.92	0.02	[0.92, 0.92]	0.87	0.98	0.88	0.92	0.97	0
BFB$_{60}$	0.92	0.02	[0.92, 0.92]	0.87	0.98	0.88	0.92	0.97	0
BFB$_{80}$	0.92	0.02	[0.92, 0.92]	0.87	0.98	0.88	0.92	0.97	0
BFB$_{100}$	0.92	0.02	[0.92, 0.92]	0.87	0.98	0.88	0.92	0.97	0
BFB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{5,B}$	0.97	0.01	[0.97, 0.97]	0.93	1.02	0.94	0.97	1	0
BFB$_{10,B}$	0.96	0.02	[0.96, 0.96]	0.92	1	0.93	0.96	1	0
BFB$_{20,B}$	0.95	0.02	[0.95, 0.95]	0.89	1	0.91	0.95	0.98	0
BFB$_{40,B}$	0.94	0.02	[0.94, 0.94]	0.89	0.98	0.89	0.94	0.98	0
BFB$_{60,B}$	0.93	0.02	[0.93, 0.93]	0.87	0.98	0.89	0.93	0.97	0
BFB$_{80,B}$	0.93	0.02	[0.93, 0.93]	0.87	0.98	0.88	0.93	0.97	0
BFB$_{100,B}$	0.92	0.02	[0.92, 0.92]	0.87	0.98	0.88	0.92	0.97	0
FFB$_1$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_5$	0.94	0.02	[0.94, 0.94]	0.88	0.98	0.9	0.94	0.97	0
FFB$_{10}$	0.92	0.02	[0.92, 0.92]	0.85	0.98	0.88	0.92	0.97	0
FFB$_{20}$	0.91	0.02	[0.91, 0.91]	0.84	0.97	0.87	0.92	0.96	0
FFB$_{40}$	0.91	0.02	[0.91, 0.91]	0.84	0.97	0.86	0.91	0.95	0
FFB$_{60}$	0.91	0.02	[0.91, 0.91]	0.84	0.97	0.86	0.91	0.95	0
FFB$_{80}$	0.91	0.02	[0.91, 0.91]	0.84	0.97	0.86	0.91	0.95	0
FFB$_{100}$	0.91	0.02	[0.91, 0.91]	0.84	0.97	0.86	0.91	0.95	0
FFB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_{5,B}$	0.97	0.02	[0.97, 0.97]	0.92	1	0.93	0.97	1	0
FFB$_{10,B}$	0.95	0.02	[0.95, 0.95]	0.89	1	0.91	0.95	0.98	0
FFB$_{20,B}$	0.94	0.02	[0.94, 0.94]	0.87	1	0.89	0.93	0.98	0
FFB$_{40,B}$	0.93	0.02	[0.93, 0.93]	0.86	0.98	0.88	0.93	0.97	0
FFB$_{60,B}$	0.92	0.02	[0.92, 0.92]	0.84	0.97	0.87	0.92	0.96	0
FFB$_{80,B}$	0.92	0.02	[0.92, 0.92]	0.85	0.97	0.87	0.92	0.96	0
FFB$_{100,B}$	0.91	0.02	[0.91, 0.91]	0.84	0.97	0.86	0.91	0.95	0

Table A.35: Performance ratios of costs relative to the online version of an algorithm in the bounded-space bin packing problem when item permutations are allowed.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
BFB$_1$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_5$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{10}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{20}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{40}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{60}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{80}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{100}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
BFB$_{5,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.31
BFB$_{10,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.3
BFB$_{20,B}$	1	0.02	[1, 1]	0.95	1.06	0.96	1	1.04	0.3
BFB$_{40,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{60,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{80,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
BFB$_{100,B}$	1	0.02	[1, 1]	0.95	1.05	0.96	1	1.04	0.29
FFB$_1$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_5$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{10}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{20}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{40}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{60}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{80}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{100}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
FFB$_{5,B}$	1	0.02	[1, 1]	0.93	1.05	0.96	1	1.03	0.19
FFB$_{10,B}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.21
FFB$_{20,B}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{40,B}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{60,B}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{80,B}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2
FFB$_{100,B}$	1	0.02	[1, 1]	0.95	1.05	0.95	1	1.03	0.2

Table A.36: Performance ratios of costs relative to the online version of an algorithm in the bounded-space bin packing problem when item permutations are forbidden.

A.2.4 Online Traveling Salesman with Lookahead

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
NN$_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
NN$_{5}$	8.09	0.1	[8.04, 8.14]	6	10.2	6.4	8	9.95	0
NN$_{10}$	6.43	0.11	[6.39, 6.47]	4.2	8.8	5	6.4	8	0.02
NN$_{15}$	5.77	0.1	[5.73, 5.81]	3.8	7.9	4.5	5.8	7.3	0.13
NN$_{20}$	5.48	0.1	[5.45, 5.51]	3.6	7.2	4.25	5.5	6.8	0.23
NN$_{25}$	5.39	0.1	[5.36, 5.42]	3.8	7.2	4.1	5.4	6.6	0.09
NN$_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
NN$_{5,B}$	9.76	0.09	[9.71, 9.81]	7	13	7.65	9.8	11.95	0
NN$_{10,B}$	8.06	0.1	[8.01, 8.11]	5.6	10.8	6.25	8.1	10	0.02
NN$_{15,B}$	6.94	0.1	[6.9, 6.98]	5	9	5.45	6.9	8.5	0.08
NN$_{20,B}$	6.57	0.1	[6.53, 6.61]	4.4	9	5.2	6.6	8.1	0.26
NN$_{25,B}$	5.39	0.1	[5.36, 5.42]	3.8	7.2	4.1	5.4	6.6	0.03
INS$_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
INS$_{5}$	8.61	0.11	[8.55, 8.67]	6.1	11.9	6.6	8.6	11	0
INS$_{10}$	6.93	0.12	[6.88, 6.98]	4.1	10.1	5.15	6.9	9.1	0.06
INS$_{15}$	5.9	0.11	[5.86, 5.94]	4.2	8.4	4.5	5.9	7.5	0.13
INS$_{20}$	5.56	0.08	[5.53, 5.59]	4.2	7	4.5	5.5	6.6	0.28
INS$_{25}$	5.55	0.06	[5.53, 5.57]	4.4	6.5	4.7	5.6	6.3	0.46
INS$_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
INS$_{5,B}$	9.91	0.1	[9.85, 9.97]	7.1	13	7.8	9.9	12.3	0
INS$_{10,B}$	8.29	0.09	[8.24, 8.34]	6	10.4	6.55	8.3	10.05	0.02
INS$_{15,B}$	6.52	0.08	[6.49, 6.55]	4.4	8.2	5.2	6.5	7.7	0.01
INS$_{20,B}$	6.17	0.09	[6.14, 6.2]	4.4	8.1	5	6.2	7.45	0.19
INS$_{25,B}$	5.55	0.06	[5.53, 5.57]	4.4	6.5	4.7	5.6	6.3	0.06
2OPT$_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
2OPT$_{5}$	8.81	0.12	[8.74, 8.88]	5.7	12.2	6.45	8.7	11.5	0
2OPT$_{10}$	6.24	0.11	[6.2, 6.28]	4.2	8.8	4.85	6.2	8	0.01
2OPT$_{15}$	5.27	0.1	[5.24, 5.3]	3.6	7.2	4.2	5.3	6.5	0.05
2OPT$_{20}$	4.79	0.08	[4.77, 4.81]	3.6	6.2	3.95	4.8	5.7	0.1
2OPT$_{25}$	4.66	0.07	[4.64, 4.68]	3.6	5.8	3.9	4.7	5.5	0.18
2OPT$_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
2OPT$_{5,B}$	9.99	0.09	[9.93, 10.05]	7.6	12.9	7.9	10	12	0
2OPT$_{10,B}$	7.93	0.08	[7.89, 7.97]	6	9.9	6.5	7.9	9.4	0
2OPT$_{15,B}$	6.42	0.08	[6.39, 6.45]	5	8	5.3	6.4	7.7	0.01
2OPT$_{20,B}$	6.29	0.08	[6.26, 6.32]	4.6	7.8	4.95	6.3	7.5	0.32
2OPT$_{25,B}$	4.66	0.07	[4.64, 4.68]	3.6	5.8	3.9	4.7	5.5	0
3OPT$_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
3OPT$_{5}$	8.75	0.12	[8.68, 8.82]	5.7	12.3	6.4	8.7	11.4	0
3OPT$_{10}$	6.15	0.1	[6.11, 6.19]	4.2	8.8	4.8	6.1	7.85	0
3OPT$_{15}$	5.16	0.09	[5.13, 5.19]	3.6	6.5	4.2	5.1	6.2	0.03
3OPT$_{20}$	4.68	0.07	[4.66, 4.7]	3.6	6.2	3.9	4.7	5.5	0.05
3OPT$_{25}$	4.57	0.07	[4.55, 4.59]	3.6	5.6	3.9	4.6	5.2	0.12
3OPT$_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
3OPT$_{5,B}$	9.96	0.09	[9.9, 10.02]	7.6	12.9	7.85	10	12	0
3OPT$_{10,B}$	7.88	0.08	[7.84, 7.92]	6	9.8	6.5	7.9	9.3	0
3OPT$_{15,B}$	6.36	0.07	[6.33, 6.39]	5	7.9	5.3	6.4	7.45	0
3OPT$_{20,B}$	6.23	0.08	[6.2, 6.26]	4.6	7.7	5	6.2	7.35	0.3
3OPT$_{25,B}$	4.57	0.07	[4.55, 4.59]	3.6	5.6	3.9	4.6	5.2	0
SA$_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
SA$_{5}$	8.92	0.13	[8.85, 8.99]	6.1	12.9	6.6	8.8	11.95	0
SA$_{10}$	6.24	0.11	[6.2, 6.28]	4.2	9.2	4.85	6.2	8.05	0
SA$_{15}$	5.27	0.1	[5.24, 5.3]	3.6	7.2	4.2	5.3	6.5	0.05
SA$_{20}$	4.79	0.08	[4.77, 4.81]	3.6	6.2	3.95	4.8	5.7	0.1
SA$_{25}$	4.66	0.07	[4.64, 4.68]	3.6	5.8	3.9	4.7	5.5	0.18
SA$_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
SA$_{5,B}$	10.08	0.09	[10.02, 10.14]	7.6	13.1	8	10.1	12.25	0
SA$_{10,B}$	7.92	0.08	[7.88, 7.96]	6	9.9	6.5	7.9	9.4	0
SA$_{15,B}$	6.42	0.08	[6.39, 6.45]	5	8	5.3	6.4	7.7	0.01
SA$_{20,B}$	6.28	0.08	[6.25, 6.31]	4.6	7.8	4.95	6.3	7.5	0.31
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\textsc{Sa}_{25,B}$	4.66	0.07	[4.64, 4.68]	3.6	5.8	3.9	4.7	5.5	0
$\textsc{Ts}_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
$\textsc{Ts}_{5}$	8.9	0.12	[8.83, 8.97]	6.1	13.1	6.6	8.8	11.85	0
$\textsc{Ts}_{10}$	6.22	0.11	[6.18, 6.26]	4.2	8.6	4.8	6.2	7.95	0.01
$\textsc{Ts}_{15}$	5.21	0.09	[5.18, 5.24]	3.6	6.8	4.2	5.2	6.4	0.05
$\textsc{Ts}_{20}$	4.76	0.08	[4.74, 4.78]	3.6	6.5	3.95	4.75	5.6	0.11
$\textsc{Ts}_{25}$	4.64	0.07	[4.62, 4.66]	3.6	5.7	3.9	4.7	5.4	0.2
$\textsc{Ts}_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
$\textsc{Ts}_{5,B}$	10.07	0.09	[10.01, 10.13]	7.6	13.1	8	10.1	12.2	0
$\textsc{Ts}_{10,B}$	7.97	0.08	[7.93, 8.01]	6	9.6	6.5	8	9.4	0
$\textsc{Ts}_{15,B}$	6.42	0.08	[6.39, 6.45]	5	7.9	5.3	6.4	7.6	0
$\textsc{Ts}_{20,B}$	6.3	0.08	[6.27, 6.33]	4.6	7.9	4.95	6.3	7.45	0.33
$\textsc{Ts}_{25,B}$	4.65	0.07	[4.63, 4.67]	3.6	5.7	3.9	4.7	5.4	0
$\textsc{Opt}_{1}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
$\textsc{Opt}_{5}$	9.55	0.1	[9.49, 9.61]	6.8	13	7.55	9.5	11.8	0
$\textsc{Opt}_{10}$	6.89	0.1	[6.85, 6.93]	5.1	9.6	5.3	6.9	8.5	0
$\textsc{Opt}_{15}$	5.67	0.1	[5.63, 5.71]	4.1	11.2	4.4	5.6	7.15	0.01
$\textsc{Opt}_{20}$	4.89	0.08	[4.87, 4.91]	3.6	6.5	4	4.9	5.9	0.02
$\textsc{Opt}_{25}$	4.52	0.06	[4.5, 4.54]	3.6	5.5	3.85	4.5	5.15	0.02
$\textsc{Opt}_{1,B}$	14.05	0.1	[13.96, 14.14]	10.5	19.2	10.8	14.05	17.5	
$\textsc{Opt}_{5,B}$	10.41	0.09	[10.35, 10.47]	7.4	13.8	8.35	10.4	12.6	0
$\textsc{Opt}_{10,B}$	8.13	0.08	[8.09, 8.17]	6	10	6.7	8.1	9.5	0
$\textsc{Opt}_{15,B}$	6.55	0.07	[6.52, 6.58]	5.1	7.9	5.4	6.6	7.6	0
$\textsc{Opt}_{20,B}$	6.4	0.08	[6.37, 6.43]	4.8	7.7	5	6.4	7.5	0.3
$\textsc{Opt}_{25,B}$	4.53	0.06	[4.51, 4.55]	3.6	5.4	3.85	4.5	5.2	0

Table A.37: Costs in the TSP.

Performance ratios of costs relative to OPT for $n = 25$ (1000 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
NN$_{1}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
NN$_{5}$	1.79	0.1	[1.78, 1.8]	1.25	2.49	1.4	1.78	2.21	0
NN$_{10}$	1.42	0.1	[1.41, 1.43]	1.07	1.98	1.12	1.42	1.74	0
NN$_{15}$	1.28	0.09	[1.27, 1.29]	1	1.78	1.05	1.27	1.56	0
NN$_{20}$	1.21	0.09	[1.2, 1.22]	1	1.68	1.02	1.21	1.48	0
NN$_{25}$	1.19	0.08	[1.18, 1.2]	1	1.5	1	1.19	1.43	0
NN$_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
NN$_{5,B}$	2.16	0.1	[2.15, 2.17]	1.49	2.93	1.69	2.15	2.63	0
NN$_{10,B}$	1.78	0.09	[1.77, 1.79]	1.24	2.35	1.43	1.77	2.22	0
NN$_{15,B}$	1.54	0.09	[1.53, 1.55]	1.14	1.93	1.25	1.53	1.85	0
NN$_{20,B}$	1.45	0.09	[1.44, 1.46]	1.08	2	1.18	1.45	1.76	0
NN$_{25,B}$	1.19	0.08	[1.18, 1.2]	1	1.5	1	1.19	1.43	0
INS$_{1}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
INS$_{5}$	1.91	0.11	[1.9, 1.92]	1.37	2.8	1.48	1.89	2.44	0
INS$_{10}$	1.53	0.11	[1.52, 1.54]	1.11	2.2	1.18	1.52	1.97	0
INS$_{15}$	1.31	0.11	[1.3, 1.32]	0.96	1.77	1.02	1.3	1.67	0
INS$_{20}$	1.23	0.08	[1.22, 1.24]	0.92	1.67	1.03	1.23	1.5	0.01
INS$_{25}$	1.23	0.05	[1.23, 1.23]	1.02	1.45	1.08	1.23	1.38	0
INS$_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
INS$_{5,B}$	2.19	0.1	[2.18, 2.2]	1.62	3.05	1.69	2.19	2.75	0
INS$_{10,B}$	1.84	0.09	[1.83, 1.85]	1.29	2.41	1.41	1.84	2.24	0
INS$_{15,B}$	1.44	0.07	[1.43, 1.45]	1.15	1.81	1.2	1.44	1.69	0
INS$_{20,B}$	1.37	0.08	[1.36, 1.38]	1.09	1.67	1.14	1.36	1.62	0
INS$_{25,B}$	1.23	0.05	[1.23, 1.23]	1.02	1.45	1.08	1.23	1.38	0
2OPT$_{1}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
2OPT$_{5}$	1.95	0.13	[1.93, 1.97]	1.33	3.05	1.43	1.93	2.56	0
2OPT$_{10}$	1.38	0.1	[1.37, 1.39]	1.04	2.02	1.11	1.37	1.74	0
2OPT$_{15}$	1.17	0.08	[1.16, 1.18]	0.95	1.51	1	1.16	1.39	0
2OPT$_{20}$	1.06	0.05	[1.06, 1.06]	0.9	1.3	0.98	1.05	1.22	0.02
2OPT$_{25}$	1.03	0.04	[1.03, 1.03]	0.87	1.23	0.96	1.02	1.14	0.03
2OPT$_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
2OPT$_{5,B}$	2.21	0.1	[2.2, 2.22]	1.62	3.15	1.73	2.2	2.78	0
2OPT$_{10,B}$	1.76	0.08	[1.75, 1.77]	1.31	2.25	1.43	1.75	2.12	0
2OPT$_{15,B}$	1.42	0.07	[1.41, 1.43]	1.15	1.69	1.21	1.41	1.66	0
2OPT$_{20,B}$	1.39	0.07	[1.38, 1.4]	1.09	1.71	1.15	1.4	1.62	0
2OPT$_{25,B}$	1.03	0.04	[1.03, 1.03]	0.87	1.23	0.96	1.02	1.14	0.03
3OPT$_{1}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
3OPT$_{5}$	1.94	0.13	[1.92, 1.96]	1.33	3.05	1.43	1.93	2.59	0
3OPT$_{10}$	1.36	0.09	[1.35, 1.37]	1.04	1.91	1.09	1.35	1.7	0
3OPT$_{15}$	1.14	0.08	[1.13, 1.15]	0.89	1.44	1	1.13	1.37	0.01
3OPT$_{20}$	1.03	0.04	[1.03, 1.03]	0.87	1.28	0.96	1.02	1.18	0.03
3OPT$_{25}$	1.01	0.02	[1.01, 1.01]	0.85	1.13	0.94	1	1.08	0.06
3OPT$_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
3OPT$_{5,B}$	2.21	0.1	[2.2, 2.22]	1.56	3.15	1.73	2.19	2.76	0
3OPT$_{10,B}$	1.75	0.08	[1.74, 1.76]	1.33	2.2	1.43	1.74	2.1	0
3OPT$_{15,B}$	1.41	0.06	[1.4, 1.42]	1.13	1.69	1.2	1.4	1.61	0
3OPT$_{20,B}$	1.38	0.07	[1.37, 1.39]	1.06	1.71	1.15	1.38	1.6	0
3OPT$_{25,B}$	1.01	0.02	[1.01, 1.01]	0.85	1.13	0.94	1	1.08	0.06
SA$_{1}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
SA$_{5}$	1.98	0.13	[1.96, 2]	1.4	3.05	1.48	1.95	2.75	0
SA$_{10}$	1.38	0.1	[1.37, 1.39]	1.04	2.02	1.11	1.37	1.74	0
SA$_{15}$	1.17	0.08	[1.16, 1.18]	0.95	1.51	1	1.16	1.39	0
SA$_{20}$	1.06	0.05	[1.06, 1.06]	0.9	1.3	0.98	1.05	1.22	0.02
SA$_{25}$	1.03	0.04	[1.03, 1.03]	0.87	1.23	0.96	1.02	1.14	0.03
SA$_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
SA$_{5,B}$	2.23	0.1	[2.22, 2.24]	1.6	2.98	1.74	2.23	2.75	0
SA$_{10,B}$	1.75	0.08	[1.74, 1.76]	1.33	2.25	1.43	1.74	2.12	0
SA$_{15,B}$	1.42	0.07	[1.41, 1.43]	1.15	1.69	1.21	1.41	1.66	0
SA$_{20,B}$	1.39	0.07	[1.38, 1.4]	1.06	1.71	1.15	1.39	1.62	0
SA$_{25,B}$	1.03	0.04	[1.03, 1.03]	0.87	1.23	0.96	1.02	1.14	0.03
TS$_{1}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
TS$_{5}$	1.97	0.13	[1.95, 1.99]	1.4	3.05	1.48	1.95	2.72	0
TS$_{10}$	1.38	0.1	[1.37, 1.39]	1.02	1.87	1.1	1.36	1.71	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of costs relative to OPT for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
TS_{15}	1.15	0.08	[1.14, 1.16]	0.96	1.49	1	1.14	1.41	0
TS_{20}	1.05	0.05	[1.05, 1.05]	0.88	1.25	0.97	1.04	1.19	0.02
TS_{25}	1.03	0.03	[1.03, 1.03]	0.87	1.23	0.95	1.02	1.13	0.03
$\text{TS}_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
$\text{TS}_{5,B}$	2.23	0.1	[2.22, 2.24]	1.6	2.98	1.75	2.22	2.74	0
$\text{TS}_{10,B}$	1.77	0.08	[1.76, 1.78]	1.33	2.24	1.43	1.76	2.13	0
$\text{TS}_{15,B}$	1.42	0.07	[1.41, 1.43]	1.13	1.68	1.21	1.42	1.64	0
$\text{TS}_{20,B}$	1.39	0.07	[1.38, 1.4]	1.06	1.71	1.15	1.4	1.62	0
$\text{TS}_{25,B}$	1.03	0.03	[1.03, 1.03]	0.87	1.2	0.95	1.02	1.13	0.03
OPT_{1}	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
OPT_{5}	2.12	0.1	[2.11, 2.13]	1.55	2.88	1.66	2.09	2.64	0
OPT_{10}	1.52	0.1	[1.51, 1.53]	1.1	2.02	1.23	1.52	1.91	0
OPT_{15}	1.25	0.09	[1.24, 1.26]	0.96	2.24	1.04	1.25	1.54	0
OPT_{20}	1.08	0.06	[1.08, 1.08]	0.87	1.34	0.98	1.07	1.27	0.02
OPT_{25}	1	0	[1, 1]	1	1	1	1	1	0
$\text{OPT}_{1,B}$	3.11	0.11	[3.09, 3.13]	2.19	4.68	2.37	3.1	3.98	0
$\text{OPT}_{5,B}$	2.31	0.1	[2.3, 2.32]	1.66	3	1.82	2.3	2.89	0
$\text{OPT}_{10,B}$	1.8	0.08	[1.79, 1.81]	1.4	2.34	1.48	1.79	2.16	0
$\text{OPT}_{15,B}$	1.45	0.07	[1.44, 1.46]	1.13	1.74	1.21	1.45	1.66	0
$\text{OPT}_{20,B}$	1.42	0.07	[1.41, 1.43]	1.11	1.71	1.16	1.42	1.63	0
$\text{OPT}_{25,B}$	1	0.01	[1, 1]	0.9	1.08	1	1	1.02	0.01

Table A.38: Performance ratios of costs relative to OPT in the TSP.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
NN$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
NN$_{5}$	0.58	0.11	[0.58, 0.58]	0.41	0.79	0.45	0.57	0.74	0
NN$_{10}$	0.46	0.12	[0.46, 0.46]	0.29	0.66	0.33	0.46	0.6	0
NN$_{15}$	0.41	0.13	[0.41, 0.41]	0.27	0.62	0.29	0.41	0.56	0
NN$_{20}$	0.39	0.14	[0.39, 0.39]	0.23	0.61	0.28	0.39	0.54	0
NN$_{25}$	0.39	0.13	[0.39, 0.39]	0.23	0.61	0.27	0.38	0.53	0
NN$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
NN$_{5,B}$	0.7	0.1	[0.7, 0.7]	0.52	0.9	0.55	0.7	0.86	0
NN$_{10,B}$	0.58	0.11	[0.58, 0.58]	0.39	0.78	0.44	0.57	0.74	0
NN$_{15,B}$	0.5	0.12	[0.5, 0.5]	0.33	0.71	0.37	0.49	0.66	0
NN$_{20,B}$	0.47	0.12	[0.47, 0.47]	0.31	0.68	0.36	0.47	0.62	0
NN$_{25,B}$	0.39	0.13	[0.39, 0.39]	0.23	0.61	0.27	0.38	0.53	0
INS$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
INS$_{5}$	0.62	0.12	[0.62, 0.62]	0.44	0.87	0.46	0.61	0.79	0
INS$_{10}$	0.5	0.14	[0.5, 0.5]	0.29	0.82	0.35	0.49	0.67	0
INS$_{15}$	0.42	0.15	[0.42, 0.42]	0.27	0.68	0.29	0.42	0.57	0
INS$_{20}$	0.4	0.12	[0.4, 0.4]	0.27	0.57	0.31	0.39	0.54	0
INS$_{25}$	0.4	0.11	[0.4, 0.4]	0.29	0.57	0.32	0.4	0.52	0
INS$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
INS$_{5,B}$	0.71	0.09	[0.71, 0.71]	0.51	0.92	0.56	0.71	0.86	0
INS$_{10,B}$	0.59	0.1	[0.59, 0.59]	0.43	0.8	0.47	0.59	0.73	0
INS$_{15,B}$	0.47	0.11	[0.47, 0.47]	0.31	0.66	0.36	0.47	0.59	0
INS$_{20,B}$	0.44	0.11	[0.44, 0.44]	0.31	0.61	0.34	0.44	0.57	0
INS$_{25,B}$	0.4	0.11	[0.4, 0.4]	0.29	0.57	0.32	0.4	0.52	0
2OPT$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{5}$	0.63	0.12	[0.63, 0.63]	0.42	0.85	0.47	0.63	0.81	0
2OPT$_{10}$	0.45	0.13	[0.45, 0.45]	0.29	0.72	0.33	0.44	0.6	0
2OPT$_{15}$	0.38	0.13	[0.38, 0.38]	0.25	0.63	0.28	0.38	0.5	0
2OPT$_{20}$	0.34	0.12	[0.34, 0.34]	0.22	0.52	0.26	0.34	0.46	0
2OPT$_{25}$	0.33	0.12	[0.33, 0.33]	0.22	0.49	0.26	0.33	0.44	0
2OPT$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{5,B}$	0.71	0.09	[0.71, 0.71]	0.54	0.95	0.57	0.71	0.87	0
2OPT$_{10,B}$	0.57	0.1	[0.57, 0.57]	0.42	0.75	0.45	0.57	0.71	0
2OPT$_{15,B}$	0.46	0.1	[0.46, 0.46]	0.31	0.65	0.35	0.46	0.58	0
2OPT$_{20,B}$	0.45	0.1	[0.45, 0.45]	0.31	0.62	0.36	0.45	0.57	0
2OPT$_{25,B}$	0.33	0.12	[0.33, 0.33]	0.22	0.49	0.26	0.33	0.44	0
3OPT$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
3OPT$_{5}$	0.63	0.12	[0.63, 0.63]	0.42	0.85	0.47	0.62	0.8	0
3OPT$_{10}$	0.44	0.12	[0.44, 0.44]	0.29	0.71	0.33	0.44	0.58	0
3OPT$_{15}$	0.37	0.12	[0.37, 0.37]	0.25	0.55	0.28	0.37	0.48	0
3OPT$_{20}$	0.34	0.11	[0.34, 0.34]	0.22	0.47	0.26	0.33	0.45	0
3OPT$_{25}$	0.33	0.11	[0.33, 0.33]	0.22	0.46	0.25	0.33	0.43	0
3OPT$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
3OPT$_{5,B}$	0.71	0.09	[0.71, 0.71]	0.54	0.95	0.58	0.71	0.87	0
3OPT$_{10,B}$	0.56	0.09	[0.56, 0.56]	0.43	0.76	0.45	0.56	0.7	0
3OPT$_{15,B}$	0.46	0.1	[0.46, 0.46]	0.31	0.65	0.35	0.45	0.57	0
3OPT$_{20,B}$	0.45	0.1	[0.45, 0.45]	0.31	0.59	0.36	0.45	0.56	0
3OPT$_{25,B}$	0.33	0.11	[0.33, 0.33]	0.22	0.46	0.25	0.33	0.43	0
SA$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{5}$	0.64	0.12	[0.64, 0.64]	0.45	0.89	0.48	0.63	0.82	0
SA$_{10}$	0.45	0.13	[0.45, 0.45]	0.29	0.72	0.33	0.44	0.6	0
SA$_{15}$	0.38	0.13	[0.38, 0.38]	0.25	0.63	0.28	0.38	0.5	0
SA$_{20}$	0.34	0.12	[0.34, 0.34]	0.22	0.52	0.26	0.34	0.46	0
SA$_{25}$	0.33	0.12	[0.33, 0.33]	0.22	0.49	0.26	0.33	0.44	0
SA$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{5,B}$	0.72	0.09	[0.72, 0.72]	0.53	0.95	0.58	0.72	0.87	0
SA$_{10,B}$	0.57	0.1	[0.57, 0.57]	0.42	0.75	0.45	0.57	0.71	0
SA$_{15,B}$	0.46	0.1	[0.46, 0.46]	0.31	0.65	0.35	0.46	0.58	0
SA$_{20,B}$	0.45	0.1	[0.45, 0.45]	0.3	0.62	0.35	0.45	0.57	0
SA$_{25,B}$	0.33	0.12	[0.33, 0.33]	0.22	0.49	0.26	0.33	0.44	0
TS$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
TS$_{5}$	0.64	0.12	[0.64, 0.64]	0.43	0.88	0.47	0.63	0.82	0
TS$_{10}$	0.45	0.13	[0.45, 0.45]	0.29	0.71	0.33	0.44	0.59	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ts_{15}	0.37	0.13	[0.37, 0.37]	0.24	0.6	0.28	0.37	0.5	0
Ts_{20}	0.34	0.12	[0.34, 0.34]	0.21	0.49	0.26	0.34	0.46	0
Ts_{25}	0.33	0.11	[0.33, 0.33]	0.22	0.48	0.26	0.33	0.44	0
$\text{Ts}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{Ts}_{5,B}$	0.72	0.09	[0.72, 0.72]	0.54	0.95	0.58	0.72	0.87	0
$\text{Ts}_{10,B}$	0.57	0.09	[0.57, 0.57]	0.43	0.75	0.45	0.57	0.71	0
$\text{Ts}_{15,B}$	0.46	0.1	[0.46, 0.46]	0.31	0.63	0.35	0.46	0.58	0
$\text{Ts}_{20,B}$	0.45	0.1	[0.45, 0.45]	0.3	0.6	0.35	0.45	0.57	0
$\text{Ts}_{25,B}$	0.33	0.11	[0.33, 0.33]	0.22	0.48	0.26	0.33	0.44	0
Opt_{1}	1	0	[1, 1]	1	1	1	1	1	0
Opt_{5}	0.68	0.09	[0.68, 0.68]	0.5	0.87	0.54	0.68	0.83	0
Opt_{10}	0.49	0.12	[0.49, 0.49]	0.35	0.7	0.37	0.49	0.64	0
Opt_{15}	0.41	0.13	[0.41, 0.41]	0.28	0.67	0.3	0.4	0.54	0
Opt_{20}	0.35	0.12	[0.35, 0.35]	0.23	0.53	0.26	0.35	0.47	0
Opt_{25}	0.32	0.11	[0.32, 0.32]	0.21	0.46	0.25	0.32	0.42	0
$\text{Opt}_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
$\text{Opt}_{5,B}$	0.74	0.09	[0.74, 0.74]	0.54	0.97	0.6	0.74	0.89	0
$\text{Opt}_{10,B}$	0.58	0.09	[0.58, 0.58]	0.43	0.78	0.47	0.58	0.72	0
$\text{Opt}_{15,B}$	0.47	0.1	[0.47, 0.47]	0.31	0.64	0.37	0.47	0.58	0
$\text{Opt}_{20,B}$	0.46	0.1	[0.46, 0.46]	0.33	0.59	0.36	0.46	0.57	0
$\text{Opt}_{25,B}$	0.32	0.11	[0.32, 0.32]	0.21	0.46	0.25	0.32	0.42	0

Table A.39: Performance ratios of costs relative to the online version of an algorithm in the TSP.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
NN$_{1}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
NN$_{5}$	28.36	0.05	[28.27, 28.45]	24.1	34.1	25.4	28.3	31.75	0
NN$_{10}$	20.93	0.05	[20.87, 20.99]	18	24.8	18.6	20.9	23.65	0
NN$_{20}$	15.72	0.06	[15.66, 15.78]	13.1	18.1	13.8	15.7	17.7	0
NN$_{40}$	12.22	0.06	[12.17, 12.27]	10.2	15	10.6	12.2	14.05	0
NN$_{60}$	10.82	0.06	[10.78, 10.86]	8.7	13	9.5	10.8	12.3	0.05
NN$_{80}$	10.15	0.06	[10.11, 10.19]	8.5	12	8.8	10.1	11.6	0.14
NN$_{100}$	9.95	0.06	[9.91, 9.99]	8.5	11.9	8.7	9.9	11.3	0.2
NN$_{1,B}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
NN$_{5,B}$	36.26	0.05	[36.15, 36.37]	31.1	43.3	32.5	36.2	40.65	0
NN$_{10,B}$	27.89	0.05	[27.8, 27.98]	23.3	31.9	25	27.9	30.8	0
NN$_{20,B}$	20.74	0.05	[20.68, 20.8]	17.5	24.4	18.3	20.8	23.15	0
NN$_{40,B}$	16.28	0.05	[16.23, 16.33]	13.5	19	14.2	16.3	18.35	0
NN$_{60,B}$	13.58	0.06	[13.53, 13.63]	11	16.2	11.9	13.6	15.4	0
NN$_{80,B}$	12.92	0.06	[12.87, 12.97]	11	15.5	11.3	12.9	14.8	0.21
NN$_{100,B}$	9.95	0.06	[9.91, 9.99]	8.5	11.9	8.7	9.9	11.3	0
INS$_{1}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
INS$_{5}$	31.14	0.06	[31.02, 31.26]	25.8	37.4	26.8	31.1	35.6	0
INS$_{10}$	25.11	0.07	[25, 25.22]	19.8	31.3	21.25	25.1	29.5	0.01
INS$_{20}$	19.11	0.08	[19.02, 19.2]	14.7	24.7	15.7	19.1	23.1	0.01
INS$_{40}$	13.88	0.09	[13.8, 13.96]	10.9	18.3	11.5	13.8	17.4	0.01
INS$_{60}$	11.24	0.08	[11.18, 11.3]	8.9	14.6	9.4	11.2	13.45	0.02
INS$_{80}$	9.95	0.05	[9.92, 9.98]	8.5	11.8	8.8	9.9	11.1	0.06
INS$_{100}$	9.65	0.03	[9.63, 9.67]	8.5	10.4	8.9	9.7	10.3	0.23
INS$_{1,B}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
INS$_{5,B}$	36.37	0.05	[36.26, 36.48]	30.7	43.5	32.05	36.3	41.1	0
INS$_{10,B}$	27.73	0.05	[27.64, 27.82]	23.8	32.7	24.9	27.8	31	0
INS$_{20,B}$	20.31	0.04	[20.26, 20.36]	17.3	22.7	18.3	20.4	22.05	0
INS$_{40,B}$	15.66	0.04	[15.62, 15.7]	14.1	17.4	14.4	15.7	16.9	0
INS$_{60,B}$	11.94	0.04	[11.91, 11.97]	10.4	13.2	11	11.9	12.9	0
INS$_{80,B}$	11.51	0.04	[11.48, 11.54]	10	13.6	10.5	11.5	12.6	0.14
INS$_{100,B}$	9.65	0.03	[9.63, 9.67]	8.5	10.4	8.9	9.7	10.3	0
2OPT$_{1}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
2OPT$_{5}$	42.03	0.09	[41.8, 42.26]	32.2	55.6	33.75	42	50.75	0
2OPT$_{10}$	30.22	0.11	[30.01, 30.43]	21.1	44.6	22.95	30.15	38.9	0
2OPT$_{20}$	19.01	0.11	[18.88, 19.14]	14.1	30.5	15.3	18.8	24.4	0
2OPT$_{40}$	12.47	0.09	[12.4, 12.54]	9.7	16.9	10.4	12.3	15.65	0
2OPT$_{60}$	10.08	0.06	[10.04, 10.12]	8.5	12.8	8.9	10	11.75	0.01
2OPT$_{80}$	8.87	0.05	[8.84, 8.9]	7.7	10.4	8	8.9	9.9	0.01
2OPT$_{100}$	8.39	0.03	[8.37, 8.41]	7.6	9.3	7.7	8.4	9.05	0.08
2OPT$_{1,B}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
2OPT$_{5,B}$	38.31	0.05	[38.19, 38.43]	32.9	45	34.15	38.3	42.5	0
2OPT$_{10,B}$	27.61	0.04	[27.54, 27.68]	24.2	31.1	25.1	27.6	30.4	0
2OPT$_{20,B}$	19.39	0.04	[19.34, 19.44]	16.6	21.8	17.6	19.4	21.3	0
2OPT$_{40,B}$	14.78	0.04	[14.74, 14.82]	13.1	16.9	13.5	14.8	16.1	0
2OPT$_{60,B}$	11.95	0.04	[11.92, 11.98]	10.4	13.4	10.95	11.9	13	0
2OPT$_{80,B}$	11.51	0.04	[11.48, 11.54]	10.1	13.1	10.55	11.5	12.5	0.14
2OPT$_{100,B}$	8.39	0.03	[8.37, 8.41]	7.6	9.3	7.7	8.4	9.05	0
3OPT$_{1}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
3OPT$_{5}$	41.59	0.09	[41.36, 41.82]	31.2	53.9	33.15	41.75	50.75	0
3OPT$_{10}$	29.23	0.11	[29.03, 29.43]	20.3	40.8	22.4	29	37.95	0
3OPT$_{20}$	18.33	0.1	[18.22, 18.44]	13.9	26.1	14.8	18.2	23.3	0
3OPT$_{40}$	12.17	0.09	[12.1, 12.24]	10	17.4	10.2	12.1	15.15	0
3OPT$_{60}$	9.81	0.06	[9.77, 9.85]	8.3	12	8.8	9.7	11.4	0.01
3OPT$_{80}$	8.62	0.04	[8.6, 8.64]	7.7	9.9	7.9	8.6	9.5	0.01
3OPT$_{100}$	8.2	0.03	[8.18, 8.22]	7.4	9	7.6	8.2	8.8	0.05
3OPT$_{1,B}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
3OPT$_{5,B}$	38.23	0.05	[38.11, 38.35]	32.5	44.9	34.05	38.3	42.6	0
3OPT$_{10,B}$	27.42	0.04	[27.35, 27.49]	24.2	30.8	24.9	27.4	30.1	0
3OPT$_{20,B}$	19.07	0.04	[19.02, 19.12]	16.6	21.5	17.2	19.1	20.8	0
3OPT$_{40,B}$	14.46	0.04	[14.42, 14.5]	13	16.2	13.35	14.5	15.6	0
3OPT$_{60,B}$	11.66	0.03	[11.64, 11.68]	10.3	12.8	10.8	11.7	12.6	0
3OPT$_{80,B}$	11.26	0.03	[11.24, 11.28]	10	12.9	10.4	11.25	12.2	0.13
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3OPT$_{100,B}$	8.2	0.03	[8.18, 8.22]	7.4	9	7.6	8.2	8.8	0
SA$_{1}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
SA$_{5}$	41.63	0.09	[41.4, 41.86]	29.8	54.6	33.7	41.4	50.5	0
SA$_{10}$	30.19	0.11	[29.98, 30.4]	21.1	44.6	22.95	30	38.55	0
SA$_{20}$	19.02	0.11	[18.89, 19.15]	14.1	30.5	15.3	18.8	24.4	0
SA$_{40}$	12.47	0.09	[12.4, 12.54]	9.7	16.9	10.4	12.3	15.65	0
SA$_{60}$	10.08	0.06	[10.04, 10.12]	8.5	12.8	8.9	10	11.75	0.01
SA$_{80}$	8.87	0.05	[8.84, 8.9]	7.7	10.4	8	8.9	9.9	0.01
SA$_{100}$	8.39	0.03	[8.37, 8.41]	7.6	9.3	7.7	8.4	9.05	0.08
SA$_{1,B}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
SA$_{5,B}$	38.34	0.05	[38.22, 38.46]	32.9	44.7	34.25	38.3	42.65	0
SA$_{10,B}$	27.6	0.04	[27.53, 27.67]	24.2	31.1	25.1	27.6	30.4	0
SA$_{20,B}$	19.39	0.04	[19.34, 19.44]	16.6	21.8	17.6	19.4	21.3	0
SA$_{40,B}$	14.78	0.04	[14.74, 14.82]	13.1	16.9	13.5	14.8	16.1	0
SA$_{60,B}$	11.95	0.04	[11.92, 11.98]	10.4	13.4	10.95	11.9	13	0
SA$_{80,B}$	11.51	0.04	[11.48, 11.54]	10.1	13.1	10.55	11.5	12.5	0.14
SA$_{100,B}$	8.39	0.03	[8.37, 8.41]	7.6	9.3	7.7	8.4	9.05	0
TS$_{1}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
TS$_{5}$	41.62	0.09	[41.39, 41.85]	31.3	54.6	33.8	41.4	50.95	0
TS$_{10}$	29.84	0.11	[29.64, 30.04]	20.3	41.1	22.85	29.7	38	0
TS$_{20}$	18.88	0.11	[18.75, 19.01]	14.9	26.6	15.4	18.5	24.4	0
TS$_{40}$	12.46	0.09	[12.39, 12.53]	9.7	16.9	10.4	12.3	15.65	0
TS$_{60}$	10.08	0.06	[10.04, 10.12]	8.5	12.8	8.9	10	11.75	0.01
TS$_{80}$	8.87	0.05	[8.84, 8.9]	7.7	10.4	8	8.9	9.9	0.01
TS$_{100}$	8.39	0.03	[8.37, 8.41]	7.6	9.3	7.7	8.4	9.05	0.08
TS$_{1,B}$	53.11	0.05	[52.95, 53.27]	45.3	64.3	47.1	53.1	60.05	
TS$_{5,B}$	38.33	0.05	[38.21, 38.45]	32.9	44.7	34.15	38.3	42.7	0
TS$_{10,B}$	27.48	0.04	[27.41, 27.55]	24	30.8	24.9	27.5	30.15	0
TS$_{20,B}$	19.32	0.04	[19.27, 19.37]	16.6	21.7	17.6	19.3	21.1	0
TS$_{40,B}$	14.76	0.04	[14.72, 14.8]	13.1	16.4	13.5	14.8	16.1	0
TS$_{60,B}$	11.95	0.04	[11.92, 11.98]	10.4	13.4	10.95	11.9	13	0
TS$_{80,B}$	11.5	0.04	[11.47, 11.53]	10.1	13	10.5	11.5	12.5	0.13
TS$_{100,B}$	8.39	0.03	[8.37, 8.41]	7.6	9.3	7.7	8.4	9.05	0

Table A.40: Costs in the TSP.

