Performance Analysis and Capacity Planning of Multi-stage Stochastic Order Fulfilment Systems with Levelled Order Release and Order Deadlines

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. Uta Mohring
geb. in Kirchheim unter Teck

Tag der mündlichen Prüfung:	15.02.2022
Hauptreferent:	Prof. Dr.-Ing. Kai Furmans
Korreferenten:	Prof. Dr.-Ing. Gisela Lanza, Prof. Dr. Raik Stolletz

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Institut für Fördertechnik und Logistiksysteme (IFL) des Karlsruher Instituts für Technologie. Ich möchte mich an dieser Stelle bei allen Personen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Zunächst danke ich Prof. Dr.-Ing. Kai Furmans, Leiter des IFL, für die Übernahme des Hauptreferats. Er hat mir viele Freiheiten gewährt und mir stets großes Vertrauen entgegengebracht. Dies hat – gepaart mit den kreativen Diskussionen auf Augenhöhe – zum erfolgreichen Abschluss dieser Arbeit beigetragen. Prof. Dr.-Ing. Gisela Lanza und Prof. Dr. Raik Stolletz danke ich für die Übernahme des Korreferats. Univ.-Prof. Dr.-Ing. Dr. h. c. Albert Albers danke ich für die Übernahme des Prüfungsvorsitzes meiner Promotionsprüfung.

Allen aktiven und ehemaligen Kollegen am IFL danke ich für die angenehme und inspirierende Arbeitsatmosphäre und die gegenseitige Unterstützung. Dies hat nicht nur das Erstellen dieser Arbeit erleichtert, sondern auch meinen Lebensabschnitt am IFL zu einer prägenden und unvergesslichen Zeit gemacht. Insbesondere danke ich Dr.-Ing. Katharina Fleischer-Dörr für ihren Zuspruch und Rückhalt in der Anfangsphase meiner Promotion sowie die kritische Korrektur meiner Arbeit. Bei Christoph Jacobi möchte ich mich für die vielen intensiven Diskussionen und sein stets ehrliches und kritisches Feedback bedanken. Er und Julia Fleischmann haben mich stets darin bestärkt und unterstützt, mich für meine Visionen einzusetzen und meinen eigenen Weg zu gehen. Für diesen Rückhalt und alles, was wir in unserer gemeinsamen Zeit am IFL miteinander und voneinander gelernt haben, bin ich sehr dankbar.

Ein ganz besonderer Dank gilt meinen Eltern und meiner Schwester für ihr Vertrauen, ihr Verständnis und ihre grenzenlose Unterstützung auf meinem bisherigen Lebensweg, aber auch für die notwendige Ablenkung. All das hat mir stets viel Kraft und Rückhalt gegeben. Ich widme diese Arbeit meinem Opa (†2019), dessen Ehrgeiz und Disziplin mir ein großes Vorbild sind.

Karlsruhe, Februar 2022 Uta Mohring

Kurzfassung

Kundenorientierte Auftragsbearbeitungsprozesse in Logistik- und Produktionssystemen sind heutzutage mit einem kontinuierlich steigenden Auftragsvolumen zunehmend kleinvolumiger Aufträge, hohen Kundenanforderungen hinsichtlich kurzfristiger und individueller Lieferfristen und einer stark stochastisch schwankenden Kundennachfrage konfrontiert. Um trotz der volatilen Kundennachfrage eine effiziente Auftragsbearbeitung und die Einhaltung der kundenindividuellen Lieferfristen gewährleisten zu können, muss die Arbeitslast kundenorientierter Auftragsbearbeitungsprozesse auf geeignete Weise geglättet werden. Hopp and Spearman (2004) unterscheiden zur Kompensation von Schwankungen in Produktionssystemen zwischen den Dimensionen Bestand, Zeit und Kapazität. Diese stellen auch einen guten Ausgangspunkt für die Entwicklung von Glättungskonzepten für stochastische, kundenorientierte Bearbeitungsprozesse dar. In dieser Arbeit werden die Potentiale der Dimensionen Zeit und Kapazität in der *Strategie der nivellierten Auftragseinlastung* zusammengeführt, um die Arbeitslast mehrstufiger, stochastischer Auftragsbearbeitungsprozesse mit kundenindividuellen Fälligkeitsfristen auf taktischer Ebene zeitlich zu glätten. Ziel dieser Arbeit ist (1) die Entwicklung eines Glättungskonzeptes, der so genannten *Strategie der nivellierten Auftragseinlastung*, (2) die Entwicklung eines zeitdiskreten analytischen Modells zur Leistungsanalyse und (3) die Entwicklung eines Algorithmus zur Kapazitätsplanung unter Gewährleistung bestimmter Leistungsanforderungen für mehrstufige, stochastische Auftragsbearbeitungsprozesse mit nivellierter Auftragseinlastung und kundenindividuellen Fälligkeitsfristen.

Die *Strategie der nivellierten Auftragseinlastung* zeichnet sich durch die Bereitstellung zeitlich konstanter Kapazitäten für die Auftragsbearbeitung und eine Auftragsbearbeitung gemäß aufsteigender Fälligkeitsfristen aus. Auf diese Weise

wird der zeitliche Spielraum jedes Auftrags zwischen dessen Auftragseingang und dessen Fälligkeitsfrist systematisch zur Kompensation der stochastischen Nachfrageschwankungen genutzt. Die verbleibende Variabilität wird in Abhängigkeit der Leistungsanforderungen der Kunden durch die Höhe der bereitgestellten Kapazität kompensiert. Das analytische Modell zur Leistungsanalyse mehrstufiger, stochastischer Auftragsbearbeitungsprozesse mit nivellierter Auftragseinlastung und kundenindividuellen Fälligkeitsfristen bildet die Auftragsbearbeitung als zeitdiskrete Markov-Kette ab und berechnet verschiedene stochastische und deterministische Leistungskenngrößen auf Basis deren asymptotischer Zustandsverteilung. Diese Kenngrößen, wie beispielsweise Durchsatz, Servicegrad, Auslastung, Anzahl Lost Sales sowie Zeitpuffer und Rückstandsdauer eines Auftrags, ermöglichen eine umfassende und exakte Leistungsanalyse von mehrstufigen, stochastischen Auftragsbearbeitungsprozessen mit nivellierter Auftragseinlastung und kundenindividuellen Fälligkeitsfristen. Der Zusammenhang zwischen der bereitgestellten Kapazität und der damit erreichbaren Leistungsfähigkeit kann nicht explizit durch eine mathematische Gleichung beschrieben werden, sondern ist implizit durch das analytische Modell gegeben. Daher ist das Entscheidungsproblem der Kapazitätsplanung unter Gewährleistung bestimmter Leistungsanforderungen ein Blackbox-Optimierungsproblem. Die problemspezifischen Konfigurationen der Blackbox-Optimierungsalgorithmen *Mesh Adaptive Direct Search* und *Surrogate Optimisation Integer* ermöglichen eine zielgerichtete Bestimmung des minimalen prozessspezifischen Kapazitätsbedarfs, der zur Gewährleistung der Leistungsanforderungen der Kunden bereitgestellt werden muss. Diese werden anhand einer oder mehrerer Leistungskenngrößen des Auftragsbearbeitungsprozesses spezifiziert.

Numerische Untersuchungen zur Beurteilung der Leistungsfähigkeit der *Strategie der nivellierten Auftragseinlastung* zeigen, dass in Systemen mit einer Auslastung größer als 0,6 durch den Einsatz der *Strategie der nivellierten Auftragseinlastung* ein deutlich höherer α- und β-Servicegrad erreicht werden kann als mit *First come first serve*. Außerdem ist der Kapazitätsbedarf zur Gewährleistung eines bestimmten α-Servicegrads bei Einsatz der *Strategie der nivellierten Auftragseinlastung* höchstens so hoch wie bei Einsatz von *First come first serve*.

Abstract

Order fulfilment systems are forced to manage a highly volatile customer demand consisting of low-volume orders effectively and efficiently while simultaneously meeting customer-required, short order deadlines. Hopp and Spearman (2004) provide three buffer types – inventory buffer, time buffer, and capacity buffer – to handle the volatile workload in production systems. These buffer types also provide several potentials for workload balancing in order fulfilment. In this thesis, we combine the potentials of time buffer and capacity buffer to develop and analytically investigate the *Strategy of Levelled Order Release* to balance workload in multi-stage, stochastic order fulfilment systems with customer-required order deadlines over time on a tactical level. The contribution of this thesis is three-fold: We develop (1) a workload balancing concept, the so-called *Strategy of Levelled Order Release*, (2) a discrete-time analytical model for performance analysis, and (3) an algorithm for capacity planning under performance constraints in multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines.

The *Strategy of Levelled Order Release* is characterised by (1) a fixed capacity reserved for order processing in each time period, and (2) an order processing according to ascending due dates in each time period. In this way, the time buffer of each order between its time of arrival and its deadline is used to balance the variability of the customer demand. The remaining variability is compensated by using the capacity buffer depending on the performance requirements of the customers. The developed analytical model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines models system behaviour of such order fulfilment systems as a discrete-time Markov chain. It calculates multiple stochastic and deterministic, system-

and customer-related performance measures of order fulfilment systems based on the limiting distribution of the Markov chain. These performance measures, such as system throughput, service level, utilisation, number of lost sales, and time buffer and backlog duration of a completed order, enable a comprehensive and exact performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. The relationship between the provided capacity and the performance that is achieved with this capacity cannot be specified by a mathematical equation, but it is given by the analytical model. Thus, the decision problem of capacity planning in multi-stage order fulfilment systems with performance requirements is a blackbox optimisation problem. The problem-specific configurations of the blackbox optimisation algorithms *Mesh Adaptive Direct Search* and *Surrogate Optimisation Integer* enable a target-oriented determination of the minimum required, process-specific capacity to meet any performance requirement of the customers that is specified based on one or multiple performance measures of the order fulfilment system.

Numerical studies on the performance evaluation of the *Strategy of Levelled Order Release* show that one can achieve significantly higher values of α- and β-service level when using the *Strategy of Levelled Order Release* instead of *First come first serve* in order fulfilment systems with a utilisation higher than 0.6. Furthermore, the total required capacity to guarantee a predefined α-service level in the order fulfilment system when using the *Strategy of Levelled Order Release* is at most as high as the one when using *First come first serve*.

Contents

1 Introduction

Intensified competition due to globalisation and e-commerce increases the pressure on order fulfilment systems to operate in an effective and efficient way (Van Gils et al. 2018; Kundu et al. 2020). Additionally, total order volume constantly increases, while the character of orders has changed from a small number of high-volume orders to a large number of low-volume orders (De Koster et al. 2007). At German Amazon warehouses, for example, an average customer order consists of 1.6 items (Boysen et al. 2019), and in production systems, the trend from mass products to customised products increases the relevance of low-volume/high-variety make-to-order production systems (Kundu et al. 2020). Moreover, promising short, individual, and reliable order deadlines has become a crucial competitive requirement in order fulfilment due to increasing customer orientation (Öner-Közen and Minner 2017). Individual order deadlines depend, for example, on the shipping distance and the service type chosen by the customer (Yan et al. 2010), and delivery promises, such as next-day or same-day delivery, are widespread in e-commerce (Yaman et al. 2012; Boysen et al. 2019). Furthermore, customer demand in order fulfilment systems is highly volatile, independent of the considered industry. Figure 1.1 shows an exemplary time series of incoming customer orders in a German company of the automotive aftermarket sector in 2015. The corresponding squared coefficient of variation of order income is 0.3. The high variability of the number of incoming customer orders results in strong fluctuations of system workload over time (Hopp and Spearman 2004; Boysen et al. 2019). In conclusion, order fulfilment systems are forced to manage a highly volatile customer demand consisting of low-volume orders in an effective and efficient way while simultaneously meeting customer-required, short order deadlines.

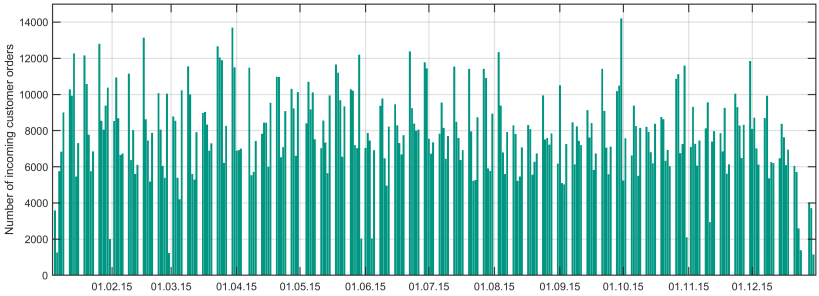


Figure 1.1: Time series of incoming customer orders of a German company of the automotive aftermarket sector in 2015 (Gaps in the time series correspond to weekends and public holidays).

Investigating production systems, Hopp and Spearman (2004) claim that "all variability [...] will be buffered". They provide three buffer types to handle the volatile workload in production systems:

- Inventory buffer (e.g. safety stocks),
- Time buffer (e.g. safety lead times), and
- Capacity buffer (e.g. excess capacity).

In make-to-stock production systems, system workload and customer demand are decoupled by the finished-goods-inventory, so that inventory buffer provides several potentials to compensate the variability of customer demand. In contrast, in the main application fields of order fulfilment – make-to-order production systems and warehouses –, system workload is directly driven by the customer demand since the orders are customer-specific and unknown in advance. Apart from some preparatory tasks, for instance, pre-picking, it is not possible to process the orders before they arrive at the order fulfilment system. Thus, in order fulfilment systems, inventory buffer only provides limited potential to handle the variability of customer demand. Despite the short lead time requirements of customer orders, there is some time buffer in order fulfilment since the processing time of an order is still much shorter than its lead time. However, it is impossible to flexibly adapt order deadlines depending on the current workload since they are

given by the customers. Capacity buffer, such as a certain proportion of additional permanent workers or temporary hired workers, provides safety capacity to handle temporary peaks of customer demand. However, using capacity buffer faces the following limitations in order fulfilment: First, due to space and equipment constraints, there is an upper limit on the number of workers that can work simultaneously at the same processing step of the order fulfilment system. Second, due to the increasing probability of blocking, a higher number of assigned workers does not necessarily increase the overall processing performance (Furmans et al. 2009). Finally, providing safety capacity is expensive and might become a competitive drawback (Van Gils et al. 2018; Kundu et al. 2020). In conclusion, we state that inventory, time, and capacity buffer provide multiple, but limited potentials to handle volatile customer demand in order fulfilment systems.

In this thesis, we combine the potentials of time buffer and capacity buffer to develop and analytically investigate the *Strategy of Levelled Order Release* to balance workload in multi-stage, stochastic order fulfilment systems with customer-required order deadlines. The focus is on the two main application fields of order fulfilment: Warehouses and make-to-order production systems. The contribution of this thesis is three-fold and comprises

- a workload balancing concept – the *Strategy of Levelled Order Release* – for multi-stage, stochastic order fulfilment systems with customer-required order deadlines,
- a discrete-time analytical model for performance analysis and performance evaluation of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines, and
- an algorithm for capacity planning while meeting performance requirements in multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines.

1.1 Problem Description

Following the contribution of the thesis, we differentiate three major research segments of the analysis of multi-stage, stochastic order fulfilment systems, resulting in three research questions.

Workload Balancing in Order Fulfilment Systems Following the concept of variability buffers of Hopp and Spearman (2004), time buffer and capacity buffer are suitable instruments to balance variability in order fulfilment systems. By combining the potentials of these buffers, we develop a workload balancing concept for order fulfilment systems that balances system workload over time on a tactical level. The concept is inspired by the concept of Heijunka levelling. Hence, our workload balancing approach is called *Strategy of Levelled Order Release*. The related research question is the following:

> **How can we balance workload over time in multi-stage, stochastic order fulfilment systems with customer-required order deadlines?**

Performance Analysis of Order Fulfilment Systems To compare the *Strategy of Levelled Order Release* with alternative strategies and to show the benefit of workload balancing in order fulfilment systems, we model system behaviour of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines in a suitable analytical model. Based on this analytical model, we derive exact formulas for various stochastic and deterministic, system- and customer-related performance measures of order fulfilment systems. The related research question is the following:

> **How can we determine the performance of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines?**

Capacity Planning in Order Fulfilment Systems The analytical model enables the performance analysis of any order fulfilment system with a given system configuration. However, the focus of operations managers is not on the performance of a given system configuration but on adapting the current system configuration to guarantee promised performance requirements to their customers. In particular, they have to decide on the capacity that is provided at the processing steps of the order fulfilment system to meet the performance requirements of the customers. For this purpose, we provide an algorithm to solve the capacity planning problem while meeting performance requirements in order fulfilment systems based on the developed analytical model. The related research question is the following:

> **How can we determine the capacity required to meet specific performance requirements in multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines?**

1.2 Scope of the Thesis

This thesis is divided into thirteen chapters. Figure 1.2 depicts the overall structure of the thesis and assigns the chapters to the research segments of the thesis.

Initially, Chapter 1 provides the motivation and the problem description of this thesis. Chapter 2 gives a literature review on related topics, such as workload balancing, performance analysis, and capacity planning in order fulfilment systems, and states the research gap of this thesis.

The first research segment on workload balancing in order fulfilment systems comprises Chapters 3 and 4: Chapter 3 introduces a general, formal description of order fulfilment systems that forms the basis for all concepts and models developed in the subsequent chapters. In Chapter 4, we develop the *Strategy of Levelled Order Release* to balance the workload of order fulfilment systems over

time based on the fundamental ideas of Heijunka levelling in production systems and the specific characteristics and requirements of order fulfilment.

The second research segment on performance analysis of order fulfilment systems comprises Chapters 5 to 8: Chapter 5 provides methodological fundamentals and problem-specific specifications of the chosen modelling approach. In Chapters 6 and 7, we introduce an exact and a simplified analytical model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines, respectively. In Chapter 8, we compare these analytical models regarding modelling accuracy, accuracy of performance analysis, and memory and computation time requirements.

The third research segment on capacity planning in order fulfilment systems is discussed in Chapter 9. In this chapter, we formulate the decision problem of capacity planning in order fulfilment systems as a mathematical optimisation model and propose two algorithms to solve the capacity planning problem based on the analytical models introduced in the second research segment.

The last part of this thesis incorporates several numerical studies applying the models and algorithms developed in this thesis: In Chapter 10, we evaluate multiple approaches of runtime and memory optimisation to reduce computation time and memory usage of the analytical models. In Chapter 11, we evaluate the solution algorithms proposed for capacity planning regarding solution quality and runtime efficiency. In Chapter 12, we evaluate the performance of the *Strategy of Levelled Order Release* in order fulfilment systems compared to alternative strategies.

Finally, Chapter 13 summarises the contribution and the main results of this thesis and presents an outlook on future research directions.

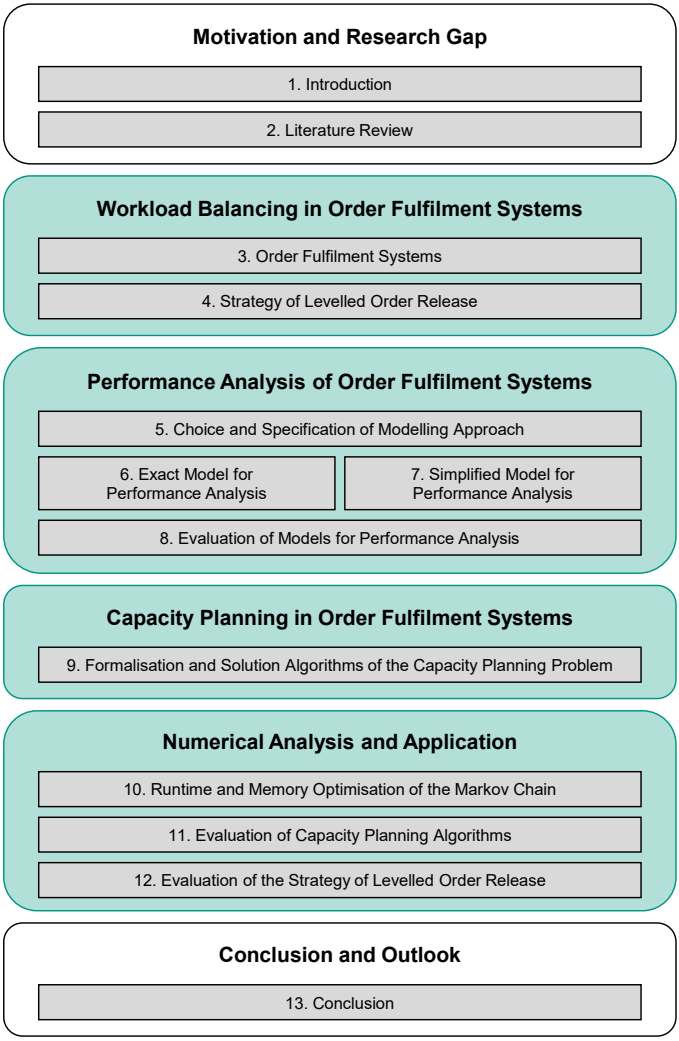


Figure 1.2: Outline of the thesis.

2 Literature Review

This chapter presents the state of the art of academic literature in the three research segments of this thesis: Workload balancing in order fulfilment systems in Section 2.1, analytical models for performance analysis of order fulfilment systems in Section 2.2, and capacity planning approaches in the context of order fulfilment in Section 2.3. The focus of each section is on the two main application fields of order fulfilment: Make-to-order production systems and warehouses. The state of the art of academic literature results from a keyword-based systematic literature review in the Scopus database. The methodology of the literature review and the keywords are described in Appendix A. The results of the literature review are summarised in Section 2.4 by defining the research gap of this thesis.

2.1 Workload Balancing

Workload balancing is defined as managing the variability of system workload over a specific time horizon (Irastorza and Deane 1974). Academic literature discusses workload balancing in different application fields: Project scheduling, vehicle routing, facility balancing, assembly line balancing, and Heijunka levelling. Table 2.1 provides a classification of workload balancing approaches in these application fields.

In *project scheduling*, resource capacity is fixed, and there are precedence constraints between the project tasks. The decision problem is to determine the start times of the project tasks provided that maximum project duration and precedence constraints are met, and provided that resource utilisation is balanced over

Table 2.1: Workload balancing approaches in different application fields (extended representation based on Vanheusden et al. (2020)).

Application field	Capacity	Tasks
Project scheduling	Fixed	Constrained by precedences
Vehicle routing	Fixed	Constrained by time windows
Facility balancing	Fixed	Unconstrained
Assembly line balancing	Fixed	Constrained by precedences
Heijunka levelling	Fixed	Constrained by production sequence
Order fulfilment	Flexible	Constrained by order deadlines

time (Rieck et al. 2012). *Vehicle routing* determines the routes of a given set of vehicles so that customer demand is fulfilled within the required time windows, and vehicles' capacity utilisation is balanced. Matl et al. (2018) provide a review of workload balancing approaches in vehicle routing. In *facility balancing*, the workload is balanced among multiple facilities and over time by adequately assigning a given set of independent tasks to a given set of facilities (Huang et al. 2006). *Assembly line balancing* problems assign tasks to stations of the assembly line so that the total workload for manufacturing one product unit is balanced among the stations and provided that precedence constraints between the tasks and the fixed number of available machines are met (Becker and Scholl 2006). *Heijunka levelling* balances both production volume and product mix of a multi-product production line over time. The decision problem of Heijunka levelling is on determining the buffer size of finished-goods-inventory that is required to guarantee a predefined service level (Matzka et al. 2012).

Workload balancing in order fulfilment differs from the above mentioned balancing approaches to some extent (see Table 2.1): First, resource capacity in order fulfilment is flexible since a high proportion of tasks is still manual (De Koster et al. 2007; Marchet et al. 2015; Peeters and Van Ooijen 2020), and it is possible to quickly hire additional temporary workers in case of shortage (Van Gils et al. 2017). Hence, we initially balance system workload over time. Subsequently, we determine the capacity that is required to meet the performance requirements.

Second, the tasks in order fulfilment are time-constrained by clear, individual, and externally given order deadlines (Öner-Közen and Minner 2017).

In the following, we provide the state of the art of workload balancing in the two main application fields of order fulfilment: Make-to-order systems in Section 2.1.1 and warehouses in Section 2.1.2.

2.1.1 Workload Balancing in Make-To-Order Systems

The classification of order review and release (ORR) policies of Bergamaschi et al. (1997) mentions workload balancing as a possible decision criterion of workload control in make-to-order production systems. *Balanced workload control* focuses on balancing the workload among the workstations of the production system. Optimisation approaches are used to minimise the deviations of the current workload from its target workload level at every workstation (Kundu et al. 2020). This lean-based approach differs from the majority of ORR policies in make-to-order production systems, which predominantly focus on strictly limiting the workload of every workstation (Kundu and Staudacher 2017). Kundu et al. (2020) provide a literature review of ORR strategies incorporating workload balancing.