Performance ratios of costs relative to 3OPT$_{100}$ for $n = 100$ (1000 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
NN$_{1}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
NN$_{5}$	3.46	0.06	[3.45, 3.47]	2.84	4.16	3.03	3.45	3.97	0
NN$_{10}$	2.56	0.06	[2.55, 2.57]	2.14	3	2.23	2.55	2.91	0
NN$_{20}$	1.92	0.06	[1.91, 1.93]	1.52	2.35	1.67	1.92	2.18	0
NN$_{40}$	1.49	0.06	[1.48, 1.5]	1.24	1.78	1.29	1.49	1.72	0
NN$_{60}$	1.32	0.06	[1.32, 1.32]	1.1	1.56	1.16	1.32	1.49	0
NN$_{80}$	1.24	0.06	[1.24, 1.24]	1.05	1.46	1.09	1.24	1.4	0
NN$_{100}$	1.21	0.05	[1.21, 1.21]	1.06	1.4	1.09	1.21	1.37	0
NN$_{1,B}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
NN$_{5,B}$	4.43	0.06	[4.41, 4.45]	3.74	5.28	3.9	4.42	5.09	0
NN$_{10,B}$	3.41	0.05	[3.4, 3.42]	2.77	4.04	3	3.4	3.85	0
NN$_{20,B}$	2.53	0.06	[2.52, 2.54]	2.08	3.03	2.21	2.53	2.88	0
NN$_{40,B}$	1.99	0.05	[1.98, 2]	1.66	2.38	1.73	1.99	2.26	0
NN$_{60,B}$	1.66	0.06	[1.65, 1.67]	1.38	2.01	1.45	1.65	1.88	0
NN$_{80,B}$	1.58	0.06	[1.57, 1.59]	1.31	1.85	1.39	1.57	1.8	0
NN$_{100,B}$	1.21	0.05	[1.21, 1.21]	1.06	1.4	1.09	1.21	1.37	0
INS$_{1}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
INS$_{5}$	3.8	0.07	[3.78, 3.82]	3.1	4.64	3.26	3.79	4.47	0
INS$_{10}$	3.07	0.07	[3.06, 3.08]	2.28	3.79	2.55	3.06	3.61	0
INS$_{20}$	2.33	0.09	[2.32, 2.34]	1.78	3.09	1.91	2.33	2.87	0
INS$_{40}$	1.69	0.09	[1.68, 1.7]	1.34	2.3	1.4	1.68	2.14	0
INS$_{60}$	1.37	0.08	[1.36, 1.38]	1.12	1.73	1.15	1.37	1.63	0
INS$_{80}$	1.21	0.05	[1.21, 1.21]	1.05	1.4	1.08	1.21	1.36	0
INS$_{100}$	1.18	0.03	[1.18, 1.18]	1.08	1.35	1.11	1.18	1.25	0
INS$_{1,B}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
INS$_{5,B}$	4.44	0.06	[4.42, 4.46]	3.7	5.42	3.87	4.43	5.1	0
INS$_{10,B}$	3.39	0.05	[3.38, 3.4]	2.83	4.03	2.98	3.38	3.83	0
INS$_{20,B}$	2.48	0.05	[2.47, 2.49]	2.07	3.07	2.19	2.48	2.75	0
INS$_{40,B}$	1.91	0.04	[1.91, 1.91]	1.64	2.2	1.74	1.91	2.08	0
INS$_{60,B}$	1.46	0.04	[1.46, 1.46]	1.29	1.69	1.33	1.46	1.59	0
INS$_{80,B}$	1.41	0.04	[1.41, 1.41]	1.19	1.62	1.27	1.41	1.54	0
INS$_{100,B}$	1.18	0.03	[1.18, 1.18]	1.08	1.35	1.11	1.18	1.25	0
2OPT$_{1}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
2OPT$_{5}$	5.13	0.1	[5.1, 5.16]	3.79	7.01	4.08	5.13	6.33	0
2OPT$_{10}$	3.69	0.12	[3.66, 3.72]	2.51	5.37	2.75	3.67	4.86	0
2OPT$_{20}$	2.32	0.11	[2.3, 2.34]	1.7	3.63	1.85	2.29	2.99	0
2OPT$_{40}$	1.52	0.09	[1.51, 1.53]	1.2	2.14	1.28	1.51	1.9	0
2OPT$_{60}$	1.23	0.06	[1.23, 1.23]	1.05	1.6	1.09	1.22	1.44	0
2OPT$_{80}$	1.08	0.04	[1.08, 1.08]	0.98	1.25	0.99	1.08	1.19	0.02
2OPT$_{100}$	1.02	0.02	[1.02, 1.02]	1	1.12	1	1.02	1.07	0
2OPT$_{1,B}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
2OPT$_{5,B}$	4.68	0.06	[4.66, 4.7]	3.93	5.58	4.1	4.68	5.32	0
2OPT$_{10,B}$	3.37	0.05	[3.36, 3.38]	2.86	4.2	3.01	3.37	3.79	0
2OPT$_{20,B}$	2.37	0.05	[2.36, 2.38]	2.06	2.92	2.13	2.36	2.65	0
2OPT$_{40,B}$	1.8	0.04	[1.8, 1.8]	1.6	2.06	1.63	1.8	1.99	0
2OPT$_{60,B}$	1.46	0.04	[1.46, 1.46]	1.24	1.64	1.33	1.46	1.59	0
2OPT$_{80,B}$	1.41	0.04	[1.41, 1.41]	1.23	1.59	1.29	1.4	1.54	0
2OPT$_{100,B}$	1.02	0.02	[1.02, 1.02]	1	1.12	1	1.02	1.07	0
3OPT$_{1}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
3OPT$_{5}$	5.08	0.1	[5.05, 5.11]	3.65	7.01	4.01	5.09	6.21	0
3OPT$_{10}$	3.57	0.12	[3.54, 3.6]	2.36	5.1	2.71	3.53	4.76	0
3OPT$_{20}$	2.24	0.1	[2.23, 2.25]	1.65	3.11	1.79	2.22	2.86	0
3OPT$_{40}$	1.49	0.09	[1.48, 1.5]	1.19	2.1	1.26	1.47	1.85	0
3OPT$_{60}$	1.2	0.06	[1.2, 1.2]	1.02	1.44	1.07	1.19	1.4	0
3OPT$_{80}$	1.05	0.04	[1.05, 1.05]	0.96	1.21	0.98	1.05	1.14	0.05
3OPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
3OPT$_{1,B}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
3OPT$_{5,B}$	4.67	0.06	[4.65, 4.69]	3.93	5.49	4.1	4.67	5.3	0
3OPT$_{10,B}$	3.35	0.05	[3.34, 3.36]	2.92	4.16	3	3.34	3.75	0
3OPT$_{20,B}$	2.33	0.05	[2.32, 2.34]	2.04	2.82	2.11	2.33	2.58	0
3OPT$_{40,B}$	1.77	0.04	[1.77, 1.77]	1.57	2.01	1.61	1.77	1.92	0
3OPT$_{60,B}$	1.42	0.04	[1.42, 1.42]	1.24	1.61	1.31	1.42	1.54	0
3OPT$_{80,B}$	1.38	0.04	[1.38, 1.38]	1.22	1.53	1.27	1.38	1.5	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of costs relative to 3OPT$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3OPT$_{100,B}$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{1}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
SA$_{5}$	5.09	0.1	[5.06, 5.12]	3.47	6.83	4.05	5.07	6.31	0
SA$_{10}$	3.69	0.12	[3.66, 3.72]	2.51	5.31	2.79	3.66	4.86	0
SA$_{20}$	2.32	0.11	[2.3, 2.34]	1.7	3.63	1.86	2.3	2.99	0
SA$_{40}$	1.52	0.09	[1.51, 1.53]	1.2	2.14	1.28	1.51	1.9	0
SA$_{60}$	1.23	0.06	[1.23, 1.23]	1.05	1.6	1.09	1.22	1.44	0
SA$_{80}$	1.08	0.04	[1.08, 1.08]	0.98	1.25	0.99	1.08	1.19	0.02
SA$_{100}$	1.02	0.02	[1.02, 1.02]	1	1.12	1	1.02	1.07	0
SA$_{1,B}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
SA$_{5,B}$	4.68	0.06	[4.66, 4.7]	3.93	5.49	4.12	4.69	5.34	0
SA$_{10,B}$	3.37	0.05	[3.36, 3.38]	2.86	4.2	3.01	3.37	3.79	0
SA$_{20,B}$	2.37	0.05	[2.36, 2.38]	2.06	2.92	2.13	2.36	2.65	0
SA$_{40,B}$	1.8	0.04	[1.8, 1.8]	1.6	2.06	1.63	1.8	1.99	0
SA$_{60,B}$	1.46	0.04	[1.46, 1.46]	1.24	1.64	1.33	1.46	1.59	0
SA$_{80,B}$	1.41	0.04	[1.41, 1.41]	1.23	1.59	1.29	1.4	1.54	0
SA$_{100,B}$	1.02	0.02	[1.02, 1.02]	1	1.12	1	1.02	1.07	0
TS$_{1}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
TS$_{5}$	5.08	0.1	[5.05, 5.11]	3.64	6.83	4.04	5.06	6.33	0
TS$_{10}$	3.65	0.12	[3.62, 3.68]	2.42	5.14	2.8	3.62	4.69	0
TS$_{20}$	2.3	0.11	[2.28, 2.32]	1.8	3.47	1.85	2.26	2.95	0
TS$_{40}$	1.52	0.09	[1.51, 1.53]	1.2	2.14	1.28	1.51	1.9	0
TS$_{60}$	1.23	0.06	[1.23, 1.23]	1.05	1.6	1.09	1.22	1.44	0
TS$_{80}$	1.08	0.04	[1.08, 1.08]	0.98	1.25	0.99	1.08	1.19	0.02
TS$_{100}$	1.02	0.02	[1.02, 1.02]	1	1.12	1	1.02	1.07	0
TS$_{1,B}$	6.49	0.06	[6.47, 6.51]	5.27	8.13	5.64	6.48	7.52	0
TS$_{5,B}$	4.68	0.06	[4.66, 4.7]	3.93	5.49	4.12	4.69	5.34	0
TS$_{10,B}$	3.36	0.05	[3.35, 3.37]	2.9	4.11	2.99	3.35	3.75	0
TS$_{20,B}$	2.36	0.05	[2.35, 2.37]	2.06	2.89	2.13	2.35	2.63	0
TS$_{40,B}$	1.8	0.04	[1.8, 1.8]	1.6	2.06	1.63	1.8	1.99	0
TS$_{60,B}$	1.46	0.04	[1.46, 1.46]	1.24	1.64	1.33	1.46	1.59	0
TS$_{80,B}$	1.4	0.04	[1.4, 1.4]	1.23	1.58	1.29	1.4	1.53	0
TS$_{100,B}$	1.02	0.02	[1.02, 1.02]	1	1.12	1	1.02	1.07	0

Table A.41: Performance ratios of costs relative to 3OPT$_{100}$ in the TSP.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
NN$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
NN$_{5}$	0.53	0.06	[0.53, 0.53]	0.43	0.63	0.47	0.53	0.6	0
NN$_{10}$	0.39	0.07	[0.39, 0.39]	0.32	0.48	0.34	0.39	0.46	0
NN$_{20}$	0.3	0.07	[0.3, 0.3]	0.24	0.37	0.25	0.3	0.35	0
NN$_{40}$	0.23	0.08	[0.23, 0.23]	0.18	0.3	0.19	0.23	0.28	0
NN$_{60}$	0.2	0.08	[0.2, 0.2]	0.16	0.27	0.17	0.2	0.25	0
NN$_{80}$	0.19	0.08	[0.19, 0.19]	0.15	0.26	0.16	0.19	0.23	0
NN$_{100}$	0.19	0.08	[0.19, 0.19]	0.14	0.24	0.16	0.19	0.22	0
NN$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
NN$_{5,B}$	0.68	0.05	[0.68, 0.68]	0.59	0.79	0.61	0.68	0.76	0
NN$_{10,B}$	0.53	0.06	[0.53, 0.53]	0.43	0.63	0.46	0.53	0.6	0
NN$_{20,B}$	0.39	0.07	[0.39, 0.39]	0.32	0.49	0.33	0.39	0.46	0
NN$_{40,B}$	0.31	0.07	[0.31, 0.31]	0.24	0.38	0.26	0.31	0.36	0
NN$_{60,B}$	0.26	0.08	[0.26, 0.26]	0.2	0.33	0.21	0.26	0.31	0
NN$_{80,B}$	0.24	0.08	[0.24, 0.24]	0.18	0.33	0.2	0.24	0.29	0
NN$_{100,B}$	0.19	0.08	[0.19, 0.19]	0.14	0.24	0.16	0.19	0.22	0
INS$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
INS$_{5}$	0.59	0.06	[0.59, 0.59]	0.49	0.7	0.51	0.59	0.67	0
INS$_{10}$	0.47	0.08	[0.47, 0.47]	0.36	0.63	0.4	0.47	0.56	0
INS$_{20}$	0.36	0.09	[0.36, 0.36]	0.26	0.5	0.29	0.36	0.44	0
INS$_{40}$	0.26	0.1	[0.26, 0.26]	0.2	0.36	0.21	0.26	0.33	0
INS$_{60}$	0.21	0.09	[0.21, 0.21]	0.15	0.29	0.17	0.21	0.26	0
INS$_{80}$	0.19	0.07	[0.19, 0.19]	0.15	0.23	0.16	0.19	0.22	0
INS$_{100}$	0.18	0.06	[0.18, 0.18]	0.14	0.22	0.16	0.18	0.21	0
INS$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
INS$_{5,B}$	0.69	0.05	[0.69, 0.69]	0.59	0.8	0.61	0.68	0.77	0
INS$_{10,B}$	0.52	0.05	[0.52, 0.52]	0.44	0.64	0.46	0.52	0.6	0
INS$_{20,B}$	0.38	0.06	[0.38, 0.38]	0.32	0.45	0.33	0.38	0.44	0
INS$_{40,B}$	0.3	0.06	[0.3, 0.3]	0.24	0.36	0.26	0.29	0.34	0
INS$_{60,B}$	0.23	0.06	[0.23, 0.23]	0.18	0.27	0.19	0.22	0.26	0
INS$_{80,B}$	0.22	0.07	[0.22, 0.22]	0.17	0.26	0.19	0.22	0.25	0
INS$_{100,B}$	0.18	0.06	[0.18, 0.18]	0.14	0.22	0.16	0.18	0.21	0
2OPT$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{5}$	0.79	0.07	[0.79, 0.79]	0.6	0.94	0.64	0.79	0.91	0
2OPT$_{10}$	0.57	0.11	[0.57, 0.57]	0.4	0.79	0.45	0.57	0.72	0
2OPT$_{20}$	0.36	0.12	[0.36, 0.36]	0.27	0.54	0.28	0.35	0.48	0
2OPT$_{40}$	0.24	0.1	[0.24, 0.24]	0.17	0.32	0.19	0.23	0.29	0
2OPT$_{60}$	0.19	0.08	[0.19, 0.19]	0.15	0.24	0.16	0.19	0.23	0
2OPT$_{80}$	0.17	0.07	[0.17, 0.17]	0.13	0.21	0.14	0.17	0.2	0
2OPT$_{100}$	0.16	0.06	[0.16, 0.16]	0.12	0.2	0.14	0.16	0.18	0
2OPT$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{5,B}$	0.72	0.05	[0.72, 0.72]	0.61	0.82	0.64	0.72	0.81	0
2OPT$_{10,B}$	0.52	0.05	[0.52, 0.52]	0.45	0.61	0.46	0.52	0.58	0
2OPT$_{20,B}$	0.37	0.06	[0.37, 0.37]	0.31	0.43	0.32	0.36	0.42	0
2OPT$_{40,B}$	0.28	0.06	[0.28, 0.28]	0.23	0.34	0.24	0.28	0.32	0
2OPT$_{60,B}$	0.23	0.06	[0.23, 0.23]	0.18	0.28	0.2	0.23	0.26	0
2OPT$_{80,B}$	0.22	0.06	[0.22, 0.22]	0.18	0.26	0.19	0.22	0.25	0
2OPT$_{100,B}$	0.16	0.06	[0.16, 0.16]	0.12	0.2	0.14	0.16	0.18	0
3OPT$_{1}$	1	0	[1, 1]	1	1	1	1	1	0
3OPT$_{5}$	0.78	0.07	[0.78, 0.78]	0.61	0.94	0.65	0.79	0.9	0
3OPT$_{10}$	0.55	0.11	[0.55, 0.55]	0.4	0.75	0.43	0.55	0.7	0
3OPT$_{20}$	0.35	0.11	[0.35, 0.35]	0.25	0.49	0.26	0.34	0.45	0
3OPT$_{40}$	0.23	0.1	[0.23, 0.23]	0.17	0.33	0.19	0.23	0.3	0
3OPT$_{60}$	0.19	0.08	[0.19, 0.19]	0.13	0.23	0.16	0.18	0.22	0
3OPT$_{80}$	0.16	0.07	[0.16, 0.16]	0.13	0.2	0.14	0.16	0.19	0
3OPT$_{100}$	0.15	0.07	[0.15, 0.15]	0.12	0.19	0.13	0.15	0.18	0
3OPT$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
3OPT$_{5,B}$	0.72	0.05	[0.72, 0.72]	0.6	0.82	0.64	0.72	0.81	0
3OPT$_{10,B}$	0.52	0.05	[0.52, 0.52]	0.44	0.61	0.46	0.52	0.58	0
3OPT$_{20,B}$	0.36	0.06	[0.36, 0.36]	0.3	0.42	0.31	0.36	0.41	0
3OPT$_{40,B}$	0.27	0.06	[0.27, 0.27]	0.22	0.33	0.24	0.27	0.31	0
3OPT$_{60,B}$	0.22	0.06	[0.22, 0.22]	0.18	0.27	0.19	0.22	0.25	0
3OPT$_{80,B}$	0.21	0.06	[0.21, 0.21]	0.18	0.26	0.18	0.21	0.24	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3OPT$_{100,B}$	0.15	0.07	[0.15, 0.15]	0.12	0.19	0.13	0.15	0.18	0
SA$_1$	1	0	[1, 1]	1	1	1	1	1	0
SA$_5$	0.78	0.08	[0.78, 0.78]	0.59	0.94	0.65	0.79	0.9	0
SA$_{10}$	0.57	0.11	[0.57, 0.57]	0.4	0.75	0.45	0.57	0.71	0
SA$_{20}$	0.36	0.12	[0.36, 0.36]	0.27	0.54	0.28	0.35	0.48	0
SA$_{40}$	0.24	0.1	[0.24, 0.24]	0.17	0.32	0.19	0.23	0.29	0
SA$_{60}$	0.19	0.08	[0.19, 0.19]	0.15	0.24	0.16	0.19	0.23	0
SA$_{80}$	0.17	0.07	[0.17, 0.17]	0.13	0.21	0.14	0.17	0.2	0
SA$_{100}$	0.16	0.06	[0.16, 0.16]	0.12	0.2	0.14	0.16	0.18	0
SA$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{5,B}$	0.72	0.05	[0.72, 0.72]	0.6	0.83	0.65	0.72	0.81	0
SA$_{10,B}$	0.52	0.05	[0.52, 0.52]	0.45	0.61	0.46	0.52	0.58	0
SA$_{20,B}$	0.37	0.06	[0.37, 0.37]	0.31	0.43	0.32	0.36	0.42	0
SA$_{40,B}$	0.28	0.06	[0.28, 0.28]	0.23	0.34	0.24	0.28	0.32	0
SA$_{60,B}$	0.23	0.06	[0.23, 0.23]	0.18	0.28	0.2	0.23	0.26	0
SA$_{80,B}$	0.22	0.06	[0.22, 0.22]	0.18	0.26	0.19	0.22	0.25	0
SA$_{100,B}$	0.16	0.06	[0.16, 0.16]	0.12	0.2	0.14	0.16	0.18	0
TS$_1$	1	0	[1, 1]	1	1	1	1	1	0
TS$_5$	0.78	0.08	[0.78, 0.78]	0.58	0.93	0.65	0.78	0.91	0
TS$_{10}$	0.56	0.11	[0.56, 0.56]	0.39	0.75	0.43	0.56	0.71	0
TS$_{20}$	0.36	0.11	[0.36, 0.36]	0.26	0.52	0.28	0.35	0.47	0
TS$_{40}$	0.24	0.1	[0.24, 0.24]	0.17	0.32	0.19	0.23	0.29	0
TS$_{60}$	0.19	0.08	[0.19, 0.19]	0.15	0.24	0.16	0.19	0.23	0
TS$_{80}$	0.17	0.07	[0.17, 0.17]	0.13	0.21	0.14	0.17	0.2	0
TS$_{100}$	0.16	0.06	[0.16, 0.16]	0.12	0.2	0.14	0.16	0.18	0
TS$_{1,B}$	1	0	[1, 1]	1	1	1	1	1	0
TS$_{5,B}$	0.72	0.05	[0.72, 0.72]	0.6	0.84	0.65	0.72	0.81	0
TS$_{10,B}$	0.52	0.05	[0.52, 0.52]	0.45	0.61	0.45	0.52	0.58	0
TS$_{20,B}$	0.36	0.06	[0.36, 0.36]	0.31	0.43	0.32	0.36	0.42	0
TS$_{40,B}$	0.28	0.06	[0.28, 0.28]	0.23	0.34	0.24	0.28	0.32	0
TS$_{60,B}$	0.23	0.06	[0.23, 0.23]	0.18	0.28	0.2	0.23	0.26	0
TS$_{80,B}$	0.22	0.06	[0.22, 0.22]	0.18	0.26	0.19	0.22	0.25	0
TS$_{100,B}$	0.16	0.06	[0.16, 0.16]	0.12	0.2	0.14	0.16	0.18	0

Table A.42: Performance ratios of costs relative to the online version of an algorithm in the TSP.

A.2.5 Online Scheduling with Lookahead

A.2.5.1 Single Machine Problem

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Spt$_{0}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	
Spt$_{5}$	1198.64	0.12	[1189.72, 1207.56]	734	1602	877.5	1196	1538.5	0
Spt$_{10}$	1089.86	0.12	[1081.75, 1097.97]	666	1482	797	1083	1419	0
Spt$_{25}$	825.93	0.12	[819.78, 832.08]	561	1136	617.5	819	1083.5	0
Spt$_{50}$	584.61	0.09	[581.35, 587.87]	426	753	466	581	711.5	0
Spt$_{100}$	516.11	0.1	[512.91, 519.31]	359	706	399.5	514	647	0
Opt$_{0}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	
Opt$_{5}$	1198.64	0.12	[1189.72, 1207.56]	734	1602	877.5	1196	1538.5	0
Opt$_{10}$	1089.86	0.12	[1081.75, 1097.97]	666	1482	797	1083	1419	0
Opt$_{25}$	825.93	0.12	[819.78, 832.08]	561	1136	617.5	819	1083.5	0
Opt$_{50}$	584.61	0.09	[581.35, 587.87]	426	753	466	581	711.5	0
Opt$_{100}$	516.11	0.1	[512.91, 519.31]	359	706	399.5	514	647	0

Table A.43: Costs in the single machine scheduling problem when immediate processing is allowed.

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Spt$_{0}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	
Spt$_{5}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	0
Spt$_{10}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	0
Spt$_{25}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	0
Spt$_{50}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	0
Spt$_{100}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	0
Opt$_{0}$	1317.65	0.11	[1308.66, 1326.64]	838	1727	994.5	1316.5	1660	
Opt$_{5}$	1317.64	0.11	[1308.65, 1326.63]	838	1727	994.5	1316.5	1660	0.01
Opt$_{10}$	1317.61	0.11	[1308.62, 1326.6]	838	1727	994.5	1316.5	1660	0
Opt$_{25}$	1317.61	0.11	[1308.62, 1326.6]	838	1727	994.5	1316.5	1660	0
Opt$_{50}$	1317.61	0.11	[1308.62, 1326.6]	838	1727	994.5	1316.5	1660	0
Opt$_{100}$	1317.61	0.11	[1308.62, 1326.6]	838	1727	994.5	1316.5	1660	0

Table A.44: Costs in the single machine scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to Opt for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Spt$_{0}$	2.58	0.15	[2.56, 2.6]	1.52	3.85	1.75	2.55	3.66	0
Spt$_{5}$	2.35	0.15	[2.33, 2.37]	1.33	3.55	1.57	2.32	3.37	0
Spt$_{10}$	2.13	0.16	[2.11, 2.15]	1.21	3.25	1.44	2.11	3.07	0
Spt$_{25}$	1.61	0.15	[1.6, 1.62]	1.07	2.5	1.18	1.59	2.3	0
Spt$_{50}$	1.14	0.07	[1.14, 1.14]	1	1.52	1.02	1.13	1.41	0
Spt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{0}$	2.58	0.15	[2.56, 2.6]	1.52	3.85	1.75	2.55	3.66	0
Opt$_{5}$	2.35	0.15	[2.33, 2.37]	1.33	3.55	1.57	2.32	3.37	0
Opt$_{10}$	2.13	0.16	[2.11, 2.15]	1.21	3.25	1.44	2.11	3.07	0
Opt$_{25}$	1.61	0.15	[1.6, 1.62]	1.07	2.5	1.18	1.59	2.3	0
Opt$_{50}$	1.14	0.07	[1.14, 1.14]	1	1.52	1.02	1.13	1.41	0
Opt$_{100}$	1	0	[1, 1]	1	1.01	1	1	1	0

Table A.45: Performance ratios of costs relative to Opt in the single machine scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to Opt for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Spt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_5$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_5$	1	0	[1, 1]	1	1.01	1	1	1	0.04
Opt$_{10}$	1	0	[1, 1]	1	1	1	1	1	0.04
Opt$_{25}$	1	0	[1, 1]	1	1	1	1	1	0.04
Opt$_{50}$	1	0	[1, 1]	1	1	1	1	1	0.04
Opt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0.04

Table A.46: Performance ratios of costs relative to Opt in the single machine scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Spt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_5$	0.91	0.01	[0.91, 0.91]	0.88	0.95	0.88	0.91	0.93	0
Spt$_{10}$	0.83	0.02	[0.83, 0.83]	0.77	0.91	0.78	0.83	0.86	0
Spt$_{25}$	0.63	0.05	[0.63, 0.63]	0.51	0.8	0.56	0.63	0.7	0
Spt$_{50}$	0.45	0.11	[0.45, 0.45]	0.32	0.68	0.35	0.44	0.6	0
Spt$_{100}$	0.4	0.15	[0.4, 0.4]	0.26	0.66	0.27	0.39	0.57	0
Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_5$	0.91	0.01	[0.91, 0.91]	0.88	0.95	0.88	0.91	0.93	0
Opt$_{10}$	0.83	0.02	[0.83, 0.83]	0.77	0.91	0.78	0.83	0.86	0
Opt$_{25}$	0.63	0.05	[0.63, 0.63]	0.51	0.8	0.56	0.63	0.7	0
Opt$_{50}$	0.45	0.11	[0.45, 0.45]	0.32	0.68	0.35	0.44	0.6	0
Opt$_{100}$	0.4	0.15	[0.4, 0.4]	0.26	0.66	0.27	0.39	0.57	0

Table A.47: Performance ratios of costs relative to the online version of an algorithm in the single machine scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Spt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_5$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
Spt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_5$	1	0	[1, 1]	1	1.01	1	1	1	0.01
Opt$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.48: Performance ratios of costs relative to the online version of an algorithm in the single machine scheduling problem when immediate processing is forbidden.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
SPT$_0$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	
SPT$_5$	8433.25	0.05	[8407.1, 8459.4]	7187	9659	7462	8414	9393.5	0
SPT$_{10}$	8323.98	0.05	[8298.17, 8349.79]	7066	9572	7360.5	8304	9297	0
SPT$_{25}$	8120.26	0.05	[8095.08, 8145.44]	6801	9428	7175	8107	9115	0
SPT$_{50}$	7931.43	0.05	[7906.84, 7956.02]	6625	9336	6962.5	7914.5	8933.5	0
SPT$_{100}$	7888.5	0.05	[7864.04, 7912.96]	6625	9307	6943.5	7861	8841.5	0

Table A.49: Costs in the single machine scheduling problem when immediate processing is allowed.

Costs $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
SPT$_0$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	
SPT$_5$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	0
SPT$_{10}$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	0
SPT$_{25}$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	0
SPT$_{50}$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	0
SPT$_{100}$	8638.98	0.05	[8612.19, 8665.77]	7435	9858	7643	8627.5	9647.5	0

Table A.50: Costs in the single machine scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to SPT$_{100}$ $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
SPT$_0$	1.1	0.02	[1.1, 1.1]	1.04	1.18	1.05	1.09	1.16	0
SPT$_5$	1.07	0.02	[1.07, 1.07]	1.03	1.12	1.04	1.07	1.11	0
SPT$_{10}$	1.06	0.01	[1.06, 1.06]	1.02	1.1	1.03	1.05	1.09	0
SPT$_{25}$	1.03	0.01	[1.03, 1.03]	1.01	1.06	1.01	1.03	1.05	0
SPT$_{50}$	1.01	0.01	[1.01, 1.01]	1	1.02	1	1.01	1.01	0
SPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.51: Performance ratios of costs relative to SPT$_{100}$ in the single machine scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to SPT$_{100}$ $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
SPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_5$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.52: Performance ratios of costs relative to SPT$_{100}$ in the single machine scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to online version $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
SPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_5$	0.98	0.01	[0.98, 0.98]	0.94	0.99	0.95	0.98	0.99	0
SPT$_{10}$	0.96	0.01	[0.96, 0.96]	0.9	0.99	0.92	0.97	0.99	0
SPT$_{25}$	0.94	0.02	[0.94, 0.94]	0.87	0.98	0.89	0.94	0.97	0
SPT$_{50}$	0.92	0.02	[0.92, 0.92]	0.85	0.97	0.87	0.92	0.96	0
SPT$_{100}$	0.91	0.02	[0.91, 0.91]	0.85	0.96	0.86	0.91	0.95	0

Table A.53: Performance ratios of costs relative to the online version of an algorithm in the single machine scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to online version $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
SPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_5$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
SPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.54: Performance ratios of costs relative to the online version of an algorithm in the single machine scheduling problem when immediate processing is forbidden.

A.2.5.2 Parallel Machines Problem

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
LPT$_0$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	
LPT$_5$	93.79	0.04	[93.56, 94.02]	73	99	80	95	98	0
LPT$_{10}$	88.79	0.04	[88.57, 89.01]	68	94	75	90	93	0
LPT$_{25}$	73.79	0.05	[73.56, 74.02]	53	79	60	75	78	0
LPT$_{50}$	48.79	0.08	[48.55, 49.03]	28	54	35	50	53	0
LPT$_{100}$	25.23	0.08	[25.1, 25.36]	17	31	20	25	30	0
OPT$_0$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	
OPT$_5$	93.79	0.04	[93.56, 94.02]	73	99	80	95	98	0
OPT$_{10}$	88.79	0.04	[88.57, 89.01]	68	94	75	90	93	0
OPT$_{25}$	73.79	0.05	[73.56, 74.02]	53	79	60	75	78	0
OPT$_{50}$	48.79	0.08	[48.55, 49.03]	28	54	35	50	53	0
OPT$_{100}$	25.23	0.08	[25.1, 25.36]	17	31	20	25	30	0

Table A.55: Costs in the parallel machines scheduling problem when immediate processing is allowed.

Costs for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
LPT$_0$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	
LPT$_5$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
LPT$_{10}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
LPT$_{25}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
LPT$_{50}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
LPT$_{100}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
OPT$_0$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	
OPT$_5$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
OPT$_{10}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
OPT$_{25}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
OPT$_{50}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0
OPT$_{100}$	98.79	0.04	[98.54, 99.04]	78	104	85	100	103	0

Table A.56: Costs in the parallel machines scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to OPT for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
LPT$_0$	3.94	0.09	[3.92, 3.96]	2.9	5.76	3.19	3.92	4.83	0
LPT$_5$	3.74	0.09	[3.72, 3.76]	2.73	5.47	3	3.73	4.6	0
LPT$_{10}$	3.54	0.09	[3.52, 3.56]	2.57	5.18	2.82	3.54	4.36	0
LPT$_{25}$	2.94	0.1	[2.92, 2.96]	2.07	4.29	2.27	2.96	3.64	0
LPT$_{50}$	1.95	0.11	[1.94, 1.96]	1.12	2.82	1.32	1.96	2.44	0
LPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
OPT$_0$	3.94	0.09	[3.92, 3.96]	2.9	5.76	3.19	3.92	4.83	0
OPT$_5$	3.74	0.09	[3.72, 3.76]	2.73	5.47	3	3.73	4.6	0
OPT$_{10}$	3.54	0.09	[3.52, 3.56]	2.57	5.18	2.82	3.54	4.36	0
OPT$_{25}$	2.94	0.1	[2.92, 2.96]	2.07	4.29	2.27	2.96	3.64	0
OPT$_{50}$	1.95	0.11	[1.94, 1.96]	1.12	2.82	1.32	1.96	2.44	0
OPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.57: Performance ratios of costs relative to OPT in the parallel machines scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to Opt for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Lpt$_0$	1	0	[1, 1]	1	1.01	1	1	1	0
Lpt$_5$	1	0	[1, 1]	1	1.01	1	1	1	0
Lpt$_{10}$	1	0	[1, 1]	1	1.01	1	1	1	0
Lpt$_{25}$	1	0	[1, 1]	1	1.01	1	1	1	0
Lpt$_{50}$	1	0	[1, 1]	1	1.01	1	1	1	0
Lpt$_{100}$	1	0	[1, 1]	1	1.01	1	1	1	0
Opt$_0$	1	0	[1, 1]	1	1.01	1	1	1	0
Opt$_5$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.58: Performance ratios of costs relative to Opt in the parallel machines scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Lpt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Lpt$_5$	0.95	0	[0.95, 0.95]	0.94	0.95	0.94	0.95	0.95	0
Lpt$_{10}$	0.9	0	[0.9, 0.9]	0.87	0.9	0.88	0.9	0.9	0
Lpt$_{25}$	0.75	0.01	[0.75, 0.75]	0.68	0.76	0.71	0.75	0.76	0
Lpt$_{50}$	0.49	0.04	[0.49, 0.49]	0.36	0.52	0.41	0.5	0.51	0
Lpt$_{100}$	0.26	0.09	[0.26, 0.26]	0.17	0.34	0.21	0.25	0.31	0
Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_5$	0.95	0	[0.95, 0.95]	0.94	0.95	0.94	0.95	0.95	0
Opt$_{10}$	0.9	0	[0.9, 0.9]	0.87	0.9	0.88	0.9	0.9	0
Opt$_{25}$	0.75	0.01	[0.75, 0.75]	0.68	0.76	0.71	0.75	0.76	0
Opt$_{50}$	0.49	0.04	[0.49, 0.49]	0.36	0.52	0.41	0.5	0.51	0
Opt$_{100}$	0.26	0.09	[0.26, 0.26]	0.17	0.34	0.21	0.25	0.31	0

Table A.59: Performance ratios of costs relative to the online version of an algorithm in the parallel machines scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to online version for $n = 25$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Lpt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Lpt$_5$	1	0	[1, 1]	1	1	1	1	1	0
Lpt$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
Lpt$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
Lpt$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
Lpt$_{100}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_5$	1	0	[1, 1]	0.99	1	1	1	1	0
Opt$_{10}$	1	0	[1, 1]	0.99	1	1	1	1	0
Opt$_{25}$	1	0	[1, 1]	0.99	1	1	1	1	0
Opt$_{50}$	1	0	[1, 1]	0.99	1	1	1	1	0
Opt$_{100}$	1	0	[1, 1]	0.99	1	1	1	1	0

Table A.60: Performance ratios of costs relative to the online version of an algorithm in the parallel machines scheduling problem when immediate processing is forbidden.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
LPT$_0$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	
LPT$_5$	103.01	0.04	[102.75, 103.27]	95	118	96	103	113	0
LPT$_{10}$	100.73	0.04	[100.48, 100.98]	92	113	92	101	110	0
LPT$_{25}$	100.14	0.04	[99.89, 100.39]	87	113	90	100	109	0
LPT$_{50}$	100.14	0.04	[99.89, 100.39]	87	113	90	100	109	0
LPT$_{100}$	100.14	0.04	[99.89, 100.39]	87	113	90	100	109	0

Table A.61: Costs in the parallel machines scheduling problem when immediate processing is allowed.