While balanced workload control in job shops is widely discussed in the literature, there are only few publications on balanced workload control in flow shops: Portioli-Staudacher and Tantardini (2012) introduce a lean-based ORR policy for flow shops that aims at levelling workload along the flow. The proposed optimisation model minimises the unbalancing of workload among workstations and the unbalancing over time. Fernandes et al. (2020) develop an optimisation-based ORR policy that combines characteristics of balancing and limiting order release approaches. The corresponding optimisation model minimises the differences between the released workload and a predefined workload norm at every workstation so that lower and upper bounds of the stations' workload are met. Kundu et al. (2020) introduce a modular design of ORR policies, conduct a simulation study to better understand the interaction effects between the parameters of an ORR policy in flow shops, and develop new ORR policies for flow shops with high processing

time variability. The simulation results show that an ORR policy which uses an optimisation approach to balance the workload among the workstations enables higher system performance than ORR policies focusing on limiting the workload at the workstations.

The order fulfilment systems studied in this thesis can be seen as flow shops. However, the above mentioned publications on balanced workload control differ from our workload balancing concept regarding the balancing unit since we study workload balancing over time and not among workstations (see Table 2.2).

Table 2.2: Workload balancing concepts in the context of order fulfilment in make-to-order systems (above), in warehouses (middle), and in this thesis (below) (extended representation based on Vanheusden et al. (2020)).

	Decision level	Balancing unit	Variability buffer
Balanced workload control	Tactical	Among workstations	–
Zone sizing	Tactical	Among zones	–
Zone assignment	Tactical	Among zones	–
Bucket brigade	Operational	Among workers	–
Flexible workforce planning	Tactical	Among workers, over time	Capacity buffer
Operational Workload Balancing Problem	Operational	Over time	Time buffer, capacity buffer
Strategy of Levelled Order Release	Tactical	Over time	Time buffer, capacity buffer

[1] The concept of variability buffers only applies to workload balancing concepts using the balancing unit "time".

2.1.2 Workload Balancing in Warehouses

In warehousing, workload balancing is applied in different settings (see Table 2.2). Zone picking is a classical approach to reduce the time needed for order processing in order picking systems. For this purpose, the order picking system is subdivided into multiple zones, and each order can be simultaneously processed in the different zones. Jane (2000) and Jane and Laih (2005) provide two approaches for workload balancing among the zones of a zone order picking system in the long term: *Zone sizing* and *zone assignment*. Jane (2000) adjusts the zone size in order to balance the workload among the pickers, whereas Jane and Laih (2005) develop a clustering algorithm that assigns items to zones in order to balance the workload among the pickers. Dynamic picking systems, such as *bucket brigades*, consider workload balancing among workers in manual order picking systems in the short term. In the concept of bucket brigade, multiple pickers work in a flow line, whereby every worker follows one simple rule: Go on processing an order until your successor in the flow line is met, who takes over the order. Then, go back upstream and take over the current order of your predecessor in the flow line. If the pickers are sequenced according to their picking performance from the slowest to the fastest, the workload of the order picking system is balanced, independent of the starting positions of the workers (Bartholdi III and Eisenstein 1996). Academic literature studies different approaches for throughput improvement of bucket brigades and different configurations of order picking systems. Hong (2019) provides a current literature review on bucket brigades. In contrast to the above mentioned workload balancing concepts, this thesis focuses on workload balancing over time (see Table 2.2).

Flexible workforce planning represents a reasonable approach to balance temporary fluctuations in customer demand: Workers of a pool of fixed and temporary workers are flexibly scheduled to balance system workload over time and among workers. This approach expects a precise forecast of the daily workload. Based on forecasted workload and worker productivity, the total number of required workers and their allocation is determined. Van Gils et al. (2017) propose this

workload balancing approach for a zone order picking system and present forecasting methods to precisely forecast the daily number of incoming order lines. Flexible workforce planning concentrates on the capacity buffer for workload balancing, whereas the *Strategy of Levelled Order Release* regards a combination of capacity and time buffer (see Table 2.2).

Vanheusden et al. (2020) and Vanheusden et al. (2021) study workforce balancing as a planning step in daily resource capacity planning of warehouses:

1. Workload forecasting: Forecast the workload of the following day;
2. Workload balancing: Schedule forecasted tasks within the day so that workload peaks are avoided, and release and deadline of each task are met;
3. Workload level: Determine the number of workers required to perform the scheduled tasks.

The authors introduce the *Operational Workload Balancing Problem* (OWBP) to evenly distribute the tasks in an order picking system over the time periods of one day, whereby the number of tasks, their release, and their deadline are given. The objective function minimises the unbalancing of the scheduled workload over all time periods of one day, which is measured by the range (Vanheusden et al. 2020), the maximum, the lexicographic maximum, the variance, or the Gini coefficient (Vanheusden et al. 2021) of the scheduled workload. The OWBP is solved by an iterated local search algorithm. This approach differs from the *Strategy of Levelled Order Release* regarding decision level and the mathematical modelling of system parameters: Vanheusden et al. (2020) and Vanheusden et al. (2021) analyse a planning horizon of one day, such that customer demand and order deadlines are deterministic, and the workload is balanced on an hourly level. In contrast, we examine a planning period of multiple days. Thus, customer demand and order deadlines are stochastic, and the workload is balanced on a daily or a shift-based level (see Table 2.2).

2.2 Performance Analysis of Order Fulfilment Systems

The literature review on analytical models for performance analysis of order fulfilment systems is limited to the two main application fields of order fulfilment: Make-to-order systems in Section 2.2.1 and warehouses in Section 2.2.2.

2.2.1 Performance Analysis of Make-To-Order Systems

Analytical models, especially queueing models, for performance analysis of stochastic manufacturing systems are widely discussed in academic literature. Dallery and Gershwin (1992), Buzacott and Shanthikumar (1993), Papadopoulos and Heavey (1996), Govil and Fu (1999), and Papadopoulos et al. (2019) provide comprehensive reviews on queueing models in manufacturing. However, analytical models that focus particularly on performance analysis of make-to-order production systems are barely studied in the literature. Haskose et al. (2002) and Haskose et al. (2004) introduce queueing models for performance analysis of workload control production planning in make-to-order systems with limited buffer capacities that provide exact results for small arbitrary make-to-order systems. To analyse make-to-order systems of any size and complexity, the authors provide an approximation approach. In contrast, simulation models are commonly used to analyse make-to-order systems (Haskose et al. 2004).

Furthermore, performance analysis of make-to-order systems is often limited to system-related performance measures, such as utilisation, throughput, and inventory levels, although they are interrelated with customer-related performance measures, such as service level and order tardiness (Altendorfer and Jodlbauer 2011). Operations managers in make-to-order systems are often confronted with a trade-off between system- and customer-related performance measures: While high capacity and inventory levels are reasonable from the customer-related perspective to guarantee high service levels, they are adverse from the system-related perspective due to low system utilisation and high operational costs. Although

customer-related performance measures are key to quantify the perceived quality of service, they are often neglected in analytical models for performance analysis of make-to-order systems (Altendorfer and Jodlbauer 2011).

Customer service level measures to which extent the delivery time requirements of the customer orders are met. Thus, knowing the order deadlines is key when calculating customer service level. In the research fields of order acceptance and due date setting in make-to-order systems, order deadlines are modelled as endogenous variables: In the job entry phase of workload control, customer enquiry management determines whether to bid for an order or not and, if so, what due date and price should be (Thürer et al. 2011). Common strategies of customer enquiry management are total acceptance, acceptance based on present and future workload, and due date negotiation (Mezzogori et al. 2021), whereby the focus of academic literature is on the latter two. These two strategies have in common that orders can be rejected, and that order deadlines are determined internally. Slotnick (2011) and Thürer et al. (2019) provide literature reviews of order acceptance and due date setting, respectively. In contrast, order deadlines that result from the shipping distance, the service type chosen by the customer or delivery promises, such as next-day or same-day delivery (Yan et al. 2010; Yaman et al. 2012) are exogenous variables. These customer-required order deadlines are predominantly defined to be deterministic and constant for every order (Altendorfer 2014). Liu and Yuan (2001) specify order deadlines by a deterministic maximum delivery lead time in their queueing model for performance analysis of a simple assembly network where different components are processed at two workstations before being merged at the assembly station. Assuming exponentially distributed interarrival times, processing times, and machine breakdowns, the authors provide exact formulas for the probability distribution of the flow time, the throughput, the expected work-in-process, and the service level, and introduce a mathematical program to maximise system throughput while maintaining the required customer service level. Alternatively, Souza and Ketzenberg (2002) model order deadlines by a deterministic upper bound of the average order lead time when analysing a two-stage make-to-order remanufacturing system in order to determine the long-run production mix that maximises profit subject to a service level constraint.

However, real make-to-order systems face random, customer-required order deadlines due to different products ordered by different customers and the planning uncertainty the customers of a manufacturing company face (Altendorfer and Minner 2011). There are only few publications regarding both stochastic, customer-required order deadlines and customer-related performance measures in make-to-order systems: The heavy traffic analysis of single-stage stochastic systems with the scheduling policy *Earliest due date*, conducted by Doytchinov et al. (2001), incorporates generally distributed customer-required order deadlines, interarrival times, and processing times. The authors provide approximate formulas of customer service level and the probability distribution of customer lateness. Moreover, the priority sequencing problem in make-to-order systems with respect to minimum tardiness cost, which is modelled as Markovian decision process by Öner-Közen and Minner (2017), incorporates stochastic, customer-required order deadlines as well as the customer-related performance measures service level and expected order tardiness. The model is primarily used to evaluate the impact of priority sequencing decisions on the performance measures. However, exact performance analysis of single-stage, stochastic make-to-order systems can be conducted based on the model when using a simple due date-based priority sequencing policy, such as *Earliest due date*. Altendorfer and Jodlbauer (2011) introduce an analytical model for performance analysis of single-stage, stochastic make-to-order systems with stochastic, customer-required order deadlines that incorporates multiple customer-related performance measures, such as service level, expected lead time, expected finished-goods-inventory, and expected order tardiness. Assuming exponentially distributed interarrival times, processing times, and order deadlines, the computed performance measures are exact. For systems with arbitrary distributed parameters, the performance analysis is limited to approximations. Altendorfer and Minner (2011) extend this model to two-stage make-to-order systems in order to analyse the simultaneous optimisation of manufacturing capacities and planned lead times with respect to total inventory holding and customer order tardiness cost.

Table 2.3: Analytical models for performance analysis of stochastic make-to-order systems.

	Modelling approach	Modelling of order deadlines	Number of stages	Performance measures	Solution quality
Haskose et al. (2002, 2004)	Continuous-time	No order deadlines	Multiple	Work in process, lead time, utilisation, proportion of rejected jobs	Exact for small systems, approximate for arbitrary systems
Liu and Yuan (2001)	Continuous-time	Deterministic maximum lead time	Two	Flow time, work in process, throughput, service level	Exact for exponentially distributed parameters
Souza and Ketzenberg (2002)	Continuous-time	Deterministic upper bound of average lead time	Two	Lead time	Approximate
Doytchinov et al. (2001)	Continuous-time	Stochastic, customer-required deadlines	One	Customer lateness, service level	Approximate
Öner-Közen and Minner (2017)	Discrete-time	Stochastic, customer-required deadlines	One	Order tardiness, service level	Exact
Altendorfer and Jodlbauer (2011)	Continuous-time	Stochastic, customer-required deadlines	One	Lead time, finished-goods-inventory, order tardiness, service level	Exact for exponentially distributed parameters, approximate for arbitrary distributed parameters
Altendorfer and Minner (2012)	Continuous-time	Stochastic, customer-required deadlines	Two	Lead time, finished-goods-inventory, order tardiness, service level	Exact for exponentially distributed parameters
This thesis	Discrete-time	Stochastic, customer-required deadlines	Multiple	Service level etc. (see Table 6.1)	Exact

Table 2.3 summarises the relevant analytical models for performance analysis of stochastic make-to-order systems by classifying them regarding modelling approach, modelling of order deadlines, number of stages, performance measures, and solution quality. To the best of our knowledge, the model of Altendorfer and Minner (2011) is the only one for performance analysis of multi-stage make-to-order systems incorporating stochastic, customer-required order deadlines and customer-related performance measures. However, it differs from our model regarding the modelling approach: Altendorfer and Minner (2011) provide a continuous-time model whose solution quality depends on the shape of the probability distributions of the stochastic parameters, whereas our discrete-time model provides exact results for any arbitrary distributed parameters. Furthermore, discrete-time models enable the calculation of complete probability distributions of relevant stochastic performance measures (Schleyer and Furmans 2007).