Costs for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
LPT$_0$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	
LPT$_5$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	0
LPT$_{10}$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	0
LPT$_{25}$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	0
LPT$_{50}$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	0
LPT$_{100}$	107.35	0.04	[107.08, 107.62]	99	123	101	107	118	0

Table A.62: Costs in the parallel machines scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to LPT$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
LPT$_0$	1.07	0.04	[1.07, 1.07]	1	1.22	1	1.07	1.18	0
LPT$_5$	1.03	0.03	[1.03, 1.03]	1	1.16	1	1.02	1.12	0
LPT$_{10}$	1.01	0.02	[1.01, 1.01]	1	1.11	1	1	1.07	0
LPT$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.63: Performance ratios of costs relative to LPT$_{100}$ in the parallel machines scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to LPT$_{100}$ for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
LPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_5$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.64: Performance ratios of costs relative to LPT$_{100}$ in the parallel machines scheduling problem when immediate processing is forbidden.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
LPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_5$	0.96	0.01	[0.96, 0.96]	0.95	1	0.95	0.95	1	0
LPT$_{10}$	0.94	0.03	[0.94, 0.94]	0.9	1	0.9	0.93	1	0
LPT$_{25}$	0.93	0.04	[0.93, 0.93]	0.82	1	0.85	0.93	1	0
LPT$_{50}$	0.93	0.04	[0.93, 0.93]	0.82	1	0.85	0.93	1	0
LPT$_{100}$	0.93	0.04	[0.93, 0.93]	0.82	1	0.85	0.93	1	0

Table A.65: Performance ratios of costs relative to the online version of an algorithm in the parallel machines scheduling problem when immediate processing is allowed.

Performance ratios of costs relative to online version for $n = 100$ (1000 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
LPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_5$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{10}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{25}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{50}$	1	0	[1, 1]	1	1	1	1	1	0
LPT$_{100}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.66: Performance ratios of costs relative to the online version of an algorithm in the parallel machines scheduling problem when immediate processing is forbidden.

A.3 Numerical Results from Chapter 6

This section contains a detailed statistical summary of the numerical results gathered during the experimental analysis in Chapter 6. The presentation is carried out in the same form as described at the beginning of Appendix A.2. Because of space restrictions, we omit results for lookahead durations $D \in \{180, 240, 300, 420, 480, 540\}$ minutes and only list results for $D \in \{0, 60, 120, 360, 600\}$ minutes.

A.3.1 Online Order Picking with Lookahead

Makespan for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Prio,Ret$_{0}$	624.81	0.05	[615.97, 633.65]	608.2	742.6	608.2	614.9	742.6	
Prio,Ret$_{60}$	562.51	0.03	[557.74, 567.28]	548.9	680	548.9	556.7	680	0.02
Prio,Ret$_{120}$	534.27	0.07	[523.69, 544.85]	493.6	725.7	493.6	524.3	725.7	0.08
Prio,Ret$_{360}$	519.58	0.06	[510.76, 528.4]	445.3	664.9	445.3	517.6	664.9	0.43
Prio,Ret$_{600}$	519.9	0.06	[511.08, 528.72]	443.4	664.9	443.4	519.2	664.9	0.45
Prio,S$_{0}$	619.17	0.04	[612.16, 626.18]	607.9	725.7	607.9	614.1	725.7	
Prio,S$_{60}$	567.5	0.06	[557.87, 577.13]	547.7	698.5	547.7	555.6	698.5	0.06
Prio,S$_{120}$	534.15	0.09	[520.55, 547.75]	491.2	725.7	491.2	526.2	725.7	0.12
Prio,S$_{360}$	505.79	0.09	[492.91, 518.67]	430.3	664.9	430.3	502.1	664.9	0.31
Prio,S$_{600}$	507.94	0.09	[495.01, 520.87]	425.7	664.9	425.7	505.5	664.9	0.41
Prio,Gap$_{0}$	622.08	0.04	[615.04, 629.12]	606.9	742.6	606.9	615.2	742.6	
Prio,Gap$_{60}$	568.84	0.06	[559.18, 578.5]	550.4	693	550.4	556.5	693	0.08
Prio,Gap$_{120}$	531.74	0.08	[519.71, 543.77]	490.2	725.7	490.2	524.4	725.7	0.08
Prio,Gap$_{360}$	513.43	0.08	[501.81, 525.05]	437.3	664.9	437.3	511.9	664.9	0.31
Prio,Gap$_{600}$	513.83	0.08	[502.2, 525.46]	434.9	664.9	434.9	510.1	664.9	0.39
Prio,Opt$_{0}$	619.94	0.04	[612.92, 626.96]	607.6	742.6	607.6	613.3	742.6	
Prio,Opt$_{60}$	558.79	0.03	[554.05, 563.53]	547.1	657.6	547.1	553.6	657.6	0.02
Prio,Opt$_{120}$	531.51	0.09	[517.98, 545.04]	488	725.7	488	522.8	725.7	0.08
Prio,Opt$_{360}$	495.92	0.11	[480.49, 511.35]	419.2	669.5	419.2	486	669.5	0.37
Prio,Opt$_{600}$	500.44	0.11	[484.87, 516.01]	414.9	669.5	414.9	495.8	669.5	0.45
Seed,Ret$_{0}$	616.8	0.03	[611.57, 622.03]	608.5	725.7	608.5	614.4	725.7	
Seed,Ret$_{60}$	563.42	0.05	[555.45, 571.39]	546	712.5	546	555.8	712.5	0.04
Seed,Ret$_{120}$	535.86	0.09	[522.22, 549.5]	490.2	725.7	490.2	527.5	725.7	0.12
Seed,Ret$_{360}$	493.33	0.12	[476.58, 510.08]	408.4	669.5	408.4	486.3	669.5	0.2
Seed,Ret$_{600}$	491.71	0.1	[477.8, 505.62]	408.6	664.9	408.6	490	664.9	0.35
Seed,S$_{0}$	620.27	0.04	[613.25, 627.29]	609.5	725.7	609.5	613.1	725.7	
Seed,S$_{60}$	569.04	0.07	[557.77, 580.31]	546.2	712.5	546.2	555.1	712.5	0.08
Seed,S$_{120}$	528.66	0.08	[516.7, 540.62]	488.3	725.7	488.3	521.4	725.7	0.04
Seed,S$_{360}$	465.52	0.12	[449.72, 481.32]	357.6	664.9	357.6	462.8	664.9	0.43
Seed,S$_{600}$	470.2	0.13	[452.91, 487.49]	373.2	669.5	373.2	462.8	669.5	0.27
Seed,Gap$_{0}$	619.82	0.03	[614.56, 625.08]	607.4	725.7	607.4	614.4	725.7	
Seed,Gap$_{60}$	562.44	0.04	[556.08, 568.8]	548.4	698.5	548.4	556	698.5	0.04
Seed,Gap$_{120}$	533.08	0.08	[521.02, 545.14]	491.3	725.7	491.3	520.6	725.7	0.08
Seed,Gap$_{360}$	514.45	0.09	[501.35, 527.55]	434.4	669.5	434.4	513.7	669.5	0.35
Seed,Gap$_{600}$	508.14	0.08	[496.64, 519.64]	437.4	664.9	437.4	509.7	664.9	0.33
Seed,Opt$_{0}$	619.73	0.04	[612.72, 626.74]	606.4	725.7	606.4	613	725.7	
Seed,Opt$_{60}$	557.79	0.03	[553.06, 562.52]	546	680	546	553.7	680	0
Seed,Opt$_{120}$	528.99	0.08	[517.02, 540.96]	488.8	725.7	488.8	525.1	725.7	0.08
Seed,Opt$_{360}$	468.29	0.13	[451.07, 485.51]	357.4	664.9	357.4	467.8	664.9	0.33
Seed,Opt$_{600}$	463.41	0.12	[447.68, 479.14]	374.5	664.9	374.5	458.8	664.9	0.41
Svg,Ret$_{0}$	619.45	0.04	[612.44, 626.46]	607.5	741	607.5	614.7	741	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Makespan for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Svg,Ret$_{60}$	561.11	0.03	[556.35, 565.87]	548.2	680	548.2	555.9	680	0.02
Svg,Ret$_{120}$	532.02	0.08	[519.98, 544.06]	492.7	725.7	492.7	527	725.7	0.06
Svg,Ret$_{360}$	507.09	0.1	[492.74, 521.44]	438.8	669.5	438.8	496.7	669.5	0.45
Svg,Ret$_{600}$	514.56	0.1	[500, 529.12]	431.3	669.5	431.3	511.6	669.5	0.37
Svg,S$_{0}$	615.81	0.03	[610.58, 621.04]	607.5	725.7	607.5	613.5	725.7	
Svg,S$_{60}$	557.53	0.02	[554.38, 560.68]	548.1	623.2	548.1	554.8	623.2	0
Svg,S$_{120}$	531.09	0.08	[519.07, 543.11]	488.7	725.7	488.7	525.4	725.7	0.08
Svg,S$_{360}$	492.56	0.12	[475.84, 509.28]	409.4	669.5	409.4	487.8	669.5	0.27
Svg,S$_{600}$	494.32	0.11	[478.94, 509.7]	415.1	669.5	415.1	483.3	669.5	0.24
Svg,Gap$_{0}$	621.23	0.04	[614.2, 628.26]	609.2	742.6	609.2	614.4	742.6	
Svg,Gap$_{60}$	560.55	0.04	[554.21, 566.89]	549.3	712.5	549.3	555.7	712.5	0
Svg,Gap$_{120}$	531.5	0.08	[519.47, 543.53]	491.8	725.7	491.8	526.3	725.7	0.08
Svg,Gap$_{360}$	508.96	0.08	[497.44, 520.48]	430.5	664.9	430.5	505.8	664.9	0.47
Svg,Gap$_{600}$	514.31	0.08	[502.67, 525.95]	429.4	664.9	429.4	517.9	664.9	0.47
Svg,Opt$_{0}$	622.92	0.05	[614.11, 631.73]	607.6	742.6	607.6	613.8	742.6	
Svg,Opt$_{60}$	558.43	0.04	[552.11, 564.75]	548.1	698.5	548.1	553.3	698.5	0.02
Svg,Opt$_{120}$	531.93	0.08	[519.89, 543.97]	491.6	725.7	491.6	525.5	725.7	0.12
Svg,Opt$_{360}$	489.01	0.12	[472.41, 505.61]	409.7	669.5	409.7	483.8	669.5	0.31
Svg,Opt$_{600}$	492.16	0.1	[478.24, 506.08]	412.4	664.9	412.4	489.4	664.9	0.39
Ls,Ret$_{0}$	620.74	0.04	[613.72, 627.76]	608.6	741	608.6	614.8	741	
Ls,Ret$_{60}$	562.53	0.05	[554.57, 570.49]	546.2	693	546.2	556.3	693	0.04
Ls,Ret$_{120}$	530	0.08	[518, 542]	490.8	725.7	490.8	523.4	725.7	0.06
Ls,Ret$_{360}$	491.63	0.12	[474.94, 508.32]	406.2	669.5	406.2	481.4	669.5	0.29
Ls,Ret$_{600}$	492.09	0.12	[475.38, 508.8]	407.8	669.5	407.8	487.9	669.5	0.2
Ls,S$_{0}$	621.5	0.05	[612.71, 630.29]	608.7	741	608.7	613.3	741	
Ls,S$_{60}$	557.83	0.03	[553.1, 562.56]	547.2	657.6	547.2	554.1	657.6	0.02
Ls,S$_{120}$	528.88	0.08	[516.91, 540.85]	488.6	725.7	488.6	525.7	725.7	0.06
Ls,S$_{360}$	467.4	0.12	[451.53, 483.27]	379	664.9	379	463.3	664.9	0.33
Ls,S$_{600}$	464.93	0.12	[449.15, 480.71]	357.7	664.9	357.7	462.8	664.9	0.33
Ls,Gap$_{0}$	622.76	0.05	[613.95, 631.57]	608.6	741	608.6	614.3	741	
Ls,Gap$_{60}$	562.47	0.05	[554.51, 570.43]	547.8	712.5	547.8	554.5	712.5	0
Ls,Gap$_{120}$	532.31	0.09	[518.76, 545.86]	491.6	725.7	491.6	523.3	725.7	0.08
Ls,Gap$_{360}$	484.77	0.11	[469.68, 499.86]	404.6	664.9	404.6	480.5	664.9	0.24
Ls,Gap$_{600}$	487.55	0.11	[472.38, 502.72]	403.5	664.9	403.5	481	664.9	0.31
Ls,Opt$_{0}$	617.52	0.03	[612.28, 622.76]	607.6	725.7	607.6	613	725.7	
Ls,Opt$_{60}$	562.69	0.05	[554.73, 570.65]	548.4	712.5	548.4	554.4	712.5	0.04
Ls,Opt$_{120}$	528.22	0.08	[516.27, 540.17]	489.7	725.7	489.7	520.4	725.7	0.08
Ls,Opt$_{360}$	467.18	0.12	[451.32, 483.04]	373.3	664.9	373.3	462.7	664.9	0.31
Ls,Opt$_{600}$	472.86	0.12	[456.81, 488.91]	392.7	664.9	392.7	472	664.9	0.2
Ts,Ret$_{0}$	619.58	0.03	[614.32, 624.84]	608.2	725.7	608.2	615	725.7	
Ts,Ret$_{60}$	567.43	0.06	[557.8, 577.06]	547.6	712.5	547.6	556.3	712.5	0.08
Ts,Ret$_{120}$	530.88	0.08	[518.87, 542.89]	493.8	725.7	493.8	523.4	725.7	0.1
Ts,Ret$_{360}$	489.94	0.12	[473.31, 506.57]	411.7	669.5	411.7	484.4	669.5	0.29
Ts,Ret$_{600}$	487.26	0.1	[473.48, 501.04]	411	664.9	411	485.7	664.9	0.33
Ts,S$_{0}$	622.39	0.05	[613.59, 631.19]	608.5	742.6	608.5	613.6	742.6	
Ts,S$_{60}$	559.28	0.04	[552.95, 565.61]	549	698.5	549	554.8	698.5	0
Ts,S$_{120}$	527.52	0.08	[515.58, 539.46]	488.4	725.7	488.4	523.8	725.7	0.06
Ts,S$_{360}$	469.72	0.13	[452.45, 486.99]	371.6	669.5	371.6	462.8	669.5	0.29
Ts,S$_{600}$	466.62	0.12	[450.78, 482.46]	372.1	664.9	372.1	462.8	664.9	0.27
Ts,Gap$_{0}$	621.1	0.04	[614.07, 628.13]	608.3	742.6	608.3	614.4	742.6	
Ts,Gap$_{60}$	563.89	0.06	[554.32, 573.46]	548.1	712.5	548.1	556.4	712.5	0.06
Ts,Gap$_{120}$	532.84	0.09	[519.27, 546.41]	488.7	725.7	488.7	525.1	725.7	0.1
Ts,Gap$_{360}$	489.69	0.12	[473.07, 506.31]	405.7	669.5	405.7	481.6	669.5	0.39
Ts,Gap$_{600}$	488.31	0.12	[471.73, 504.89]	414.3	669.5	414.3	476.4	669.5	0.35
Ts,Opt$_{0}$	614.88	0.03	[609.66, 620.1]	606.6	725.7	606.6	612.5	725.7	
Ts,Opt$_{60}$	561.7	0.05	[553.75, 569.65]	547.4	712.5	547.4	554.2	712.5	0.04
Ts,Opt$_{120}$	531.77	0.09	[518.23, 545.31]	488	725.7	488	526.3	725.7	0.08
Ts,Opt$_{360}$	474.58	0.14	[455.78, 493.38]	357.4	669.5	357.4	469.1	669.5	0.31
Ts,Opt$_{600}$	469.68	0.13	[452.41, 486.95]	376.9	669.5	376.9	464.9	669.5	0.31
Opt,Opt$_{0}$	615.99	0.06	[605.53, 626.45]	507.7	741	507.7	613.3	741	
Opt,Opt$_{60}$	566.2	0.06	[556.59, 575.81]	507.7	712.5	507.7	554.5	712.5	0.08
Opt,Opt$_{120}$	533.08	0.08	[521.02, 545.14]	489.9	725.7	489.9	525	725.7	0.14
Opt,Opt$_{360}$	513.23	0.09	[500.16, 526.3]	434.4	669.5	434.4	510.8	669.5	0.31
Opt,Opt$_{600}$	507.97	0.08	[496.47, 519.47]	436.3	664.9	436.3	509.7	664.9	0.27

Table A.67: Makespans in the order picking system.

Performance ratios of makespan relative to Ts,Opt_{600} for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Prio,Ret_{0}	1.35	0.13	[1.3, 1.4]	0.92	1.66	0.92	1.39	1.66	0.04
Prio,Ret_{60}	1.22	0.13	[1.18, 1.26]	0.83	1.67	0.83	1.2	1.67	0.04
Prio,Ret_{120}	1.15	0.14	[1.1, 1.2]	0.78	1.66	0.78	1.13	1.66	0.02
Prio,Ret_{360}	1.12	0.12	[1.08, 1.16]	0.77	1.39	0.77	1.06	1.39	0.02
Prio,Ret_{600}	1.12	0.12	[1.08, 1.16]	0.77	1.39	0.77	1.08	1.39	0.02
Prio,S_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.67	0.92	1.32	1.67	0.04
Prio,S_{60}	1.23	0.15	[1.18, 1.28]	0.83	1.68	0.83	1.2	1.68	0.04
Prio,S_{120}	1.15	0.13	[1.11, 1.19]	0.99	1.66	0.99	1.12	1.66	0.02
Prio,S_{360}	1.09	0.14	[1.05, 1.13]	0.73	1.61	0.73	1.02	1.61	0.04
Prio,S_{600}	1.1	0.14	[1.06, 1.14]	0.74	1.61	0.74	1.02	1.61	0.06
Prio,Gap_{0}	1.35	0.12	[1.3, 1.4]	0.92	1.66	0.92	1.37	1.66	0.04
Prio,Gap_{60}	1.23	0.14	[1.18, 1.28]	0.82	1.68	0.82	1.2	1.68	0.04
Prio,Gap_{120}	1.15	0.14	[1.1, 1.2]	0.76	1.66	0.76	1.11	1.66	0.02
Prio,Gap_{360}	1.11	0.13	[1.07, 1.15]	0.76	1.61	0.76	1.05	1.61	0.02
Prio,Gap_{600}	1.11	0.13	[1.07, 1.15]	0.76	1.61	0.76	1.04	1.61	0.04
Prio,Opt_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.36	1.66	0.04
Prio,Opt_{60}	1.21	0.13	[1.17, 1.25]	0.85	1.54	0.85	1.2	1.54	0.04
Prio,Opt_{120}	1.15	0.14	[1.1, 1.2]	0.98	1.66	0.98	1.07	1.66	0.02
Prio,Opt_{360}	1.06	0.12	[1.02, 1.1]	0.99	1.61	0.99	1.02	1.61	0.1
Prio,Opt_{600}	1.08	0.12	[1.04, 1.12]	0.99	1.61	0.99	1.02	1.61	0.04
Seed,Ret_{0}	1.33	0.13	[1.28, 1.38]	0.92	1.66	0.92	1.33	1.66	0.04
Seed,Ret_{60}	1.22	0.13	[1.18, 1.26]	0.85	1.53	0.85	1.2	1.53	0.04
Seed,Ret_{120}	1.16	0.13	[1.12, 1.2]	0.99	1.66	0.99	1.12	1.66	0.02
Seed,Ret_{360}	1.06	0.12	[1.02, 1.1]	0.98	1.61	0.98	1.01	1.61	0.06
Seed,Ret_{600}	1.06	0.13	[1.02, 1.1]	0.73	1.61	0.73	1.01	1.61	0.06
Seed,S_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.68	0.92	1.36	1.68	0.04
Seed,S_{60}	1.23	0.14	[1.18, 1.28]	0.85	1.67	0.85	1.23	1.67	0.04
Seed,S_{120}	1.14	0.14	[1.09, 1.19]	0.75	1.66	0.75	1.07	1.66	0.04
Seed,S_{360}	0.99	0.05	[0.98, 1]	0.73	1.03	0.73	1	1.03	0.29
Seed,S_{600}	1	0.01	[1, 1]	0.98	1.03	0.98	1	1.03	0.31
Seed,Gap_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.33	1.66	0.04
Seed,Gap_{60}	1.22	0.13	[1.18, 1.26]	0.83	1.5	0.83	1.2	1.5	0.04
Seed,Gap_{120}	1.15	0.13	[1.11, 1.19]	1	1.66	1	1.08	1.66	0
Seed,Gap_{360}	1.11	0.13	[1.07, 1.15]	0.99	1.61	0.99	1.02	1.61	0.02
Seed,Gap_{600}	1.1	0.13	[1.06, 1.14]	0.74	1.61	0.74	1.03	1.61	0.04
Seed,Opt_{0}	1.34	0.13	[1.29, 1.39]	0.91	1.68	0.91	1.32	1.68	0.04
Seed,Opt_{60}	1.21	0.13	[1.17, 1.25]	0.82	1.67	0.82	1.19	1.67	0.04
Seed,Opt_{120}	1.14	0.14	[1.09, 1.19]	0.81	1.66	0.81	1.11	1.66	0.04
Seed,Opt_{360}	1	0.11	[0.97, 1.03]	0.74	1.61	0.74	1	1.61	0.39
Seed,Opt_{600}	0.99	0.05	[0.98, 1]	0.73	1.07	0.73	1	1.07	0.43
Svg,Ret_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.32	1.66	0.04
Svg,Ret_{60}	1.21	0.13	[1.17, 1.25]	0.83	1.67	0.83	1.2	1.67	0.04
Svg,Ret_{120}	1.15	0.14	[1.1, 1.2]	0.81	1.66	0.81	1.09	1.66	0.04
Svg,Ret_{360}	1.09	0.12	[1.05, 1.13]	0.98	1.61	0.98	1.03	1.61	0.04
Svg,Ret_{600}	1.11	0.13	[1.07, 1.15]	1	1.61	1	1.03	1.61	0.02
Svg,S_{0}	1.33	0.13	[1.28, 1.38]	0.92	1.66	0.92	1.32	1.66	0.04
Svg,S_{60}	1.21	0.13	[1.17, 1.25]	0.82	1.5	0.82	1.19	1.5	0.04
Svg,S_{120}	1.15	0.14	[1.1, 1.2]	0.81	1.66	0.81	1.09	1.66	0.04
Svg,S_{360}	1.06	0.12	[1.02, 1.1]	0.99	1.61	0.99	1	1.61	0.08
Svg,S_{600}	1.06	0.12	[1.02, 1.1]	0.98	1.61	0.98	1	1.61	0.04
Svg,Gap_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.36	1.66	0.04
Svg,Gap_{60}	1.21	0.13	[1.17, 1.25]	0.82	1.53	0.82	1.2	1.53	0.04
Svg,Gap_{120}	1.15	0.14	[1.1, 1.2]	0.75	1.66	0.75	1.12	1.66	0.04
Svg,Gap_{360}	1.1	0.13	[1.06, 1.14]	0.73	1.61	0.73	1.03	1.61	0.02
Svg,Gap_{600}	1.11	0.13	[1.07, 1.15]	0.75	1.61	0.75	1.06	1.61	0.02
Svg,Opt_{0}	1.35	0.13	[1.3, 1.4]	0.91	1.67	0.91	1.36	1.67	0.04
Svg,Opt_{60}	1.21	0.13	[1.17, 1.25]	0.83	1.49	0.83	1.2	1.49	0.04
Svg,Opt_{120}	1.15	0.14	[1.1, 1.2]	0.81	1.66	0.81	1.12	1.66	0.04
Svg,Opt_{360}	1.05	0.11	[1.02, 1.08]	0.98	1.61	0.98	1.01	1.61	0.06
Svg,Opt_{600}	1.06	0.13	[1.02, 1.1]	0.73	1.61	0.73	1.01	1.61	0.1
Ls,Ret_{0}	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.36	1.66	0.04
Ls,Ret_{60}	1.22	0.14	[1.17, 1.27]	0.83	1.68	0.83	1.2	1.68	0.04
Ls,Ret_{120}	1.15	0.14	[1.1, 1.2]	0.74	1.66	0.74	1.07	1.66	0.04
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of makespan relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	1.05	0.12	[1.01, 1.09]	0.99	1.61	0.99	1	1.61	0.1
Ls,Ret$_{600}$	1.06	0.12	[1.02, 1.1]	0.98	1.61	0.98	1.01	1.61	0.12
Ls,S$_{0}$	1.35	0.13	[1.3, 1.4]	0.92	1.67	0.92	1.36	1.67	0.04
Ls,S$_{60}$	1.21	0.13	[1.17, 1.25]	0.83	1.49	0.83	1.19	1.49	0.04
Ls,S$_{120}$	1.14	0.14	[1.09, 1.19]	0.74	1.66	0.74	1.06	1.66	0.04
Ls,S$_{360}$	1	0.04	[0.99, 1.01]	0.74	1.07	0.74	1	1.07	0.27
Ls,S$_{600}$	0.99	0.05	[0.98, 1]	0.73	1.02	0.73	1	1.02	0.31
Ls,Gap$_{0}$	1.35	0.13	[1.3, 1.4]	0.92	1.66	0.92	1.39	1.66	0.04
Ls,Gap$_{60}$	1.22	0.13	[1.18, 1.26]	0.85	1.53	0.85	1.23	1.53	0.04
Ls,Gap$_{120}$	1.15	0.13	[1.11, 1.19]	0.98	1.66	0.98	1.06	1.66	0.02
Ls,Gap$_{360}$	1.04	0.12	[1, 1.08]	0.73	1.61	0.73	1	1.61	0.12
Ls,Gap$_{600}$	1.05	0.13	[1.01, 1.09]	0.73	1.61	0.73	1	1.61	0.16
Ls,Opt$_{0}$	1.34	0.13	[1.29, 1.39]	0.91	1.67	0.91	1.32	1.67	0.04
Ls,Opt$_{60}$	1.22	0.13	[1.18, 1.26]	0.85	1.54	0.85	1.23	1.54	0.04
Ls,Opt$_{120}$	1.14	0.14	[1.09, 1.19]	0.74	1.66	0.74	1.07	1.66	0.04
Ls,Opt$_{360}$	1	0.04	[0.99, 1.01]	0.73	1.03	0.73	1	1.03	0.27
Ls,Opt$_{600}$	1.01	0.1	[0.98, 1.04]	0.73	1.61	0.73	1	1.61	0.24
Ts,Ret$_{0}$	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.33	1.66	0.04
Ts,Ret$_{60}$	1.23	0.15	[1.18, 1.28]	0.83	1.68	0.83	1.2	1.68	0.04
Ts,Ret$_{120}$	1.15	0.14	[1.1, 1.2]	0.75	1.66	0.75	1.12	1.66	0.04
Ts,Ret$_{360}$	1.05	0.12	[1.01, 1.09]	0.99	1.61	0.99	1.01	1.61	0.08
Ts,Ret$_{600}$	1.05	0.12	[1.01, 1.09]	0.73	1.61	0.73	1.01	1.61	0.08
Ts,S$_{0}$	1.35	0.13	[1.3, 1.4]	0.91	1.67	0.91	1.36	1.67	0.04
Ts,S$_{60}$	1.21	0.13	[1.17, 1.25]	0.83	1.49	0.83	1.19	1.49	0.04
Ts,S$_{120}$	1.14	0.14	[1.09, 1.19]	0.74	1.66	0.74	1.07	1.66	0.04
Ts,S$_{360}$	1	0.01	[1, 1]	0.97	1.02	0.97	1	1.02	0.24
Ts,S$_{600}$	1	0.04	[0.99, 1.01]	0.73	1.07	0.73	1	1.07	0.31
Ts,Gap$_{0}$	1.34	0.13	[1.29, 1.39]	0.92	1.66	0.92	1.37	1.66	0.04
Ts,Gap$_{60}$	1.22	0.14	[1.17, 1.27]	0.83	1.67	0.83	1.2	1.67	0.04
Ts,Gap$_{120}$	1.15	0.13	[1.11, 1.19]	0.99	1.66	0.99	1.08	1.66	0.02
Ts,Gap$_{360}$	1.05	0.12	[1.01, 1.09]	0.99	1.61	0.99	1	1.61	0.14
Ts,Gap$_{600}$	1.05	0.11	[1.02, 1.08]	0.97	1.61	0.97	1	1.61	0.12
Ts,Opt$_{0}$	1.33	0.13	[1.28, 1.38]	0.91	1.66	0.91	1.32	1.66	0.04
Ts,Opt$_{60}$	1.22	0.14	[1.17, 1.27]	0.82	1.68	0.82	1.2	1.68	0.04
Ts,Opt$_{120}$	1.15	0.13	[1.11, 1.19]	0.98	1.66	0.98	1.07	1.66	0.02
Ts,Opt$_{360}$	1.01	0.09	[0.98, 1.04]	0.83	1.61	0.83	1	1.61	0.31
Ts,Opt$_{600}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{0}$	1.33	0.13	[1.28, 1.38]	0.92	1.73	0.92	1.32	1.73	0.02
Opt,Opt$_{60}$	1.23	0.15	[1.18, 1.28]	0.85	1.68	0.85	1.24	1.68	0.06
Opt,Opt$_{120}$	1.15	0.13	[1.11, 1.19]	0.92	1.66	0.92	1.11	1.66	0.04
Opt,Opt$_{360}$	1.11	0.13	[1.07, 1.15]	0.99	1.61	0.99	1.02	1.61	0.04
Opt,Opt$_{600}$	1.1	0.13	[1.06, 1.14]	0.74	1.61	0.74	1.03	1.61	0.04

Table A.68: Performance ratios of makespan relative to Ts,Opt$_{600}$ in the order picking system.

Performance ratios of makespan relative to online version for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Prio,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Ret$_{60}$	0.9	0.06	[0.88, 0.92]	0.74	1.1	0.74	0.9	1.1	0.02
Prio,Ret$_{120}$	0.86	0.08	[0.84, 0.88]	0.68	1.08	0.68	0.86	1.08	0.02
Prio,Ret$_{360}$	0.83	0.08	[0.81, 0.85]	0.66	1.08	0.66	0.84	1.08	0.02
Prio,Ret$_{600}$	0.83	0.08	[0.81, 0.85]	0.66	1.08	0.66	0.84	1.08	0.02
Prio,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,S$_{60}$	0.92	0.06	[0.9, 0.94]	0.77	1.15	0.77	0.9	1.15	0.06
Prio,S$_{120}$	0.86	0.08	[0.84, 0.88]	0.7	1.09	0.7	0.85	1.09	0.04
Prio,S$_{360}$	0.82	0.1	[0.8, 0.84]	0.62	1.08	0.62	0.82	1.08	0.04
Prio,S$_{600}$	0.82	0.1	[0.8, 0.84]	0.62	1.08	0.62	0.82	1.08	0.04
Prio,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Gap$_{60}$	0.92	0.07	[0.9, 0.94]	0.74	1.12	0.74	0.9	1.12	0.08
Prio,Gap$_{120}$	0.86	0.07	[0.84, 0.88]	0.7	1.07	0.7	0.85	1.07	0.02
Prio,Gap$_{360}$	0.83	0.09	[0.81, 0.85]	0.63	1.07	0.63	0.82	1.07	0.02
Prio,Gap$_{600}$	0.83	0.09	[0.81, 0.85]	0.65	1.07	0.65	0.82	1.07	0.02
Prio,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Opt$_{60}$	0.9	0.05	[0.89, 0.91]	0.74	1.07	0.74	0.9	1.07	0.02
Prio,Opt$_{120}$	0.86	0.08	[0.84, 0.88]	0.68	1.09	0.68	0.85	1.09	0.04
Prio,Opt$_{360}$	0.8	0.12	[0.77, 0.83]	0.6	1.09	0.6	0.79	1.09	0.04
Prio,Opt$_{600}$	0.81	0.12	[0.78, 0.84]	0.61	1.09	0.61	0.8	1.09	0.04
Seed,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Ret$_{60}$	0.91	0.05	[0.9, 0.92]	0.78	1.16	0.78	0.9	1.16	0.04
Seed,Ret$_{120}$	0.87	0.07	[0.85, 0.89]	0.8	1.09	0.8	0.86	1.09	0.04
Seed,Ret$_{360}$	0.8	0.12	[0.77, 0.83]	0.6	1.09	0.6	0.8	1.09	0.04
Seed,Ret$_{600}$	0.8	0.11	[0.78, 0.82]	0.6	1.08	0.6	0.8	1.08	0.02
Seed,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,S$_{60}$	0.92	0.07	[0.9, 0.94]	0.8	1.16	0.8	0.9	1.16	0.08
Seed,S$_{120}$	0.85	0.07	[0.83, 0.87]	0.71	1.09	0.71	0.85	1.09	0.02
Seed,S$_{360}$	0.75	0.13	[0.72, 0.78]	0.58	1.09	0.58	0.75	1.09	0.02
Seed,S$_{600}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.74	1.09	0.04
Seed,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Gap$_{60}$	0.91	0.05	[0.9, 0.92]	0.78	1.14	0.78	0.9	1.14	0.04
Seed,Gap$_{120}$	0.86	0.08	[0.84, 0.88]	0.71	1.09	0.71	0.85	1.09	0.04
Seed,Gap$_{360}$	0.83	0.1	[0.81, 0.85]	0.64	1.09	0.64	0.83	1.09	0.04
Seed,Gap$_{600}$	0.82	0.09	[0.8, 0.84]	0.62	1.08	0.62	0.83	1.08	0.02
Seed,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Opt$_{60}$	0.9	0.04	[0.89, 0.91]	0.77	1	0.77	0.9	1	0
Seed,Opt$_{120}$	0.85	0.07	[0.83, 0.87]	0.69	1.08	0.69	0.85	1.08	0.02
Seed,Opt$_{360}$	0.76	0.14	[0.73, 0.79]	0.59	1.08	0.59	0.77	1.08	0.02
Seed,Opt$_{600}$	0.75	0.13	[0.72, 0.78]	0.58	1.08	0.58	0.73	1.08	0.02
Svg,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Ret$_{60}$	0.91	0.05	[0.9, 0.92]	0.75	1.11	0.75	0.91	1.11	0.02
Svg,Ret$_{120}$	0.86	0.07	[0.84, 0.88]	0.67	1.08	0.67	0.86	1.08	0.02
Svg,Ret$_{360}$	0.82	0.11	[0.79, 0.85]	0.61	1.09	0.61	0.81	1.09	0.04
Svg,Ret$_{600}$	0.83	0.11	[0.8, 0.86]	0.62	1.09	0.62	0.83	1.09	0.06
Svg,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,S$_{60}$	0.91	0.03	[0.9, 0.92]	0.78	1	0.78	0.9	1	0
Svg,S$_{120}$	0.86	0.07	[0.84, 0.88]	0.8	1.08	0.8	0.86	1.08	0.02
Svg,S$_{360}$	0.8	0.12	[0.77, 0.83]	0.59	1.09	0.59	0.8	1.09	0.04
Svg,S$_{600}$	0.8	0.12	[0.77, 0.83]	0.59	1.09	0.59	0.79	1.09	0.04
Svg,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Gap$_{60}$	0.9	0.04	[0.89, 0.91]	0.74	1	0.74	0.9	1	0
Svg,Gap$_{120}$	0.86	0.07	[0.84, 0.88]	0.7	1.08	0.7	0.85	1.08	0.02
Svg,Gap$_{360}$	0.82	0.1	[0.8, 0.84]	0.62	1.08	0.62	0.82	1.08	0.02
Svg,Gap$_{600}$	0.83	0.09	[0.81, 0.85]	0.65	1.08	0.65	0.83	1.08	0.02
Svg,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Opt$_{60}$	0.9	0.06	[0.88, 0.92]	0.74	1.15	0.74	0.9	1.15	0.02
Svg,Opt$_{120}$	0.86	0.08	[0.84, 0.88]	0.67	1.08	0.67	0.86	1.08	0.02
Svg,Opt$_{360}$	0.79	0.13	[0.76, 0.82]	0.59	1.09	0.59	0.78	1.09	0.04
Svg,Opt$_{600}$	0.79	0.12	[0.76, 0.82]	0.59	1.08	0.59	0.79	1.08	0.02
Ls,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Ret$_{60}$	0.91	0.06	[0.89, 0.93]	0.74	1.13	0.74	0.9	1.13	0.04
Ls,Ret$_{120}$	0.85	0.07	[0.83, 0.87]	0.67	1.08	0.67	0.85	1.08	0.02
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of makespan relative to online version for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	0.79	0.13	[0.76, 0.82]	0.6	1.09	0.6	0.78	1.09	0.04
Ls,Ret$_{600}$	0.79	0.13	[0.76, 0.82]	0.61	1.09	0.61	0.78	1.09	0.04
Ls,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ls,S$_{60}$	0.9	0.05	[0.89, 0.91]	0.75	1.07	0.75	0.9	1.07	0.02
Ls,S$_{120}$	0.85	0.08	[0.83, 0.87]	0.67	1.08	0.67	0.86	1.08	0.02
Ls,S$_{360}$	0.75	0.13	[0.72, 0.78]	0.59	1.08	0.59	0.74	1.08	0.02
Ls,S$_{600}$	0.75	0.13	[0.72, 0.78]	0.59	1.08	0.59	0.73	1.08	0.02
Ls,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Gap$_{60}$	0.9	0.04	[0.89, 0.91]	0.75	1	0.75	0.9	1	0
Ls,Gap$_{120}$	0.86	0.09	[0.84, 0.88]	0.67	1.09	0.67	0.85	1.09	0.04
Ls,Gap$_{360}$	0.78	0.12	[0.75, 0.81]	0.61	1.08	0.61	0.78	1.08	0.02
Ls,Gap$_{600}$	0.78	0.12	[0.75, 0.81]	0.59	1.08	0.59	0.78	1.08	0.02
Ls,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Opt$_{60}$	0.91	0.06	[0.89, 0.93]	0.78	1.16	0.78	0.9	1.16	0.04
Ls,Opt$_{120}$	0.86	0.07	[0.84, 0.88]	0.75	1.08	0.75	0.85	1.08	0.02
Ls,Opt$_{360}$	0.76	0.12	[0.73, 0.79]	0.6	1.08	0.6	0.76	1.08	0.02
Ls,Opt$_{600}$	0.77	0.13	[0.74, 0.8]	0.59	1.08	0.59	0.77	1.08	0.02
Ts,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Ret$_{60}$	0.92	0.07	[0.9, 0.94]	0.78	1.16	0.78	0.9	1.16	0.08
Ts,Ret$_{120}$	0.86	0.07	[0.84, 0.88]	0.71	1.08	0.71	0.85	1.08	0.02
Ts,Ret$_{360}$	0.79	0.12	[0.76, 0.82]	0.6	1.09	0.6	0.78	1.09	0.04
Ts,Ret$_{600}$	0.79	0.11	[0.77, 0.81]	0.6	1.08	0.6	0.79	1.08	0.02
Ts,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ts,S$_{60}$	0.9	0.04	[0.89, 0.91]	0.75	1	0.75	0.9	1	0
Ts,S$_{120}$	0.85	0.07	[0.83, 0.87]	0.69	1.09	0.69	0.85	1.09	0.02
Ts,S$_{360}$	0.76	0.14	[0.73, 0.79]	0.58	1.09	0.58	0.75	1.09	0.04
Ts,S$_{600}$	0.75	0.13	[0.72, 0.78]	0.59	1.09	0.59	0.74	1.09	0.02
Ts,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Gap$_{60}$	0.91	0.07	[0.89, 0.93]	0.74	1.16	0.74	0.9	1.16	0.06
Ts,Gap$_{120}$	0.86	0.08	[0.84, 0.88]	0.69	1.09	0.69	0.86	1.09	0.04
Ts,Gap$_{360}$	0.79	0.12	[0.76, 0.82]	0.6	1.09	0.6	0.78	1.09	0.04
Ts,Gap$_{600}$	0.79	0.12	[0.76, 0.82]	0.6	1.09	0.6	0.77	1.09	0.04
Ts,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Opt$_{60}$	0.91	0.06	[0.89, 0.93]	0.78	1.17	0.78	0.9	1.17	0.04
Ts,Opt$_{120}$	0.86	0.07	[0.84, 0.88]	0.8	1.09	0.8	0.86	1.09	0.04
Ts,Opt$_{360}$	0.77	0.14	[0.74, 0.8]	0.59	1.09	0.59	0.77	1.09	0.04
Ts,Opt$_{600}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.76	1.09	0.04
Opt,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{60}$	0.92	0.08	[0.9, 0.94]	0.76	1.16	0.76	0.9	1.16	0.08
Opt,Opt$_{120}$	0.87	0.07	[0.85, 0.89]	0.67	1	0.67	0.86	1	0.02
Opt,Opt$_{360}$	0.84	0.11	[0.81, 0.87]	0.59	1.09	0.59	0.84	1.09	0.04
Opt,Opt$_{600}$	0.83	0.09	[0.81, 0.85]	0.6	1	0.6	0.84	1	0.02

Table A.69: Performance ratios of makespan relative to the online version of an algorithm in the order picking system.

Distance for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Prio,Ret$_{0}$	96.91	0.02	[96.36, 97.46]	92.7	100.6	92.7	96.8	100.6	
Prio,Ret$_{60}$	96.21	0.02	[95.67, 96.75]	92.3	99.5	92.3	95.8	99.5	0.22
Prio,Ret$_{120}$	95.74	0.02	[95.2, 96.28]	91.1	98.1	91.1	95.8	98.1	0.35
Prio,Ret$_{360}$	95.52	0.02	[94.98, 96.06]	92.1	98.7	92.1	95.3	98.7	0.61
Prio,Ret$_{600}$	95.29	0.02	[94.75, 95.83]	90.9	97.8	90.9	95.4	97.8	0.43
Prio,S$_{0}$	86.42	0.02	[85.93, 86.91]	82.7	88.9	82.7	86.3	88.9	
Prio,S$_{60}$	85.76	0.02	[85.27, 86.25]	82.8	88.3	82.8	85.9	88.3	0.2
Prio,S$_{120}$	85.13	0.02	[84.65, 85.61]	82.2	88	82.2	85	88	0.2
Prio,S$_{360}$	84.75	0.02	[84.27, 85.23]	81.9	87.5	81.9	84.9	87.5	0.55
Prio,S$_{600}$	84.56	0.02	[84.08, 85.04]	81.8	87.2	81.8	84.9	87.2	0.59
Prio,Gap$_{0}$	85.05	0.01	[84.81, 85.29]	82.7	87.9	82.7	84.9	87.9	
Prio,Gap$_{60}$	84.47	0.01	[84.23, 84.71]	81.6	86.9	81.6	84.4	86.9	0.31
Prio,Gap$_{120}$	83.85	0.02	[83.38, 84.32]	81.5	86.8	81.5	83.9	86.8	0.2
Prio,Gap$_{360}$	83.61	0.02	[83.14, 84.08]	80.6	86.9	80.6	83.5	86.9	0.55
Prio,Gap$_{600}$	83.47	0.02	[83, 83.94]	80.8	86	80.8	83.6	86	0.51
Prio,Opt$_{0}$	75.56	0.01	[75.35, 75.77]	72.8	78.5	72.8	75.8	78.5	
Prio,Opt$_{60}$	74.99	0.01	[74.78, 75.2]	72.6	76.8	72.6	75	76.8	0.16
Prio,Opt$_{120}$	74.51	0.01	[74.3, 74.72]	72.5	76.7	72.5	74.8	76.7	0.2
Prio,Opt$_{360}$	73.89	0.02	[73.47, 74.31]	71.6	76.1	71.6	74	76.1	0.47
Prio,Opt$_{600}$	73.73	0.01	[73.52, 73.94]	71.4	75.4	71.4	73.9	75.4	0.47
Seed,Ret$_{0}$	94.42	0.02	[93.89, 94.95]	89.6	98.8	89.6	94.3	98.8	
Seed,Ret$_{60}$	89.65	0.02	[89.14, 90.16]	85.1	94	85.1	89.6	94	0.02
Seed,Ret$_{120}$	82.08	0.02	[81.62, 82.54]	78.5	88.3	78.5	82.1	88.3	0
Seed,Ret$_{360}$	67.49	0.02	[67.11, 67.87]	64.8	69.9	64.8	67.4	69.9	0.33
Seed,Ret$_{600}$	67.49	0.02	[67.11, 67.87]	65.4	70.1	65.4	67.3	70.1	0.49
Seed,S$_{0}$	84.67	0.02	[84.19, 85.15]	79.6	87.7	79.6	85.2	87.7	
Seed,S$_{60}$	80.39	0.02	[79.94, 80.84]	76.6	84	76.6	80.5	84	0
Seed,S$_{120}$	73.92	0.03	[73.29, 74.55]	69.3	79.8	69.3	74	79.8	0
Seed,S$_{360}$	59.62	0.02	[59.28, 59.96]	57.4	62	57.4	59.7	62	0.33
Seed,S$_{600}$	59.63	0.02	[59.29, 59.97]	57.1	61.8	57.1	59.9	61.8	0.49
Seed,Gap$_{0}$	83.25	0.02	[82.78, 83.72]	78.9	86.4	78.9	83.1	86.4	
Seed,Gap$_{60}$	79.19	0.02	[78.74, 79.64]	76.4	83.3	76.4	79.3	83.3	0.02
Seed,Gap$_{120}$	72.26	0.03	[71.65, 72.87]	68.2	77.5	68.2	72.2	77.5	0
Seed,Gap$_{360}$	63.49	0.02	[63.13, 63.85]	61.1	66.2	61.1	63.5	66.2	0.45
Seed,Gap$_{600}$	63.52	0.02	[63.16, 63.88]	60.6	65.8	60.6	63.5	65.8	0.59
Seed,Opt$_{0}$	74.76	0.02	[74.34, 75.18]	71.5	77.5	71.5	74.8	77.5	
Seed,Opt$_{60}$	72.01	0.02	[71.6, 72.42]	69.1	74.7	69.1	72.1	74.7	0
Seed,Opt$_{120}$	67.8	0.02	[67.42, 68.18]	64	71.8	64	67.9	71.8	0
Seed,Opt$_{360}$	57.75	0.02	[57.42, 58.08]	55.6	59.8	55.6	57.9	59.8	0.29
Seed,Opt$_{600}$	57.83	0.02	[57.5, 58.16]	55.5	59.8	55.5	57.9	59.8	0.47
Svg,Ret$_{0}$	93.57	0.02	[93.04, 94.1]	88.5	97.9	88.5	93.3	97.9	
Svg,Ret$_{60}$	89.19	0.02	[88.69, 89.69]	85.7	93.9	85.7	89	93.9	0.02
Svg,Ret$_{120}$	83.2	0.02	[82.73, 83.67]	78.6	89.1	78.6	83.3	89.1	0
Svg,Ret$_{360}$	78.51	0.02	[78.07, 78.95]	75.3	82.3	75.3	78.2	82.3	0.65
Svg,Ret$_{600}$	80.94	0.02	[80.48, 81.4]	77.1	85.6	77.1	80.8	85.6	0.59
Svg,S$_{0}$	83.98	0.02	[83.5, 84.46]	80.2	88	80.2	84.2	88	
Svg,S$_{60}$	80.46	0.02	[80, 80.92]	76.5	84	76.5	80.2	84	0
Svg,S$_{120}$	76.14	0.02	[75.71, 76.57]	72.8	80.9	72.8	76.3	80.9	0.02
Svg,S$_{360}$	70.33	0.02	[69.93, 70.73]	66.9	74.6	66.9	70.2	74.6	0.59
Svg,S$_{600}$	72.72	0.02	[72.31, 73.13]	69.8	76.3	69.8	72.9	76.3	0.53
Svg,Gap$_{0}$	82.91	0.02	[82.44, 83.38]	79.2	86.6	79.2	82.9	86.6	
Svg,Gap$_{60}$	79.84	0.02	[79.39, 80.29]	77.1	82.7	77.1	79.9	82.7	0
Svg,Gap$_{120}$	76.61	0.02	[76.18, 77.04]	72.7	83.2	72.7	76.5	83.2	0.02
Svg,Gap$_{360}$	74.28	0.03	[73.65, 74.91]	69.1	81.9	69.1	74.2	81.9	0.59
Svg,Gap$_{600}$	76.74	0.03	[76.09, 77.39]	71.6	80.8	71.6	76.8	80.8	0.67
Svg,Opt$_{0}$	74.46	0.02	[74.04, 74.88]	71.9	77.3	71.9	74.6	77.3	
Svg,Opt$_{60}$	72.37	0.02	[71.96, 72.78]	70.2	75.1	70.2	72.3	75.1	0.04
Svg,Opt$_{120}$	70.12	0.02	[69.72, 70.52]	67.1	73.5	67.1	70.1	73.5	0
Svg,Opt$_{360}$	67.43	0.03	[66.86, 68]	64.6	72.7	64.6	67.3	72.7	0.67
Svg,Opt$_{600}$	70.24	0.02	[69.84, 70.64]	66.6	72.5	66.6	70.2	72.5	0.63
Ls,Ret$_{0}$	92.7	0.03	[91.91, 93.49]	85.7	97.3	85.7	92.7	97.3	
Ls,Ret$_{60}$	87.26	0.02	[86.77, 87.75]	83.2	92	83.2	87.1	92	0.02
Ls,Ret$_{120}$	79.57	0.02	[79.12, 80.02]	75.7	84.7	75.7	79.4	84.7	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Distance for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	66.09	0.02	[65.72, 66.46]	63.7	68.7	63.7	66.1	68.7	0.45
Ls,Ret$_{600}$	65.97	0.02	[65.6, 66.34]	63.8	68.6	63.8	65.9	68.6	0.37
Ls,S$_0$	83.17	0.02	[82.7, 83.64]	79.3	87.2	79.3	83	87.2	
Ls,S$_{60}$	78.13	0.02	[77.69, 78.57]	74.3	81.6	74.3	78.1	81.6	0
Ls,S$_{120}$	71.87	0.02	[71.46, 72.28]	68.4	75.2	68.4	72	75.2	0
Ls,S$_{360}$	58.4	0.02	[58.07, 58.73]	55.5	60.8	55.5	58.4	60.8	0.37
Ls,S$_{600}$	58.42	0.02	[58.09, 58.75]	56.2	60.7	56.2	58.5	60.7	0.45
Ls,Gap$_0$	81.22	0.02	[80.76, 81.68]	76.8	85	76.8	81.1	85	
Ls,Gap$_{60}$	76.74	0.02	[76.31, 77.17]	72.9	79.9	72.9	76.8	79.9	0
Ls,Gap$_{120}$	70.63	0.03	[70.03, 71.23]	66.4	74.5	66.4	70.7	74.5	0
Ls,Gap$_{360}$	59.8	0.02	[59.46, 60.14]	57	62.4	57	59.8	62.4	0.35
Ls,Gap$_{600}$	59.89	0.02	[59.55, 60.23]	56.8	62.2	56.8	59.8	62.2	0.47
Ls,Opt$_0$	73.6	0.02	[73.18, 74.02]	70.3	76.4	70.3	73.8	76.4	
Ls,Opt$_{60}$	70.86	0.02	[70.46, 71.26]	68.1	73.3	68.1	71.1	73.3	0.02
Ls,Opt$_{120}$	66.92	0.02	[66.54, 67.3]	64.3	69.3	64.3	67	69.3	0
Ls,Opt$_{360}$	58.26	0.02	[57.93, 58.59]	55.5	60.6	55.5	58.2	60.6	0.24
Ls,Opt$_{600}$	58.36	0.02	[58.03, 58.69]	55.7	60.4	55.7	58.2	60.4	0.49
Ts,Ret$_0$	93.23	0.03	[92.44, 94.02]	85.9	98.6	85.9	93.3	98.6	
Ts,Ret$_{60}$	87.51	0.03	[86.77, 88.25]	82.9	92.3	82.9	87.2	92.3	0
Ts,Ret$_{120}$	80.27	0.03	[79.59, 80.95]	75.9	86.2	75.9	80.3	86.2	0
Ts,Ret$_{360}$	66.01	0.02	[65.64, 66.38]	63.3	68.6	63.3	66.1	68.6	0.39
Ts,Ret$_{600}$	65.99	0.02	[65.62, 66.36]	63.8	68.5	63.8	65.9	68.5	0.51
Ts,S$_0$	83.48	0.03	[82.77, 84.19]	78.8	87.6	78.8	83.6	87.6	
Ts,S$_{60}$	78.54	0.02	[78.1, 78.98]	75.1	82.3	75.1	78.7	82.3	0.02
Ts,S$_{120}$	72.38	0.03	[71.77, 72.99]	67.6	76.6	67.6	72.2	76.6	0.02
Ts,S$_{360}$	58.38	0.02	[58.05, 58.71]	56	60.6	56	58.4	60.6	0.37
Ts,S$_{600}$	58.36	0.02	[58.03, 58.69]	56.1	60.4	56.1	58.3	60.4	0.35
Ts,Gap$_0$	81.98	0.02	[81.52, 82.44]	77.3	86	77.3	81.9	86	
Ts,Gap$_{60}$	77.41	0.02	[76.97, 77.85]	74.6	80.7	74.6	77.3	80.7	0
Ts,Gap$_{120}$	70.95	0.02	[70.55, 71.35]	67.5	74.9	67.5	71.2	74.9	0
Ts,Gap$_{360}$	59.28	0.02	[58.94, 59.62]	57	62.1	57	59.2	62.1	0.35
Ts,Gap$_{600}$	59.46	0.02	[59.12, 59.8]	56.9	61.8	56.9	59.4	61.8	0.43
Ts,Opt$_0$	74.12	0.02	[73.7, 74.54]	70.4	76.6	70.4	74.1	76.6	
Ts,Opt$_{60}$	71.24	0.02	[70.84, 71.64]	68.7	73.8	68.7	71.4	73.8	0.02
Ts,Opt$_{120}$	67.17	0.02	[66.79, 67.55]	63.8	70.6	63.8	67.1	70.6	0
Ts,Opt$_{360}$	57.69	0.02	[57.36, 58.02]	55.3	60	55.3	57.9	60	0.41
Ts,Opt$_{600}$	57.77	0.02	[57.44, 58.1]	55	60.5	55	57.8	60.5	0.35
Opt,Opt$_0$	75	0.02	[74.58, 75.42]	69.3	78.2	69.3	75.1	78.2	
Opt,Opt$_{60}$	73.66	0.02	[73.24, 74.08]	69.3	76.2	69.3	73.7	76.2	0.14
Opt,Opt$_{120}$	69.94	0.02	[69.54, 70.34]	67	74.6	67	69.8	74.6	0.06
Opt,Opt$_{360}$	63.37	0.02	[63.01, 63.73]	61	66	61	63.3	66	0.33
Opt,Opt$_{600}$	63.39	0.02	[63.03, 63.75]	60.3	65.6	60.3	63.4	65.6	0.51

Table A.70: Distances in the order picking system.

Performance ratios of distance relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Prio,Ret$_{0}$	1.68	0.02	[1.67, 1.69]	1.63	1.74	1.63	1.68	1.74	0
Prio,Ret$_{60}$	1.67	0.02	[1.66, 1.68]	1.62	1.72	1.62	1.66	1.72	0
Prio,Ret$_{120}$	1.66	0.01	[1.66, 1.66]	1.62	1.7	1.62	1.65	1.7	0
Prio,Ret$_{360}$	1.65	0.01	[1.65, 1.65]	1.59	1.72	1.59	1.65	1.72	0
Prio,Ret$_{600}$	1.65	0.01	[1.65, 1.65]	1.6	1.7	1.6	1.65	1.7	0
Prio,S$_{0}$	1.5	0.01	[1.5, 1.5]	1.45	1.54	1.45	1.5	1.54	0
Prio,S$_{60}$	1.48	0.01	[1.48, 1.48]	1.45	1.54	1.45	1.48	1.54	0
Prio,S$_{120}$	1.47	0.01	[1.47, 1.47]	1.43	1.52	1.43	1.47	1.52	0
Prio,S$_{360}$	1.47	0.01	[1.47, 1.47]	1.43	1.53	1.43	1.47	1.53	0
Prio,S$_{600}$	1.46	0.01	[1.46, 1.46]	1.42	1.51	1.42	1.46	1.51	0
Prio,Gap$_{0}$	1.47	0.01	[1.47, 1.47]	1.43	1.52	1.43	1.47	1.52	0
Prio,Gap$_{60}$	1.46	0.01	[1.46, 1.46]	1.42	1.5	1.42	1.46	1.5	0
Prio,Gap$_{120}$	1.45	0.01	[1.45, 1.45]	1.4	1.5	1.4	1.45	1.5	0
Prio,Gap$_{360}$	1.45	0.01	[1.45, 1.45]	1.4	1.49	1.4	1.45	1.49	0
Prio,Gap$_{600}$	1.45	0.01	[1.45, 1.45]	1.4	1.49	1.4	1.45	1.49	0
Prio,Opt$_{0}$	1.31	0.01	[1.31, 1.31]	1.28	1.35	1.28	1.31	1.35	0
Prio,Opt$_{60}$	1.3	0.01	[1.3, 1.3]	1.27	1.33	1.27	1.3	1.33	0
Prio,Opt$_{120}$	1.29	0.01	[1.29, 1.29]	1.24	1.33	1.24	1.29	1.33	0
Prio,Opt$_{360}$	1.28	0.01	[1.28, 1.28]	1.25	1.32	1.25	1.28	1.32	0
Prio,Opt$_{600}$	1.28	0.01	[1.28, 1.28]	1.24	1.31	1.24	1.28	1.31	0
Seed,Ret$_{0}$	1.63	0.02	[1.62, 1.64]	1.54	1.71	1.54	1.63	1.71	0
Seed,Ret$_{60}$	1.55	0.03	[1.54, 1.56]	1.47	1.65	1.47	1.55	1.65	0
Seed,Ret$_{120}$	1.42	0.03	[1.41, 1.43]	1.35	1.49	1.35	1.42	1.49	0
Seed,Ret$_{360}$	1.17	0.01	[1.17, 1.17]	1.14	1.2	1.14	1.17	1.2	0
Seed,Ret$_{600}$	1.17	0.01	[1.17, 1.17]	1.14	1.19	1.14	1.17	1.19	0
Seed,S$_{0}$	1.47	0.02	[1.46, 1.48]	1.38	1.54	1.38	1.47	1.54	0
Seed,S$_{60}$	1.39	0.02	[1.38, 1.4]	1.31	1.48	1.31	1.4	1.48	0
Seed,S$_{120}$	1.28	0.03	[1.27, 1.29]	1.18	1.35	1.18	1.28	1.35	0
Seed,S$_{360}$	1.03	0.01	[1.03, 1.03]	1.01	1.05	1.01	1.03	1.05	0
Seed,S$_{600}$	1.03	0.01	[1.03, 1.03]	1.02	1.05	1.02	1.03	1.05	0
Seed,Gap$_{0}$	1.44	0.02	[1.43, 1.45]	1.38	1.49	1.38	1.45	1.49	0
Seed,Gap$_{60}$	1.37	0.02	[1.36, 1.38]	1.3	1.44	1.3	1.38	1.44	0
Seed,Gap$_{120}$	1.25	0.03	[1.24, 1.26]	1.19	1.32	1.19	1.25	1.32	0
Seed,Gap$_{360}$	1.1	0.01	[1.1, 1.1]	1.07	1.12	1.07	1.1	1.12	0
Seed,Gap$_{600}$	1.1	0.01	[1.1, 1.1]	1.08	1.12	1.08	1.1	1.12	0
Seed,Opt$_{0}$	1.29	0.01	[1.29, 1.29]	1.25	1.33	1.25	1.29	1.33	0
Seed,Opt$_{60}$	1.25	0.02	[1.24, 1.26]	1.19	1.29	1.19	1.25	1.29	0
Seed,Opt$_{120}$	1.17	0.02	[1.16, 1.18]	1.11	1.21	1.11	1.18	1.21	0
Seed,Opt$_{360}$	1	0.01	[1, 1]	0.98	1.01	0.98	1	1.01	0.41
Seed,Opt$_{600}$	1	0.01	[1, 1]	0.99	1.02	0.99	1	1.02	0.43
Svg,Ret$_{0}$	1.62	0.02	[1.61, 1.63]	1.53	1.68	1.53	1.63	1.68	0
Svg,Ret$_{60}$	1.54	0.02	[1.53, 1.55]	1.47	1.64	1.47	1.54	1.64	0
Svg,Ret$_{120}$	1.44	0.03	[1.43, 1.45]	1.37	1.53	1.37	1.44	1.53	0
Svg,Ret$_{360}$	1.36	0.02	[1.35, 1.37]	1.31	1.41	1.31	1.36	1.41	0
Svg,Ret$_{600}$	1.4	0.02	[1.39, 1.41]	1.35	1.44	1.35	1.41	1.44	0
Svg,S$_{0}$	1.45	0.02	[1.44, 1.46]	1.39	1.51	1.39	1.45	1.51	0
Svg,S$_{60}$	1.39	0.02	[1.38, 1.4]	1.32	1.45	1.32	1.39	1.45	0
Svg,S$_{120}$	1.32	0.02	[1.31, 1.33]	1.26	1.38	1.26	1.32	1.38	0
Svg,S$_{360}$	1.22	0.02	[1.21, 1.23]	1.18	1.25	1.18	1.22	1.25	0
Svg,S$_{600}$	1.26	0.02	[1.25, 1.27]	1.21	1.31	1.21	1.26	1.31	0
Svg,Gap$_{0}$	1.44	0.02	[1.43, 1.45]	1.38	1.48	1.38	1.44	1.48	0
Svg,Gap$_{60}$	1.38	0.02	[1.37, 1.39]	1.33	1.43	1.33	1.38	1.43	0
Svg,Gap$_{120}$	1.33	0.02	[1.32, 1.34]	1.27	1.4	1.27	1.33	1.4	0
Svg,Gap$_{360}$	1.29	0.03	[1.28, 1.3]	1.23	1.38	1.23	1.28	1.38	0
Svg,Gap$_{600}$	1.33	0.02	[1.32, 1.34]	1.28	1.39	1.28	1.33	1.39	0
Svg,Opt$_{0}$	1.29	0.01	[1.29, 1.29]	1.25	1.32	1.25	1.29	1.32	0
Svg,Opt$_{60}$	1.25	0.01	[1.25, 1.25]	1.2	1.29	1.2	1.26	1.29	0
Svg,Opt$_{120}$	1.21	0.02	[1.2, 1.22]	1.16	1.25	1.16	1.22	1.25	0
Svg,Opt$_{360}$	1.17	0.02	[1.16, 1.18]	1.13	1.23	1.13	1.16	1.23	0
Svg,Opt$_{600}$	1.22	0.01	[1.22, 1.22]	1.18	1.25	1.18	1.21	1.25	0
Ls,Ret$_{0}$	1.6	0.02	[1.59, 1.61]	1.48	1.67	1.48	1.61	1.67	0
Ls,Ret$_{60}$	1.51	0.03	[1.5, 1.52]	1.44	1.59	1.44	1.51	1.59	0
Ls,Ret$_{120}$	1.38	0.03	[1.37, 1.39]	1.29	1.46	1.29	1.37	1.46	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of distance relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	1.14	0.01	[1.14, 1.14]	1.12	1.17	1.12	1.14	1.17	0
Ls,Ret$_{600}$	1.14	0.01	[1.14, 1.14]	1.12	1.17	1.12	1.14	1.17	0
Ls,S$_{0}$	1.44	0.02	[1.43, 1.45]	1.37	1.5	1.37	1.44	1.5	0
Ls,S$_{60}$	1.35	0.03	[1.34, 1.36]	1.28	1.44	1.28	1.35	1.44	0
Ls,S$_{120}$	1.24	0.03	[1.23, 1.25]	1.16	1.32	1.16	1.24	1.32	0
Ls,S$_{360}$	1.01	0.01	[1.01, 1.01]	0.99	1.03	0.99	1.01	1.03	0.08
Ls,S$_{600}$	1.01	0.01	[1.01, 1.01]	1	1.03	1	1.01	1.03	0.06
Ls,Gap$_{0}$	1.41	0.02	[1.4, 1.42]	1.33	1.46	1.33	1.41	1.46	0
Ls,Gap$_{60}$	1.33	0.02	[1.32, 1.34]	1.26	1.41	1.26	1.33	1.41	0
Ls,Gap$_{120}$	1.22	0.03	[1.21, 1.23]	1.13	1.29	1.13	1.22	1.29	0
Ls,Gap$_{360}$	1.04	0.01	[1.04, 1.04]	1.01	1.06	1.01	1.04	1.06	0
Ls,Gap$_{600}$	1.04	0.01	[1.04, 1.04]	1.02	1.06	1.02	1.04	1.06	0
Ls,Opt$_{0}$	1.27	0.02	[1.26, 1.28]	1.21	1.31	1.21	1.28	1.31	0
Ls,Opt$_{60}$	1.23	0.02	[1.22, 1.24]	1.18	1.27	1.18	1.23	1.27	0
Ls,Opt$_{120}$	1.16	0.02	[1.15, 1.17]	1.1	1.2	1.1	1.16	1.2	0
Ls,Opt$_{360}$	1.01	0.01	[1.01, 1.01]	0.99	1.04	0.99	1.01	1.04	0.2
Ls,Opt$_{600}$	1.01	0.01	[1.01, 1.01]	0.99	1.03	0.99	1.01	1.03	0.08
Ts,Ret$_{0}$	1.61	0.02	[1.6, 1.62]	1.48	1.69	1.48	1.61	1.69	0
Ts,Ret$_{60}$	1.52	0.03	[1.51, 1.53]	1.43	1.61	1.43	1.52	1.61	0
Ts,Ret$_{120}$	1.39	0.03	[1.38, 1.4]	1.3	1.48	1.3	1.39	1.48	0
Ts,Ret$_{360}$	1.14	0.01	[1.14, 1.14]	1.12	1.17	1.12	1.14	1.17	0
Ts,Ret$_{600}$	1.14	0.01	[1.14, 1.14]	1.12	1.17	1.12	1.14	1.17	0
Ts,S$_{0}$	1.45	0.03	[1.44, 1.46]	1.37	1.51	1.37	1.45	1.51	0
Ts,S$_{60}$	1.36	0.03	[1.35, 1.37]	1.28	1.44	1.28	1.36	1.44	0
Ts,S$_{120}$	1.25	0.03	[1.24, 1.26]	1.15	1.35	1.15	1.25	1.35	0
Ts,S$_{360}$	1.01	0.01	[1.01, 1.01]	0.99	1.03	0.99	1.01	1.03	0.1
Ts,S$_{600}$	1.01	0.01	[1.01, 1.01]	0.99	1.03	0.99	1.01	1.03	0.1
Ts,Gap$_{0}$	1.42	0.02	[1.41, 1.43]	1.35	1.47	1.35	1.43	1.47	0
Ts,Gap$_{60}$	1.34	0.02	[1.33, 1.35]	1.29	1.4	1.29	1.34	1.4	0
Ts,Gap$_{120}$	1.23	0.03	[1.22, 1.24]	1.15	1.31	1.15	1.22	1.31	0
Ts,Gap$_{360}$	1.03	0.01	[1.03, 1.03]	1	1.05	1	1.03	1.05	0.02
Ts,Gap$_{600}$	1.03	0.01	[1.03, 1.03]	1.02	1.04	1.02	1.03	1.04	0
Ts,Opt$_{0}$	1.28	0.02	[1.27, 1.29]	1.23	1.32	1.23	1.29	1.32	0
Ts,Opt$_{60}$	1.23	0.02	[1.22, 1.24]	1.17	1.27	1.17	1.23	1.27	0
Ts,Opt$_{120}$	1.16	0.02	[1.15, 1.17]	1.08	1.21	1.08	1.17	1.21	0
Ts,Opt$_{360}$	1	0.01	[1, 1]	0.98	1.01	0.98	1	1.01	0.51
Ts,Opt$_{600}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{0}$	1.3	0.02	[1.29, 1.31]	1.19	1.39	1.19	1.3	1.39	0
Opt,Opt$_{60}$	1.28	0.02	[1.27, 1.29]	1.19	1.33	1.19	1.28	1.33	0
Opt,Opt$_{120}$	1.21	0.03	[1.2, 1.22]	1.13	1.3	1.13	1.21	1.3	0
Opt,Opt$_{360}$	1.1	0.01	[1.1, 1.1]	1.07	1.12	1.07	1.1	1.12	0
Opt,Opt$_{600}$	1.1	0.01	[1.1, 1.1]	1.08	1.12	1.08	1.1	1.12	0

Table A.71: Performance ratios of distance relative to Ts,Opt$_{600}$ in the order picking system.

Performance ratios of distance relative to online version for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Prio,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Ret$_{60}$	0.99	0.01	[0.99, 0.99]	0.98	1.01	0.98	0.99	1.01	0.22
Prio,Ret$_{120}$	0.99	0.01	[0.99, 0.99]	0.97	1.01	0.97	0.99	1.01	0.14
Prio,Ret$_{360}$	0.99	0.01	[0.99, 0.99]	0.96	1	0.96	0.99	1	0.1
Prio,Ret$_{600}$	0.98	0.01	[0.98, 0.98]	0.96	1.01	0.96	0.98	1.01	0.06
Prio,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,S$_{60}$	0.99	0.01	[0.99, 0.99]	0.97	1.02	0.97	0.99	1.02	0.2
Prio,S$_{120}$	0.99	0.01	[0.99, 0.99]	0.97	1.01	0.97	0.98	1.01	0.08
Prio,S$_{360}$	0.98	0.01	[0.98, 0.98]	0.96	1.02	0.96	0.98	1.02	0.04
Prio,S$_{600}$	0.98	0.01	[0.98, 0.98]	0.95	1.01	0.95	0.98	1.01	0.06
Prio,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Gap$_{60}$	0.99	0.01	[0.99, 0.99]	0.97	1.02	0.97	0.99	1.02	0.31
Prio,Gap$_{120}$	0.99	0.01	[0.99, 0.99]	0.96	1.01	0.96	0.99	1.01	0.08
Prio,Gap$_{360}$	0.98	0.01	[0.98, 0.98]	0.96	1.01	0.96	0.98	1.01	0.04
Prio,Gap$_{600}$	0.98	0.01	[0.98, 0.98]	0.95	1.02	0.95	0.98	1.02	0.04
Prio,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Opt$_{60}$	0.99	0.01	[0.99, 0.99]	0.97	1.01	0.97	0.99	1.01	0.16
Prio,Opt$_{120}$	0.99	0.01	[0.99, 0.99]	0.96	1	0.96	0.99	1	0.02
Prio,Opt$_{360}$	0.98	0.01	[0.98, 0.98]	0.96	0.99	0.96	0.98	0.99	0
Prio,Opt$_{600}$	0.98	0.01	[0.98, 0.98]	0.96	1	0.96	0.98	1	0.02
Seed,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Ret$_{60}$	0.95	0.02	[0.94, 0.96]	0.89	1.01	0.89	0.95	1.01	0.02
Seed,Ret$_{120}$	0.87	0.03	[0.86, 0.88]	0.81	0.94	0.81	0.87	0.94	0
Seed,Ret$_{360}$	0.71	0.02	[0.71, 0.71]	0.69	0.76	0.69	0.71	0.76	0
Seed,Ret$_{600}$	0.71	0.02	[0.71, 0.71]	0.69	0.75	0.69	0.71	0.75	0
Seed,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,S$_{60}$	0.95	0.02	[0.94, 0.96]	0.89	1	0.89	0.95	1	0
Seed,S$_{120}$	0.87	0.03	[0.86, 0.88]	0.82	0.94	0.82	0.87	0.94	0
Seed,S$_{360}$	0.7	0.02	[0.7, 0.7]	0.68	0.74	0.68	0.7	0.74	0
Seed,S$_{600}$	0.7	0.02	[0.7, 0.7]	0.68	0.74	0.68	0.7	0.74	0
Seed,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Gap$_{60}$	0.95	0.02	[0.94, 0.96]	0.89	1	0.89	0.95	1	0.02
Seed,Gap$_{120}$	0.87	0.03	[0.86, 0.88]	0.81	0.93	0.81	0.87	0.93	0
Seed,Gap$_{360}$	0.76	0.02	[0.76, 0.76]	0.73	0.81	0.73	0.76	0.81	0
Seed,Gap$_{600}$	0.76	0.02	[0.76, 0.76]	0.74	0.8	0.74	0.76	0.8	0
Seed,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Opt$_{60}$	0.96	0.02	[0.95, 0.97]	0.92	1	0.92	0.97	1	0
Seed,Opt$_{120}$	0.91	0.02	[0.9, 0.92]	0.87	0.94	0.87	0.91	0.94	0
Seed,Opt$_{360}$	0.77	0.01	[0.77, 0.77]	0.75	0.79	0.75	0.77	0.79	0
Seed,Opt$_{600}$	0.77	0.01	[0.77, 0.77]	0.75	0.8	0.75	0.77	0.8	0
Svg,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Ret$_{60}$	0.95	0.02	[0.94, 0.96]	0.88	1.02	0.88	0.95	1.02	0.02
Svg,Ret$_{120}$	0.89	0.03	[0.88, 0.9]	0.84	0.95	0.84	0.89	0.95	0
Svg,Ret$_{360}$	0.84	0.02	[0.84, 0.84]	0.8	0.91	0.8	0.84	0.91	0
Svg,Ret$_{600}$	0.87	0.02	[0.87, 0.87]	0.83	0.93	0.83	0.86	0.93	0
Svg,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,S$_{60}$	0.96	0.02	[0.95, 0.97]	0.91	1	0.91	0.96	1	0
Svg,S$_{120}$	0.91	0.03	[0.9, 0.92]	0.85	0.96	0.85	0.91	0.96	0
Svg,S$_{360}$	0.84	0.02	[0.84, 0.84]	0.8	0.89	0.8	0.84	0.89	0
Svg,S$_{600}$	0.87	0.02	[0.87, 0.87]	0.84	0.93	0.84	0.87	0.93	0
Svg,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Gap$_{60}$	0.96	0.02	[0.95, 0.97]	0.92	1	0.92	0.97	1	0
Svg,Gap$_{120}$	0.92	0.03	[0.91, 0.93]	0.87	0.98	0.87	0.93	0.98	0
Svg,Gap$_{360}$	0.9	0.03	[0.89, 0.91]	0.85	0.97	0.85	0.89	0.97	0
Svg,Gap$_{600}$	0.93	0.02	[0.92, 0.94]	0.89	0.97	0.89	0.92	0.97	0
Svg,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Opt$_{60}$	0.97	0.02	[0.96, 0.98]	0.93	1.01	0.93	0.97	1.01	0.04
Svg,Opt$_{120}$	0.94	0.02	[0.93, 0.95]	0.91	0.98	0.91	0.94	0.98	0
Svg,Opt$_{360}$	0.91	0.02	[0.9, 0.92]	0.88	0.96	0.88	0.9	0.96	0
Svg,Opt$_{600}$	0.94	0.02	[0.93, 0.95]	0.91	1	0.91	0.94	1	0
Ls,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Ret$_{60}$	0.94	0.03	[0.93, 0.95]	0.9	1	0.9	0.95	1	0.02
Ls,Ret$_{120}$	0.86	0.03	[0.85, 0.87]	0.81	0.92	0.81	0.85	0.92	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of distance relative to online version for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	0.71	0.02	[0.71, 0.71]	0.68	0.77	0.68	0.71	0.77	0
Ls,Ret$_{600}$	0.71	0.02	[0.71, 0.71]	0.69	0.77	0.69	0.71	0.77	0
Ls,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,S$_{60}$	0.94	0.02	[0.93, 0.95]	0.88	0.98	0.88	0.94	0.98	0
Ls,S$_{120}$	0.86	0.03	[0.85, 0.87]	0.81	0.92	0.81	0.86	0.92	0
Ls,S$_{360}$	0.7	0.02	[0.7, 0.7]	0.68	0.74	0.68	0.7	0.74	0
Ls,S$_{600}$	0.7	0.02	[0.7, 0.7]	0.68	0.74	0.68	0.7	0.74	0
Ls,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Gap$_{60}$	0.95	0.02	[0.94, 0.96]	0.89	1	0.89	0.94	1	0
Ls,Gap$_{120}$	0.87	0.03	[0.86, 0.88]	0.81	0.94	0.81	0.87	0.94	0
Ls,Gap$_{360}$	0.74	0.02	[0.74, 0.74]	0.71	0.78	0.71	0.74	0.78	0
Ls,Gap$_{600}$	0.74	0.02	[0.74, 0.74]	0.71	0.77	0.71	0.74	0.77	0
Ls,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Opt$_{60}$	0.96	0.02	[0.95, 0.97]	0.91	1	0.91	0.96	1	0.02
Ls,Opt$_{120}$	0.91	0.02	[0.9, 0.92]	0.87	0.96	0.87	0.91	0.96	0
Ls,Opt$_{360}$	0.79	0.02	[0.79, 0.79]	0.76	0.83	0.76	0.79	0.83	0
Ls,Opt$_{600}$	0.79	0.02	[0.79, 0.79]	0.77	0.83	0.77	0.79	0.83	0
Ts,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Ret$_{60}$	0.94	0.03	[0.93, 0.95]	0.88	1	0.88	0.94	1	0
Ts,Ret$_{120}$	0.86	0.03	[0.85, 0.87]	0.81	0.93	0.81	0.86	0.93	0
Ts,Ret$_{360}$	0.71	0.02	[0.71, 0.71]	0.68	0.76	0.68	0.7	0.76	0
Ts,Ret$_{600}$	0.71	0.02	[0.71, 0.71]	0.68	0.76	0.68	0.7	0.76	0
Ts,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,S$_{60}$	0.94	0.03	[0.93, 0.95]	0.86	1	0.86	0.95	1	0.02
Ts,S$_{120}$	0.87	0.04	[0.86, 0.88]	0.78	0.94	0.78	0.87	0.94	0
Ts,S$_{360}$	0.7	0.02	[0.7, 0.7]	0.67	0.74	0.67	0.7	0.74	0
Ts,S$_{600}$	0.7	0.02	[0.7, 0.7]	0.67	0.74	0.67	0.7	0.74	0
Ts,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Gap$_{60}$	0.94	0.02	[0.93, 0.95]	0.9	1	0.9	0.94	1	0
Ts,Gap$_{120}$	0.87	0.03	[0.86, 0.88]	0.81	0.93	0.81	0.86	0.93	0
Ts,Gap$_{360}$	0.72	0.02	[0.72, 0.72]	0.7	0.77	0.7	0.72	0.77	0
Ts,Gap$_{600}$	0.73	0.02	[0.73, 0.73]	0.7	0.77	0.7	0.72	0.77	0
Ts,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Opt$_{60}$	0.96	0.02	[0.95, 0.97]	0.92	1	0.92	0.96	1	0.02
Ts,Opt$_{120}$	0.91	0.02	[0.9, 0.92]	0.86	0.95	0.86	0.9	0.95	0
Ts,Opt$_{360}$	0.78	0.02	[0.78, 0.78]	0.76	0.82	0.76	0.78	0.82	0
Ts,Opt$_{600}$	0.78	0.02	[0.78, 0.78]	0.76	0.81	0.76	0.78	0.81	0
Opt,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{60}$	0.98	0.02	[0.97, 0.99]	0.93	1.02	0.93	0.98	1.02	0.14
Opt,Opt$_{120}$	0.93	0.03	[0.92, 0.94]	0.86	1	0.86	0.93	1	0
Opt,Opt$_{360}$	0.85	0.02	[0.85, 0.85]	0.8	0.93	0.8	0.84	0.93	0
Opt,Opt$_{600}$	0.85	0.03	[0.84, 0.86]	0.79	0.94	0.79	0.84	0.94	0

Table A.72: Performance ratios of distance relative to the online version of an algorithm in the order picking system.