2.2.2 Performance Analysis of Warehouses

The focus of research in warehousing is on the analysis and optimisation of specific planning and control problems, especially in order picking, such as batching, routing, zoning, sequencing, worker assignment, and storage assignment. There are numerous analytical models analysing the impact of specific control strategies on warehouse performance and determining optimal parameter values and strategy configurations in order to maximise warehouse performance. Rouwenhorst et al. (2000), Gu et al. (2007), De Koster et al. (2007), and Davarzani and Norrman (2015) provide comprehensive reviews on warehouse operations. Despite the existing interdependencies between different processes and decisions in warehouse operations, integrated modelling and analysis of multiple warehouse planning problems as well as a holistic performance analysis of warehouses are barely studied in the literature so far (Davarzani and Norrman 2015; Van Gils et al. 2017). Van Gils et al. (2018) state the potentials of integrated analysis and simultaneous solution approaches of multiple operational planning problems and provide a classification of combined planning problems in warehouse operations. The authors conclude that integrated modelling and analysis focuses on

batching and storage assignment, routing and storage assignment, and batching and routing. However, these integrated models regard combinations of multiple specific decision problems. None of them studies a holistic performance analysis of warehouses.

Furthermore, stochastic problems are rarely studied in warehouse literature, although warehouse managers suffer from several stochastic influences and uncertainties resulting from both the outside supply chain and internal processes. Deterministic models can provide good approximations of stable stochastic systems. However, when analysing warehouses with highly fluctuating parameters, deterministic models may strongly deviate from actual system behaviour and lead to wrong conclusions (Gong and De Koster 2011). Gong and De Koster (2011) state the potentials of stochastic models and optimisation approaches and provide an overview of stochastic research in warehouse operations.

The existing stochastic models for performance analysis of warehouses, which especially investigate different configurations of order picking systems, focus on system-related performance measures, such as travel time (Chew and Tang 1999; Le-Duc and De Koster 2007; Pan et al. 2014), throughput time (Tang and Chew 1997; Gong and De Koster 2011; Le-Duc and De Koster 2007; Yu and De Koster 2009; Van Nieuwenhuyse and De Koster 2009; Pan and Wu 2012; Xu et al. 2014), and throughput capacity (Van Der Gaast et al. 2020). The provided models are approximations, and they are restricted to expected values and variances of the performance measures. In contrast, customer-related performance measures, such as service level and order tardiness, are rarely considered despite their importance with respect to customer satisfaction (Van Gils et al. 2018).

Consequently, order deadlines that are essential for quantifying customer-related performance measures are barely incorporated into stochastic warehouse models. In the wave picking models of Çeven and Gue (2017) and MacCarthy et al. (2019), customer orders that arrive before a certain cutoff time are due to the next deadline corresponding to a truck departure in the shipping area of the warehouse. Thus, each wave is specified by a cutoff time and a deadline, and service level is measured as the proportion of orders that is completed until their deadline. Based on this

performance measure, called *Next Scheduled Deadline*, Çeven and Gue (2017) optimise the timing and the number of order waves, and MacCarthy et al. (2019) provide best performance frontiers to determine minimum picking rate as well as optimum timing and number of order waves. Furthermore, the discrete-time model for performance analysis of a one-block warehouse with order batching of Schleyer and Gue (2012) specifies order deadlines by a deterministic maximum throughput time. Based on this model, the authors calculate the probability distribution of the throughput time and the service level and derive the optimal batch size with respect to a service level constraint.

Table 2.4: Analytical models for performance analysis of stochastic systems in warehousing.

	Modelling approach	Modelling of order deadlines	Number of stages	Performance measures	Solution quality
Çeven and Gue (2017)	Continuous-time	Deterministic, process-oriented deadlines	One	Service level	Exact
MacCarthy et al. (2019)	Continuous-time	Deterministic, process-oriented deadlines	One	Service level	Exact
Schleyer and Gue (2012)	Discrete-time	Deterministic, maximum throughput time	One	Throughput time, service level	Exact
This thesis	Discrete-time	Stochastic, customer-required deadlines	Multiple	Service level etc. (see Table 6.1)	Exact

Table 2.4 summarises the relevant analytical models for performance analysis of stochastic systems in warehousing. Although the aforementioned models incorporate order deadlines, they are restricted to deterministic order deadlines. Furthermore, they are limited to order picking, that is, single-stage systems. Thus, there is no analytical model in warehouse literature for performance analysis

of stochastic, multi-stage order fulfilment processes with stochastic, customer-required order deadlines with respect to customer-related performance measures.

2.3 Capacity Planning in Order Fulfilment Systems

Capacity planning determines the resource requirements to meet a given customer demand over a planning horizon (Chen et al. 2009). Capacity planning problems with customer-required deadlines and due date-related performance constraints are barely studied in the literature. Table 2.5 presents an overview of the related literature that can be subdivided into performance analysis and comparison of capacity planning policies and optimisation of capacity planning. Furthermore, the publications differ regarding the number of stages, the modelling of order deadlines, and the chosen performance metric.

Bertrand and Sridharan (2001) evaluate subcontracting policies in single-stage systems with stochastic, customer-required lead times, Poisson-distributed order arrivals, and negative exponentially distributed processing times. Subcontracting is necessary since the order arrival rate is greater than the service rate. The developed subcontracting policies specify when and which orders should be subcontracted and vary regarding informational needs and complexity. In a numerical study, these policies are compared regarding the percentage of tardy orders, the expected tardiness of an order, capacity utilisation, and throughput time. Mincsovics and Dellaert (2009) propose workload-dependent capacity planning policies in a single-stage system with exponentially distributed interarrival and service times and a fixed lead time. The number of unprocessed orders is constrained by a constant upper bound, and all arriving orders exceeding this bound are refused. The workload-dependent capacity planning policies are defined by two switching points. A switching point is a specific workload value, for which an order arrival triggers a switch in capacity. The authors present one exact and one approximate analytical approach to evaluate the due date-related performance of the policies, measured by the expected number of backorders and the expected

Table 2.5: Capacity planning approaches with due date-related performance constraints: Capacity planning policies (above) and optimisation of capacity planning (below).

	Number of stages	Modelling of order deadlines	Performance metric
Bertrand and Sridharan (2001)	One	Stochastic, customer-required lead times	Percentage of tardy orders, expected tardiness of an order
Mincsovics and Dellaert (2009)	One	Deterministic lead time	Expected number of backorders, expected number of early orders
Altendorfer et al. (2014)	Multiple	Stochastic, customer-required deadlines	Service level, expected tardiness of an order
Altendorfer and Minner (2011, 2012)	Multiple	Stochastic, customer-required deadlines	Expected number of backorders
Buyukkaramikli et al. (2013)	One	Stochastic, customer-required lead times	Probability distribution of throughput time
Ma et al. (2019)	One	Deterministic maximum response time	Expected sojourn time
This thesis	Multiple	Stochastic, customer-required deadlines	Service level, probability distribution of a performance measure (see Table 6.1)

number of early orders. A policy with constant provided capacity is used as a benchmark in the numerical study. Altendorfer et al. (2014) investigate periodic capacity planning policies regarding potential improvements of service level and the expected tardiness of an order in multi-stage systems with stochastic customer demand, processing times, and customer-required deadlines. The policies differ regarding the amount of available information on the stochastic system parameters and the approach used to derive the amount of provided capacity from the amount of demanded capacity. A policy with constant provided capacity is used as a benchmark for these policies in a simulation study.

Altendorfer and Minner (2011) and Altendorfer and Minner (2012) investigate capacity planning in multi-stage systems with exponentially distributed interarrival times, processing times, and customer-required deadlines. Altendorfer and Minner (2011) examine the planned lead time release policy that releases orders to the next stage whenever the remaining time to due date is shorter than the sum of planned lead times of the remaining stages. In contrast, Altendorfer and Minner (2012) examine the work-ahead window release policy that releases an order when its remaining time to due date is shorter than the work-ahead window. Using a continuous-time queueing model, Altendorfer and Minner (2011) model the system by a series of M|M|1 queueing systems, whereby the provided capacity per stage is modelled by the stage-dependent processing rate. Altendorfer and Minner (2012) model the system by a series of M|M|m queueing systems and the provided capacity per stage by the processing rate and the number of servers per stage. Based on the queueing model, the authors derive analytic formulas for the expected values of work-in-process per stage, finished-goods-inventory, and the number of backorders. The capacity planning problem is formulated as an unconstrained optimisation problem that minimises the expected costs of work-in-process per stage, finished-goods-inventory, backorders, and capacity. Besides the provided processing rate per stage, further decision variables are considered: The planned lead time per stage in Altendorfer and Minner (2011) and the number of servers per stage and the work-ahead window in Altendorfer and Minner (2012). Ma et al. (2019) study capacity planning with a response time constraint in single-stage systems. Customer demand is Poisson-distributed, processing times are generally distributed, and there is a deterministic maximum response time per order. The system is modelled as a M|G|1 queueing system, whose processing rate specifies the required capacity. Based on the Pollaczek-Khintchine formula, the expected sojourn time of an order is calculated. The response time constraint is modelled by penalty costs that occur if the expected sojourn time of an order exceeds the maximum response time per order. Thus, the capacity planning problem is to minimise the sum of capacity cost and expected penalty cost. Buyukkaramikli et al. (2013) examine capacity planning with a throughput time constraint in single-stage systems with exponentially distributed interarrival times, processing times, and customer-required lead times. Capacity can be changed periodically,

whereby two capacity levels are possible: Permanent capacity level and permanent plus contingent capacity level. The throughput time constraint guarantees that each order is completed before its predefined lead time with a certain probability. The optimisation problem for capacity planning determines the period length, the permanent capacity level, the contingent capacity level, and the switching point at which contingent capacity is deployed, such that the capacity-related cost is minimised, and the throughput time constraint is satisfied. For this, the system is modelled as a queueing system, and the periodic capacity policy is reflected in the change of the processing rate. The capacity planning problem is solved using a search algorithm.

Except Buyukkaramikli et al. (2013), the literature on capacity planning with customer-required deadlines and due date-related performance constraints is limited to average-based performance metrics, such as the expected number of backorders, the expected sojourn time, or the expected tardiness (see Table 2.5). In contrast, in this thesis, the performance constraint of capacity planning is specified by the probability distribution of a performance measure, such as system throughput or backlog duration, or the service level of the order fulfilment system.

2.4 Chapter Conclusion

The results of the literature review state the research gap of this thesis. First, there is no workload balancing concept in the literature to balance the workload of order fulfilment systems over time on a tactical level by using a combination of time buffer and capacity buffer (see Table 2.2). Second, the discrete-time analytical model provided in this thesis is the first analytical model for performance analysis of workload balancing in order fulfilment systems that ensures a real-life representation of order fulfilment systems (see Tables 2.3, 2.4). It regards customer-required order deadlines and the interdependencies between the processing steps, and it models customer demand, customer-required deadlines, and processing performance as stochastic parameters. Furthermore, the discrete-time

analytical model enables the exact calculation of complete probability distributions for several system- and customer-related performance measures of the order fulfilment system. Third, capacity planning problems with due date-related performance constraints, which rely on probability distributions of corresponding performance measures and service level, in multi-stage systems with stochastic, customer-required order deadlines have not been studied in literature so far (see Table 2.5).

3 Order Fulfilment Systems

This chapter aims at introducing a general and formal description of order fulfilment systems, as they form the basis for all concepts and models developed in the following. In Section 3.1, we give an overview of the definition of order fulfilment in the literature. Section 3.2 introduces a formal description of order fulfilment systems. Section 3.3 summarises the results of this chapter.

3.1 Definition of Order Fulfilment

The Global Supply Chain Forum identifies order fulfilment as one of the eight key processes of supply chain management, along with customer relationship management, customer service management, demand management, manufacturing flow management, procurement, product development and commercialisation, and returns (Cooper et al. 1997). The process of order fulfilment aims at designing a logistics network that permits a company to meet customer requests while minimising total costs. It involves all activities on defining customer requirements, designing the logistics network, and filling customer orders (Croxton 2003). The activities of order fulfilment are subdivided into

- Strategic order fulfilment and
- Operational order fulfilment.