Utilization for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Prio,Ret$_{0}$	0.79	0.05	[0.78, 0.8]	0.6	0.8	0.6	0.8	0.8	
Prio,Ret$_{60}$	0.88	0.05	[0.87, 0.89]	0.7	0.9	0.7	0.9	0.9	0.02
Prio,Ret$_{120}$	0.92	0.09	[0.9, 0.94]	0.6	1	0.6	0.9	1	0.06
Prio,Ret$_{360}$	0.94	0.07	[0.92, 0.96]	0.7	1	0.7	0.9	1	0.04
Prio,Ret$_{600}$	0.94	0.07	[0.92, 0.96]	0.7	1	0.7	0.9	1	0.06
Prio,S$_{0}$	0.73	0.07	[0.72, 0.74]	0.6	0.8	0.6	0.7	0.8	
Prio,S$_{60}$	0.81	0.07	[0.79, 0.83]	0.6	0.9	0.6	0.8	0.9	0.04
Prio,S$_{120}$	0.86	0.1	[0.84, 0.88]	0.6	1	0.6	0.9	1	0.1
Prio,S$_{360}$	0.91	0.1	[0.88, 0.94]	0.6	1	0.6	0.9	1	0.04
Prio,S$_{600}$	0.91	0.1	[0.88, 0.94]	0.7	1	0.7	0.9	1	0.04
Prio,Gap$_{0}$	0.78	0.06	[0.77, 0.79]	0.6	0.8	0.6	0.8	0.8	
Prio,Gap$_{60}$	0.85	0.08	[0.83, 0.87]	0.7	0.9	0.7	0.9	0.9	0.06
Prio,Gap$_{120}$	0.91	0.09	[0.89, 0.93]	0.6	1	0.6	0.9	1	0.08
Prio,Gap$_{360}$	0.92	0.09	[0.9, 0.94]	0.7	1	0.7	0.9	1	0.1
Prio,Gap$_{600}$	0.93	0.09	[0.91, 0.95]	0.7	1	0.7	0.9	1	0.08
Prio,Opt$_{0}$	0.71	0.06	[0.7, 0.72]	0.6	0.8	0.6	0.7	0.8	
Prio,Opt$_{60}$	0.8	0.05	[0.79, 0.81]	0.7	0.9	0.7	0.8	0.9	0
Prio,Opt$_{120}$	0.84	0.09	[0.82, 0.86]	0.6	1	0.6	0.8	1	0.08
Prio,Opt$_{360}$	0.9	0.11	[0.87, 0.93]	0.6	1	0.6	0.9	1	0.02
Prio,Opt$_{600}$	0.89	0.11	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.08
Seed,Ret$_{0}$	0.79	0.05	[0.78, 0.8]	0.6	0.8	0.6	0.8	0.8	
Seed,Ret$_{60}$	0.84	0.07	[0.82, 0.86]	0.7	0.9	0.7	0.8	0.9	0.04
Seed,Ret$_{120}$	0.86	0.09	[0.84, 0.88]	0.6	1	0.6	0.9	1	0.2
Seed,Ret$_{360}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.06
Seed,Ret$_{600}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.1
Seed,S$_{0}$	0.73	0.07	[0.72, 0.74]	0.6	0.8	0.6	0.7	0.8	
Seed,S$_{60}$	0.79	0.08	[0.77, 0.81]	0.6	0.9	0.6	0.8	0.9	0.08
Seed,S$_{120}$	0.84	0.09	[0.82, 0.86]	0.6	0.9	0.6	0.8	0.9	0.06
Seed,S$_{360}$	0.89	0.13	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.08
Seed,S$_{600}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.02
Seed,Gap$_{0}$	0.77	0.06	[0.76, 0.78]	0.6	0.8	0.6	0.8	0.8	
Seed,Gap$_{60}$	0.84	0.06	[0.83, 0.85]	0.7	0.9	0.7	0.8	0.9	0.02
Seed,Gap$_{120}$	0.88	0.09	[0.86, 0.9]	0.6	1	0.6	0.9	1	0.1
Seed,Gap$_{360}$	0.91	0.1	[0.88, 0.94]	0.7	1	0.7	0.9	1	0.12
Seed,Gap$_{600}$	0.92	0.09	[0.9, 0.94]	0.7	1	0.7	0.9	1	0.06
Seed,Opt$_{0}$	0.7	0.06	[0.69, 0.71]	0.6	0.8	0.6	0.7	0.8	
Seed,Opt$_{60}$	0.8	0.05	[0.79, 0.81]	0.6	0.9	0.6	0.8	0.9	0
Seed,Opt$_{120}$	0.82	0.09	[0.8, 0.84]	0.6	0.9	0.6	0.8	0.9	0.14
Seed,Opt$_{360}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.08
Seed,Opt$_{600}$	0.89	0.12	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.08
Svg,Ret$_{0}$	0.79	0.05	[0.78, 0.8]	0.6	0.8	0.6	0.8	0.8	
Svg,Ret$_{60}$	0.85	0.06	[0.84, 0.86]	0.7	0.9	0.7	0.9	0.9	0.02
Svg,Ret$_{120}$	0.87	0.09	[0.85, 0.89]	0.6	1	0.6	0.9	1	0.14
Svg,Ret$_{360}$	0.9	0.11	[0.87, 0.93]	0.6	1	0.6	0.9	1	0.06
Svg,Ret$_{600}$	0.9	0.11	[0.87, 0.93]	0.6	1	0.6	0.9	1	0.1
Svg,S$_{0}$	0.73	0.07	[0.72, 0.74]	0.6	0.8	0.6	0.7	0.8	
Svg,S$_{60}$	0.8	0.04	[0.79, 0.81]	0.7	0.9	0.7	0.8	0.9	0.02
Svg,S$_{120}$	0.84	0.09	[0.82, 0.86]	0.6	1	0.6	0.9	1	0.06
Svg,S$_{360}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.06
Svg,S$_{600}$	0.9	0.11	[0.87, 0.93]	0.6	1	0.6	0.9	1	0.04
Svg,Gap$_{0}$	0.77	0.07	[0.75, 0.79]	0.6	0.8	0.6	0.8	0.8	
Svg,Gap$_{60}$	0.83	0.06	[0.82, 0.84]	0.7	0.9	0.7	0.8	0.9	0
Svg,Gap$_{120}$	0.88	0.09	[0.86, 0.9]	0.6	1	0.6	0.9	1	0.08
Svg,Gap$_{360}$	0.92	0.09	[0.9, 0.94]	0.7	1	0.7	0.9	1	0.08
Svg,Gap$_{600}$	0.92	0.08	[0.9, 0.94]	0.7	1	0.7	0.9	1	0.08
Svg,Opt$_{0}$	0.71	0.06	[0.7, 0.72]	0.6	0.8	0.6	0.7	0.8	
Svg,Opt$_{60}$	0.79	0.05	[0.78, 0.8]	0.6	0.9	0.6	0.8	0.9	0.02
Svg,Opt$_{120}$	0.83	0.1	[0.81, 0.85]	0.6	0.9	0.6	0.8	0.9	0.1
Svg,Opt$_{360}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.04
Svg,Opt$_{600}$	0.89	0.11	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.06
Ls,Ret$_{0}$	0.78	0.06	[0.77, 0.79]	0.6	0.8	0.6	0.8	0.8	
Ls,Ret$_{60}$	0.83	0.06	[0.82, 0.84]	0.7	0.9	0.7	0.8	0.9	0.04
Ls,Ret$_{120}$	0.86	0.08	[0.84, 0.88]	0.6	1	0.6	0.9	1	0.16
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Utilization for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.02
Ls,Ret$_{600}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.08
Ls,S$_{0}$	0.71	0.06	[0.7, 0.72]	0.6	0.8	0.6	0.7	0.8	
Ls,S$_{60}$	0.8	0.04	[0.79, 0.81]	0.7	0.9	0.7	0.8	0.9	0
Ls,S$_{120}$	0.83	0.08	[0.81, 0.85]	0.6	0.9	0.6	0.8	0.9	0.1
Ls,S$_{360}$	0.89	0.12	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.06
Ls,S$_{600}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.04
Ls,Gap$_{0}$	0.75	0.07	[0.74, 0.76]	0.6	0.8	0.6	0.8	0.8	
Ls,Gap$_{60}$	0.8	0.05	[0.79, 0.81]	0.6	0.9	0.6	0.8	0.9	0.02
Ls,Gap$_{120}$	0.84	0.08	[0.82, 0.86]	0.6	0.9	0.6	0.8	0.9	0.1
Ls,Gap$_{360}$	0.89	0.13	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.06
Ls,Gap$_{600}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.08
Ls,Opt$_{0}$	0.72	0.07	[0.71, 0.73]	0.6	0.8	0.6	0.7	0.8	
Ls,Opt$_{60}$	0.79	0.07	[0.77, 0.81]	0.6	0.9	0.6	0.8	0.9	0.04
Ls,Opt$_{120}$	0.82	0.09	[0.8, 0.84]	0.6	0.9	0.6	0.8	0.9	0.1
Ls,Opt$_{360}$	0.89	0.12	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.04
Ls,Opt$_{600}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.04
Ts,Ret$_{0}$	0.78	0.05	[0.77, 0.79]	0.6	0.8	0.6	0.8	0.8	
Ts,Ret$_{60}$	0.84	0.07	[0.82, 0.86]	0.7	0.9	0.7	0.8	0.9	0.04
Ts,Ret$_{120}$	0.86	0.08	[0.84, 0.88]	0.6	1	0.6	0.9	1	0.16
Ts,Ret$_{360}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.06
Ts,Ret$_{600}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.06
Ts,S$_{0}$	0.72	0.07	[0.71, 0.73]	0.6	0.8	0.6	0.7	0.8	
Ts,S$_{60}$	0.8	0.05	[0.79, 0.81]	0.6	0.9	0.6	0.8	0.9	0.02
Ts,S$_{120}$	0.84	0.08	[0.82, 0.86]	0.6	0.9	0.6	0.8	0.9	0.06
Ts,S$_{360}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.06
Ts,S$_{600}$	0.89	0.12	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.02
Ts,Gap$_{0}$	0.75	0.08	[0.73, 0.77]	0.6	0.8	0.6	0.8	0.8	
Ts,Gap$_{60}$	0.8	0.06	[0.79, 0.81]	0.6	0.9	0.6	0.8	0.9	0.04
Ts,Gap$_{120}$	0.84	0.09	[0.82, 0.86]	0.6	1	0.6	0.8	1	0.08
Ts,Gap$_{360}$	0.87	0.13	[0.84, 0.9]	0.6	1	0.6	0.9	1	0.06
Ts,Gap$_{600}$	0.89	0.12	[0.86, 0.92]	0.6	1	0.6	0.9	1	0.06
Ts,Opt$_{0}$	0.71	0.06	[0.7, 0.72]	0.6	0.8	0.6	0.7	0.8	
Ts,Opt$_{60}$	0.79	0.06	[0.78, 0.8]	0.6	0.9	0.6	0.8	0.9	0.04
Ts,Opt$_{120}$	0.82	0.09	[0.8, 0.84]	0.6	0.9	0.6	0.8	0.9	0.12
Ts,Opt$_{360}$	0.87	0.14	[0.84, 0.9]	0.6	1	0.6	0.9	1	0.08
Ts,Opt$_{600}$	0.88	0.13	[0.85, 0.91]	0.6	1	0.6	0.9	1	0.04
Opt,Opt$_{0}$	0.72	0.09	[0.7, 0.74]	0.6	0.9	0.6	0.7	0.9	
Opt,Opt$_{60}$	0.79	0.07	[0.77, 0.81]	0.6	0.9	0.6	0.8	0.9	0.04
Opt,Opt$_{120}$	0.86	0.09	[0.84, 0.88]	0.6	1	0.6	0.9	1	0.1
Opt,Opt$_{360}$	0.91	0.1	[0.88, 0.94]	0.7	1	0.7	0.9	1	0.08
Opt,Opt$_{600}$	0.92	0.09	[0.9, 0.94]	0.7	1	0.7	0.9	1	0.06

Table A.73: Picker utilizations in the order picking system.

Performance ratios of utilization relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Prio,Ret$_{0}$	0.91	0.16	[0.87, 0.95]	0.67	1.33	0.67	0.89	1.33	0.12
Prio,Ret$_{60}$	1.02	0.16	[0.97, 1.07]	0.7	1.5	0.7	1	1.5	0.37
Prio,Ret$_{120}$	1.07	0.17	[1.02, 1.12]	0.67	1.67	0.67	1.11	1.67	0.53
Prio,Ret$_{360}$	1.09	0.15	[1.04, 1.14]	0.8	1.67	0.8	1.11	1.67	0.57
Prio,Ret$_{600}$	1.09	0.15	[1.04, 1.14]	0.8	1.67	0.8	1.11	1.67	0.57
Prio,S$_{0}$	0.85	0.2	[0.8, 0.9]	0.67	1.33	0.67	0.8	1.33	0.1
Prio,S$_{60}$	0.94	0.18	[0.89, 0.99]	0.6	1.5	0.6	0.9	1.5	0.2
Prio,S$_{120}$	1	0.14	[0.96, 1.04]	0.67	1.29	0.67	1	1.29	0.41
Prio,S$_{360}$	1.05	0.15	[1.01, 1.09]	0.7	1.67	0.7	1.11	1.67	0.57
Prio,S$_{600}$	1.05	0.16	[1, 1.1]	0.7	1.67	0.7	1.11	1.67	0.55
Prio,Gap$_{0}$	0.9	0.17	[0.86, 0.94]	0.67	1.33	0.67	0.89	1.33	0.12
Prio,Gap$_{60}$	0.99	0.18	[0.94, 1.04]	0.7	1.5	0.7	1	1.5	0.35
Prio,Gap$_{120}$	1.05	0.16	[1, 1.1]	0.67	1.67	0.67	1.11	1.67	0.51
Prio,Gap$_{360}$	1.07	0.16	[1.02, 1.12]	0.7	1.67	0.7	1.11	1.67	0.59
Prio,Gap$_{600}$	1.08	0.15	[1.03, 1.13]	0.7	1.67	0.7	1.11	1.67	0.59
Prio,Opt$_{0}$	0.82	0.17	[0.78, 0.86]	0.67	1.17	0.67	0.78	1.17	0.06
Prio,Opt$_{60}$	0.93	0.17	[0.89, 0.97]	0.7	1.5	0.7	0.89	1.5	0.16
Prio,Opt$_{120}$	0.97	0.14	[0.93, 1.01]	0.67	1.17	0.67	1	1.17	0.33
Prio,Opt$_{360}$	1.03	0.11	[1, 1.06]	0.7	1.25	0.7	1	1.25	0.41
Prio,Opt$_{600}$	1.02	0.11	[0.99, 1.05]	0.7	1.25	0.7	1	1.25	0.39
Seed,Ret$_{0}$	0.92	0.16	[0.88, 0.96]	0.67	1.33	0.67	0.89	1.33	0.12
Seed,Ret$_{60}$	0.98	0.18	[0.93, 1.03]	0.7	1.5	0.7	1	1.5	0.33
Seed,Ret$_{120}$	1	0.14	[0.96, 1.04]	0.67	1.29	0.67	1	1.29	0.41
Seed,Ret$_{360}$	1.01	0.11	[0.98, 1.04]	0.7	1.25	0.7	1	1.25	0.27
Seed,Ret$_{600}$	1.01	0.13	[0.97, 1.05]	0.6	1.5	0.6	1	1.5	0.27
Seed,S$_{0}$	0.85	0.19	[0.8, 0.9]	0.67	1.33	0.67	0.8	1.33	0.08
Seed,S$_{60}$	0.92	0.19	[0.87, 0.97]	0.6	1.5	0.6	0.89	1.5	0.16
Seed,S$_{120}$	0.97	0.16	[0.93, 1.01]	0.67	1.5	0.67	1	1.5	0.33
Seed,S$_{360}$	1.01	0.07	[0.99, 1.03]	0.87	1.33	0.87	1	1.33	0.1
Seed,S$_{600}$	1.01	0.04	[1, 1.02]	0.89	1.14	0.89	1	1.14	0.1
Seed,Gap$_{0}$	0.9	0.18	[0.85, 0.95]	0.67	1.33	0.67	0.89	1.33	0.12
Seed,Gap$_{60}$	0.98	0.17	[0.93, 1.03]	0.78	1.5	0.78	1	1.5	0.29
Seed,Gap$_{120}$	1.02	0.15	[0.98, 1.06]	0.67	1.29	0.67	1	1.29	0.47
Seed,Gap$_{360}$	1.05	0.14	[1.01, 1.09]	0.7	1.29	0.7	1.11	1.29	0.59
Seed,Gap$_{600}$	1.07	0.15	[1.02, 1.12]	0.7	1.67	0.7	1.11	1.67	0.59
Seed,Opt$_{0}$	0.82	0.18	[0.78, 0.86]	0.6	1.33	0.6	0.78	1.33	0.04
Seed,Opt$_{60}$	0.93	0.16	[0.89, 0.97]	0.6	1.33	0.6	0.89	1.33	0.12
Seed,Opt$_{120}$	0.95	0.17	[0.9, 1]	0.67	1.5	0.67	1	1.5	0.22
Seed,Opt$_{360}$	1.01	0.1	[0.98, 1.04]	0.6	1.33	0.6	1	1.33	0.1
Seed,Opt$_{600}$	1.01	0.09	[0.98, 1.04]	0.87	1.5	0.87	1	1.5	0.12
Svg,Ret$_{0}$	0.91	0.16	[0.87, 0.95]	0.67	1.33	0.67	0.89	1.33	0.12
Svg,Ret$_{60}$	0.99	0.17	[0.94, 1.04]	0.7	1.5	0.7	1	1.5	0.33
Svg,Ret$_{120}$	1.01	0.17	[0.96, 1.06]	0.67	1.67	0.67	1	1.67	0.43
Svg,Ret$_{360}$	1.04	0.12	[1, 1.08]	0.7	1.25	0.7	1.11	1.25	0.53
Svg,Ret$_{600}$	1.04	0.14	[1, 1.08]	0.7	1.29	0.7	1.11	1.29	0.57
Svg,S$_{0}$	0.85	0.18	[0.81, 0.89]	0.67	1.33	0.67	0.8	1.33	0.08
Svg,S$_{60}$	0.93	0.16	[0.89, 0.97]	0.7	1.33	0.7	0.89	1.33	0.12
Svg,S$_{120}$	0.98	0.17	[0.93, 1.03]	0.67	1.5	0.67	1	1.5	0.37
Svg,S$_{360}$	1.01	0.11	[0.98, 1.04]	0.6	1.25	0.6	1	1.25	0.31
Svg,S$_{600}$	1.03	0.11	[1, 1.06]	0.7	1.25	0.7	1	1.25	0.37
Svg,Gap$_{0}$	0.89	0.18	[0.84, 0.94]	0.67	1.33	0.67	0.87	1.33	0.12
Svg,Gap$_{60}$	0.97	0.16	[0.93, 1.01]	0.78	1.5	0.78	1	1.5	0.22
Svg,Gap$_{120}$	1.02	0.17	[0.97, 1.07]	0.67	1.67	0.67	1	1.67	0.43
Svg,Gap$_{360}$	1.06	0.15	[1.02, 1.1]	0.7	1.67	0.7	1.11	1.67	0.59
Svg,Gap$_{600}$	1.07	0.15	[1.02, 1.12]	0.7	1.67	0.7	1.11	1.67	0.57
Svg,Opt$_{0}$	0.82	0.17	[0.78, 0.86]	0.67	1.33	0.67	0.78	1.33	0.04
Svg,Opt$_{60}$	0.92	0.16	[0.88, 0.96]	0.67	1.33	0.67	0.89	1.33	0.14
Svg,Opt$_{120}$	0.97	0.17	[0.92, 1.02]	0.67	1.5	0.67	1	1.5	0.31
Svg,Opt$_{360}$	1.01	0.1	[0.98, 1.04]	0.7	1.17	0.7	1	1.17	0.29
Svg,Opt$_{600}$	1.02	0.12	[0.99, 1.05]	0.7	1.5	0.7	1	1.5	0.33
Ls,Ret$_{0}$	0.91	0.16	[0.87, 0.95]	0.67	1.33	0.67	0.89	1.33	0.12
Ls,Ret$_{60}$	0.97	0.18	[0.92, 1.02]	0.7	1.5	0.7	0.9	1.5	0.27
Ls,Ret$_{120}$	0.99	0.17	[0.94, 1.04]	0.67	1.67	0.67	1	1.67	0.35
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of utilization relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	1.01	0.1	[0.98, 1.04]	0.7	1.25	0.7	1	1.25	0.29
Ls,Ret$_{600}$	1	0.1	[0.97, 1.03]	0.6	1.17	0.6	1	1.17	0.24
Ls,S$_{0}$	0.82	0.16	[0.78, 0.86]	0.67	1.17	0.67	0.78	1.17	0.04
Ls,S$_{60}$	0.93	0.16	[0.89, 0.97]	0.7	1.33	0.7	0.89	1.33	0.16
Ls,S$_{120}$	0.96	0.16	[0.92, 1]	0.67	1.5	0.67	1	1.5	0.24
Ls,S$_{360}$	1.02	0.08	[1, 1.04]	0.87	1.5	0.87	1	1.5	0.1
Ls,S$_{600}$	1.01	0.09	[0.98, 1.04]	0.87	1.5	0.87	1	1.5	0.08
Ls,Gap$_{0}$	0.87	0.17	[0.83, 0.91]	0.67	1.33	0.67	0.87	1.33	0.1
Ls,Gap$_{60}$	0.93	0.17	[0.89, 0.97]	0.67	1.5	0.67	0.89	1.5	0.14
Ls,Gap$_{120}$	0.97	0.15	[0.93, 1.01]	0.67	1.17	0.67	1	1.17	0.35
Ls,Gap$_{360}$	1.02	0.13	[0.98, 1.06]	0.6	1.5	0.6	1	1.5	0.27
Ls,Gap$_{600}$	1.01	0.13	[0.97, 1.05]	0.6	1.5	0.6	1	1.5	0.27
Ls,Opt$_{0}$	0.84	0.19	[0.79, 0.89]	0.6	1.33	0.6	0.8	1.33	0.08
Ls,Opt$_{60}$	0.91	0.17	[0.87, 0.95]	0.67	1.33	0.67	0.89	1.33	0.12
Ls,Opt$_{120}$	0.95	0.17	[0.9, 1]	0.67	1.5	0.67	1	1.5	0.24
Ls,Opt$_{360}$	1.01	0.08	[0.99, 1.03]	0.89	1.5	0.89	1	1.5	0.08
Ls,Opt$_{600}$	1.01	0.1	[0.98, 1.04]	0.6	1.5	0.6	1	1.5	0.08
Ts,Ret$_{0}$	0.91	0.17	[0.87, 0.95]	0.67	1.33	0.67	0.89	1.33	0.12
Ts,Ret$_{60}$	0.98	0.18	[0.93, 1.03]	0.7	1.5	0.7	1	1.5	0.29
Ts,Ret$_{120}$	1	0.17	[0.95, 1.05]	0.67	1.67	0.67	1	1.67	0.37
Ts,Ret$_{360}$	1.01	0.1	[0.98, 1.04]	0.7	1.25	0.7	1	1.25	0.27
Ts,Ret$_{600}$	1.01	0.12	[0.98, 1.04]	0.6	1.5	0.6	1	1.5	0.24
Ts,S$_{0}$	0.84	0.18	[0.8, 0.88]	0.67	1.33	0.67	0.78	1.33	0.08
Ts,S$_{60}$	0.93	0.16	[0.89, 0.97]	0.67	1.33	0.67	0.89	1.33	0.14
Ts,S$_{120}$	0.97	0.17	[0.92, 1.02]	0.67	1.5	0.67	1	1.5	0.31
Ts,S$_{360}$	1.01	0.04	[1, 1.02]	0.9	1.13	0.9	1	1.13	0.08
Ts,S$_{600}$	1.01	0.08	[0.99, 1.03]	0.87	1.5	0.87	1	1.5	0.08
Ts,Gap$_{0}$	0.87	0.19	[0.82, 0.92]	0.67	1.33	0.67	0.87	1.33	0.12
Ts,Gap$_{60}$	0.93	0.17	[0.89, 0.97]	0.67	1.5	0.67	0.89	1.5	0.16
Ts,Gap$_{120}$	0.97	0.14	[0.93, 1.01]	0.67	1.17	0.67	1	1.17	0.35
Ts,Gap$_{360}$	1	0.11	[0.97, 1.03]	0.6	1.25	0.6	1	1.25	0.18
Ts,Gap$_{600}$	1.02	0.11	[0.99, 1.05]	0.6	1.17	0.6	1	1.17	0.33
Ts,Opt$_{0}$	0.83	0.17	[0.79, 0.87]	0.67	1.33	0.67	0.78	1.33	0.04
Ts,Opt$_{60}$	0.92	0.17	[0.88, 0.96]	0.6	1.33	0.6	0.89	1.33	0.14
Ts,Opt$_{120}$	0.94	0.15	[0.9, 0.98]	0.67	1.17	0.67	1	1.17	0.22
Ts,Opt$_{360}$	0.99	0.08	[0.97, 1.01]	0.6	1.25	0.6	1	1.25	0.06
Ts,Opt$_{600}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{0}$	0.83	0.18	[0.79, 0.87]	0.67	1.33	0.67	0.78	1.33	0.06
Opt,Opt$_{60}$	0.93	0.18	[0.88, 0.98]	0.6	1.5	0.6	0.89	1.5	0.16
Opt,Opt$_{120}$	1	0.15	[0.96, 1.04]	0.67	1.33	0.67	1	1.33	0.41
Opt,Opt$_{360}$	1.05	0.14	[1.01, 1.09]	0.7	1.29	0.7	1.11	1.29	0.59
Opt,Opt$_{600}$	1.07	0.15	[1.02, 1.12]	0.7	1.67	0.7	1.11	1.67	0.59

Table A.74: Performance ratios of picker utilization relative to Ts,Opt$_{600}$ in the order picking system.

Performance ratios of utilization relative to online version for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Prio,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Ret$_{60}$	1.12	0.07	[1.1, 1.14]	0.87	1.33	0.87	1.13	1.33	0.02
Prio,Ret$_{120}$	1.17	0.09	[1.14, 1.2]	0.87	1.43	0.87	1.13	1.43	0.02
Prio,Ret$_{360}$	1.2	0.1	[1.17, 1.23]	0.87	1.67	0.87	1.25	1.67	0.02
Prio,Ret$_{600}$	1.2	0.1	[1.17, 1.23]	0.87	1.67	0.87	1.13	1.67	0.02
Prio,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,S$_{60}$	1.11	0.09	[1.08, 1.14]	0.86	1.33	0.86	1.14	1.33	0.04
Prio,S$_{120}$	1.19	0.1	[1.16, 1.22]	0.87	1.5	0.87	1.14	1.5	0.04
Prio,S$_{360}$	1.25	0.14	[1.2, 1.3]	0.75	1.67	0.75	1.29	1.67	0.02
Prio,S$_{600}$	1.25	0.13	[1.2, 1.3]	0.87	1.67	0.87	1.29	1.67	0.02
Prio,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Gap$_{60}$	1.1	0.09	[1.07, 1.13]	0.87	1.33	0.87	1.13	1.33	0.06
Prio,Gap$_{120}$	1.17	0.09	[1.14, 1.2]	0.87	1.43	0.87	1.13	1.43	0.02
Prio,Gap$_{360}$	1.2	0.11	[1.16, 1.24]	0.87	1.67	0.87	1.25	1.67	0.04
Prio,Gap$_{600}$	1.2	0.12	[1.16, 1.24]	0.87	1.67	0.87	1.25	1.67	0.04
Prio,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Opt$_{60}$	1.14	0.07	[1.12, 1.16]	1	1.33	1	1.14	1.33	0
Prio,Opt$_{120}$	1.19	0.1	[1.16, 1.22]	0.86	1.5	0.86	1.14	1.5	0.02
Prio,Opt$_{360}$	1.28	0.14	[1.23, 1.33]	0.86	1.67	0.86	1.29	1.67	0.02
Prio,Opt$_{600}$	1.26	0.13	[1.21, 1.31]	0.86	1.67	0.86	1.29	1.67	0.02
Seed,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Ret$_{60}$	1.07	0.08	[1.05, 1.09]	0.87	1.33	0.87	1.13	1.33	0.04
Seed,Ret$_{120}$	1.09	0.07	[1.07, 1.11]	0.87	1.25	0.87	1.13	1.25	0.04
Seed,Ret$_{360}$	1.12	0.14	[1.08, 1.16]	0.75	1.67	0.75	1.13	1.67	0.1
Seed,Ret$_{600}$	1.12	0.13	[1.08, 1.16]	0.75	1.5	0.75	1.13	1.5	0.08
Seed,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,S$_{60}$	1.09	0.09	[1.06, 1.12]	0.86	1.29	0.86	1.14	1.29	0.08
Seed,S$_{120}$	1.16	0.09	[1.13, 1.19]	0.75	1.29	0.75	1.14	1.29	0.02
Seed,S$_{360}$	1.23	0.16	[1.17, 1.29]	0.75	1.5	0.75	1.25	1.5	0.06
Seed,S$_{600}$	1.22	0.16	[1.16, 1.28]	0.75	1.5	0.75	1.25	1.5	0.06
Seed,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Gap$_{60}$	1.09	0.07	[1.07, 1.11]	0.87	1.33	0.87	1.13	1.33	0.02
Seed,Gap$_{120}$	1.14	0.09	[1.11, 1.17]	0.87	1.43	0.87	1.13	1.43	0.04
Seed,Gap$_{360}$	1.18	0.13	[1.14, 1.22]	0.87	1.67	0.87	1.13	1.67	0.04
Seed,Gap$_{600}$	1.2	0.11	[1.16, 1.24]	0.87	1.67	0.87	1.25	1.67	0.02
Seed,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Opt$_{60}$	1.14	0.06	[1.12, 1.16]	1	1.33	1	1.14	1.33	0
Seed,Opt$_{120}$	1.17	0.1	[1.14, 1.2]	0.86	1.5	0.86	1.14	1.5	0.02
Seed,Opt$_{360}$	1.26	0.16	[1.2, 1.32]	0.86	1.67	0.86	1.29	1.67	0.04
Seed,Opt$_{600}$	1.27	0.14	[1.22, 1.32]	0.86	1.67	0.86	1.29	1.67	0.02
Svg,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Ret$_{60}$	1.09	0.07	[1.07, 1.11]	0.87	1.33	0.87	1.13	1.33	0.02
Svg,Ret$_{120}$	1.11	0.09	[1.08, 1.14]	0.87	1.43	0.87	1.13	1.43	0.02
Svg,Ret$_{360}$	1.15	0.13	[1.11, 1.19]	0.75	1.67	0.75	1.13	1.67	0.06
Svg,Ret$_{600}$	1.16	0.13	[1.12, 1.2]	0.75	1.67	0.75	1.13	1.67	0.08
Svg,S$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,S$_{60}$	1.11	0.07	[1.09, 1.13]	0.87	1.33	0.87	1.14	1.33	0.02
Svg,S$_{120}$	1.16	0.09	[1.13, 1.19]	0.86	1.43	0.86	1.14	1.43	0.04
Svg,S$_{360}$	1.22	0.15	[1.17, 1.27]	0.86	1.67	0.86	1.25	1.67	0.08
Svg,S$_{600}$	1.24	0.14	[1.19, 1.29]	0.86	1.67	0.86	1.29	1.67	0.06
Svg,Gap$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Gap$_{60}$	1.09	0.09	[1.06, 1.12]	1	1.5	1	1.13	1.5	0
Svg,Gap$_{120}$	1.15	0.1	[1.12, 1.18]	0.87	1.5	0.87	1.13	1.5	0.02
Svg,Gap$_{360}$	1.2	0.13	[1.16, 1.24]	0.87	1.67	0.87	1.25	1.67	0.04
Svg,Gap$_{600}$	1.21	0.12	[1.17, 1.25]	0.87	1.67	0.87	1.25	1.67	0.04
Svg,Opt$_0$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Opt$_{60}$	1.13	0.07	[1.11, 1.15]	0.86	1.33	0.86	1.14	1.33	0.02
Svg,Opt$_{120}$	1.18	0.11	[1.14, 1.22]	0.86	1.5	0.86	1.14	1.5	0.02
Svg,Opt$_{360}$	1.26	0.14	[1.21, 1.31]	0.86	1.67	0.86	1.29	1.67	0.04
Svg,Opt$_{600}$	1.27	0.14	[1.22, 1.32]	0.86	1.67	0.86	1.29	1.67	0.02
Ls,Ret$_0$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Ret$_{60}$	1.06	0.08	[1.04, 1.08]	0.87	1.33	0.87	1	1.33	0.04
Ls,Ret$_{120}$	1.1	0.08	[1.08, 1.12]	0.87	1.29	0.87	1.13	1.29	0.02
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of utilization relative to online version for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	1.14	0.15	[1.09, 1.19]	0.75	1.67	0.75	1.13	1.67	0.12
Ls,Ret$_{600}$	1.12	0.15	[1.07, 1.17]	0.75	1.5	0.75	1.13	1.5	0.12
Ls,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,S$_{60}$	1.14	0.06	[1.12, 1.16]	1	1.33	1	1.14	1.33	0
Ls,S$_{120}$	1.17	0.1	[1.14, 1.2]	1	1.5	1	1.14	1.5	0
Ls,S$_{360}$	1.26	0.14	[1.21, 1.31]	0.86	1.5	0.86	1.29	1.5	0.02
Ls,S$_{600}$	1.25	0.14	[1.2, 1.3]	0.86	1.5	0.86	1.29	1.5	0.02
Ls,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Gap$_{60}$	1.07	0.08	[1.05, 1.09]	0.86	1.33	0.86	1	1.33	0.02
Ls,Gap$_{120}$	1.12	0.09	[1.09, 1.15]	0.87	1.29	0.87	1.13	1.29	0.02
Ls,Gap$_{360}$	1.19	0.15	[1.14, 1.24]	0.86	1.67	0.86	1.14	1.67	0.08
Ls,Gap$_{600}$	1.18	0.15	[1.13, 1.23]	0.86	1.67	0.86	1.14	1.67	0.06
Ls,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Opt$_{60}$	1.09	0.08	[1.07, 1.11]	0.86	1.33	0.86	1.14	1.33	0.04
Ls,Opt$_{120}$	1.14	0.1	[1.11, 1.17]	0.75	1.33	0.75	1.14	1.33	0.04
Ls,Opt$_{360}$	1.24	0.16	[1.18, 1.3]	0.75	1.67	0.75	1.25	1.67	0.06
Ls,Opt$_{600}$	1.22	0.16	[1.16, 1.28]	0.75	1.67	0.75	1.14	1.67	0.06
Ts,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Ret$_{60}$	1.07	0.08	[1.05, 1.09]	0.87	1.33	0.87	1.13	1.33	0.04
Ts,Ret$_{120}$	1.1	0.08	[1.08, 1.12]	0.87	1.29	0.87	1.13	1.29	0.02
Ts,Ret$_{360}$	1.13	0.15	[1.08, 1.18]	0.75	1.67	0.75	1.13	1.67	0.12
Ts,Ret$_{600}$	1.13	0.14	[1.09, 1.17]	0.75	1.5	0.75	1.13	1.5	0.1
Ts,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,S$_{60}$	1.11	0.07	[1.09, 1.13]	0.86	1.33	0.86	1.14	1.33	0.02
Ts,S$_{120}$	1.17	0.08	[1.14, 1.2]	1	1.5	1	1.14	1.5	0
Ts,S$_{360}$	1.23	0.16	[1.17, 1.29]	0.75	1.5	0.75	1.29	1.5	0.08
Ts,S$_{600}$	1.24	0.15	[1.19, 1.29]	0.86	1.5	0.86	1.29	1.5	0.04
Ts,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Gap$_{60}$	1.07	0.09	[1.04, 1.1]	0.86	1.33	0.86	1	1.33	0.04
Ts,Gap$_{120}$	1.12	0.12	[1.08, 1.16]	0.75	1.5	0.75	1.13	1.5	0.04
Ts,Gap$_{360}$	1.17	0.16	[1.12, 1.22]	0.75	1.5	0.75	1.14	1.5	0.12
Ts,Gap$_{600}$	1.19	0.16	[1.14, 1.24]	0.75	1.67	0.75	1.25	1.67	0.08
Ts,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Opt$_{60}$	1.11	0.07	[1.09, 1.13]	0.86	1.17	0.86	1.14	1.17	0.04
Ts,Opt$_{120}$	1.14	0.09	[1.11, 1.17]	0.75	1.29	0.75	1.14	1.29	0.02
Ts,Opt$_{360}$	1.22	0.16	[1.16, 1.28]	0.75	1.5	0.75	1.29	1.5	0.06
Ts,Opt$_{600}$	1.23	0.15	[1.18, 1.28]	0.75	1.5	0.75	1.29	1.5	0.04
Opt,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{60}$	1.11	0.1	[1.08, 1.14]	0.86	1.33	0.86	1.14	1.33	0.04
Opt,Opt$_{120}$	1.21	0.1	[1.18, 1.24]	1	1.5	1	1.17	1.5	0
Opt,Opt$_{360}$	1.28	0.12	[1.24, 1.32]	0.87	1.67	0.87	1.29	1.67	0.02
Opt,Opt$_{600}$	1.29	0.11	[1.25, 1.33]	1	1.67	1	1.29	1.67	0

Table A.75: Performance ratios of picker utilization relative to the online version of an algorithm in the order picking system.

Throughput for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Prio,Ret$_{0}$	120.03	0.05	[118.33, 121.73]	100.6	125.9	100.6	122.1	125.9	
Prio,Ret$_{60}$	133.14	0.03	[132.01, 134.27]	109.2	137.7	109.2	133.9	137.7	0.02
Prio,Ret$_{120}$	140.71	0.07	[137.92, 143.5]	101.4	153.4	101.4	142.1	153.4	0.06
Prio,Ret$_{360}$	144.51	0.06	[142.06, 146.96]	109.6	164	109.6	144.3	164	0.43
Prio,Ret$_{600}$	144.39	0.06	[141.94, 146.84]	109.6	164.7	109.6	145	164.7	0.45
Prio,S$_{0}$	120.99	0.04	[119.62, 122.36]	101.4	126.3	101.4	122.4	126.3	
Prio,S$_{60}$	132.29	0.06	[130.04, 134.54]	105.9	138.3	105.9	134.4	138.3	0.06
Prio,S$_{120}$	140.96	0.08	[137.77, 144.15]	101.4	153.9	101.4	142.6	153.9	0.12
Prio,S$_{360}$	149.07	0.09	[145.27, 152.87]	109.6	172.2	109.6	149	172.2	0.29
Prio,S$_{600}$	148.47	0.09	[144.69, 152.25]	109.6	173.1	109.6	149	173.1	0.41
Prio,Gap$_{0}$	120.47	0.04	[119.11, 121.83]	101.4	126.5	101.4	121.5	126.5	
Prio,Gap$_{60}$	131.93	0.05	[130.06, 133.8]	105.9	138.2	105.9	133.9	138.2	0.08
Prio,Gap$_{120}$	141.43	0.07	[138.63, 144.23]	101.4	154.1	101.4	143.4	154.1	0.08
Prio,Gap$_{360}$	146.48	0.07	[143.58, 149.38]	109.6	167	109.6	146.2	167	0.31
Prio,Gap$_{600}$	146.34	0.07	[143.44, 149.24]	109.6	167.9	109.6	148	167.9	0.39
Prio,Opt$_{0}$	120.89	0.04	[119.52, 122.26]	101.4	126.6	101.4	122.5	126.6	
Prio,Opt$_{60}$	134	0.03	[132.86, 135.14]	112.8	139	112.8	134.7	139	0.02
Prio,Opt$_{120}$	141.69	0.08	[138.48, 144.9]	101.4	154.3	101.4	143.4	154.3	0.08
Prio,Opt$_{360}$	152.51	0.1	[148.2, 156.82]	109.6	179	109.6	154.9	179	0.37
Prio,Opt$_{600}$	151.01	0.1	[146.74, 155.28]	109.6	177.6	109.6	151.8	177.6	0.43
Seed,Ret$_{0}$	121.37	0.03	[120.34, 122.4]	101.4	125.7	101.4	122.1	125.7	
Seed,Ret$_{60}$	133.05	0.04	[131.54, 134.56]	102.7	138.5	102.7	134.7	138.5	0.04
Seed,Ret$_{120}$	140.5	0.08	[137.32, 143.68]	101.4	153.6	101.4	141.9	153.6	0.12
Seed,Ret$_{360}$	153.43	0.1	[149.09, 157.77]	109.6	180.4	109.6	153.7	180.4	0.2
Seed,Ret$_{600}$	153.62	0.09	[149.71, 157.53]	109.6	180.3	109.6	153	180.3	0.35
Seed,S$_{0}$	120.79	0.04	[119.42, 122.16]	101.4	126	101.4	122.1	126	
Seed,S$_{60}$	132	0.06	[129.76, 134.24]	102.7	138.4	102.7	134.6	138.4	0.08
Seed,S$_{120}$	142.29	0.07	[139.47, 145.11]	101.4	154.2	101.4	143.8	154.2	0.04
Seed,S$_{360}$	162.96	0.11	[157.89, 168.03]	109.6	204.2	109.6	158.6	204.2	0.43
Seed,S$_{600}$	161.61	0.12	[156.12, 167.1]	109.6	200.3	109.6	159	200.3	0.27
Seed,Gap$_{0}$	120.85	0.04	[119.48, 122.22]	101.4	125.8	101.4	122.1	125.8	
Seed,Gap$_{60}$	133.22	0.04	[131.71, 134.73]	107	137.9	107	134.6	137.9	0.04
Seed,Gap$_{120}$	141.21	0.07	[138.41, 144.01]	101.4	153.7	101.4	143.1	153.7	0.08
Seed,Gap$_{360}$	146.56	0.08	[143.24, 149.88]	109.6	170.5	109.6	147.1	170.5	0.35
Seed,Gap$_{600}$	148.07	0.07	[145.14, 151]	109.6	168.4	109.6	148.4	168.4	0.31
Seed,Opt$_{0}$	120.93	0.04	[119.56, 122.3]	101.4	126.4	101.4	122.6	126.4	
Seed,Opt$_{60}$	134.26	0.03	[133.12, 135.4]	109.2	139.4	109.2	135.3	139.4	0
Seed,Opt$_{120}$	142.21	0.07	[139.39, 145.03]	101.4	154.2	101.4	143.4	154.2	0.08
Seed,Opt$_{360}$	162.27	0.12	[156.76, 167.78]	109.6	204.3	109.6	159.7	204.3	0.33
Seed,Opt$_{600}$	163.68	0.11	[158.59, 168.77]	109.6	199.6	109.6	164.5	199.6	0.39
Svg,Ret$_{0}$	120.93	0.04	[119.56, 122.3]	100.6	125.9	100.6	121.9	125.9	
Svg,Ret$_{60}$	133.46	0.03	[132.33, 134.59]	109.2	137.9	109.2	134.2	137.9	0.02
Svg,Ret$_{120}$	141.33	0.07	[138.53, 144.13]	101.4	153.2	101.4	142.3	153.2	0.06
Svg,Ret$_{360}$	148.82	0.09	[145.03, 152.61]	109.6	171.1	109.6	149.7	171.1	0.45
Svg,Ret$_{600}$	146.66	0.09	[142.93, 150.39]	109.6	171.9	109.6	147.4	171.9	0.37
Svg,S$_{0}$	121.57	0.03	[120.54, 122.6]	101.4	126.1	101.4	122.3	126.1	
Svg,S$_{60}$	134.26	0.03	[133.12, 135.4]	115.4	139.1	115.4	134.9	139.1	0
Svg,S$_{120}$	141.7	0.07	[138.89, 144.51]	101.4	154.1	101.4	143.4	154.1	0.08
Svg,S$_{360}$	153.73	0.1	[149.38, 158.08]	109.6	181	109.6	154.4	181	0.27
Svg,S$_{600}$	153.06	0.1	[148.73, 157.39]	109.6	177.5	109.6	153.7	177.5	0.24
Svg,Gap$_{0}$	120.63	0.04	[119.26, 122]	101.4	126.2	101.4	122.2	126.2	
Svg,Gap$_{60}$	133.66	0.04	[132.15, 135.17]	102.7	138.3	102.7	134.7	138.3	0
Svg,Gap$_{120}$	141.49	0.07	[138.69, 144.29]	101.4	153.9	101.4	142.7	153.9	0.08
Svg,Gap$_{360}$	147.91	0.08	[144.56, 151.26]	109.6	173.4	109.6	147.4	173.4	0.45
Svg,Gap$_{600}$	146.27	0.07	[143.37, 149.17]	109.6	171.6	109.6	146.3	171.6	0.47
Svg,Opt$_{0}$	120.37	0.05	[118.67, 122.07]	100.6	126.3	100.6	122.3	126.3	
Svg,Opt$_{60}$	134.14	0.04	[132.62, 135.66]	107	139	107	135.3	139	0.02
Svg,Opt$_{120}$	141.44	0.07	[138.64, 144.24]	101.4	153.6	101.4	142.6	153.6	0.12
Svg,Opt$_{360}$	154.87	0.11	[150.05, 159.69]	109.6	180	109.6	155	180	0.31
Svg,Opt$_{600}$	153.44	0.09	[149.53, 157.35]	109.6	178.6	109.6	152.8	178.6	0.39
Ls,Ret$_{0}$	120.72	0.04	[119.35, 122.09]	100.6	125.9	100.6	121.9	125.9	
Ls,Ret$_{60}$	133.25	0.04	[131.74, 134.76]	105.9	138.4	105.9	134.8	138.4	0.04
Ls,Ret$_{120}$	141.92	0.07	[139.11, 144.73]	101.4	153.4	101.4	142.8	153.4	0.06
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Throughput for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	153.99	0.1	[149.63, 158.35]	109.6	181.4	109.6	154.9	181.4	0.29
Ls,Ret$_{600}$	153.88	0.1	[149.53, 158.23]	109.6	181.1	109.6	153.7	181.1	0.18
Ls,S$_{0}$	120.62	0.05	[118.91, 122.33]	100.6	125.8	100.6	122.3	125.8	
Ls,S$_{60}$	134.21	0.03	[133.07, 135.35]	112.8	138.8	112.8	134.7	138.8	0.02
Ls,S$_{120}$	142.22	0.07	[139.4, 145.04]	101.4	154.1	101.4	143.7	154.1	0.06
Ls,S$_{360}$	162.22	0.11	[157.17, 167.27]	109.6	197.3	109.6	159.3	197.3	0.33
Ls,S$_{600}$	163.22	0.12	[157.68, 168.76]	109.6	204.2	109.6	159.1	204.2	0.31
Ls,Gap$_{0}$	120.38	0.05	[118.68, 122.08]	100.6	126	100.6	122.2	126	
Ls,Gap$_{60}$	133.34	0.05	[131.45, 135.23]	102.7	138.8	102.7	134.7	138.8	0
Ls,Gap$_{120}$	141.44	0.08	[138.24, 144.64]	101.4	153.9	101.4	142.7	153.9	0.08
Ls,Gap$_{360}$	155.96	0.1	[151.55, 160.37]	109.6	182.5	109.6	156.3	182.5	0.24
Ls,Gap$_{600}$	155	0.1	[150.62, 159.38]	109.6	183.2	109.6	156.4	183.2	0.31
Ls,Opt$_{0}$	121.27	0.03	[120.24, 122.3]	101.4	125.9	101.4	122.5	125.9	
Ls,Opt$_{60}$	133.29	0.05	[131.4, 135.18]	102.7	138.8	102.7	134.9	138.8	0.04
Ls,Opt$_{120}$	142.41	0.07	[139.59, 145.23]	101.4	153.9	101.4	143.7	153.9	0.08
Ls,Opt$_{360}$	162.2	0.11	[157.15, 167.25]	109.6	200.3	109.6	158.2	200.3	0.31
Ls,Opt$_{600}$	160.3	0.11	[155.31, 165.29]	109.6	189.8	109.6	157.3	189.8	0.2
Ts,Ret$_{0}$	120.88	0.03	[119.85, 121.91]	101.4	125.8	101.4	122.1	125.8	
Ts,Ret$_{60}$	132.31	0.06	[130.06, 134.56]	102.7	138.3	102.7	134.7	138.3	0.08
Ts,Ret$_{120}$	141.67	0.07	[138.86, 144.48]	101.4	153.3	101.4	142.6	153.3	0.1
Ts,Ret$_{360}$	154.57	0.1	[150.2, 158.94]	109.6	179	109.6	154.4	179	0.29
Ts,Ret$_{600}$	155.05	0.1	[150.66, 159.44]	109.6	179.4	109.6	153.8	179.4	0.33
Ts,S$_{0}$	120.43	0.04	[119.07, 121.79]	101.4	125.9	101.4	122.1	125.9	
Ts,S$_{60}$	133.93	0.03	[132.79, 135.07]	107	138.8	107	134.7	138.8	0
Ts,S$_{120}$	142.58	0.07	[139.76, 145.4]	101.4	154.6	101.4	143.4	154.6	0.06
Ts,S$_{360}$	161.82	0.12	[156.33, 167.31]	109.6	201.2	109.6	158.7	201.2	0.29
Ts,S$_{600}$	162.57	0.11	[157.51, 167.63]	109.6	200.9	109.6	158.6	200.9	0.27
Ts,Gap$_{0}$	120.65	0.04	[119.28, 122.02]	101.4	125.8	101.4	121.8	125.8	
Ts,Gap$_{60}$	133.05	0.05	[131.17, 134.93]	102.7	139.3	102.7	135	139.3	0.06
Ts,Gap$_{120}$	141.32	0.08	[138.12, 144.52]	101.4	153.9	101.4	143	153.9	0.1
Ts,Gap$_{360}$	154.67	0.11	[149.86, 159.48]	109.6	183.3	109.6	155.1	183.3	0.37
Ts,Gap$_{600}$	155.08	0.1	[150.69, 159.47]	109.6	178.3	109.6	157.3	178.3	0.35
Ts,Opt$_{0}$	121.76	0.03	[120.73, 122.79]	101.4	126	101.4	122.6	126	
Ts,Opt$_{60}$	133.53	0.05	[131.64, 135.42]	102.7	139.3	102.7	135.2	139.3	0.04
Ts,Opt$_{120}$	141.61	0.08	[138.41, 144.81]	101.4	154.3	101.4	143.3	154.3	0.08
Ts,Opt$_{360}$	160.41	0.13	[154.51, 166.31]	109.6	204.3	109.6	159	204.3	0.31
Ts,Opt$_{600}$	161.83	0.12	[156.34, 167.32]	109.6	198.4	109.6	159.8	198.4	0.31
Opt,Opt$_{0}$	121.91	0.06	[119.84, 123.98]	100.6	149.7	100.6	122.4	149.7	
Opt,Opt$_{60}$	132.63	0.06	[130.38, 134.88]	102.7	149.7	102.7	134.3	149.7	0.06
Opt,Opt$_{120}$	141.13	0.07	[138.34, 143.92]	101.4	153.6	101.4	143.2	153.6	0.16
Opt,Opt$_{360}$	146.93	0.08	[143.6, 150.26]	109.6	170.1	109.6	147.6	170.1	0.33
Opt,Opt$_{600}$	148.13	0.07	[145.2, 151.06]	109.6	168.9	109.6	148.4	168.9	0.24

Table A.76: Box throughputs in the order picking system.

Performance ratios of throughput relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Prio,Ret$_{0}$	0.75	0.14	[0.72, 0.78]	0.6	1.09	0.6	0.72	1.09	0.04
Prio,Ret$_{60}$	0.84	0.14	[0.81, 0.87]	0.6	1.2	0.6	0.83	1.2	0.04
Prio,Ret$_{120}$	0.88	0.14	[0.85, 0.91]	0.6	1.29	0.6	0.89	1.29	0.02
Prio,Ret$_{360}$	0.9	0.12	[0.87, 0.93]	0.72	1.3	0.72	0.94	1.3	0.02
Prio,Ret$_{600}$	0.9	0.12	[0.87, 0.93]	0.72	1.3	0.72	0.93	1.3	0.02
Prio,S$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.75	1.09	0.04
Prio,S$_{60}$	0.83	0.15	[0.79, 0.87]	0.59	1.21	0.59	0.83	1.21	0.04
Prio,S$_{120}$	0.88	0.12	[0.85, 0.91]	0.6	1.01	0.6	0.9	1.01	0.02
Prio,S$_{360}$	0.93	0.13	[0.9, 0.96]	0.62	1.37	0.62	0.98	1.37	0.04
Prio,S$_{600}$	0.93	0.13	[0.9, 0.96]	0.62	1.35	0.62	0.98	1.35	0.06
Prio,Gap$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.08	0.6	0.73	1.08	0.04
Prio,Gap$_{60}$	0.83	0.15	[0.79, 0.87]	0.59	1.21	0.59	0.83	1.21	0.04
Prio,Gap$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.31	0.6	0.9	1.31	0.02
Prio,Gap$_{360}$	0.92	0.12	[0.89, 0.95]	0.62	1.31	0.62	0.96	1.31	0.02
Prio,Gap$_{600}$	0.92	0.12	[0.89, 0.95]	0.62	1.31	0.62	0.96	1.31	0.04
Prio,Opt$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.74	1.09	0.04
Prio,Opt$_{60}$	0.84	0.13	[0.81, 0.87]	0.65	1.18	0.65	0.84	1.18	0.04
Prio,Opt$_{120}$	0.89	0.12	[0.86, 0.92]	0.6	1.02	0.6	0.93	1.02	0.02
Prio,Opt$_{360}$	0.95	0.09	[0.93, 0.97]	0.62	1.01	0.62	0.99	1.01	0.1
Prio,Opt$_{600}$	0.94	0.1	[0.91, 0.97]	0.62	1.01	0.62	0.98	1.01	0.04
Seed,Ret$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.75	1.09	0.04
Seed,Ret$_{60}$	0.83	0.14	[0.8, 0.86]	0.65	1.18	0.65	0.84	1.18	0.04
Seed,Ret$_{120}$	0.88	0.12	[0.85, 0.91]	0.6	1.01	0.6	0.9	1.01	0.02
Seed,Ret$_{360}$	0.95	0.09	[0.93, 0.97]	0.62	1.02	0.62	0.99	1.02	0.06
Seed,Ret$_{600}$	0.96	0.11	[0.93, 0.99]	0.62	1.37	0.62	0.99	1.37	0.06
Seed,S$_{0}$	0.76	0.14	[0.73, 0.79]	0.59	1.09	0.59	0.73	1.09	0.04
Seed,S$_{60}$	0.83	0.15	[0.79, 0.87]	0.6	1.18	0.6	0.81	1.18	0.04
Seed,S$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.34	0.6	0.93	1.34	0.04
Seed,S$_{360}$	1.01	0.06	[0.99, 1.03]	0.97	1.37	0.97	1	1.37	0.29
Seed,S$_{600}$	1	0.01	[1, 1]	0.97	1.02	0.97	1	1.02	0.31
Seed,Gap$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.75	1.09	0.04
Seed,Gap$_{60}$	0.84	0.14	[0.81, 0.87]	0.67	1.2	0.67	0.83	1.2	0.04
Seed,Gap$_{120}$	0.88	0.12	[0.85, 0.91]	0.6	1	0.6	0.92	1	0
Seed,Gap$_{360}$	0.92	0.11	[0.89, 0.95]	0.62	1.01	0.62	0.98	1.01	0.02
Seed,Gap$_{600}$	0.93	0.12	[0.9, 0.96]	0.62	1.34	0.62	0.97	1.34	0.04
Seed,Opt$_{0}$	0.76	0.14	[0.73, 0.79]	0.59	1.09	0.59	0.76	1.09	0.04
Seed,Opt$_{60}$	0.84	0.14	[0.81, 0.87]	0.6	1.21	0.6	0.84	1.21	0.04
Seed,Opt$_{120}$	0.89	0.13	[0.86, 0.92]	0.6	1.24	0.6	0.9	1.24	0.04
Seed,Opt$_{360}$	1.01	0.09	[0.98, 1.04]	0.62	1.36	0.62	1	1.36	0.39
Seed,Opt$_{600}$	1.01	0.06	[0.99, 1.03]	0.93	1.36	0.93	1	1.36	0.43
Svg,Ret$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.76	1.09	0.04
Svg,Ret$_{60}$	0.84	0.14	[0.81, 0.87]	0.6	1.2	0.6	0.84	1.2	0.04
Svg,Ret$_{120}$	0.89	0.13	[0.86, 0.92]	0.6	1.24	0.6	0.92	1.24	0.04
Svg,Ret$_{360}$	0.93	0.1	[0.9, 0.96]	0.62	1.02	0.62	0.97	1.02	0.04
Svg,Ret$_{600}$	0.92	0.11	[0.89, 0.95]	0.62	1	0.62	0.97	1	0.02
Svg,S$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.75	1.09	0.04
Svg,S$_{60}$	0.84	0.14	[0.81, 0.87]	0.67	1.21	0.67	0.84	1.21	0.04
Svg,S$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.24	0.6	0.92	1.24	0.04
Svg,S$_{360}$	0.96	0.09	[0.94, 0.98]	0.62	1.01	0.62	1	1.01	0.08
Svg,S$_{600}$	0.95	0.09	[0.93, 0.97]	0.62	1.02	0.62	1	1.02	0.04
Svg,Gap$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.73	1.09	0.04
Svg,Gap$_{60}$	0.84	0.14	[0.81, 0.87]	0.65	1.21	0.65	0.84	1.21	0.04
Svg,Gap$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.33	0.6	0.89	1.33	0.04
Svg,Gap$_{360}$	0.93	0.12	[0.9, 0.96]	0.62	1.37	0.62	0.97	1.37	0.02
Svg,Gap$_{600}$	0.92	0.12	[0.89, 0.95]	0.62	1.32	0.62	0.94	1.32	0.02
Svg,Opt$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.74	1.09	0.04
Svg,Opt$_{60}$	0.84	0.14	[0.81, 0.87]	0.67	1.21	0.67	0.83	1.21	0.04
Svg,Opt$_{120}$	0.89	0.13	[0.86, 0.92]	0.6	1.24	0.6	0.9	1.24	0.04
Svg,Opt$_{360}$	0.96	0.08	[0.94, 0.98]	0.62	1.02	0.62	0.99	1.02	0.06
Svg,Opt$_{600}$	0.96	0.11	[0.93, 0.99]	0.62	1.37	0.62	0.99	1.37	0.08
Ls,Ret$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.74	1.09	0.04
Ls,Ret$_{60}$	0.84	0.14	[0.81, 0.87]	0.59	1.21	0.59	0.84	1.21	0.04
Ls,Ret$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.35	0.6	0.93	1.35	0.04
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of throughput relative to Ts,Opt$_{600}$ for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	0.96	0.09	[0.94, 0.98]	0.62	1.01	0.62	1	1.01	0.1
Ls,Ret$_{600}$	0.96	0.09	[0.94, 0.98]	0.62	1.02	0.62	0.99	1.02	0.12
Ls,S$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.73	1.09	0.04
Ls,S$_{60}$	0.84	0.14	[0.81, 0.87]	0.67	1.21	0.67	0.84	1.21	0.04
Ls,S$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.35	0.6	0.94	1.35	0.04
Ls,S$_{360}$	1	0.05	[0.99, 1.01]	0.93	1.36	0.93	1	1.36	0.24
Ls,S$_{600}$	1.01	0.06	[0.99, 1.03]	0.98	1.37	0.98	1	1.37	0.31
Ls,Gap$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.72	1.09	0.04
Ls,Gap$_{60}$	0.84	0.14	[0.81, 0.87]	0.65	1.18	0.65	0.81	1.18	0.04
Ls,Gap$_{120}$	0.88	0.12	[0.85, 0.91]	0.6	1.02	0.6	0.94	1.02	0.02
Ls,Gap$_{360}$	0.97	0.1	[0.94, 1]	0.62	1.36	0.62	1	1.36	0.12
Ls,Gap$_{600}$	0.97	0.1	[0.94, 1]	0.62	1.36	0.62	1	1.36	0.16
Ls,Opt$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.76	1.09	0.04
Ls,Opt$_{60}$	0.84	0.14	[0.81, 0.87]	0.65	1.18	0.65	0.82	1.18	0.04
Ls,Opt$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.36	0.6	0.94	1.36	0.04
Ls,Opt$_{360}$	1	0.05	[0.99, 1.01]	0.97	1.37	0.97	1	1.37	0.27
Ls,Opt$_{600}$	1	0.08	[0.98, 1.02]	0.62	1.37	0.62	1	1.37	0.24
Ts,Ret$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.75	1.09	0.04
Ts,Ret$_{60}$	0.83	0.15	[0.79, 0.87]	0.59	1.21	0.59	0.83	1.21	0.04
Ts,Ret$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.33	0.6	0.89	1.33	0.04
Ts,Ret$_{360}$	0.96	0.09	[0.94, 0.98]	0.62	1.01	0.62	0.99	1.01	0.08
Ts,Ret$_{600}$	0.97	0.1	[0.94, 1]	0.62	1.36	0.62	0.99	1.36	0.08
Ts,S$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.73	1.09	0.04
Ts,S$_{60}$	0.84	0.14	[0.81, 0.87]	0.67	1.21	0.67	0.84	1.21	0.04
Ts,S$_{120}$	0.89	0.14	[0.85, 0.93]	0.6	1.36	0.6	0.94	1.36	0.04
Ts,S$_{360}$	1	0.01	[1, 1]	0.98	1.03	0.98	1	1.03	0.24
Ts,S$_{600}$	1.01	0.05	[1, 1.02]	0.93	1.36	0.93	1	1.36	0.31
Ts,Gap$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.73	1.09	0.04
Ts,Gap$_{60}$	0.84	0.15	[0.8, 0.88]	0.6	1.21	0.6	0.83	1.21	0.04
Ts,Gap$_{120}$	0.88	0.12	[0.85, 0.91]	0.6	1.01	0.6	0.93	1.01	0.02
Ts,Gap$_{360}$	0.96	0.09	[0.94, 0.98]	0.62	1.02	0.62	1	1.02	0.14
Ts,Gap$_{600}$	0.96	0.08	[0.94, 0.98]	0.62	1.03	0.62	1	1.03	0.12
Ts,Opt$_{0}$	0.76	0.14	[0.73, 0.79]	0.6	1.09	0.6	0.75	1.09	0.04
Ts,Opt$_{60}$	0.84	0.14	[0.81, 0.87]	0.59	1.21	0.59	0.83	1.21	0.04
Ts,Opt$_{120}$	0.89	0.12	[0.86, 0.92]	0.6	1.02	0.6	0.94	1.02	0.02
Ts,Opt$_{360}$	0.99	0.07	[0.97, 1.01]	0.62	1.2	0.62	1	1.2	0.31
Ts,Opt$_{600}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{0}$	0.76	0.14	[0.73, 0.79]	0.59	1.09	0.59	0.73	1.09	0.02
Opt,Opt$_{60}$	0.83	0.15	[0.79, 0.87]	0.59	1.18	0.59	0.8	1.18	0.06
Opt,Opt$_{120}$	0.88	0.12	[0.85, 0.91]	0.6	1.09	0.6	0.89	1.09	0.02
Opt,Opt$_{360}$	0.92	0.11	[0.89, 0.95]	0.62	1.01	0.62	0.98	1.01	0.04
Opt,Opt$_{600}$	0.93	0.12	[0.9, 0.96]	0.62	1.34	0.62	0.97	1.34	0.04

Table A.77: Performance ratios of box throughput relative to Ts,Opt$_{600}$ in the order picking system.

Performance ratios of throughput relative to online version for $n = 625$ (50 samples)

Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Prio,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Ret$_{60}$	1.11	0.06	[1.09, 1.13]	0.91	1.34	0.91	1.11	1.34	0.02
Prio,Ret$_{120}$	1.17	0.08	[1.14, 1.2]	0.93	1.46	0.93	1.17	1.46	0.02
Prio,Ret$_{360}$	1.21	0.09	[1.18, 1.24]	0.93	1.51	0.93	1.19	1.51	0.02
Prio,Ret$_{600}$	1.21	0.08	[1.18, 1.24]	0.93	1.52	0.93	1.19	1.52	0.02
Prio,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,S$_{60}$	1.09	0.06	[1.07, 1.11]	0.87	1.29	0.87	1.11	1.29	0.06
Prio,S$_{120}$	1.17	0.07	[1.15, 1.19]	0.92	1.43	0.92	1.17	1.43	0.04
Prio,S$_{360}$	1.23	0.1	[1.2, 1.26]	0.92	1.62	0.92	1.22	1.62	0.04
Prio,S$_{600}$	1.23	0.1	[1.2, 1.26]	0.92	1.6	0.92	1.22	1.6	0.04
Prio,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Gap$_{60}$	1.1	0.07	[1.08, 1.12]	0.89	1.34	0.89	1.11	1.34	0.08
Prio,Gap$_{120}$	1.17	0.07	[1.15, 1.19]	0.93	1.43	0.93	1.18	1.43	0.02
Prio,Gap$_{360}$	1.22	0.09	[1.19, 1.25]	0.93	1.58	0.93	1.21	1.58	0.02
Prio,Gap$_{600}$	1.22	0.09	[1.19, 1.25]	0.93	1.54	0.93	1.21	1.54	0.02
Prio,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Prio,Opt$_{60}$	1.11	0.05	[1.09, 1.13]	0.93	1.35	0.93	1.11	1.35	0.02
Prio,Opt$_{120}$	1.17	0.08	[1.14, 1.2]	0.92	1.46	0.92	1.17	1.46	0.04
Prio,Opt$_{360}$	1.26	0.12	[1.22, 1.3]	0.92	1.67	0.92	1.26	1.67	0.04
Prio,Opt$_{600}$	1.25	0.11	[1.21, 1.29]	0.92	1.63	0.92	1.25	1.63	0.04
Seed,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Ret$_{60}$	1.1	0.04	[1.09, 1.11]	0.86	1.27	0.86	1.11	1.27	0.04
Seed,Ret$_{120}$	1.16	0.06	[1.14, 1.18]	0.92	1.25	0.92	1.16	1.25	0.04
Seed,Ret$_{360}$	1.27	0.11	[1.23, 1.31]	0.92	1.67	0.92	1.26	1.67	0.04
Seed,Ret$_{600}$	1.27	0.1	[1.23, 1.31]	0.93	1.66	0.93	1.25	1.66	0.02
Seed,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,S$_{60}$	1.09	0.06	[1.07, 1.11]	0.87	1.25	0.87	1.11	1.25	0.08
Seed,S$_{120}$	1.18	0.07	[1.16, 1.2]	0.92	1.42	0.92	1.17	1.42	0.02
Seed,S$_{360}$	1.35	0.13	[1.3, 1.4]	0.92	1.71	0.92	1.33	1.71	0.02
Seed,S$_{600}$	1.34	0.13	[1.29, 1.39]	0.92	1.68	0.92	1.36	1.68	0.04
Seed,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Gap$_{60}$	1.1	0.05	[1.08, 1.12]	0.87	1.29	0.87	1.11	1.29	0.04
Seed,Gap$_{120}$	1.17	0.07	[1.15, 1.19]	0.92	1.41	0.92	1.18	1.41	0.04
Seed,Gap$_{360}$	1.22	0.1	[1.19, 1.25]	0.92	1.56	0.92	1.2	1.56	0.04
Seed,Gap$_{600}$	1.23	0.09	[1.2, 1.26]	0.92	1.6	0.92	1.2	1.6	0.02
Seed,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Seed,Opt$_{60}$	1.11	0.04	[1.1, 1.12]	1	1.29	1	1.11	1.29	0
Seed,Opt$_{120}$	1.18	0.07	[1.16, 1.2]	0.92	1.44	0.92	1.17	1.44	0.02
Seed,Opt$_{360}$	1.35	0.14	[1.3, 1.4]	0.92	1.71	0.92	1.3	1.71	0.02
Seed,Opt$_{600}$	1.36	0.13	[1.31, 1.41]	0.92	1.71	0.92	1.36	1.71	0.02
Svg,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Ret$_{60}$	1.1	0.05	[1.08, 1.12]	0.9	1.34	0.9	1.11	1.34	0.02
Svg,Ret$_{120}$	1.17	0.07	[1.15, 1.19]	0.93	1.49	0.93	1.17	1.49	0.02
Svg,Ret$_{360}$	1.23	0.1	[1.2, 1.26]	0.92	1.64	0.92	1.24	1.64	0.04
Svg,Ret$_{600}$	1.22	0.11	[1.18, 1.26]	0.92	1.61	0.92	1.21	1.61	0.06
Svg,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,S$_{60}$	1.1	0.03	[1.09, 1.11]	1	1.27	1	1.11	1.27	0
Svg,S$_{120}$	1.16	0.06	[1.14, 1.18]	0.92	1.25	0.92	1.17	1.25	0.02
Svg,S$_{360}$	1.27	0.12	[1.23, 1.31]	0.92	1.68	0.92	1.25	1.68	0.04
Svg,S$_{600}$	1.26	0.11	[1.22, 1.3]	0.92	1.68	0.92	1.27	1.68	0.04
Svg,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Gap$_{60}$	1.11	0.04	[1.1, 1.12]	1	1.34	1	1.11	1.34	0
Svg,Gap$_{120}$	1.17	0.07	[1.15, 1.19]	0.93	1.44	0.93	1.17	1.44	0.02
Svg,Gap$_{360}$	1.23	0.1	[1.2, 1.26]	0.93	1.62	0.93	1.22	1.62	0.02
Svg,Gap$_{600}$	1.22	0.09	[1.19, 1.25]	0.93	1.53	0.93	1.2	1.53	0.02
Svg,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Svg,Opt$_{60}$	1.12	0.06	[1.1, 1.14]	0.87	1.35	0.87	1.11	1.35	0.02
Svg,Opt$_{120}$	1.18	0.08	[1.15, 1.21]	0.92	1.5	0.92	1.17	1.5	0.02
Svg,Opt$_{360}$	1.29	0.12	[1.25, 1.33]	0.91	1.69	0.91	1.28	1.69	0.04
Svg,Opt$_{600}$	1.28	0.11	[1.24, 1.32]	0.92	1.69	0.92	1.27	1.69	0.02
Ls,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Ret$_{60}$	1.11	0.06	[1.09, 1.13]	0.88	1.34	0.88	1.11	1.34	0.04
Ls,Ret$_{120}$	1.18	0.07	[1.16, 1.2]	0.92	1.5	0.92	1.18	1.5	0.02
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Performance ratios of throughput relative to online version for $n = 625$ (50 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ls,Ret$_{360}$	1.28	0.12	[1.24, 1.32]	0.92	1.67	0.92	1.28	1.67	0.04
Ls,Ret$_{600}$	1.28	0.12	[1.24, 1.32]	0.92	1.65	0.92	1.28	1.65	0.04
Ls,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,S$_{60}$	1.11	0.05	[1.09, 1.13]	0.94	1.34	0.94	1.11	1.34	0.02
Ls,S$_{120}$	1.18	0.08	[1.15, 1.21]	0.92	1.5	0.92	1.17	1.5	0.02
Ls,S$_{360}$	1.35	0.12	[1.3, 1.4]	0.92	1.69	0.92	1.35	1.69	0.02
Ls,S$_{600}$	1.36	0.13	[1.31, 1.41]	0.92	1.7	0.92	1.36	1.7	0.02
Ls,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Gap$_{60}$	1.11	0.04	[1.1, 1.12]	1	1.33	1	1.11	1.33	0
Ls,Gap$_{120}$	1.18	0.09	[1.15, 1.21]	0.92	1.49	0.92	1.17	1.49	0.04
Ls,Gap$_{360}$	1.3	0.11	[1.26, 1.34]	0.92	1.65	0.92	1.28	1.65	0.02
Ls,Gap$_{600}$	1.29	0.12	[1.25, 1.33]	0.92	1.69	0.92	1.28	1.69	0.02
Ls,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ls,Opt$_{60}$	1.1	0.05	[1.08, 1.12]	0.86	1.27	0.86	1.11	1.27	0.04
Ls,Opt$_{120}$	1.17	0.06	[1.15, 1.19]	0.92	1.33	0.92	1.18	1.33	0.02
Ls,Opt$_{360}$	1.34	0.12	[1.29, 1.39]	0.92	1.68	0.92	1.32	1.68	0.02
Ls,Opt$_{600}$	1.32	0.12	[1.28, 1.36]	0.92	1.69	0.92	1.29	1.69	0.02
Ts,Ret$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Ret$_{60}$	1.1	0.06	[1.08, 1.12]	0.86	1.28	0.86	1.11	1.28	0.08
Ts,Ret$_{120}$	1.17	0.07	[1.15, 1.19]	0.92	1.41	0.92	1.18	1.41	0.02
Ts,Ret$_{360}$	1.28	0.12	[1.24, 1.32]	0.92	1.68	0.92	1.28	1.68	0.04
Ts,Ret$_{600}$	1.28	0.11	[1.24, 1.32]	0.92	1.67	0.92	1.27	1.67	0.02
Ts,S$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,S$_{60}$	1.11	0.04	[1.1, 1.12]	1	1.34	1	1.11	1.34	0
Ts,S$_{120}$	1.19	0.07	[1.17, 1.21]	0.92	1.45	0.92	1.17	1.45	0.02
Ts,S$_{360}$	1.35	0.13	[1.3, 1.4]	0.91	1.71	0.91	1.34	1.71	0.04
Ts,S$_{600}$	1.35	0.12	[1.3, 1.4]	0.92	1.68	0.92	1.34	1.68	0.02
Ts,Gap$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Gap$_{60}$	1.1	0.06	[1.08, 1.12]	0.86	1.35	0.86	1.11	1.35	0.06
Ts,Gap$_{120}$	1.17	0.08	[1.14, 1.2]	0.92	1.46	0.92	1.17	1.46	0.04
Ts,Gap$_{360}$	1.28	0.12	[1.24, 1.32]	0.92	1.67	0.92	1.28	1.67	0.04
Ts,Gap$_{600}$	1.29	0.11	[1.25, 1.33]	0.92	1.68	0.92	1.29	1.68	0.04
Ts,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts,Opt$_{60}$	1.1	0.05	[1.08, 1.12]	0.86	1.27	0.86	1.11	1.27	0.04
Ts,Opt$_{120}$	1.16	0.06	[1.14, 1.18]	0.91	1.25	0.91	1.17	1.25	0.04
Ts,Opt$_{360}$	1.32	0.13	[1.27, 1.37]	0.91	1.71	0.91	1.3	1.71	0.04
Ts,Opt$_{600}$	1.33	0.13	[1.28, 1.38]	0.91	1.66	0.91	1.32	1.66	0.04
Opt,Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Opt,Opt$_{60}$	1.09	0.07	[1.07, 1.11]	0.86	1.29	0.86	1.11	1.29	0.06
Opt,Opt$_{120}$	1.16	0.08	[1.13, 1.19]	1	1.5	1	1.16	1.5	0.02
Opt,Opt$_{360}$	1.21	0.11	[1.17, 1.25]	0.92	1.67	0.92	1.2	1.67	0.04
Opt,Opt$_{600}$	1.22	0.1	[1.19, 1.25]	1	1.63	1	1.2	1.63	0.02

Table A.78: Performance ratios of box throughput relative to the online version of an algorithm in the order picking system.

A.3.2 Online Pickup and Delivery with Lookahead

Makespan for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
SRH$_{0}$	819.73	0.04	[813.27, 826.19]	753.3	943	755.8	814.8	928.55	
SRH$_{60}$	819.73	0.04	[813.27, 826.19]	748.8	908.1	749.25	813.8	902.9	0.43
SRH$_{120}$	814.83	0.04	[808.41, 821.25]	736.8	898.8	745.05	812.9	894.6	0.38
SRH$_{360}$	815.41	0.04	[808.98, 821.84]	753.3	938.3	757.1	811.6	906.8	0.52
SRH$_{600}$	813.69	0.04	[807.28, 820.1]	757.7	891.1	758.8	810.25	884.05	0.42
2OPT$_{0}$	819.73	0.04	[813.27, 826.19]	753.3	943	755.8	814.8	928.55	
2OPT$_{60}$	819.73	0.04	[813.27, 826.19]	748.8	908.1	749.25	813.8	902.9	0.43
2OPT$_{120}$	814.83	0.04	[808.41, 821.25]	736.8	898.8	745.05	812.9	894.6	0.38
2OPT$_{360}$	815.41	0.04	[808.98, 821.84]	753.3	938.3	757.1	811.6	906.8	0.52
2OPT$_{600}$	813.69	0.04	[807.28, 820.1]	757.7	891.1	758.8	810.25	884.05	0.42
SA$_{0}$	819.73	0.04	[813.27, 826.19]	753.3	943	755.8	814.8	928.55	
SA$_{60}$	819.73	0.04	[813.27, 826.19]	748.8	908.1	749.25	813.8	902.9	0.43
SA$_{120}$	814.83	0.04	[808.41, 821.25]	736.8	898.8	745.05	812.9	894.6	0.38
SA$_{360}$	815.41	0.04	[808.98, 821.84]	753.3	938.3	757.1	811.6	906.8	0.52
SA$_{600}$	813.69	0.04	[807.28, 820.1]	757.7	891.1	758.8	810.25	884.05	0.42
TS$_{0}$	813.35	0.03	[808.54, 818.16]	743.9	883	750.25	810.6	878.9	
TS$_{60}$	812.15	0.04	[805.75, 818.55]	752.4	919.2	753.35	808.9	905.55	0.48
TS$_{120}$	812.46	0.03	[807.66, 817.26]	756.7	924.6	761.15	807.35	909.85	0.49
TS$_{360}$	811.49	0.03	[806.69, 816.29]	735.7	870.4	749.25	810.1	864.75	0.45
TS$_{600}$	813.64	0.03	[808.83, 818.45]	735.9	921.9	739.15	812.55	895.25	0.53
OPT$_{0}$	819.12	0.03	[814.28, 823.96]	752.5	885.1	752.9	821.3	883.15	
OPT$_{60}$	817.59	0.03	[812.76, 822.42]	750.6	894.4	758.5	812.9	889.8	0.48
OPT$_{120}$	818.08	0.04	[811.63, 824.53]	749.2	897	749.95	819.6	895.1	0.5
OPT$_{360}$	816.9	0.03	[812.07, 821.73]	727.6	879.1	743.3	814.85	876.95	0.56
OPT$_{600}$	817.7	0.03	[812.87, 822.53]	745.1	910.8	746.9	816.5	906.65	0.46

Table A.79: Makespans in the pickup and delivery service.