Strategic order fulfilment is on establishing a suitable structure for managing the order fulfilment process effectively and efficiently by incorporating the requirements of manufacturing, logistics, and marketing. Table 3.1 gives an overview

Table 3.1: Sub-processes of strategic order fulfilment (adopted from Croxton (2003)).

Sub-process	Tasks
Review marketing strategy, supply chain structure, customer service goals	• Review firm's strategies • Understand customer requirements • Determine capabilities of the supply chain • Determine the order fulfilment budget
Define requirements for order fulfilment	• Review the order-to-cash cycle and supply capabilities • Define lead time and customer service requirements for each customer segment • Define operational requirements • Evaluate core competencies
Evaluate logistics network	• Determine if the current network can support the requirements within financial constraints • Determine which plants produce which products, determine warehouse, plant and supplier locations, determine transportation modes
Define plan for order fulfilment	• Determine how to fill the orders from each customer segment • Make decisions about payment terms, order sizes, and packing requirements • Determine allocation rules • Assess the role of technology
Develop framework of metrics	• Link order fulfilment performance to the firm's financial performance • Determine appropriate metrics and set goals

of the sub-processes of strategic order fulfilment. In contrast, *operational order fulfilment* focuses on the execution of the order fulfilment process once it has been established. It determines how customer orders are generated and communicated, recorded, processed, documented, picked, and delivered (Croxton et al. 2001; Croxton 2003). Table 3.2 gives an overview of the sub-processes of operational order fulfilment.

Table 3.2: Sub-processes of operational order fulfilment (adopted from Croxton (2003)).

Sub-process	Tasks
Generate and communicate order	• Generate order • Transmit order
Enter order	• Receive order • Enter order • Edit order
Process order	• Check credit • Check inventory • Plan order flow and transportation
Handle documentation	• Acknowledge order • Prepare bill of lading, picking instructions, and packing slips • Generate invoice
Fill order	• Pick product • Pack product • Stage for loading • Prepare load confirmation
Deliver order	• Prepare shipping documents • Transmit delivery confirmation • Audit and pay freight bill
Perform post delivery activities and measure performance	• Receive and post payment • Record bad debt expense • Measure process performance

Lin and Shaw (1998) identify the order fulfilment process as one of the three pillars of a company, along with the product development process and the customer service process. The order fulfilment process "starts with receiving orders from the customers and ends with having the finished goods delivered" (Lin and Shaw 1998). It involves order management, manufacturing, and distribution. Order management is on receiving customer orders and committing order

requests. Manufacturing involves production scheduling, material planning, capacity planning, and shop floor control, and distribution focuses on inventory and transportation planning. Following Lin and Shaw (1998), the main objectives of order fulfilment are

- to deliver qualified products to fulfil customer demand at the right time and the right place, and
- to achieve the agility to handle uncertainties resulting from internal or external environments.

Okongwu et al. (2012) extend the definitions of Lin and Shaw (1998) and Croxton et al. (2001) by subdividing the order fulfilment process into the following phases:

- Order promising, and
- Order fulfilment.

Order promising verifies resource availability to commit reliable due dates to the customers. In contrast, *order fulfilment* focuses on the execution of the order fulfilment process after the due dates are determined. It incorporates operational disruptions, such as machine breakdowns, material shortage, and quality defects (Okongwu et al. 2012).

3.2 Formal Description

In this thesis, the focus is on operational order fulfilment. We concentrate on the phase of order fulfilment of the order fulfilment process since order promising is irrelevant under the assumption of customer-required order deadlines.

We describe order fulfilment from a superior, abstract, and process-oriented perspective (see Figure 3.1): Incoming, unprocessed orders with customer-required order deadlines determine the starting point of the order fulfilment process (see definition of Lin and Shaw (1998)). These orders are processed within the order

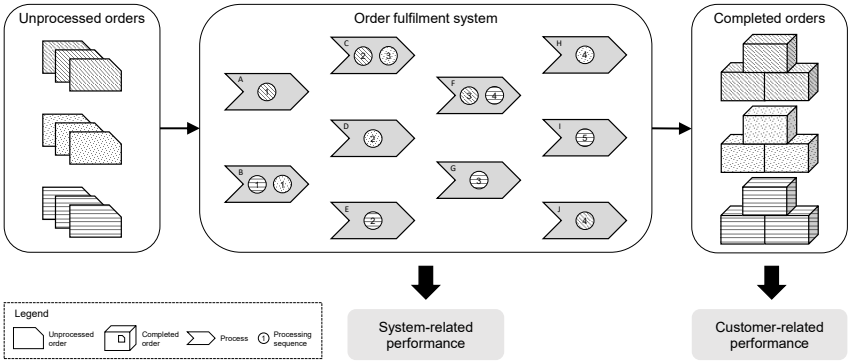


Figure 3.1: Exemplary structure of an order fulfilment system with three order types (green, red, blue) and ten processes (A-J).

fulfilment system in a sequence of multiple processing steps. We do not further specify the processing steps of the order fulfilment system in this thesis. We abstract from a subdivision into, for instance, administrative, manufacturing, and logistics tasks or a subdivision according to the sub-processes of operational order fulfilment (see Table 3.2). Instead, we model every processing step as a separate process that is specified by the parameters introduced in Section 3.2.3. Completed orders ready for delivery represent the output of the order fulfilment system (see definition of Lin and Shaw (1998)).

To evaluate the performance of the order fulfilment system regarding the overarching objectives of order fulfilment, we calculate several system- and customer-related performance measures. System-related performance measures characterise the efficiency of the order fulfilment process. Common examples are the utilisation, the number of unprocessed orders, and the number of lost sales. In contrast, customer-related performance measures specify the characteristics of the completed orders and are indicators to quantify customer satisfaction. Common examples of customer-related performance measures are the service level and the backlog duration of a completed order.

The resources that are required for order processing at each processing step of the order fulfilment system are not explicitly modelled as a separate parameter.

Instead, they are integrated into the specification of each processing step by determining the time capacity provided for order processing at each process. This approach is sufficient since we abstract from the technical design and implementation of the processing steps – manual, partly automated, or fully automated – and thus from different resource types, such as workers of different qualification levels, different variants of tools and machines, or any combination of multiple resource types. If necessary, the number of required resources at any processing step can be derived from the amount of required time capacity by considering its technical design and its specific characteristics. However, this is beyond the scope of this thesis.

We abstract from the occurrence of material shortage by assuming that all products ordered by the customers are always available in the required quantity in the order fulfilment system. Thus, aspects of inventory management and product availability are neglected in the concepts and models developed in the following.

In order fulfilment, orders are usually classified into different order types. Orders are subdivided, for instance, regarding their deadlines into standard and express orders. Furthermore, in warehouse operations, a classification of the orders in picker-to-parts and parts-to-picker can be meaningful.

In conclusion, we formally define an order fulfilment system by

- a finite set of order types $\mathcal{I} = \{1, \ldots, i_{max}\}$, each of which is specified by the parameters introduced in Section 3.2.2,
- a finite set of processes $\mathcal{P} = \{1, \ldots, p_{max}\}$, each of which is specified by the parameters introduced in Section 3.2.3, and
- a function $v : \mathcal{I} \rightarrow \{0, 1\}^{p_{max} \times p_{max}}$ defining the processing sequence of the order types in the order fulfilment system by mapping each order type

$i \in \mathcal{I}$ to an adjacency matrix, whose rows and columns correspond to the processes $p \in \mathcal{P}$ of the order fulfilment system:

$$v : \mathcal{I} \to \{0,1\}^{p_{max} \times p_{max}}$$
$$i \mapsto \mathbf{V}_i = \begin{pmatrix} v_{i,1,1} & \dots & v_{i,1,|\mathcal{P}|} \\ \vdots & \ddots & \vdots \\ v_{i,|\mathcal{P}|,1} & \dots & v_{i,|\mathcal{P}|,|\mathcal{P}|} \end{pmatrix} \tag{3.1}$$

with

$$v_{i,p,p'} = \begin{cases} 1 & \text{if process } p' \text{ succeeds process } p \text{ in} \\ & \text{processing sequence of order type } i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{I}, \forall p, p' \in \mathcal{P}. \tag{3.2}$$

3.2.1 Specification of an Order

An order represents the reference base in order fulfilment. However, an order can be specified in different ways: In the simplest case, every order corresponds to a single customer order. Furthermore, internal processing orders that result from the customer orders depending on the processing strategy of order fulfilment can represent the reference base in order fulfilment. In order batching, customer orders are consolidated to batches by some criteria, whereby every batch represents one internal processing order. Order batching is widespread in logistics and production systems since it is often more efficient to process a batch of customer orders instead of processing a single customer order. For instance, there are processing steps in production systems at which a certain number of units is processed simultaneously. Thus, it is reasonable to group multiple customer orders to one batch to efficiently use the capacity at these processing steps. Moreover, in order picking, multiple customer orders are pooled to one batch to increase the picking performance (Schleyer and Furmans 2007). In contrast, in order picking systems with zone picking, every customer order is subdivided into several internal processing orders, one per zone, to simultaneously process the customer order in the different zones (Gu et al. 2007; De Koster et al. 2007). Furthermore, in the case of high-volume

customer orders, it can be meaningful to subdivide the customer orders into their order lines, so that every order line represents one internal processing order.

The concrete specification of the reference base is not relevant for the concepts and models developed in this thesis, provided that all parameters specifying the order fulfilment system refer to the same reference base. For simplicity reasons, we sum all specifications of the reference base under the expression *order* and only use this expression in the following.

3.2.2 Specification of an Order Type

Each order type $i \in \mathcal{I}$ of an order fulfilment system is specified by the parameters *number of incoming orders* and *lead time of an order*:

Number of incoming orders The *number of incoming orders* A_i specifies the volume of orders of order type i that arrive at the order fulfilment system per time period. It is modelled as a discrete random variable with the finite range $\mathcal{A}_i = \{a_{i,min}, \ldots, a_{i,max}\}$ to ensure a real-life representation of the highly volatile customer demand in order fulfilment systems.

Lead time of an order The *lead time of an order* E_i specifies the time buffer of an order of order type i between its time of arrival and its deadline. It is measured in number of time periods. We model the lead time of an order as a discrete random variable to incorporate the real-life characteristics of order deadlines in order fulfilment: Individual and customer-required deadlines. The range of the lead time of an order is given by $\mathcal{E}_i = \{e_{i,min}, \ldots, e_{i,max}\}$. The minimum possible lead time of an order is zero. In this case, the order is due in the same time period in which it arrives at the system.

3.2.3 Specification of a Process

Each process $p \in \mathcal{P}$ of an order fulfilment system is specified by the parameters *processing performance* and *capacity*:

Processing performance The *processing performance* $L_{i,p}$ specifies the number orders of order type i that can be completely processed per time unit at process p. Operational disruptions, such as machine and tool breakdowns and rework of orders, have an impact on the processing performance. They are incorporated by modelling the processing performance as a discrete random variable. Its range is given by $\mathcal{L}_{i,p} = \{l_{i,p,min}, \ldots, l_{i,p,max}\}$.

Capacity The *capacity* $c_{i,p}$ specifies the amount of time capacity that is provided for order processing of order type i at process p in every time period. It is measured in time units, and it is modelled as a deterministic, integer, and non-negative parameter.

3.3 Chapter Conclusion

In this chapter, we presented the formal description of an order fulfilment system that forms the basis for all concepts and models developed in the following. We define an order fulfilment system by a finite set of order types, a finite set of processes, and a function mapping each order type to its processing sequence in the order fulfilment system. Each order type is characterised by the number of incoming orders and the lead time of an order. Each process is specified by its processing performance and its capacity.

4 Strategy of Levelled Order Release

This chapter aims at developing a levelling concept for order fulfilment systems that balances system workload over time. The concept is inspired by the key ideas of Heijunka levelling in production systems. In Sections 4.1 and 4.2, we present the concept of Heijunka levelling in production systems and discuss the differences between levelling in order fulfilment and levelling in production systems. Based on these findings, we develop a levelling concept for order fulfilment systems, the so-called *Strategy of Levelled Order Release*, in Section 4.3. By summarising the results of this chapter, Section 4.4 provides an answer to the first research question of this thesis.

4.1 Concept of Heijunka Levelling in Production Systems

Section 4.1 is based on Mohring et al. (2020). Parts of the following text and the figures are taken from that publication without changes.