Performance ratios of makespan relative to OPT for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
SRH$_{0}$	1	0.04	[0.99, 1.01]	0.91	1.12	0.92	1	1.12	0.54
SRH$_{60}$	1	0.04	[0.99, 1.01]	0.89	1.15	0.91	1	1.12	0.51
SRH$_{120}$	1	0.04	[0.99, 1.01]	0.92	1.13	0.92	0.99	1.11	0.59
SRH$_{360}$	1	0.03	[0.99, 1.01]	0.9	1.09	0.92	0.99	1.09	0.59
SRH$_{600}$	1	0.03	[0.99, 1.01]	0.88	1.08	0.91	1	1.07	0.59
2OPT$_{0}$	1	0.04	[0.99, 1.01]	0.91	1.12	0.92	1	1.12	0.54
2OPT$_{60}$	1	0.04	[0.99, 1.01]	0.89	1.15	0.91	1	1.12	0.51
2OPT$_{120}$	1	0.04	[0.99, 1.01]	0.92	1.13	0.92	0.99	1.11	0.59
2OPT$_{360}$	1	0.03	[0.99, 1.01]	0.9	1.09	0.92	0.99	1.09	0.59
2OPT$_{600}$	1	0.03	[0.99, 1.01]	0.88	1.08	0.91	1	1.07	0.59
SA$_{0}$	1	0.04	[0.99, 1.01]	0.91	1.12	0.92	1	1.12	0.54
SA$_{60}$	1	0.04	[0.99, 1.01]	0.89	1.15	0.91	1	1.12	0.51
SA$_{120}$	1	0.04	[0.99, 1.01]	0.92	1.13	0.92	0.99	1.11	0.59
SA$_{360}$	1	0.03	[0.99, 1.01]	0.9	1.09	0.92	0.99	1.09	0.59
SA$_{600}$	1	0.03	[0.99, 1.01]	0.88	1.08	0.91	1	1.07	0.59
TS$_{0}$	1	0.03	[0.99, 1.01]	0.91	1.07	0.92	1	1.06	0.59
TS$_{60}$	0.99	0.03	[0.98, 1]	0.89	1.14	0.9	0.99	1.11	0.61
TS$_{120}$	0.99	0.04	[0.98, 1]	0.88	1.14	0.91	0.99	1.11	0.6
TS$_{360}$	0.99	0.03	[0.98, 1]	0.89	1.05	0.91	0.99	1.05	0.64
TS$_{600}$	1	0.02	[1, 1]	0.88	1.05	0.91	1	1.04	0.52
OPT$_{0}$	1	0.03	[0.99, 1.01]	0.9	1.1	0.92	1	1.09	0.5
OPT$_{60}$	1	0.03	[0.99, 1.01]	0.91	1.1	0.92	1	1.09	0.51
OPT$_{120}$	1	0.05	[0.99, 1.01]	0.9	1.15	0.9	1	1.14	0.47
OPT$_{360}$	1	0.03	[0.99, 1.01]	0.89	1.07	0.92	1	1.07	0.54
OPT$_{600}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.80: Performance ratios of makespan relative to OPT in the pickup and delivery service.

Performance ratios of makespan relative to online version for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
SRH$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
SRH$_{60}$	1	0.04	[0.99, 1.01]	0.91	1.09	0.92	1	1.09	0.43
SRH$_{120}$	0.99	0.04	[0.98, 1]	0.89	1.1	0.9	1	1.09	0.39
SRH$_{360}$	1	0.03	[0.99, 1.01]	0.89	1.1	0.9	1	1.09	0.42
SRH$_{600}$	0.99	0.04	[0.98, 1]	0.91	1.07	0.91	1	1.07	0.39
2OPT$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{60}$	1	0.04	[0.99, 1.01]	0.91	1.09	0.92	1	1.09	0.43
2OPT$_{120}$	0.99	0.04	[0.98, 1]	0.89	1.1	0.9	1	1.09	0.39
2OPT$_{360}$	1	0.03	[0.99, 1.01]	0.89	1.1	0.9	1	1.09	0.42
2OPT$_{600}$	0.99	0.04	[0.98, 1]	0.91	1.07	0.91	1	1.07	0.39
SA$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{60}$	1	0.04	[0.99, 1.01]	0.91	1.09	0.92	1	1.09	0.43
SA$_{120}$	0.99	0.04	[0.98, 1]	0.89	1.1	0.9	1	1.09	0.39
SA$_{360}$	1	0.03	[0.99, 1.01]	0.89	1.1	0.9	1	1.09	0.42
SA$_{600}$	0.99	0.04	[0.98, 1]	0.91	1.07	0.91	1	1.07	0.39
TS$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
TS$_{60}$	1	0.03	[0.99, 1.01]	0.91	1.15	0.92	1	1.11	0.48
TS$_{120}$	1	0.03	[0.99, 1.01]	0.92	1.17	0.93	1	1.14	0.45
TS$_{360}$	1	0.03	[0.99, 1.01]	0.92	1.07	0.93	1	1.06	0.46
TS$_{600}$	1	0.03	[0.99, 1.01]	0.94	1.08	0.94	1	1.07	0.51
OPT$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
OPT$_{60}$	1	0.04	[0.99, 1.01]	0.92	1.1	0.92	1	1.09	0.48
OPT$_{120}$	1	0.05	[0.99, 1.01]	0.87	1.13	0.88	1	1.12	0.51
OPT$_{360}$	1	0.04	[0.99, 1.01]	0.91	1.1	0.91	1	1.09	0.47
OPT$_{600}$	1	0.03	[0.99, 1.01]	0.91	1.11	0.92	1	1.08	0.5

Table A.81: Performance ratios of makespan relative to the online version of an algorithm in the pickup and delivery service.

Distances for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
SRH$_{0}$	1677.4	0.07	[1654.27, 1700.53]	1420.3	2120.7	1421.8	1688.25	2022.65	
SRH$_{60}$	1645.6	0.07	[1622.91, 1668.29]	1318.2	2003.4	1353.65	1641.45	1953.05	0.35
SRH$_{120}$	1610.23	0.08	[1584.85, 1635.61]	1342.6	1898.3	1351.55	1592.05	1885.85	0.34
SRH$_{360}$	1611.39	0.07	[1589.17, 1633.61]	1364.6	1931.9	1369.95	1620.9	1930.6	0.51
SRH$_{600}$	1612.54	0.07	[1590.3, 1634.78]	1328.9	1859.6	1350.65	1609.65	1838.1	0.41
2OPT$_{0}$	1677.4	0.07	[1654.27, 1700.53]	1420.3	2120.7	1421.8	1688.25	2022.65	
2OPT$_{60}$	1645.6	0.07	[1622.91, 1668.29]	1318.2	2003.4	1353.65	1641.45	1953.05	0.35
2OPT$_{120}$	1610.23	0.08	[1584.85, 1635.61]	1342.6	1898.3	1351.55	1592.05	1885.85	0.34
2OPT$_{360}$	1611.39	0.07	[1589.17, 1633.61]	1364.6	1931.9	1369.95	1620.9	1930.6	0.51
2OPT$_{600}$	1612.54	0.07	[1590.3, 1634.78]	1328.9	1859.6	1350.65	1609.65	1838.1	0.41
SA$_{0}$	1677.4	0.07	[1654.27, 1700.53]	1420.3	2120.7	1421.8	1688.25	2022.65	
SA$_{60}$	1645.6	0.07	[1622.91, 1668.29]	1318.2	2003.4	1353.65	1641.45	1953.05	0.35
SA$_{120}$	1610.23	0.08	[1584.85, 1635.61]	1342.6	1898.3	1351.55	1592.05	1885.85	0.34
SA$_{360}$	1611.39	0.07	[1589.17, 1633.61]	1364.6	1931.9	1369.95	1620.9	1930.6	0.51
SA$_{600}$	1612.54	0.07	[1590.3, 1634.78]	1328.9	1859.6	1350.65	1609.65	1838.1	0.41
TS$_{0}$	1759.03	0.07	[1734.77, 1783.29]	1481.9	2059.4	1483.35	1753.5	2037.85	
TS$_{60}$	1756.27	0.08	[1728.59, 1783.95]	1439	2090.4	1439.1	1752.2	2070.75	0.44
TS$_{120}$	1745	0.1	[1710.63, 1779.37]	1409.4	2332.6	1414.4	1733.3	2241.65	0.42
TS$_{360}$	1706.83	0.09	[1676.57, 1737.09]	1315.9	2110.7	1351.65	1693.5	2092.05	0.4
TS$_{600}$	1698.68	0.08	[1671.91, 1725.45]	1363.9	2045	1401.85	1703.9	2044.15	0.5
OPT$_{0}$	1637.32	0.08	[1611.52, 1663.12]	1332.3	2077	1334.2	1633.45	2012.35	
OPT$_{60}$	1669.62	0.08	[1643.31, 1695.93]	1279.6	1972.8	1322.8	1672.4	1956.05	0.59
OPT$_{120}$	1695.97	0.09	[1665.9, 1726.04]	1362.4	2134.7	1364.95	1688.15	2098.35	0.52
OPT$_{360}$	1665.57	0.1	[1632.76, 1698.38]	1287.8	2035.5	1299.3	1648.2	2026.25	0.4
OPT$_{600}$	1660.2	0.08	[1634.04, 1686.36]	1334.1	2024.4	1372.75	1665.1	1992.45	0.49

Table A.82: Distances in the pickup and delivery service.

Performance ratios of distance relative to OPT for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
SRH$_0$	1.01	0.07	[1, 1.02]	0.86	1.23	0.86	1.01	1.22	0.46
SRH$_{60}$	0.99	0.08	[0.97, 1.01]	0.78	1.17	0.81	0.98	1.16	0.57
SRH$_{120}$	0.97	0.08	[0.95, 0.99]	0.72	1.16	0.77	0.98	1.14	0.62
SRH$_{360}$	0.97	0.07	[0.96, 0.98]	0.8	1.14	0.8	0.97	1.14	0.69
SRH$_{600}$	0.97	0.07	[0.96, 0.98]	0.82	1.19	0.82	0.98	1.16	0.66
2OPT$_0$	1.01	0.07	[1, 1.02]	0.86	1.23	0.86	1.01	1.22	0.46
2OPT$_{60}$	0.99	0.08	[0.97, 1.01]	0.78	1.17	0.81	0.98	1.16	0.57
2OPT$_{120}$	0.97	0.08	[0.95, 0.99]	0.72	1.16	0.77	0.98	1.14	0.62
2OPT$_{360}$	0.97	0.07	[0.96, 0.98]	0.8	1.14	0.8	0.97	1.14	0.69
2OPT$_{600}$	0.97	0.07	[0.96, 0.98]	0.82	1.19	0.82	0.98	1.16	0.66
SA$_0$	1.01	0.07	[1, 1.02]	0.86	1.23	0.86	1.01	1.22	0.46
SA$_{60}$	0.99	0.08	[0.97, 1.01]	0.78	1.17	0.81	0.98	1.16	0.57
SA$_{120}$	0.97	0.08	[0.95, 0.99]	0.72	1.16	0.77	0.98	1.14	0.62
SA$_{360}$	0.97	0.07	[0.96, 0.98]	0.8	1.14	0.8	0.97	1.14	0.69
SA$_{600}$	0.97	0.07	[0.96, 0.98]	0.82	1.19	0.82	0.98	1.16	0.66
TS$_0$	1.06	0.07	[1.05, 1.07]	0.88	1.25	0.89	1.06	1.24	0.14
TS$_{60}$	1.06	0.07	[1.05, 1.07]	0.84	1.24	0.85	1.05	1.24	0.18
TS$_{120}$	1.05	0.08	[1.03, 1.07]	0.86	1.3	0.87	1.05	1.28	0.28
TS$_{360}$	1.03	0.07	[1.02, 1.04]	0.9	1.22	0.9	1.02	1.21	0.39
TS$_{600}$	1.02	0.02	[1.02, 1.02]	0.96	1.1	0.97	1.02	1.09	0.11
OPT$_0$	0.99	0.07	[0.98, 1]	0.83	1.15	0.83	0.99	1.15	0.56
OPT$_{60}$	1.01	0.08	[0.99, 1.03]	0.83	1.24	0.84	1.01	1.21	0.46
OPT$_{120}$	1.03	0.1	[1.01, 1.05]	0.76	1.34	0.77	1.03	1.28	0.35
OPT$_{360}$	1	0.07	[0.99, 1.01]	0.87	1.21	0.88	1	1.19	0.55
OPT$_{600}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.83: Performance ratios of distance relative to OPT in the pickup and delivery service.

Performance ratios of distance relative to online version for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
SRH$_0$	1	0	[1, 1]	1	1	1	1	1	0
SRH$_{60}$	0.98	0.07	[0.97, 0.99]	0.84	1.19	0.85	0.97	1.15	0.35
SRH$_{120}$	0.96	0.07	[0.95, 0.97]	0.78	1.17	0.8	0.96	1.14	0.27
SRH$_{360}$	0.96	0.06	[0.95, 0.97]	0.8	1.17	0.81	0.96	1.12	0.26
SRH$_{600}$	0.96	0.06	[0.95, 0.97]	0.82	1.09	0.83	0.96	1.09	0.35
2OPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{60}$	0.98	0.07	[0.97, 0.99]	0.84	1.19	0.85	0.97	1.15	0.35
2OPT$_{120}$	0.96	0.07	[0.95, 0.97]	0.78	1.17	0.8	0.96	1.14	0.27
2OPT$_{360}$	0.96	0.06	[0.95, 0.97]	0.8	1.17	0.81	0.96	1.12	0.26
2OPT$_{600}$	0.96	0.06	[0.95, 0.97]	0.82	1.09	0.83	0.96	1.09	0.35
SA$_0$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{60}$	0.98	0.07	[0.97, 0.99]	0.84	1.19	0.85	0.97	1.15	0.35
SA$_{120}$	0.96	0.07	[0.95, 0.97]	0.78	1.17	0.8	0.96	1.14	0.27
SA$_{360}$	0.96	0.06	[0.95, 0.97]	0.8	1.17	0.81	0.96	1.12	0.26
SA$_{600}$	0.96	0.06	[0.95, 0.97]	0.82	1.09	0.83	0.96	1.09	0.35
TS$_0$	1	0	[1, 1]	1	1	1	1	1	0
TS$_{60}$	1	0.07	[0.99, 1.01]	0.86	1.17	0.87	0.99	1.16	0.44
TS$_{120}$	0.99	0.09	[0.97, 1.01]	0.85	1.28	0.85	0.98	1.25	0.44
TS$_{360}$	0.97	0.08	[0.95, 0.99]	0.82	1.21	0.82	0.96	1.19	0.32
TS$_{600}$	0.97	0.07	[0.96, 0.98]	0.85	1.15	0.85	0.96	1.15	0.26
OPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
OPT$_{60}$	1.02	0.08	[1, 1.04]	0.76	1.27	0.81	1.01	1.27	0.59
OPT$_{120}$	1.04	0.11	[1.02, 1.06]	0.75	1.44	0.78	1.04	1.41	0.6
OPT$_{360}$	1.02	0.07	[1.01, 1.03]	0.83	1.19	0.84	1.01	1.18	0.62
OPT$_{600}$	1.02	0.07	[1.01, 1.03]	0.87	1.21	0.87	1.01	1.2	0.56

Table A.84: Performance ratios of distance relative to the online version of an algorithm in the pickup and delivery service.

Tardiness for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
SRH$_0$	25.8	0.48	[23.36, 28.24]	7	75.6	7.45	22.75	72.15	
SRH$_{60}$	27.39	0.51	[24.64, 30.14]	6.8	76	7.35	24.2	72.55	0.53
SRH$_{120}$	26.21	0.41	[24.09, 28.33]	10.4	54.7	10.9	23.4	54.55	0.45
SRH$_{360}$	26.75	0.53	[23.96, 29.54]	8.3	120.9	9.15	24.85	93	0.54
SRH$_{600}$	26.28	0.4	[24.21, 28.35]	6.8	62.4	6.9	25.05	60	0.4
2OPT$_0$	25.8	0.48	[23.36, 28.24]	7	75.6	7.45	22.75	72.15	
2OPT$_{60}$	27.39	0.51	[24.64, 30.14]	6.8	76	7.35	24.2	72.55	0.53
2OPT$_{120}$	26.21	0.41	[24.09, 28.33]	10.4	54.7	10.9	23.4	54.55	0.45
2OPT$_{360}$	26.75	0.53	[23.96, 29.54]	8.3	120.9	9.15	24.85	93	0.54
2OPT$_{600}$	26.28	0.4	[24.21, 28.35]	6.8	62.4	6.9	25.05	60	0.4
SA$_0$	25.8	0.48	[23.36, 28.24]	7	75.6	7.45	22.75	72.15	
SA$_{60}$	27.39	0.51	[24.64, 30.14]	6.8	76	7.35	24.2	72.55	0.53
SA$_{120}$	26.21	0.41	[24.09, 28.33]	10.4	54.7	10.9	23.4	54.55	0.45
SA$_{360}$	26.75	0.53	[23.96, 29.54]	8.3	120.9	9.15	24.85	93	0.54
SA$_{600}$	26.28	0.4	[24.21, 28.35]	6.8	62.4	6.9	25.05	60	0.4
TS$_0$	21.08	0.43	[19.29, 22.87]	5.5	43.5	6.2	19.6	43.15	
TS$_{60}$	21.77	0.39	[20.1, 23.44]	6.3	45.9	6.85	21.4	44.4	0.51
TS$_{120}$	21.66	0.42	[19.87, 23.45]	7.9	58.8	8.2	19.6	56.5	0.5
TS$_{360}$	21.04	0.4	[19.38, 22.7]	8.8	55.4	8.8	19.35	50.1	0.55
TS$_{600}$	21.96	0.5	[19.8, 24.12]	3.9	59	4.3	19.75	57.15	0.44
OPT$_0$	20.74	0.38	[19.19, 22.29]	7.3	48.8	7.6	20.05	43.65	
OPT$_{60}$	20.81	0.43	[19.05, 22.57]	6.5	48.3	7	19.25	47.35	0.5
OPT$_{120}$	21.75	0.41	[19.99, 23.51]	8.7	58.8	9.05	20.15	55.15	0.51
OPT$_{360}$	21.52	0.4	[19.82, 23.22]	8.5	56.1	8.95	20.4	51.6	0.57
OPT$_{600}$	22.33	0.49	[20.17, 24.49]	4.3	59.3	4.5	20.15	57.2	0.46

Table A.85: Tardinesses in the pickup and delivery service.

Performance ratios of tardiness relative to OPT for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
SRH$_0$	1.32	0.52	[1.18, 1.46]	0.42	3.84	0.46	1.12	3.65	0.39
SRH$_{60}$	1.44	0.61	[1.27, 1.61]	0.21	4.28	0.29	1.19	4.18	0.36
SRH$_{120}$	1.4	0.64	[1.22, 1.58]	0.3	6.16	0.4	1.21	5.84	0.37
SRH$_{360}$	1.34	0.49	[1.21, 1.47]	0.46	4.84	0.46	1.2	4.01	0.35
SRH$_{600}$	1.4	0.65	[1.22, 1.58]	0.4	5.76	0.43	1.17	5.55	0.35
2OPT$_0$	1.32	0.52	[1.18, 1.46]	0.42	3.84	0.46	1.12	3.65	0.39
2OPT$_{60}$	1.44	0.61	[1.27, 1.61]	0.21	4.28	0.29	1.19	4.18	0.36
2OPT$_{120}$	1.4	0.64	[1.22, 1.58]	0.3	6.16	0.4	1.21	5.84	0.37
2OPT$_{360}$	1.34	0.49	[1.21, 1.47]	0.46	4.84	0.46	1.2	4.01	0.35
2OPT$_{600}$	1.4	0.65	[1.22, 1.58]	0.4	5.76	0.43	1.17	5.55	0.35
SA$_0$	1.32	0.52	[1.18, 1.46]	0.42	3.84	0.46	1.12	3.65	0.39
SA$_{60}$	1.44	0.61	[1.27, 1.61]	0.21	4.28	0.29	1.19	4.18	0.36
SA$_{120}$	1.4	0.64	[1.22, 1.58]	0.3	6.16	0.4	1.21	5.84	0.37
SA$_{360}$	1.34	0.49	[1.21, 1.47]	0.46	4.84	0.46	1.2	4.01	0.35
SA$_{600}$	1.4	0.65	[1.22, 1.58]	0.4	5.76	0.43	1.17	5.55	0.35
TS$_0$	1.07	0.48	[0.97, 1.17]	0.33	2.68	0.33	0.94	2.53	0.57
TS$_{60}$	1.11	0.46	[1.01, 1.21]	0.34	2.89	0.34	1.03	2.75	0.48
TS$_{120}$	1.1	0.48	[1, 1.2]	0.4	3.98	0.41	0.97	3.2	0.54
TS$_{360}$	1.1	0.6	[0.97, 1.23]	0.23	5.91	0.37	0.92	4.51	0.55
TS$_{600}$	0.98	0.05	[0.97, 0.99]	0.74	1.07	0.76	0.99	1.06	0.63
OPT$_0$	1.09	0.55	[0.97, 1.21]	0.19	4.3	0.26	0.95	3.81	0.52
OPT$_{60}$	1.07	0.47	[0.97, 1.17]	0.12	3.38	0.16	0.94	2.9	0.54
OPT$_{120}$	1.25	0.8	[1.05, 1.45]	0.16	8.57	0.22	1.01	6.33	0.47
OPT$_{360}$	1.12	0.63	[0.98, 1.26]	0.25	6.4	0.38	0.96	4.86	0.53
OPT$_{600}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.86: Performance ratios of tardiness relative to OPT in the pickup and delivery service.

Performance ratios of tardiness relative to online version for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
$\textsc{Srh}_{0}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Srh}_{60}$	1.22	0.6	[1.08, 1.36]	0.23	5.16	0.26	1.06	4.11	0.53
$\textsc{Srh}_{120}$	1.17	0.49	[1.06, 1.28]	0.33	2.9	0.35	0.99	2.76	0.49
$\textsc{Srh}_{360}$	1.14	0.46	[1.04, 1.24]	0.24	2.95	0.36	1.03	2.84	0.52
$\textsc{Srh}_{600}$	1.16	0.48	[1.05, 1.27]	0.36	2.93	0.38	1.07	2.88	0.55
$2\textsc{Opt}_{0}$	1	0	[1, 1]	1	1	1	1	1	0
$2\textsc{Opt}_{60}$	1.22	0.6	[1.08, 1.36]	0.23	5.16	0.26	1.06	4.11	0.53
$2\textsc{Opt}_{120}$	1.17	0.49	[1.06, 1.28]	0.33	2.9	0.35	0.99	2.76	0.49
$2\textsc{Opt}_{360}$	1.14	0.46	[1.04, 1.24]	0.24	2.95	0.36	1.03	2.84	0.52
$2\textsc{Opt}_{600}$	1.16	0.48	[1.05, 1.27]	0.36	2.93	0.38	1.07	2.88	0.55
$\textsc{Sa}_{0}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Sa}_{60}$	1.22	0.6	[1.08, 1.36]	0.23	5.16	0.26	1.06	4.11	0.53
$\textsc{Sa}_{120}$	1.17	0.49	[1.06, 1.28]	0.33	2.9	0.35	0.99	2.76	0.49
$\textsc{Sa}_{360}$	1.14	0.46	[1.04, 1.24]	0.24	2.95	0.36	1.03	2.84	0.52
$\textsc{Sa}_{600}$	1.16	0.48	[1.05, 1.27]	0.36	2.93	0.38	1.07	2.88	0.55
$\textsc{Ts}_{0}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Ts}_{60}$	1.15	0.48	[1.04, 1.26]	0.47	3.4	0.47	1.02	3.37	0.51
$\textsc{Ts}_{120}$	1.12	0.38	[1.04, 1.2]	0.42	2.25	0.44	1.04	2.17	0.53
$\textsc{Ts}_{360}$	1.12	0.41	[1.03, 1.21]	0.28	2.51	0.33	1.08	2.49	0.55
$\textsc{Ts}_{600}$	1.13	0.48	[1.02, 1.24]	0.37	3.06	0.38	1.04	3.04	0.53
$\textsc{Opt}_{0}$	1	0	[1, 1]	1	1	1	1	1	0
$\textsc{Opt}_{60}$	1.07	0.43	[0.98, 1.16]	0.35	2.75	0.37	1	2.66	0.5
$\textsc{Opt}_{120}$	1.2	0.58	[1.06, 1.34]	0.3	3.77	0.3	0.99	3.71	0.48
$\textsc{Opt}_{360}$	1.12	0.39	[1.03, 1.21]	0.39	2.45	0.4	1.08	2.35	0.57
$\textsc{Opt}_{600}$	1.16	0.56	[1.03, 1.29]	0.23	5.16	0.27	1.05	4.11	0.52

Table A.87: Performance ratios of tardiness relative to the online version of an algorithm in the pickup and delivery service.

Maximum tardiness for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
$\textsc{Srh}_{0}$	172.29	0.33	[161.09, 183.49]	89	409.3	92.1	161.3	372.35	
$\textsc{Srh}_{60}$	178.52	0.34	[166.56, 190.48]	88.3	342.6	88.4	163.9	334.6	0.56
$\textsc{Srh}_{120}$	176.33	0.27	[166.95, 185.71]	94.4	316	96.4	174.25	305.5	0.51
$\textsc{Srh}_{360}$	186.76	0.31	[175.36, 198.16]	87.3	401.8	88.7	179.4	389.5	0.56
$\textsc{Srh}_{600}$	181.5	0.32	[170.06, 192.94]	82.5	351.2	82.7	164.05	345.55	0.43
$2\textsc{Opt}_{0}$	172.29	0.33	[161.09, 183.49]	89	409.3	92.1	161.3	372.35	
$2\textsc{Opt}_{60}$	178.52	0.34	[166.56, 190.48]	88.3	342.6	88.4	163.9	334.6	0.56
$2\textsc{Opt}_{120}$	176.33	0.27	[166.95, 185.71]	94.4	316	96.4	174.25	305.5	0.51
$2\textsc{Opt}_{360}$	186.76	0.31	[175.36, 198.16]	87.3	401.8	88.7	179.4	389.5	0.56
$2\textsc{Opt}_{600}$	181.5	0.32	[170.06, 192.94]	82.5	351.2	82.7	164.05	345.55	0.43
$\textsc{Sa}_{0}$	172.29	0.33	[161.09, 183.49]	89	409.3	92.1	161.3	372.35	
$\textsc{Sa}_{60}$	178.52	0.34	[166.56, 190.48]	88.3	342.6	88.4	163.9	334.6	0.56
$\textsc{Sa}_{120}$	176.33	0.27	[166.95, 185.71]	94.4	316	96.4	174.25	305.5	0.51
$\textsc{Sa}_{360}$	186.76	0.31	[175.36, 198.16]	87.3	401.8	88.7	179.4	389.5	0.56
$\textsc{Sa}_{600}$	181.5	0.32	[170.06, 192.94]	82.5	351.2	82.7	164.05	345.55	0.43
$\textsc{Ts}_{0}$	139.76	0.29	[131.78, 147.74]	81.3	306.3	81.95	132.1	281.1	
$\textsc{Ts}_{60}$	142.93	0.24	[136.17, 149.69]	55.1	266.8	61.95	137.05	249.05	0.55
$\textsc{Ts}_{120}$	153.52	0.29	[144.75, 162.29]	82.6	298.4	83	145.75	280.95	0.59
$\textsc{Ts}_{360}$	144.71	0.29	[136.44, 152.98]	70.5	296.5	71.3	138.35	287.75	0.39
$\textsc{Ts}_{600}$	143.5	0.33	[134.17, 152.83]	51.5	353.9	52.55	133.45	307	0.42
$\textsc{Opt}_{0}$	144.53	0.23	[137.98, 151.08]	78.4	251.2	84.1	139.7	242.3	
$\textsc{Opt}_{60}$	136.93	0.26	[129.92, 143.94]	55.1	243.1	69.6	127.85	234.3	0.43
$\textsc{Opt}_{120}$	149.16	0.3	[140.35, 157.97]	81.2	319.9	81.65	144	314.4	0.57
$\textsc{Opt}_{360}$	148.55	0.3	[139.77, 157.33]	70.5	304.1	74.85	143.05	300.3	0.38
$\textsc{Opt}_{600}$	145.55	0.32	[136.38, 154.72]	53.3	353.9	53.45	140.4	313.95	0.42

Table A.88: Maximum tardinesses in the pickup and delivery service.

Performance ratios of maximum tardiness relative to OPT for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
SRH$_0$	1.28	0.4	[1.18, 1.38]	0.42	3.27	0.49	1.17	2.91	0.32
SRH$_{60}$	1.34	0.43	[1.23, 1.45]	0.27	3.23	0.36	1.22	3.08	0.3
SRH$_{120}$	1.35	0.52	[1.21, 1.49]	0.37	5.09	0.44	1.21	4.83	0.34
SRH$_{360}$	1.39	0.4	[1.28, 1.5]	0.41	3.32	0.47	1.32	3.16	0.25
SRH$_{600}$	1.39	0.56	[1.24, 1.54]	0.41	6.59	0.49	1.19	5.14	0.29
2OPT$_0$	1.28	0.4	[1.18, 1.38]	0.42	3.27	0.49	1.17	2.91	0.32
2OPT$_{60}$	1.34	0.43	[1.23, 1.45]	0.27	3.23	0.36	1.22	3.08	0.3
2OPT$_{120}$	1.35	0.52	[1.21, 1.49]	0.37	5.09	0.44	1.21	4.83	0.34
2OPT$_{360}$	1.39	0.4	[1.28, 1.5]	0.41	3.32	0.47	1.32	3.16	0.25
2OPT$_{600}$	1.39	0.56	[1.24, 1.54]	0.41	6.59	0.49	1.19	5.14	0.29
SA$_0$	1.28	0.4	[1.18, 1.38]	0.42	3.27	0.49	1.17	2.91	0.32
SA$_{60}$	1.34	0.43	[1.23, 1.45]	0.27	3.23	0.36	1.22	3.08	0.3
SA$_{120}$	1.35	0.52	[1.21, 1.49]	0.37	5.09	0.44	1.21	4.83	0.34
SA$_{360}$	1.39	0.4	[1.28, 1.5]	0.41	3.32	0.47	1.32	3.16	0.25
SA$_{600}$	1.39	0.56	[1.24, 1.54]	0.41	6.59	0.49	1.19	5.14	0.29
TS$_0$	1.04	0.38	[0.96, 1.12]	0.31	2.9	0.4	0.97	2.7	0.53
TS$_{60}$	1.07	0.43	[0.98, 1.16]	0.43	3.93	0.44	1	3.08	0.5
TS$_{120}$	1.15	0.44	[1.05, 1.25]	0.26	4.57	0.37	1.06	3.49	0.44
TS$_{360}$	1.09	0.54	[0.97, 1.21]	0.45	5.56	0.45	0.99	3.99	0.51
TS$_{600}$	0.99	0.07	[0.98, 1]	0.68	1.28	0.69	1	1.22	0.2
OPT$_0$	1.1	0.48	[1, 1.2]	0.26	4.71	0.39	0.99	3.79	0.51
OPT$_{60}$	1.02	0.39	[0.94, 1.1]	0.44	3.1	0.45	0.93	2.88	0.58
OPT$_{120}$	1.14	0.5	[1.03, 1.25]	0.4	4.57	0.42	1.05	3.95	0.44
OPT$_{360}$	1.12	0.53	[1, 1.24]	0.45	5.56	0.45	1.02	4.02	0.49
OPT$_{600}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.89: Performance ratios of maximum tardiness relative to OPT in the pickup and delivery service.

Performance ratios of maximum tardiness relative to online version for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
SRH$_0$	1	0	[1, 1]	1	1	1	1	1	0
SRH$_{60}$	1.12	0.4	[1.03, 1.21]	0.32	2.31	0.37	1.08	2.24	0.56
SRH$_{120}$	1.11	0.39	[1.02, 1.2]	0.36	3.06	0.41	1.01	3.01	0.52
SRH$_{360}$	1.16	0.35	[1.08, 1.24]	0.35	2.48	0.43	1.04	2.45	0.55
SRH$_{600}$	1.15	0.43	[1.05, 1.25]	0.43	3.48	0.46	1.03	2.97	0.53
2OPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{60}$	1.12	0.4	[1.03, 1.21]	0.32	2.31	0.37	1.08	2.24	0.56
2OPT$_{120}$	1.11	0.39	[1.02, 1.2]	0.36	3.06	0.41	1.01	3.01	0.52
2OPT$_{360}$	1.16	0.35	[1.08, 1.24]	0.35	2.48	0.43	1.04	2.45	0.55
2OPT$_{600}$	1.15	0.43	[1.05, 1.25]	0.43	3.48	0.46	1.03	2.97	0.53
SA$_0$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{60}$	1.12	0.4	[1.03, 1.21]	0.32	2.31	0.37	1.08	2.24	0.56
SA$_{120}$	1.11	0.39	[1.02, 1.2]	0.36	3.06	0.41	1.01	3.01	0.52
SA$_{360}$	1.16	0.35	[1.08, 1.24]	0.35	2.48	0.43	1.04	2.45	0.55
SA$_{600}$	1.15	0.43	[1.05, 1.25]	0.43	3.48	0.46	1.03	2.97	0.53
TS$_0$	1	0	[1, 1]	1	1	1	1	1	0
TS$_{60}$	1.08	0.32	[1.01, 1.15]	0.47	2.19	0.51	1.03	2.09	0.55
TS$_{120}$	1.16	0.34	[1.08, 1.24]	0.45	2.52	0.48	1.11	2.45	0.63
TS$_{360}$	1.1	0.36	[1.02, 1.18]	0.44	2.88	0.45	1.03	2.5	0.53
TS$_{600}$	1.08	0.39	[1, 1.16]	0.34	3.27	0.37	0.98	2.64	0.5
OPT$_0$	1	0	[1, 1]	1	1	1	1	1	0
OPT$_{60}$	0.99	0.31	[0.93, 1.05]	0.36	2.08	0.39	0.94	1.99	0.43
OPT$_{120}$	1.08	0.35	[1.01, 1.15]	0.53	2.15	0.54	0.99	2.03	0.49
OPT$_{360}$	1.07	0.34	[1, 1.14]	0.47	2.03	0.48	0.99	2.02	0.49
OPT$_{600}$	1.06	0.41	[0.97, 1.15]	0.21	3.9	0.28	1.01	2.91	0.51

Table A.90: Performance ratios of maximum tardiness relative to the online version of an algorithm in the pickup and delivery service.

Utilization for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
Srh$_{0}$	0.93	0.05	[0.92, 0.94]	0.9	1	0.9	0.9	1	
Srh$_{60}$	0.98	0.04	[0.97, 0.99]	0.9	1	0.9	1	1	0.01
Srh$_{120}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.05
Srh$_{360}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.04
Srh$_{600}$	0.99	0.04	[0.98, 1]	0.9	1	0.9	1	1	0.09
2Opt$_{0}$	0.93	0.05	[0.92, 0.94]	0.9	1	0.9	0.9	1	
2Opt$_{60}$	0.98	0.04	[0.97, 0.99]	0.9	1	0.9	1	1	0.01
2Opt$_{120}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.05
2Opt$_{360}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.04
2Opt$_{600}$	0.99	0.04	[0.98, 1]	0.9	1	0.9	1	1	0.09
Sa$_{0}$	0.93	0.05	[0.92, 0.94]	0.9	1	0.9	0.9	1	
Sa$_{60}$	0.98	0.04	[0.97, 0.99]	0.9	1	0.9	1	1	0.01
Sa$_{120}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.05
Sa$_{360}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.04
Sa$_{600}$	0.99	0.04	[0.98, 1]	0.9	1	0.9	1	1	0.09
Ts$_{0}$	0.91	0.04	[0.9, 0.92]	0.8	1	0.85	0.9	1	
Ts$_{60}$	0.96	0.05	[0.95, 0.97]	0.9	1	0.9	1	1	0.04
Ts$_{120}$	0.98	0.04	[0.97, 0.99]	0.9	1	0.9	1	1	0.09
Ts$_{360}$	0.99	0.04	[0.98, 1]	0.9	1	0.9	1	1	0.07
Ts$_{600}$	0.99	0.03	[0.98, 1]	0.9	1	0.9	1	1	0.05
Opt$_{0}$	0.88	0.05	[0.87, 0.89]	0.8	0.9	0.8	0.9	0.9	
Opt$_{60}$	0.91	0.04	[0.9, 0.92]	0.8	1	0.8	0.9	1	0.01
Opt$_{120}$	0.96	0.05	[0.95, 0.97]	0.9	1	0.9	1	1	0.02
Opt$_{360}$	0.96	0.05	[0.95, 0.97]	0.9	1	0.9	1	1	0.26
Opt$_{600}$	0.97	0.05	[0.96, 0.98]	0.9	1	0.9	1	1	0.13

Table A.91: Vehicle utilizations in the pickup and delivery service.

Performance ratios of utilization relative to Opt for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Srh$_{0}$	0.96	0.06	[0.95, 0.97]	0.9	1.11	0.9	1	1.11	0.04
Srh$_{60}$	1.01	0.06	[1, 1.02]	0.9	1.11	0.9	1	1.11	0.2
Srh$_{120}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.22
Srh$_{360}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.24
Srh$_{600}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.2
2Opt$_{0}$	0.96	0.06	[0.95, 0.97]	0.9	1.11	0.9	1	1.11	0.04
2Opt$_{60}$	1.01	0.06	[1, 1.02]	0.9	1.11	0.9	1	1.11	0.2
2Opt$_{120}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.22
2Opt$_{360}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.24
2Opt$_{600}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.2
Sa$_{0}$	0.96	0.06	[0.95, 0.97]	0.9	1.11	0.9	1	1.11	0.04
Sa$_{60}$	1.01	0.06	[1, 1.02]	0.9	1.11	0.9	1	1.11	0.2
Sa$_{120}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.22
Sa$_{360}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.24
Sa$_{600}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.2
Ts$_{0}$	0.94	0.06	[0.93, 0.95]	0.8	1.11	0.85	0.9	1.11	0.03
Ts$_{60}$	0.99	0.06	[0.98, 1]	0.9	1.11	0.9	1	1.11	0.15
Ts$_{120}$	1.01	0.05	[1, 1.02]	0.9	1.11	0.9	1	1.11	0.19
Ts$_{360}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.21
Ts$_{600}$	1.02	0.05	[1.01, 1.03]	0.9	1.11	0.9	1	1.11	0.2
Opt$_{0}$	0.91	0.07	[0.9, 0.92]	0.8	1	0.8	0.9	1	0
Opt$_{60}$	0.94	0.06	[0.93, 0.95]	0.8	1.11	0.8	0.9	1.11	0.02
Opt$_{120}$	0.99	0.07	[0.98, 1]	0.9	1.11	0.9	1	1.11	0.17
Opt$_{360}$	0.99	0.06	[0.98, 1]	0.9	1.11	0.9	1	1.11	0.13
Opt$_{600}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.92: Performance ratios of vehicle utilization relative to Opt in the pickup and delivery service.

Performance ratios of utilization relative to online version for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
SRH$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
SRH$_{60}$	1.06	0.05	[1.05, 1.07]	0.9	1.11	0.95	1.11	1.11	0.01
SRH$_{120}$	1.07	0.05	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.02
SRH$_{360}$	1.07	0.05	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.02
SRH$_{600}$	1.07	0.06	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.03
2OPT$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
2OPT$_{60}$	1.06	0.05	[1.05, 1.07]	0.9	1.11	0.95	1.11	1.11	0.01
2OPT$_{120}$	1.07	0.05	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.02
2OPT$_{360}$	1.07	0.05	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.02
2OPT$_{600}$	1.07	0.06	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.03
SA$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
SA$_{60}$	1.06	0.05	[1.05, 1.07]	0.9	1.11	0.95	1.11	1.11	0.01
SA$_{120}$	1.07	0.05	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.02
SA$_{360}$	1.07	0.05	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.02
SA$_{600}$	1.07	0.06	[1.06, 1.08]	0.9	1.11	0.9	1.11	1.11	0.03
TS$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
TS$_{60}$	1.05	0.06	[1.04, 1.06]	0.9	1.13	0.9	1.11	1.12	0.04
TS$_{120}$	1.08	0.05	[1.07, 1.09]	1	1.25	1	1.11	1.18	0
TS$_{360}$	1.08	0.05	[1.07, 1.09]	1	1.25	1	1.11	1.18	0
TS$_{600}$	1.08	0.05	[1.07, 1.09]	1	1.25	1	1.11	1.18	0
OPT$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
OPT$_{60}$	1.03	0.05	[1.02, 1.04]	0.89	1.13	0.94	1	1.13	0.01
OPT$_{120}$	1.09	0.07	[1.07, 1.11]	1	1.25	1	1.11	1.25	0
OPT$_{360}$	1.1	0.06	[1.09, 1.11]	1	1.25	1	1.11	1.25	0
OPT$_{600}$	1.1	0.07	[1.08, 1.12]	1	1.25	1	1.11	1.25	0

Table A.93: Performance ratios of vehicle utilization relative to the online version of an algorithm in the pickup and delivery service.

Throughput for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	% det.
SRH$_{0}$	3.67	0.04	[3.64, 3.7]	3.2	4	3.25	3.7	4	
SRH$_{60}$	3.66	0.04	[3.63, 3.69]	3.3	4	3.3	3.7	4	0.29
SRH$_{120}$	3.68	0.04	[3.65, 3.71]	3.3	4.1	3.35	3.7	4.05	0.2
SRH$_{360}$	3.69	0.04	[3.66, 3.72]	3.2	4	3.3	3.7	3.95	0.29
SRH$_{600}$	3.69	0.04	[3.66, 3.72]	3.4	4	3.4	3.7	3.95	0.27
2OPT$_{0}$	3.67	0.04	[3.64, 3.7]	3.2	4	3.25	3.7	4	
2OPT$_{60}$	3.66	0.04	[3.63, 3.69]	3.3	4	3.3	3.7	4	0.29
2OPT$_{120}$	3.68	0.04	[3.65, 3.71]	3.3	4.1	3.35	3.7	4.05	0.2
2OPT$_{360}$	3.69	0.04	[3.66, 3.72]	3.2	4	3.3	3.7	3.95	0.29
2OPT$_{600}$	3.69	0.04	[3.66, 3.72]	3.4	4	3.4	3.7	3.95	0.27
SA$_{0}$	3.67	0.04	[3.64, 3.7]	3.2	4	3.25	3.7	4	
SA$_{60}$	3.66	0.04	[3.63, 3.69]	3.3	4	3.3	3.7	4	0.29
SA$_{120}$	3.68	0.04	[3.65, 3.71]	3.3	4.1	3.35	3.7	4.05	0.2
SA$_{360}$	3.69	0.04	[3.66, 3.72]	3.2	4	3.3	3.7	3.95	0.29
SA$_{600}$	3.69	0.04	[3.66, 3.72]	3.4	4	3.4	3.7	3.95	0.27
TS$_{0}$	3.69	0.03	[3.67, 3.71]	3.4	4	3.4	3.7	4	
TS$_{60}$	3.7	0.04	[3.67, 3.73]	3.3	4	3.35	3.7	4	0.29
TS$_{120}$	3.7	0.04	[3.67, 3.73]	3.2	4	3.3	3.7	3.95	0.21
TS$_{360}$	3.7	0.03	[3.68, 3.72]	3.4	4.1	3.45	3.7	4	0.28
TS$_{600}$	3.69	0.03	[3.67, 3.71]	3.3	4.1	3.4	3.7	4.05	0.34
OPT$_{0}$	3.67	0.03	[3.65, 3.69]	3.4	4	3.4	3.65	4	
OPT$_{60}$	3.67	0.03	[3.65, 3.69]	3.4	4	3.4	3.7	3.95	0.36
OPT$_{120}$	3.67	0.04	[3.64, 3.7]	3.3	4	3.35	3.7	4	0.35
OPT$_{360}$	3.68	0.04	[3.65, 3.71]	3.4	4.1	3.4	3.7	4.05	0.39
OPT$_{600}$	3.68	0.03	[3.66, 3.7]	3.3	4	3.3	3.7	4	0.3

Table A.94: Job throughputs in the pickup and delivery service.

Performance ratios of throughput relative to Opt for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$F(1)$
Srh$_{0}$	1	0.04	[0.99, 1.01]	0.89	1.09	0.89	1	1.09	0.27
Srh$_{60}$	1	0.04	[0.99, 1.01]	0.87	1.12	0.88	1	1.1	0.29
Srh$_{120}$	1	0.04	[0.99, 1.01]	0.89	1.09	0.89	1	1.09	0.37
Srh$_{360}$	1	0.03	[0.99, 1.01]	0.91	1.12	0.92	1	1.09	0.37
Srh$_{600}$	1	0.04	[0.99, 1.01]	0.92	1.15	0.93	1	1.12	0.35
2Opt$_{0}$	1	0.04	[0.99, 1.01]	0.89	1.09	0.89	1	1.09	0.27
2Opt$_{60}$	1	0.04	[0.99, 1.01]	0.87	1.12	0.88	1	1.1	0.29
2Opt$_{120}$	1	0.04	[0.99, 1.01]	0.89	1.09	0.89	1	1.09	0.37
2Opt$_{360}$	1	0.03	[0.99, 1.01]	0.91	1.12	0.92	1	1.09	0.37
2Opt$_{600}$	1	0.04	[0.99, 1.01]	0.92	1.15	0.93	1	1.12	0.35
Sa$_{0}$	1	0.04	[0.99, 1.01]	0.89	1.09	0.89	1	1.09	0.27
Sa$_{60}$	1	0.04	[0.99, 1.01]	0.87	1.12	0.88	1	1.1	0.29
Sa$_{120}$	1	0.04	[0.99, 1.01]	0.89	1.09	0.89	1	1.09	0.37
Sa$_{360}$	1	0.03	[0.99, 1.01]	0.91	1.12	0.92	1	1.09	0.37
Sa$_{600}$	1	0.04	[0.99, 1.01]	0.92	1.15	0.93	1	1.12	0.35
Ts$_{0}$	1	0.03	[0.99, 1.01]	0.95	1.09	0.95	1	1.09	0.36
Ts$_{60}$	1.01	0.03	[1, 1.02]	0.89	1.12	0.91	1	1.1	0.36
Ts$_{120}$	1	0.04	[0.99, 1.01]	0.86	1.12	0.89	1	1.1	0.35
Ts$_{360}$	1.01	0.03	[1, 1.02]	0.95	1.12	0.95	1	1.11	0.41
Ts$_{600}$	1	0.03	[0.99, 1.01]	0.95	1.15	0.96	1	1.11	0.28
Opt$_{0}$	1	0.03	[0.99, 1.01]	0.92	1.09	0.92	1	1.08	0.3
Opt$_{60}$	1	0.03	[0.99, 1.01]	0.92	1.09	0.92	1	1.09	0.31
Opt$_{120}$	1	0.05	[0.99, 1.01]	0.87	1.14	0.87	1	1.13	0.34
Opt$_{360}$	1	0.03	[0.99, 1.01]	0.92	1.12	0.93	1	1.09	0.34
Opt$_{600}$	1	0	[1, 1]	1	1	1	1	1	0

Table A.95: Performance ratios of job throughput relative to Opt in the pickup and delivery service.

Performance ratios of throughput relative to online version for $n = 50$ (100 samples)									
Algorithm	μ	CV	95% CI	min	max	$q_{0.01}$	$q_{0.5}$	$q_{0.99}$	$1 - F(1)$
Srh$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Srh$_{60}$	1	0.04	[0.99, 1.01]	0.92	1.09	0.92	1	1.09	0.29
Srh$_{120}$	1.01	0.04	[1, 1.02]	0.89	1.12	0.91	1	1.12	0.25
Srh$_{360}$	1.01	0.03	[1, 1.02]	0.91	1.13	0.92	1	1.12	0.24
Srh$_{600}$	1.01	0.04	[1, 1.02]	0.92	1.12	0.92	1	1.1	0.29
2Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
2Opt$_{60}$	1	0.04	[0.99, 1.01]	0.92	1.09	0.92	1	1.09	0.29
2Opt$_{120}$	1.01	0.04	[1, 1.02]	0.89	1.12	0.91	1	1.12	0.25
2Opt$_{360}$	1.01	0.03	[1, 1.02]	0.91	1.13	0.92	1	1.12	0.24
2Opt$_{600}$	1.01	0.04	[1, 1.02]	0.92	1.12	0.92	1	1.1	0.29
Sa$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Sa$_{60}$	1	0.04	[0.99, 1.01]	0.92	1.09	0.92	1	1.09	0.29
Sa$_{120}$	1.01	0.04	[1, 1.02]	0.89	1.12	0.91	1	1.12	0.25
Sa$_{360}$	1.01	0.03	[1, 1.02]	0.91	1.13	0.92	1	1.12	0.24
Sa$_{600}$	1.01	0.04	[1, 1.02]	0.92	1.12	0.92	1	1.1	0.29
Ts$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Ts$_{60}$	1	0.03	[0.99, 1.01]	0.87	1.09	0.89	1	1.09	0.29
Ts$_{120}$	1	0.04	[0.99, 1.01]	0.84	1.09	0.88	1	1.09	0.3
Ts$_{360}$	1	0.03	[0.99, 1.01]	0.92	1.06	0.92	1	1.06	0.25
Ts$_{600}$	1	0.03	[0.99, 1.01]	0.92	1.06	0.93	1	1.06	0.29
Opt$_{0}$	1	0	[1, 1]	1	1	1	1	1	0
Opt$_{60}$	1	0.04	[0.99, 1.01]	0.92	1.11	0.92	1	1.1	0.36
Opt$_{120}$	1	0.05	[0.99, 1.01]	0.88	1.14	0.89	1	1.13	0.38
Opt$_{360}$	1	0.04	[0.99, 1.01]	0.89	1.11	0.91	1	1.1	0.29
Opt$_{600}$	1	0.04	[0.99, 1.01]	0.92	1.09	0.93	1	1.09	0.3

Table A.96: Performance ratios of job throughput relative to the online version of an algorithm in the pickup and delivery service.

References

[1] Ahlroth, L.; Schumacher, A.; Haanpää, H.: On the power of lookahead in online lot-sizing. In: *Operations Research Letters*, 38 (6), 522–526, 2010. 5, 89

[2] Albers, S.: On the influence of lookahead in competitive paging algorithms. In: *Algorithmica*, 18 (3), 283–305, 1997. 5, 12, 24, 81, 89

[3] Albers, S.: A competitive analysis of the list update problem with lookahead. In: *Theoretical Computer Science*, 197 (1-2), 95–109, 1998. 5, 89

[4] Albers, S.; Favrholdt, L.; Giel, O.: On paging with locality of reference. In: *Journal of Computer and System Sciences*, 70 (2), 145–175, 2005. 38, 40, 154

[5] Albers, S.; Möhring, R.; Pflug, G.; Schultz, R.: Summary. In: S. Albers; R. Möhring; G. Pflug; R. Schultz (eds.), Algorithms for Optimization with Incomplete Information, 2005. 6

[6] Albers, S.; Mitzenmacher, M.: Average-case analyses of first fit and random fit bin packing. In: *Random Structures and Algorithms*, 16 (3), 240–259, 2000. 46

[7] Allulli, L.; Ausiello, G.; Bonifaci, V.; Laura, L.: On the power of lookahead in on-line server routing problems. In: *Theoretical Computer Science*, 408 (2-3), 116–128, 2008. 5, 8, 31, 81, 89

[8] Allulli, L.; Ausiello, G.; Laura, L.: On the power of lookahead in on-line vehicle routing problems. In: L. Wang (ed.), Computing and Combinatorics, 728–736, Springer, 2005. 5, 8, 89

[9] Angelopoulos, S.; Dorrigiv, R.; López-Ortiz, A.: On the separation and equivalence of paging strategies. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, 229–237, 2007. 38, 49, 50, 266

[10] Angelopoulos, S.; Schweitzer, P.: Paging and list update under bijective analysis. In: *Journal of the ACM*, 60 (2), article no. 7, 2013. 38, 50, 266

[11] Ausiello, G.; Allulli, L.; Bonifaci, V.; Laura, L.: On-line algorithms, real time, the virtue of laziness, and the power of clairvoyance. In: J. Cai; S. Cooper; A. Li (eds.), Theory and Applications of Models of Computation, 1–20, Springer, 2006. 5, 8, 38, 40, 89

[12] Ausiello, G.; Crescenzi, P.; Kann, V.; Marchetti-Spaccalema, A.; Gambosi, G.; Spaccamela, A.: Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties. Springer, 2nd edition, 2003. 16, 17, 25, 27, 28, 29

[13] Ausiello, G.; Feuerstein, E.; Leonardi, S.; Stougie, L.; Talamo, M.: Competitive algorithms for the on-line traveling salesman. In: S. Akl; F. Dehne; J. Sack; N. Santoro (eds.), Algorithms and Data Structures, 206–217, Springer, 1995. 31

[14] Baker, B.: A new proof for the first-fit decreasing bin-packing algorithm. In: *Journal of Algorithms*, 6 (1), 49–70, 1985. 175

[15] Becchetti, L.: Modeling locality: A probabilistic analysis of LRU and FWF. In: S. Albers; T. Radzik (eds.), Algorithms ESA 2004, 98–109, Springer, 2004. 38, 47

[16] Becchetti, L.; Leonardi, S.; Marchetti-Spaccamela, A.; Schäfer, G.; Vredeveld, T.: Average-case and smoothed competitive analysis of the multilevel feedback algorithm. In: *Mathematics of Operations Research*, 31 (1), 85–108, 2006. 38, 44

[17] Belady, L.: A study of replacement algorithms for a virtual-storage computer. In: *IBM Systems Journal*, 5 (2), 78–101, 1966. 11, 156

[18] Bellman, R.: Dynamic Programming. Princeton University Press, 1957. 92

[19] Bellmore, M.; Nemhauser, G.: The traveling salesman problem: A survey. In: *Operations Research*, 16 (3), 538–558, 1968. 30

[20] Ben-David, S.; Borodin, A.: A new measure for the study of on-line algorithms. In: *Algorithmica*, 11 (1), 73–91, 1994. 5, 38, 42, 89

[21] Ben-David, S.; Borodin, A.; Karp, R.; Tardos, G.; Wigderson, A.: On the power of randomization in on-line algorithms. In: *Algorithmica*, 11 (1), 2–14, 1994. 38, 45, 73, 74

[22] Bent, R.; Van Hentenryck, P.: Regrets only! Online stochastic optimization under time constraints. In: Proceedings of the 19th National Conference on Artificial Intelligence, 501–506, 2004. 6, 266

[23] Bent, R.; Van Hentenryck, P.: The value of consensus in online stochastic scheduling. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling, 219–226, 2004. 266

[24] Bent, R.; Van Hentenryck, P.: Online stochastic and robust optimization. In: M. Maher (ed.), Advances in Computer Science - ASIAN 2004: Higher-Level Decision Making, 3237–3238, Springer, 2005. 4, 6, 266

[25] Bent, R.; Van Hentenryck, P.: Online Stochastic Combinatorial Optimization. The MIT Press, 2006. 266

[26] Bentley, J.; Johnson, D.; Leighton, F.; McGeoch, C.; McGeoch, L.: Some unexpected expected behavior results for bin packing. In: Proceedings of the 16th Annual ACM Symposium on Theory of Computing, 279–288, 1984. 46

[27] Berbeglia, G.; Cordeau, J.; Laporte, G.: Dynamic pickup and delivery problems. In: *European Journal of Operational Research*, 202 (1), 8–15, 2010. 239

[28] Berman, P.; Coulston, C.: Speed is more powerful than clairvoyance. In: *Nordic Journal of Computing*, 6 (2), 181–193, 1999. 267

[29] Bertsimas, D.; Brown, D.; Caramanis, C.: Theory and applications of robust optimization. In: *SIAM Review*, 53 (3), 464–501, 2011. 6

[30] Błażewicz, J.; Ecker, K.; Pesch, E.; Schmidt, G.; Węglarz, J. (eds.): Handbook on Scheduling: From Theory to Applications. Springer, 2007. 201

[31] Blom, M.; Krumke, S.; de Paepe, W.; Stougie, L.: The online TSP against fair adversaries. In: G. Bongiovanni; R. Petreschi; G. Gambosi (eds.), Algorithms and Complexity, 137–149, Springer, 2000. 38, 40

[32] Borodin, A.; El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, 1998. 4, 5, 11, 30, 38, 39, 45, 74, 86

[33] Borodin, A.; Irani, S.; Raghavan, P.; Schieber, B.: Competitive paging with locality of reference. In: *Journal of Computer and System Sciences*, 50 (2), 244 – 258, 1995. 38, 40, 154

[34] Bosman, P.; La Poutré, H.: Online transportation and logistics using computationally intelligent anticipation. In: A. Fink; F. Rothlauf (eds.), Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management, 185–208, Springer, 2008. 5, 89

[35] Boyar, J.; Favrholdt, L.: The relative worst order ratio for online algorithms. In: *ACM Transactions on Algorithms*, 3 (2), article no. 22, 2007. 38, 43

[36] Boyar, J.; Favrholdt, L.; Larsen, K.: The relative worst order ratio applied to paging. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 718–727, 2005. 43

[37] Boyar, J.; Favrholdt, L.; Larsen, K.; Nielsen, M.: Extending the accommodating function. In: *Acta Informatica*, 40 (1), 3–35, 2003. 38, 41

[38] Boyar, J.; Irani, S.; Larsen, K.: A comparison of performance measures for online algorithms. In: Proceedings of the 11th International Symposium on Algorithms and Data Structures, 119–130, 2009. 37, 43

[39] Boyar, J.; Larsen, K.; Nielsen, M.: The accommodating function: A generalization of the competitive ratio. In: *SIAM Journal on Computing*, 31 (1), 233–258, 2002. 38, 41

[40] Boyar, J.; Medvedev, P.: The relative worst order ratio applied to seat reservation. In: *ACM Transactions on Algorithms*, 4 (4), article no. 48, 2008. 43

[41] Breslauer, D.: On competitive on-line paging with lookahead. In: *Theoretical Computer Science*, 209 (1–2), 365 – 375, 1998. 5, 12, 13, 89

[42] Burkard, R.: Travelling salesman and assignment problems: A survey. In: Discrete Optimization I Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium, 193–215, 1979. 30

[43] Cassandras, C.; Lafortune, S.: Introduction to Discrete Event Systems. Springer, 2nd edition, 2008. 62, 63, 65, 67, 68

[44] Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976. 31

[45] Chung, F.; Graham, R.; Saks, M.: A dynamic location problem for graphs. In: *Combinatorica*, 9 (2), 111–131, 1989. 5, 89

[46] Coffman, Jr., E.; Courcoubetis, C.; Garey, M.; Johnson, D.; Shor, P.; Weber, R.; Yannakakis, M.: Perfect packing theorems and the average-case behavior of optimal and online bin packing. In: *SIAM Review*, 44 (1), 95–108, 2002. 46

[47] Coffman, Jr., E.; Garey, M.; Johnson, D.: Approximation algorithms for bin packing: A survey. In: D. Hochbaum (ed.), Approximation Algorithms for NP-Hard Problems, 46–93, PWS Publishing, 1997. 46

[48] Coffman, Jr., E.; Gilbert, E.: On the expected relative performance of list scheduling. In: *Operations Research*, 33 (3), pp. 548–561, 1985. 46

[49] Coffman, Jr., E.; Johnson, D.; Shor, P.; Weber, R.: Markov chains, computer proofs, and average-case analysis of best fit bin packing. In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 412–421, 1993. 46

[50] Coffman, Jr., E.; So, K.; Hofri, M.; Yao, A.: A stochastic model of bin-packing. In: *Information and Control*, 44 (2), 105–115, 1980. 46

[51] Coleman, B.: Quality vs. performance in lookahead scheduling. In: Proceedings of the 9th International Joint Conference on Information Science, 324–327, 2006. 5, 89

[52] Cook, S.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 151–158, 1971. 27

[53] Cordeau, J.; Laporte, G.: A tabu search heuristic for the static multi-vehicle dial-a-ride problem. In: *Transportation Research Part B: Methodological*, 37 (6), 579–594, 2003. 239, 242, 243

[54] Cormen, T.; Leiserson, C.; Rivest, R.; Stein, C.: Introduction to Algorithms. The MIT Press, 3rd edition, 2009. 15, 25, 26, 29, 267

[55] Csirik, J.; Woeginger, G.: On-line packing and covering problems. In: A. Fiat; G. Woeginger (eds.), Online Algorithms: The State of the Art, 147–177, Springer, 1998. 11, 46, 111, 112, 166, 176, 187

[56] Csirik, J.; Woeginger, G.: Resource augmentation for online bounded space bin packing. In: *Journal of Algorithms*, 44 (2), 308–320, 2002. 38, 39, 267

[57] de Koster, R.; Le-Duc, T.; Roodbergen, K.: Design and control of warehouse order picking: A literature review. In: *European Journal of Operational Research*, 182 (2), 481–501, 2007. 220

[58] Demeulemeester, E.; Herroelen, W.: Project Scheduling: A Research Handbook. Kluwer, 2002. 266

[59] Denning, P.: The working set model for program behavior. In: *Communications of the ACM*, 11 (5), 323–333, 1968. 40

[60] Denning, P.: Working sets past and present. In: *IEEE Transactions on Software Engineering*, 6 (1), 64–84, 1980. 40

[61] Desaulniers, G.; Desrosiers, J.; Solomon, M. (eds.): Column Generation. Springer, 2005. 98

[62] Deutsch, E.: Dyck path enumeration. In: *Discrete Mathematics*, 204 (1–3), 167–202, 1999. 119

[63] Deutsch, E.; Shapiro, L.: A survey of the Fine numbers. In: *Discrete Mathematics*, 241 (1-3), 241–265, 2001. 119, 121

[64] Dooly, D.; Goldman, S.; Scott, S.: On-line analysis of the TCP acknowledgment delay problem. In: *Journal of the ACM*, 48 (2), 243–273, 2001. 5, 89

[65] Dorrigiv, R.: Alternative Measures for the Analysis of Online Algorithms. Ph.D. thesis, University of Waterloo, 2010. 36, 37, 50, 262

[66] Dorrigiv, R.; López-Ortiz, A.: Closing the gap between theory and practice: New measures for on-line algorithm analysis. In: Proceedings of the 2nd International Conference on Algorithms and Computation, 13–24, 2008. 38, 41

[67] Dorrigiv, R.; López-Ortiz, A.; Munro, J.: On the relative dominance of paging algorithms. In: *Theoretical Computer Science*, 410 (38-40), 3694–3701, 2009. 38, 47, 48

[68] Dunke, F.; Nickel, S.: Simulative algorithm analysis in online optimization with lookahead. In: W. Dangelmaier; C. Laroque; A. Klaas (eds.), Simulation in Produktion und Logistik: Entscheidungsunterstützung von der Planung bis zur Steuerung, 405–416, HNI-Verlagsschriftenreihe, 2013. 38, 75, 219

[69] Dyckhoff, H.; Finke, U.: Cutting and Packing in Production and Distribution: A Typology and Bibliography. Physica, 1992. 9

[70] Ehrgott, M.: Multicriteria Optimization. Springer, 2nd edition, 2005. 248

[71] Epstein, L.; van Stee, R.: Online bin packing with resource augmentation. In: G. Persiano; R. Solis-Oba (eds.), Approximation and Online Algorithms, 23–35, Springer, 2005. 38, 39, 267

[72] Esen, M.: Design, Implementation and Analysis of Online Bin Packing Problems. Master's thesis, Technische Universität Kaiserslautern, 2000. 167, 168, 169, 266

[73] Fiat, A.; Woeginger, G.: Competitive odds and ends. In: A. Fiat; G. Woeginger (eds.), Online Algorithms: The State of the Art, 385–394, Springer, 1998. 36, 262

[74] Fiat, A.; Woeginger, G. (eds.): Online Algorithms: The State of the Art. Springer, 1998. 4, 5, 15, 154

[75] Franaszek, P.; Wagner, T.: Some distribution-free aspects of paging algorithm performance. In: *Journal of the ACM*, 21 (1), 31–39, 1974. 46

[76] Fujiwara, H.; Iwama, K.: Average-case competitive analyses for ski-rental problems. In: *Algorithmica*, 42 (1), 95–107, 2005. 46

[77] Garey, M.; Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979. 9, 16, 25, 26, 27, 28

[78] Ghiani, G.; Laporte, G.; Musmanno, R.: Introduction to Logistics Systems Planning and Control. Wiley, 2004. 1

[79] Glover, F.: Tabu search – Part I. In: *ORSA Journal on Computing*, 1 (3), 190–206, 1989. 191

[80] Graham, R.: Bounds for certain multiprocessing anomalies. In: *Bell System Technical Journal*, 45 (9), 1563–1581, 1966. 38, 39

[81] Graham, R.; Lawler, E.; Lenstra, J.; Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: A survey. In: Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium, 287 – 326, 1979. 87, 202

[82] Gross, J.; Yellen, J. (eds.): Handbook of Graph Theory. CRC Press, 2004. 267

[83] Grove, E.: Online bin packing with lookahead. In: Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, 430–436, 1995. 5, 9, 10, 11, 81, 89

[84] Grötschel, M.; Krumke, S.; Rambau, J. (eds.): Online Optimization of Large Scale Systems. Springer, 2001. 3, 4, 74, 267

[85] Grötschel, M.; Krumke, S.; Rambau, J.; Winter, T.; Zimmermann, U.: Combinatorial online optimization in real time. In: M. Grötschel; S. Krumke; J. Rambau (eds.), Online Optimization of Large Scale Systems, 679–704, Springer, 2001. 3, 4, 6, 20, 74, 267

[86] Gutin, G.; Jensen, T.; Yeo, A.: Batched bin packing. In: *Discrete Optimization*, 2 (1), 71–82, 2005. 5, 89

[87] HALLDÓRSSON, M.; SZEGEDY, M.: Lower bounds for on-line graph coloring. In: *Theoretical Computer Science*, 130 (1), 163–174, 1994. 5, 89

[88] HEINZ, S.: The Online Target Date Assignment Problem. Master's thesis, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2005. 2

[89] HENN, S.; KOCH, S.; WÄSCHER, G.: Order batching in order picking warehouses: A survey of solution approaches. In: R. MANZINI (ed.), Warehousing in the Global Supply Chain, 105–137, Springer, 2012. 220, 223, 224

[90] HILLER, B.: Online Optimization: Probabilistic Analysis and Algorithm Engineering. Ph.D. thesis, Technische Universität Berlin, 2009. 38, 47, 48, 49, 51, 68, 69, 266

[91] HILLER, B.; VREDEVELD, T.: Probabilistic alternatives for competitive analysis. In: *Computer Science - Research and Development*, 27 (3), 189–196, 2012. 37

[92] HUBER, C.: Throughput Analysis of Manual Order Picking Systems with Congestion Consideration. Ph.D. thesis, Karlsruher Institut für Technologie, 2011. 220, 221

[93] IMREH, C.; NÉMETH, T.: On time lookahead algorithms for the online data acknowledgement problem. In: L. KUCERA; A. KUCERA (eds.), Mathematical Foundations of Computer Science 2007, 288–297, Springer, 2007. 5, 89

[94] IRANI, S.: Coloring inductive graphs on-line. In: *Algorithmica*, 11 (1), 53–72, 1994. 5, 89

[95] JAILLET, P.; LU, X.: Online traveling salesman problems with service flexibility. In: *Networks*, 58 (2), 137–146, 2011. 5, 89

[96] JAILLET, P.; WAGNER, M.: Online routing problems: Value of advanced information as improved competitive ratios. In: *Transportation Science*, 40 (2), 200–210, 2006. 5, 8, 89

[97] JAYNES, E.: Information theory and statistical mechanics. In: *Physical Review*, 106 (4), 620–630, 1957. 51, 92, 262

[98] JAYNES, E.: Information theory and statistical mechanics II. In: *Physical Review*, 108 (2), 171–190, 1957. 51, 92, 262

[99] JOHNSON, D.: Near-Optimal Bin Packing Algorithms. Ph.D. thesis, Massachusetts Institute of Technology, 1973. 38, 39

[100] JOHNSON, D.: Fast algorithms for bin packing. In: *Journal of Computer and System Sciences*, 8 (3), 272–314, 1974. 38, 39

[101] KALLRATH, J.: Online Storage Systems and Transportation Problems with Applications: Optimization Models and Mathematical Solutions. Springer, 2005. 8, 10, 222, 242, 243, 247

[102] KALYANASUNDARAM, B.; PRUHS, K.: Speed is as powerful as clairvoyance. In: *Journal of the ACM*, 47 (4), 617–643, 2000. 38, 39, 267

[103] Karlin, A.: On the performance of competitive algorithms in practice. In: A. Fiat; G. Woeginger (eds.), Online Algorithms: The State of the Art, 373–384, Springer, 1998. 101

[104] Karlin, A.; Manasse, M.; Rudolph, L.; Sleator, D.: Competitive snoopy caching. In: *Algorithmica*, 3 (1-4), 79–119, 1988. 38, 39

[105] Karlin, A.; Phillips, S.; Raghavan, P.: Markov paging. In: *SIAM Journal on Computing*, 30 (3), 906–922, 2000. 46

[106] Karmarkar, N.: Probabilistic analysis of some bin-packing problems. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, 107–111, 1982. 46

[107] Karp, R.: On-line algorithms versus off-line algorithms: How much is it worth to know the future? In: Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture, 416–429, 1992. 150

[108] Kenyon, C.: Best-fit bin-packing with random order. In: Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, 359–364, 1996. 38, 44, 45

[109] Kenyon, C.; Rabani, Y.; Sinclair, A.: Biased random walks, Lyapunov functions, and stochastic analysis of best fit bin packing. In: *Journal of Algorithms*, 27 (2), 218–235, 1998. 46

[110] Khachiyan, L.: On the complexity of computing the volume of a polytope. In: *Engineering Cybernet*, 3, 45–46, 1988. 62

[111] Khachiyan, L.: Computing the volume of polytopes and polyhedra is hard. In: *Russian Mathematical Surveys*, 44 (3), 199–200, 1989. 62

[112] Khachiyan, L.: Complexity of polytope volume computation. In: *New Trends in Discrete and Computational Geometry*, 3, 91–101, 1993. 62

[113] Kiniwa, J.; Hamada, T.; Mizoguchi, D.: Lookahead scheduling requests for multisize page caching. In: *IEEE Transactions on Computers*, 50 (9), 972–983, 2001. 5, 89

[114] Königsberger, K.: Analysis 1. Springer, 5th edition, 2001. 130

[115] Koutsoupias, E.; Papadimitriou, C.: Beyond competitive analysis. In: *SIAM Journal of Computing*, 30 (1), 300–317, 2000. 5, 38, 42, 47, 89

[116] Krumke, S.: Online Optimization: Competitive Analysis and Beyond. Habilitation, Technische Universität Berlin, 2002. 20, 31

[117] Krumke, S.; de Paepe, W.; Poensgen, D.; Stougie, L.: News from the online traveling repairman. In: *Theoretical Computer Science*, 295 (1–3), 279 – 294, 2003. 31

[118] Larsen, A.; Madsen, O.; Solomon, M.: Recent developments in dynamic vehicle routing systems. In: B. Golden; S. Raghavan; E. Wasil (eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, 199–218, Springer, 2008. 7

[119] Lavrov, A.; Nickel, S.: Simulation und Optimierung zur Planung von Kommissionierungssystemen. VDI-Seminar, 2005. 2, 70

[120] Lawler, E.; Lenstra, J.; Rinnooy Kan, A.; Shmoys, D. (eds.): The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, 1985. 189, 190, 192, 243

[121] Lee, C.; Lee, D.: A simple on-line bin-packing algorithm. In: *Journal of the ACM*, 32 (3), 562–572, 1985. 46

[122] Lenstra, J.; Rinnooy Kan, A.: Some simple applications of the travelling salesman problem. In: *Operational Research Quarterly*, 26 (4), 717–733, 1975. 134

[123] Li, W.; Yuan, J.; Cao, J.; Bu, H.: Online scheduling of unit length jobs on a batching machine to maximize the number of early jobs with lookahead. In: *Theoretical Computer Science*, 410 (47–49), 5182–5187, 2009. 5, 89

[124] Lin, S.; Kernighan, B.: An effective heuristic algorithm for the travelling-salesman problem. In: *Operations Research*, 21 (2), 498–516, 1973. 31

[125] Lucas, K.; Roosen, P. (eds.): Emergence, Analysis and Evolution of Structures: Concepts and Strategies across Disciplines. Springer, 2010. 3

[126] Lunze, J.: Ereignisdiskrete Systeme: Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen. Oldenbourg, 2006. 62, 63, 64, 65, 67, 68

[127] Lévêque, O.: Random matrices and communication systems. Lecture Notes, École Polytechnique Fédérale de Lausanne, available online at `http://ipg.epfl.ch/~leveque/LectureNotes/rm_lecture_notes.pdf` (visited on January 2nd, 2014), 2012. 269

[128] Manasse, M.: Ski rental problem. In: M. Kao (ed.), Encyclopedia of Algorithms, 849–851, Springer, 2008. 101, 103

[129] Mandelbaum, M.; Shabtay, D.: Scheduling unit length jobs on parallel machines with lookahead information. In: *Journal of Scheduling*, 14 (4), 335–350, 2011. 5, 89

[130] Mao, W.; Kincaid, R.: A look-ahead heuristic for scheduling jobs with release dates on a single machine. In: *Computers and Operations Research*, 21 (10), 1041 – 1050, 1994. 5, 89

[131] Michaels, J.; Rosen, K.: Applications of Discrete Mathematics. McGraw-Hill, 1991. 119, 269

[132] Miller, C.; Tucker, A.; Zemlin, R.: Integer programming formulation of traveling salesman problems. In: *Journal of the ACM*, 7 (4), 326–329, 1960. 192, 228

[133] Müller, A.; Stoyan, D.: Comparison Methods for Stochastic Models and Risks. Wiley, 2002. 48

[134] Motwani, R.; Saraswat, V.; Torng, E.: Online scheduling with lookahead: Multipass assembly lines. In: *INFORMS Journal on Computing*, 10 (3), 331–340, 1998. 5, 89

[135] März, L.; Krug, W.: Kopplung von Simulation und Optimierung. In: W. Krug; O. Rose; G. Weigert (eds.), Simulation und Optimierung in Produktion und Logistik: Praxisorientierter Leitfaden mit Fallbeispielen, 41–45, Springer, 2011. 1, 70

[136] Naaman, N.; Rom, R.: Average case analysis of bounded space bin packing algorithms. In: *Algorithmica*, 50 (1), 72–97, 2007. 38, 45, 46

[137] Papula, L.: Mathematische Formelsammlung. Vieweg+Teubner, 10th edition, 2009. 186

[138] Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer, 4th edition, 2012. 201, 202, 203, 210

[139] Psaraftis, H.: Dynamic vehicle routing: Status and prospects. In: *Annals of Operations Research*, 61 (1), 143–164, 1995. 7

[140] Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, 2005. 92

[141] Rabe, M.; Spieckermann, S.; Wenzel, S.: Verifikation und Validierung für die Simulation in Produktion und Logistik: Vorgehensmodelle und Techniken. Springer, 2008. 70

[142] Ramanan, P.: Average-case analysis of the smart next fit algorithm. In: *Information Processing Letters*, 31 (5), 221–225, 1989. 46

[143] Regev, A.: A proof of Catalan's convolution formula. In: *Integers*, 12 (5), 929–934, 2012. 126, 128

[144] Remy, J.; Souza, A.; Steger, A.: On an online spanning tree problem in randomly weighted graphs. In: *Combinatorics, Probability and Computing*, 16 (1), 127–144, 2007. 46

[145] Scharbrodt, M.; Schickinger, T.; Steger, A.: A new average case analysis for completion time scheduling. In: *Journal of the ACM*, 53 (1), 121–146, 2006. 38, 46

[146] Shapiro, L.: A Catalan triangle. In: *Discrete Mathematics*, 14 (1), 83–90, 1976. 114, 121, 125

[147] Shor, P.: The average-case analysis of some on-line algorithms for bin packing. In: *Combinatorica*, 6 (2), 179–200, 1986. 46, 175

[148] Simchi-Levi, D.; Kaminsky, P.; Simchi-Levi, E.: Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies. McGraw-Hill, 3rd edition, 2008. 1

[149] Sleator, D.; Tarjan, R.: Amortized efficiency of list update and paging rules. In: *Communications of the ACM*, 28 (2), 202–208, 1985. 11, 38, 39, 155, 267

[150] Sleator, D.; Tarjan, R.: Self-adjusting binary search trees. In: *Journal of the ACM*, 32 (3), 652–686, 1985. 38, 39

[151] Souza, A.: Average Performance Analysis. Ph.D. thesis, Eidgenössische Technische Hochschule Zürich, 2006. 5, 38, 45, 46

[152] Souza, A.; Steger, A.: The expected competitive ratio for weighted completion time scheduling. In: V. Diekert; M. Habib (eds.), STACS 2004, 620–631, Springer, 2004. 46

[153] Spielman, D.; Teng, S.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In: *Journal of the ACM*, 51 (3), 385–463, 2004. 38, 44

[154] Stadtler, H.; Kilger, C. (eds.): Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies. Springer, 4th edition, 2008. 1

[155] Tinkl, M.: Online-Optimierung der Rundreise auf der Kreislinie mit Informationsvorlauf. Ph.D. thesis, Universität Augsburg, 2011. 5, 11, 18, 33, 80, 81, 85, 89

[156] Torng, E.: A unified analysis of paging and caching. In: *Algorithmica*, 20 (2), 175–200, 1998. 5, 89

[157] Torres, L.: Online Vehicle Routing: Set Partitioning Problems. Cuvillier, 2004. 20

[158] Toth, P.; Vigo, D. (eds.): The Vehicle Routing Problem. SIAM, 2002. 8, 239

[159] Verein Deutscher Ingenieure (VDI): VDI-Richtlinie 3633. Simulation von Logistik-, Materialfluß- und Produktionssystemen: Begriffsdefinitionen. In: VDI-Handbuch Materialfluß und Fördertechnik, Beuth, 1996. 62, 69

[160] Woeginger, G.: Exact algorithms for NP-hard problems: A survey. In: M. Jünger; G. Reinelt; G. Rinaldi (eds.), Combinatorial Optimization - Eureka, You Shrink!, 185–207, Springer, 2003. 28

[161] Yang, D.; Nair, G.; Sivaramakrishnan, B.; Jayakumar, H.: Round robin with look ahead: A new scheduling algorithm for bluetooth. In: Proceedings of the 2002 International Conference on Parallel Processing Workshops, 45–50, 2002. 5, 89

[162] Ye, Y.; Agrawal, S.; Wang, Z.: A dynamic near-optimal algorithm for online linear programming. Working Paper, available online at `http://www.stanford.edu/~yyye/onlineLPv4.pdf` (visited on January 2nd, 2014), 2009. 97, 98

[163] Yeh, T.; Kuo, C.; Lei, C.; Yen, H.: Competitive analysis of on-line disk scheduling. In: T. Asano; Y. Igarashi; H. Nagamochi; S. Miyano; S. Suri (eds.), Algorithms and Computation, 356–365, Springer, 1996. 5, 89

[164] Young, N.: Competitive Paging And Dual-Guided On-Line Weighted Caching And Matching Algorithms. Ph.D. thesis, Princeton University, 1991. 5, 12, 89, 154

[165] Young, N.: The k-server dual and loose competitiveness for paging. In: *Algorithmica*, 11 (6), 525–541, 1994. 38, 40

[166] Young, N.: Bounding the diffuse adversary. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, 420–425, 1998. 38, 47

[167] Young, N.: On-line file caching. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, 82–86, 1998. 38, 41

[168] Zheng, F.; Cheng, Y.; Liu, M.; Xu, Y.: Online interval scheduling on a single machine with finite lookahead. In: *Computers and Operations Research*, 40 (1), 180 – 191, 2013. 5, 89

[169] Zheng, F.; Xu, Y.; Zhang, E.: How much can lookahead help in online single machine scheduling. In: *Information Processing Letters*, 106 (2), 70–74, 2008. 5, 89

List of Figures

List of Tables

Erklärung

gemäß § 4, Abs. 4 der Promotionsordnung vom 15. August 2006:

Ich versichere wahrheitsgemäß, die Dissertation bis auf die in der Abhandlung angegebene Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und genau kenntlich gemacht zu haben, was aus Arbeiten anderer und aus eigenen Veröffentlichungen unverändert oder mit Abänderungen entnommen wurde.

Eggenstein, 24. Juli 2014

Fabian Dunke