Heijunka levelling is a simple and widespread concept in production systems to manage the production of several product types on one common production line. It has its origins in the Toyota Production System. The key idea of Heijunka levelling is to convert the volatile customer demand into a regular, recurring, and standardised production schedule to guarantee an even utilisation of the given production capacity. Furthermore, Heijunka levelling aims at supplying

downstream processes with a constant flow of small lots of different product types and at generating a constant demand of parts from upstream processes. Consequently, the need of excess capacity or stocks to handle peaks as well as the bullwip effect are reduced or even eliminated. Heijunka levelling smooths both the volume and the product mix of the production system (Matzka et al. 2012; Liker 2004, p.115f.).

Heijunka levelling is applied to one selected production step of the production line, the so-called *pacemaker*. This pacemaker production step controls the production sequence and the production rhythm of the whole production line: Kanban loops control upstream production steps, and downstream production steps form a continuous flow system. The choice of the pacemaker depends on several characteristics of the production line, such as product portfolio, production capacity, and customer requirements. The pacemaker production step is usually close to the customer. However, the further upstream the pacemaker the better (Furmans and Veit 2013; Smalley and Womack 2004, p.30).

The planning procedure of Heijunka levelling refers to one planning period of the production system, for instance, one month or one quarter. The planning period is subdivided into smaller scheduling intervals, such as one week, one day, or one shift. The planning procedure of Heijunka levelling consists of two planning steps: *System parametrisation* and *Operational planning* (see Figure 4.1).

Figure 4.1: Planning procedure of Heijunka levelling.

4.1.1 System Parametrisation

System parametrisation occurs at the beginning of every planning period and aims at smoothing both the production volume and the product mix.

Volume smoothing determines the production capacity per product type per scheduling interval that is reserved for producing this product type in each scheduling interval. For this, the total product type-specific customer demand of the planning period is evenly distributed on the scheduling intervals. The reserved product type-specific production capacity corresponds to the average customer demand per scheduling interval of this product type (Matzka et al. 2012; Furmans and Veit 2013).

Product mix smoothing determines the production sequence of the product types within a scheduling interval. Common objectives of production sequence planning are minimising setup times or maximising the regularity of the product mix. These decision problems are discussed in detail by the research area of level scheduling. Dhamala and Kubiak (2005) and Boysen et al. (2009) provide comprehensive reviews of level scheduling. The production sequence is characterised by the so-called *EPEI* ("each part each interval"). The EPEI corresponds to the time interval in which at least one part of each product type can be produced (Matzka et al. 2012).

Figure 4.2 presents a model of a Heijunka levelled kanban system. The levelling pattern on the Heijunka-board visualises the reserved production capacities and the production sequence per scheduling interval.

4.1.2 Operational Planning

Based on the levelling pattern, operational planning occurs at the beginning of each scheduling interval. The incoming customer orders of the current scheduling interval are completed by taking the required products from the finished-goods-supermarket. The associated kanbans return to the Heijunka-board. These

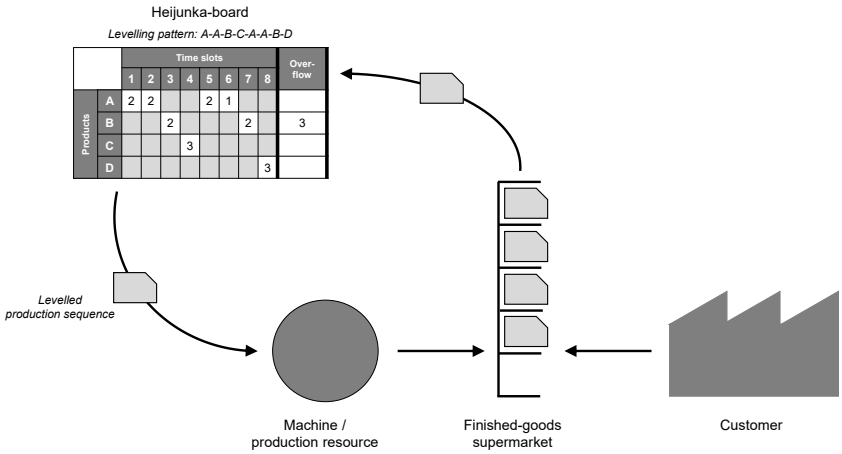


Figure 4.2: Model of a Heijunka levelled kanban system.

kanbans are assigned to the reserved production capacities of the corresponding product types on the Heijunka-board according to *First come first serve* (see Figure 4.2). If the customer demand of a product type exceeds its reserved production capacity in the current scheduling interval, the associated kanbans are stored in an overflow box. They are assigned to future scheduling intervals when the customer demand of this product type is below its reserved production capacity. If the reserved production capacity of a product type exceeds the number of free kanbans of this product type, the excess production capacity remains unused in order to avoid the production of unneeded products (Matzka et al. 2012; Furmans and Veit 2013).

4.2 Delimitation from Heijunka Levelling

The workload balancing concept of Heijunka levelling balances a volatile customer demand over time, as it is the objective of workload balancing in order fulfilment systems. However, due to differences in the system characteristics and the overarching question of the levelling concept, it is impossible to directly apply

the concept of Heijunka levelling in order fulfilment systems. Table 4.1 provides an overview of these differences, and they are discussed in detail in the following.

Table 4.1: Comparison of levelling in order fulfilment systems and Heijunka levelling in production systems regarding system characteristics (above) and overarching question (below).

	Levelling in order fulfilment	Heijunka levelling
Order lot size	Lot size of one	Lot size of at least one
Role of customer demand	System workload is directly driven by customer demand	System workload and customer demand are decoupled by a buffer
Capacity	• Determined by the number of assigned workers • Flexible in the short term	• Determined by the number of available machines • Fixed in the short term
Decision variable	Amount of provided capacity	Buffer size of finished-goods-inventory

First, production systems and order fulfilment systems differ regarding order lot size. In order fulfilment systems, orders are customer-specific in product type and product volume. The probability that multiple customers place the same order at the same time is negligible. Therefore, order lot size in order fulfilment is usually one. It is possible to have a lot size of one in production systems. However, it is predominantly more efficient to have larger lots in production systems due to the required setups between the production of different product types. Setup processes, such as tool changes, cleaning operations, and programming setups, are usually time-consuming, especially in immature production systems. Consequently, higher lot sizes lead to a decreasing frequency of setup processes and higher productivity in production systems (Dehdari 2014, p.18).

Second, customer demand plays a different role in Heijunka levelling than in the levelling concept for order fulfilment systems since the focus of Heijunka levelling is on make-to-stock production systems. Make-to-stock production systems satisfy the customer demand from the finished-goods-inventory (see Figure 4.2). Thus,

system workload and customer demand are decoupled by the finished-goods-inventory in Heijunka levelling. In contrast, in order fulfilment systems, system workload is directly driven by the customer demand since the orders are customer-specific and unknown in advance. Apart from some preparatory tasks, it is not possible to process the orders before they arrive at the order fulfilment system.

Third, there are differences regarding the characteristics of the available capacity. In order fulfilment systems, a high proportion of processing steps is still manual due to the high flexibility of workers and the high investment cost of automated systems (De Koster et al. 2007; Marchet et al. 2015; Peeters and Van Ooijen 2020). Hence, capacity in order fulfilment systems is predominantly determined by the number of assigned workers. Furthermore, capacity is flexible in the short term since it is possible to hire additional temporary workers in case of shortage (Van Gils et al. 2017). In contrast, capacity in production systems is determined by the number of available machines, which is fixed in the short term (Becker and Scholl 2006).

Finally, levelling in order fulfilment systems differs from Heijunka levelling regarding the overarching question of levelling. In both application areas, the objective is to guarantee specific promised performance requirements, such as a service level of 99%, at minimum costs. However, the starting points to achieve this objective are different due to the above mentioned differences in system characteristics: The question of Heijunka levelling is to decide on a suitable buffer size of the finished-goods-inventory to meet the promised performance requirements (Furmans 2005; Lippolt and Furmans 2008; Matzka et al. 2012; Furmans and Veit 2013). In contrast, in order fulfilment systems, the decision is on a suitable amount of provided capacity to meet the promised performance requirements.

4.3 Levelling Concept for Order Fulfilment Systems

Section 4.3 is based on Mohring et al. (2020). Parts of the following text are taken from that publication without changes.

The levelling concept for order fulfilment systems balances the volatile system workload over time and comprises two planning problems: Capacity planning and order dispatching. It is called the *Strategy of Levelled Order Release*. It is inspired by the key ideas of Heijunka levelling (see Section 4.1) and considers the specific characteristics and requirements of order fulfilment (see Section 4.2).

In order fulfilment systems, the pacemaker processing step always corresponds to the first processing step in the processing sequence of an order type since orders are customer-specific and system workload directly depends on the customer demand (see Section 4.2). The *Strategy of Levelled Order Release* is applied to the pacemaker. All downstream processing steps form a continuous flow system.

The planning procedure of the *Strategy of Levelled Order Release* consists of the planning steps of system parametrisation and operational planning, analogous to Heijunka levelling (see Figure 4.1).

4.3.1 System Parametrisation

System parametrisation occurs at the beginning of every planning period and determines the capacity per order type per scheduling interval that is reserved for order processing of this order type in each scheduling interval (smoothing of volume), and the processing sequence of the order types within a scheduling interval (smoothing of product mix).

The reserved capacity per scheduling interval $\bar{c}_{i,p}$ of order type $i \in \mathcal{I}$ at process $p \in \mathcal{P}$ is calculated as the ratio of the expected number of incoming orders per

scheduling interval $E(A_i)$ of order type i to the expected processing performance per time unit $E(L_{i,p})$ of order type i at process p:

$$\bar{c}_{i,p} = \frac{E(A_i)}{E(L_{i,p})}. \tag{4.1}$$

This approach of capacity planning should be seen as one possible, straightforward approach to determine a good initial capacity level. However, this capacity level is probably insufficient in order fulfilment systems that face high performance requirements, such as service level requirements of 99%. A solution approach to determine the minimum capacity that is required to guarantee specific performance requirements in the order fulfilment system will be presented in Chapter 9.

To determine the processing sequence of the order types, we use the same methodology as in Heijunka levelling (see Section 4.1.1). The resulting levelling pattern is visualised on the Heijunka-board (see Figure 4.3) and forms the starting point of the operational planning.

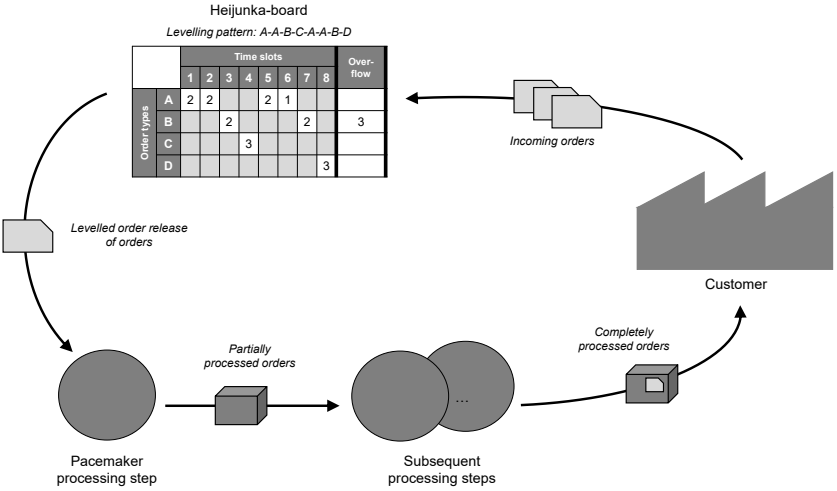


Figure 4.3: Model of an order fulfilment system with levelled order release.

4.3.2 Operational Planning

Operational planning occurs at the beginning of each scheduling interval and allocates unprocessed orders of the different order types to the corresponding reserved capacities in the levelling pattern. The pool of assignable orders comprises the order income of the current scheduling interval and the unprocessed orders remaining from previous scheduling intervals stored in the overflow box of the Heijunka-board (see Figure 4.3). To determine the processing sequence of orders of a specific order type, we consider their individual due dates as follows:

- Orders are processed according to ascending due dates.
- Orders with equal due dates are processed following *First come first serve*.

If the number of unprocessed orders of an order type exceeds its reserved capacity in the current scheduling interval, the order backlog in the overflow box increases by the corresponding number of orders. Otherwise, the remaining capacity is deployed for training, maintenance, and continuous improvement measures.

4.4 Chapter Conclusion

In this chapter, we developed a levelling concept for order fulfilment systems, the so-called *Strategy of Levelled Order Release*, based on the key ideas of Heijunka levelling in production systems. It is impossible to directly apply the concept of Heijunka levelling in order fulfilment systems due to several differences in the system characteristics between production systems and order fulfilment systems (see Table 4.1). Furthermore, the overarching question of the levelling concept is different: Heijunka levelling decides on a suitable buffer size of the finished-goods-inventory to meet the promised performance requirements, whereas in order fulfilment systems, the decision is on a suitable amount of provided capacity to meet the promised performance requirements.

The key characteristics of the *Strategy of Levelled Order Release* are the following:

- There is a fixed capacity per order type per scheduling interval reserved for order processing of orders of this order type in each scheduling interval.
- In each scheduling interval, the reserved capacity per order type is deployed to process orders of this order type according to ascending due dates.

The *Strategy of Levelled Order Release* balances volatile workload in multi-stage, stochastic order fulfilment systems with order deadlines over time by combining the potentials of time buffer and capacity buffer in order fulfilment systems: By processing the orders according to ascending due dates, the time buffer of each order between its time of arrival and its deadline is systematically exploited to balance the variability of the customer demand. The remaining variability of the customer demand is either passed on to the customer, resulting in low service levels, or it is balanced using the capacity buffer. Thus, the *Strategy of Levelled Order Release* provides an answer to the first research question of the thesis:

> **How can we balance workload over time in multi-stage, stochastic order fulfilment systems with customer-required order deadlines?**

The extent to which capacity buffer is used to balance the remaining variability depends on the specific performance requirements of the order fulfilment system. According to the *Strategy of Levelled Order Release*, the reserved capacity per order type per scheduling interval is fixed within one planning period. However, at the beginning of every planning period, capacity can be adapted individually depending on the specific performance requirements of order fulfilment since capacity in order fulfilment systems is flexible in the short term (see Section 4.2). The approach of capacity planning based on expected values presented in Section 4.3.1 should be seen as one possible, straightforward approach to determine a good initial capacity level. This capacity level is probably insufficient in order fulfilment systems that face high performance requirements, such as a service level of 99%. A solution approach to determine the minimum capacity that is required to guarantee specific performance requirements will be presented in Chapter 9.

5 Choice and Specification of Modelling Approach

This chapter aims at selecting and specifying a suitable modelling approach to model and analyse system behaviour and performance of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. We initially discuss the advantages of discrete-time analytical models compared to simulation models and continuous-time models (see Section 5.1) and introduce the theoretical basics of discrete-time Markov chains (see Section 5.2). Subsequently, Section 5.3 provides several specifications for modelling system behaviour of multi-stage order fulfilment systems with levelled order release and customer-required order deadlines as a discrete-time Markov chain. Finally, Section 5.4 summarises the results of this chapter.

5.1 Discrete-time Analytical Model

Analytical models and simulation models are widespread modelling approaches to depict and analyse system behaviour and performance of stochastic systems. Simulation models provide a high degree of freedom and enable modelling at any desired level of detail. However, modelling, validation, and performing experiments using simulation models is time-consuming, and simulation models are usually tailored to the individual characteristics of the considered system. In contrast, analytical models are of higher generality and more time efficient. However, analytical models suffer from a limited level of detail since they are often restricted to simplified representations of the real system. Furthermore,

key figures calculated based on simulation results only approximate actual system behaviour, whereas analytical models enable an exact calculation of relevant key figures (Schleyer 2007, p.17-19). Due to the drawbacks of simulation models compared to analytical models, we prefer an analytical model to analyse order fulfilment systems.

Analytical models are differentiated into continuous-time and discrete-time models. Continuous-time models depict system behaviour at any point in time $t \in \mathbb{R}_0$, whereas in discrete-time models, time is discretised into constant time intervals of length t_{inc}. System behaviour is observed at points in time t that are integer multiples of the discretisation interval t_{inc} (Stewart 1994, p.4f.):

$$t = k \cdot t_{inc} \qquad k \in \mathbb{N}_0. \tag{5.1}$$

Discrete-time models provide several advantages regarding accuracy, level of detail, and the description of real processes compared to continuous-time models (Schleyer 2007, p.13-17): First, discrete-time models provide higher accuracy in specifying the input parameters since complete probability distributions are given to specify generally distributed stochastic parameters, whereas in continuous-time models, they are only specified by the first two moments. Second, calculated key figures in continuous-time models are often limited to their expected values and their variances. However, it is essential for practitioners to know specific quantiles, for instance, when calculating the service level. Hence, complete probability distributions of the key figures are required. Discrete-time models enable the exact calculation of complete probability distributions of key figures and thus provide a more detailed analysis of system behaviour than continuous-time models. Third, real processes, such as the number of incoming orders and the processing performance in order fulfilment systems (see Section 3.2), are discrete. Furthermore, probability distributions resulting from real data that is observed in as-is analyses of the real system are discrete and of arbitrary shape. In continuous-time models, these discrete and arbitrary probability distributions are approximated by suitable parametric, theoretical probability distributions, which leads to additional computational effort and an imprecise description of the real processes. In contrast, in discrete-time models, the discrete and arbitrary

probability distributions are directly used without any additional computational effort and any loss of precision. Due to these advantages of discrete-time models, we choose a discrete-time analytical model to analyse order fulfilment systems.

When modelling an order fulfilment system as a discrete-time analytical model, one realises that the system state occurring at time $(t+1)$ only depends on the current system state at time t and the realisations of the stochastic parameters occurring between time t and time $(t+1)$. This property is called the *Markov property*. Hence, a discrete-time Markov chain is a suitable model to analyse system behaviour of order fulfilment systems.

5.2 Discrete-time Markov Chain

A *discrete-time Markov chain* is a stochastic process $\{X^t, t \in \mathcal{T}\}$ with a discrete time parameter set $\mathcal{T} = \{0, 1, \ldots\}$ and a discrete state space $\mathcal{X} = \{0, 1, \ldots\}$ whose conditional probability function satisfies the *Markov property*:

$$P(X^{t+1} = i^{t+1} \mid X^0 = i^0, \ldots, X^t = i^t) = P(X^{t+1} = i^{t+1} \mid X^t = i^t), \tag{5.2}$$

whereby X^t specifies the state of the stochastic process observed at time t. Hence, state X^t contains all relevant information concerning the history of the process (Stewart 1994, p.4f.).

The conditional probabilities $P(X^{t+1} = i^{t+1} \mid X^t = i^t)$ are called *single-step transition probabilities* or transition probabilities for short. They give the conditional probability of making a transition from state i^t at time t to state i^{t+1} at time $(t+1)$. If the transition probabilities are time-independent, the Markov chain is said to be *homogeneous*. The *transition matrix* $\mathbf{P}$ summarises the transition probabilities in a $(|\mathcal{X}| \times |\mathcal{X}|)$-dimensional stochastic matrix, whereby the entry $p_{i,j}$ in the i^{th} row and the j^{th} column corresponds to the probability of making a transition from state i at time t to state j at time $(t+1)$:

$$p_{i,j} = P(X^{t+1} = j \mid X^t = i) \qquad i, j \in \mathcal{X}. \tag{5.3}$$

The n-*step transition probability*

$$p_{i,j}^{(n)} = P(X^{t+n} = j \mid X^t = i) \qquad i, j \in \mathcal{X}, \tag{5.4}$$

is the probability of reaching state j at time $(t+n)$, in n transitions, when starting in state i at time t (Stewart 1994, p.5f.).

5.2.1 Characteristics

The states of a discrete-time Markov chain and the Markov chain itself have several mathematical characteristics. In the following, we only introduce those characteristics that become relevant in the subsequent chapters of the thesis. The following definitions are based on Privault (2013, p.117-126).

Communicating States A state $j \in \mathcal{X}$ is said to be *accessible* from state $i \in \mathcal{X}$ if it is possible to reach state j with non-zero probability in a finite number of transitions when starting in state i. If states i and j are accessible from each other, respectively, they are said to be *communicating states*. A subset of the state space $\mathcal{X}' \subset \mathcal{X}$ is called a *communicating class* if every pair of states $i, j \in \mathcal{X}'$ is communicating.

Irreducible Markov Chain A Markov chain whose state space consists of a unique communicating class is said to be an *irreducible* Markov chain. Otherwise, the Markov chain is called *reducible*.

Recurrent and Transient States A state $i \in \mathcal{X}$ is said to be a *recurrent* state if starting in state i, the Markov chain will return to state i within a finite time with probability of one. Otherwise, in the case of a probability smaller than one, state i is said to be *transient*. States of the same communicating class are either all recurrent or all transient. A communicating class of recurrent (transient) states is called a *recurrent class* (*transient class*).

Aperiodic States The period of a state $i \in \mathcal{X}$ is the greatest common divisor of $n \in \mathbb{N}$ with $p_{i,i}^{(n)} > 0$. State i is said to be *aperiodic* if it has a period of one. Otherwise, state i is *periodic*. All states of the same communicating class have the same period. A Markov chain is said to be *aperiodic* if all states are aperiodic.

5.2.2 Relevant Probability Distributions

When studying the behaviour of a discrete-time Markov chain, the following probability distributions are of particular interest.

State Probability Distribution at Time t The state probability distribution at time t

$$\begin{aligned} \pi^t &= \begin{pmatrix} \pi_0^t & \pi_1^t & \dots & \pi_{|\mathcal{X}|}^t \end{pmatrix} \\ \pi_i^t &= P(X^t = i) \qquad\qquad \forall i \in \mathcal{X}, \end{aligned} \tag{5.5}$$

specifies the probability of being in any state $i \in \mathcal{X}$ at time t. Given the initial state probability distribution π^0, the state probabilities at time t are calculated as follows (Stewart 1994, p.14)

$$\pi^t = \pi^0 \cdot \mathbf{P}^t. \tag{5.6}$$

Stationary Distribution State probabilities of a Markov chain are said to be *stationary* if any transition according to the single-step transition probabilities $\mathbf{P}$ have no effect on the state probabilities (Bolch 2006, p.41):

$$\pi = \pi \cdot \mathbf{P}. \tag{5.7}$$

In the case of an irreducible Markov chain with finite state space, the stationary distribution π is obtained by solving the following set of linear equations:

$$\pi_j = \sum_{i \in \mathcal{X}} \pi_i \cdot p_{i,j} \quad \forall j \in \mathcal{X} \tag{5.8}$$

$$\sum_{i \in \mathcal{X}} \pi_i = 1. \tag{5.9}$$

In the case of a reducible Markov chain with finite state space, its stationary distribution π consists the stationary distributions of its recurrent classes that are calculated by separately solving the set of linear equations (5.8)-(5.9) for every recurrent class. The stationary probability of every transient state is zero (Cinlar 1975, p.152-156).

Limiting Distribution The limiting distribution specifies the asymptotic behaviour of the Markov chain. Given the initial state probability distribution π^0 of the Markov chain, if the limit

$$\tilde{\pi} = \lim_{t \to \infty} \pi^t \tag{5.10}$$

exists, then this limit is called the *limiting distribution* of the Markov chain (Stewart 1994, p.15). For any aperiodic Markov chain, the limiting distribution $\tilde{\pi}$ exists. In the case of an aperiodic, irreducible Markov chain with finite state space, the limiting distribution $\tilde{\pi}$ is independent of the initial state probability distribution π^0 and corresponds to the unique stationary distribution π of the Markov chain (Bolch 2006, p.47). In the case of an aperiodic, reducible Markov chain with finite state space, the limiting distribution $\tilde{\pi}$ depends on the initial state of the Markov chain. If the initial state is recurrent, the limiting distribution $\tilde{\pi}$ corresponds to the stationary distribution of its recurrent class. Otherwise, if the initial state is transient, the limiting distribution $\tilde{\pi}$ is calculated as the weighted sum of the stationary distributions of the recurrent classes. The weights correspond to the probabilities that a certain recurrent class is accessible from the initial state. The matrix of limiting distributions $\tilde{\mathbf{\Pi}}$ summarises the limiting distributions $\tilde{\pi}$ for any initial state, whereby the i^{th} row corresponds to the limiting distribution of initial state $i \in \mathcal{X}$ (Cinlar 1975, p.152-156).

5.3 Modelling Order Fulfilment as a Discrete-time Markov Chain

We use the general description of an order fulfilment system introduced in Section 3.2 to model system behaviour of multi-stage order fulfilment systems with levelled order release and customer-required order deadlines as a discrete-time Markov chain. Section 5.3.1 introduces several specifications of the object of consideration, and Section 5.3.2 specifies the considered modelling approaches.

5.3.1 Specification of the Order Fulfilment System

We consider the different order types $i \in \mathcal{I}$ of the order fulfilment system separately since the reserved capacities per time period are order type-specific in the case of levelled order release (see Section 4.3), so that there are no interaction effects between different order types. To analyse an order fulfilment system with multiple order types $i \in \mathcal{I}$, we use one separate Markov chain for each order type.

We assume each order type $i \in \mathcal{I}$ to have a sequential processing sequence and the processes $p \in \mathcal{P}$ to be sorted according to the processing sequence of the considered order type. Thus, process $p = 1$ is the first processing step and process p_{max} is the last processing step of order type i in the order fulfilment system. An order type whose processing sequence contains splits is modelled by dividing this order type into multiple "artificial" order types, each of which has a sequential processing sequence.

We consider orders with failed due dates, but we limit the possible backlog duration of an order by the *maximum accepted backlog duration* of R time periods. Consequently, orders become lost sales once their backlog duration exceeds the maximum accepted backlog duration. The *set of due dates* $\mathcal{K}$ of an order is limited by the minimum possible due date of an order of $(-R)$ time periods and the maximum possible lead time of an order of e_{max} time periods:

$$\mathcal{K} = \{-R, \ldots, e_{max}\}\,. \tag{5.11}$$

5.3.2 Specification of the Modelling Approach

There are different approaches to model the multi-stage character of order processing and the transmission of orders between successive processing steps in a multi-stage order fulfilment system as a discrete-time Markov chain. In this thesis, we provide the following modelling approaches:

- Exact modelling approach, and
- Simplified modelling approach.

The *exact modelling approach* models each processing step of the multi-stage order fulfilment system as a separate process (see Figure 5.1). In this manner, the exact modelling approach considers the interdependencies between the processing steps by exactly recording the partially processed orders that are transmitted between successive processing steps of the multi-stage order fulfilment system. However, the high level of detail of the exact modelling approach leads to high modelling complexity and high computational effort of the corresponding discrete-time Markov chain.

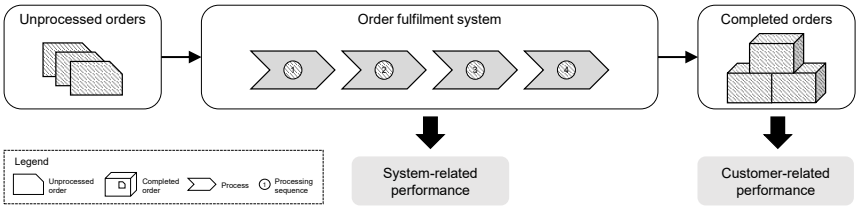


Figure 5.1: Structure of an order fulfilment system in the exact modelling approach.

In contrast, the *simplified modelling approach* abstracts from the multi-stage character of order processing by combining the multiple processing steps of the order fulfilment system to one aggregated process (see Figure 5.2). Hence, each multi-stage order fulfilment system is modelled as a single-stage order fulfilment system consisting of one aggregated process, whose processing performance results from

the processing performances of the different processing steps. Consequently, partially processed orders are neglected in the simplified model. An order is either completely unprocessed or completely processed. In this manner, this simplified modelling approach decreases both modelling complexity and computational effort of the discrete-time Markov chain.

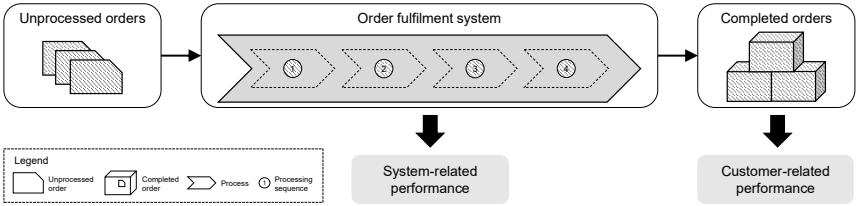


Figure 5.2: Structure of an order fulfilment system in the simplified modelling approach.

5.4 Chapter Conclusion

In this chapter, we selected and specified the modelling approach to model and analyse system behaviour and performance of a multi-stage, stochastic order fulfilment system with levelled order release and customer-required order deadlines. We chose a discrete-time Markov chain since discrete-time analytical models provide several advantages regarding accuracy, level of detail, and the description of real processes compared to continuous-time analytical models. Above all, a discrete-time Markov chain enables the exact calculation of complete probability distributions of stochastic performance measures. Thus, it provides a more detailed performance analysis of order fulfilment systems than continuous-time analytical models.

To model system behaviour of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines as a discrete-time Markov chain, we consider the order types of the order fulfilment system separately. Moreover, we assume each order type to have a sequential processing

sequence, and we limit the possible backlog duration of an order by the maximum accepted backlog duration. There are different modelling approaches to model the multi-stage character of order processing and the transmission of orders between successive processing steps in a multi-stage order fulfilment system as a discrete-time Markov chain. In this thesis, we differentiate between an exact and a simplified modelling approach. The exact and the simplified model for performance analysis will be introduced in Chapters 6 and 7. Furthermore, we will evaluate these models regarding modelling accuracy, accuracy of performance analysis, and memory and computation time requirements in Chapter 8.

6 Exact Model for Performance Analysis

This chapter aims at introducing an exact model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. Section 6.1 provides several preliminary remarks about the exact model for performance analysis. We introduce the discrete-time Markov chain in Section 6.2. Based on the limiting distribution of the Markov chain, we derive exact formulas for various stochastic and deterministic, system- and customer-related performance measures of the order fulfilment system in Section 6.3. Section 6.4 discusses the memory and computation time requirements of the provided model. By summarising the results of this chapter, Section 6.5 provides an answer to the second research question of this thesis.

6.1 Preliminaries

The exact model for performance analysis is based on the exact modelling approach. It considers a multi-stage, stochastic order fulfilment system with a finite set of processes $\mathcal{P}$, a unique order type, and levelled order release considering a finite set of due dates $\mathcal{K}$ (see Figure 5.1). The order type and each process are specified by the parameters introduced in Sections 3.2.2 and 3.2.3, respectively.

We observe the order fulfilment system at discrete-time points in time $t \in \mathbb{N}_0$ that are integer multiples of a constant time period. The time period corresponds to the scheduling interval of operational planning of the levelling concept (see

Section 4.3). It commonly has a length of one day or one shift. In each time period, the order fulfilment system is affected by two stochastic processes:

- Order processing specified by the processing performance per time period;
- Order income specified by the order income per time period.

6.1.1 Processing Performance per Time Period

The *processing performance per time period H_p of process $p \in \mathcal{P}$* specifies the number of orders that can be completely processed at process p within one time period. At least $(c_p \cdot l_{p,min})$ orders and at most $(c_p \cdot l_{p,max})$ orders can be processed at process p within one time period. Hence, the range $\mathcal{H}_p$ of the processing performance per time period H_p is defined by

$$\mathcal{H}_p = \{(c_p \cdot l_{p,min}), \ldots, (c_p \cdot l_{p,max})\}. \tag{6.1}$$

Assuming identical and independent distribution of the processing performance per time unit L_p of process p, the probability distribution of the processing performance per time period H_p of process p is computed as c_p-fold convolution of the probability distribution of L_p.

We use the random vector

$$\mathbf{H} = \begin{pmatrix} H_1 & \ldots & H_{p_{max}} \end{pmatrix} \tag{6.2}$$

to describe the *processing performance per time period of the order fulfilment system*, whereby the random variable H_p corresponds to the processing performance per time period of process $p \in \mathcal{P}$. The range of $\mathbf{H}$ is given by

$$\mathcal{H} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_{p_{max}}. \tag{6.3}$$

6.1.2 Order Income per Time Period

We specify the *order income per time period* using the random vector

$$\mathbf{G} = \begin{pmatrix} G_{-R} & \dots & G_{e_{max}} \end{pmatrix}, \tag{6.4}$$

whereby the random variable G_k corresponds to the number of incoming orders per time period with a lead time of $k \in \mathcal{K}$ time periods. Its range $\mathcal{G}$ is defined based on the following conditions:

- The lead time of each incoming order is non-negative.
- The range of every vector component G_k, $k \in \{0, \dots, e_{max}\}$, is limited downwards by zero and upwards by the maximum possible number of incoming orders per time period a_{max}.
- The total number of incoming orders per time period $(\sum_{k \in \mathcal{K}} g_k)$ corresponds to a realisation of the number of incoming orders per time period A.

$$\begin{aligned} \mathcal{G} = \Bigg\{ \mathbf{g} \in \mathbb{N}_0^{(R+e_{max}+1)} \mid\ & g_k = 0 && \forall k \in \{-R, \dots, -1\} \\ & \wedge\, g_k \in \{0, \dots, a_{max}\} && \forall k \in \{0, \dots, e_{max}\} \\ & \wedge \sum_{k \in \mathcal{K}} g_k \in \mathcal{A} \Bigg\}. \end{aligned} \tag{6.5}$$

The probability $P(\mathbf{G} = \mathbf{g})$ depends on

- the probability of having a total number of $\left(\sum_{k \in \mathcal{K}} g_k\right)$ incoming orders per time period,
- the number of combinatorial possibilities of having g_k orders with a lead time of k time periods for any lead time $k \in \mathcal{E}$, and
- the probability of having g_k orders with a lead time of k time periods for any lead time $k \in \mathcal{E}$.

$$P(\mathbf{G}=\mathbf{g}) = P\left(A=\sum_{k\in\mathcal{K}} g_k\right)\cdot\left[\prod_{k=0}^{e_{max}}\frac{(\sum_{m=k}^{e_{max}} g_m)!}{g_k!\cdot((\sum_{m=k}^{e_{max}} g_m)-g_k)!}\right] \cdot\left[\prod_{k=0}^{e_{max}} P(E=k)^{g_k}\right] \qquad \forall \mathbf{g}\in\mathcal{G}. \tag{6.6}$$

6.1.3 Stable Order Fulfilment System

We assume the order fulfilment system to be stable. An order fulfilment system is stable if its order income-related utilisation $\tilde{U}$ is smaller than one. The *order income-related utilisation* $\tilde{U}$ is calculated as the ratio of the expected value of the number of incoming orders per time period $E(A)$ to the smallest expected value of the processing performance per time period $E(H_p)$ of the processes $p \in \mathcal{P}$:

$$\tilde{U} = \frac{E(A)}{\min_{p\in\mathcal{P}} E(H_p)}. \tag{6.7}$$

6.2 Discrete-time Markov Chain

In the following, we introduce the discrete-time Markov chain that models system behaviour of a multi-stage, stochastic order fulfilment system with levelled order release and customer-required order deadlines according to the exact modelling approach.

6.2.1 System State

The system state $\mathbf{X}$ of the Markov chain depicts the number of unprocessed orders in the order fulfilment system at the beginning of any time period

$$\mathbf{X} = \begin{pmatrix} X_{1,-R} & \dots & X_{1,e_{max}} \\ \vdots & \ddots & \vdots \\ X_{p_{max},-R} & \dots & X_{p_{max},e_{max}} \end{pmatrix}, \tag{6.8}$$

whereby $X_{p,k}$, $p \in \mathcal{P}$, $k \in \mathcal{K}$, specifies the number of unprocessed orders at process p with a due date of k time periods at the beginning of the time period.

6.2.2 State Transition

The state transition from an arbitrary state $\mathbf{X}^t = \mathbf{x}$ at the beginning of time period t to state $\mathbf{X}^{t+1} = \mathbf{z}$ at the beginning of time period $(t+1)$ consists of the following sub-steps:

- Order processing in time period t:
  - Order processing at process $p = 1$,
  - $\ldots$,
  - Order processing at process p_{max},
- Due date update at the end of time period t, and
- Order income at the beginning of time period $(t+1)$.

The sub-step of order processing is further subdivided into p_{max} sub-steps since we assume that order processing at the different processes is decoupled in time. Hence, order processing at process p does not start before order processing at the previous process $(p-1)$ is completed.

Based on these $(p_{max}+2)$ sub-steps, we explain the state transition in the following. We denote the interim states of the state transition by the variables $\mathbf{y}^{(m)}$, $m \in \{0, \ldots, (p_{max}+2)\}$, whereby m is used as a counter. The initial value is given by $\mathbf{y}^{(0)} = \mathbf{x}$. Note that $\mathbf{y}^{(p-1)}$ specifies the number of unprocessed orders immediately before the start of order processing at process $p \in \mathcal{P}$, whereas $\mathbf{y}^{(p)}$ specifies the number of unprocessed orders after order processing at process $p \in \mathcal{P}$ is completed.

Sub-step of Order Processing The number of processed orders at process p in time period t depends on the realisation of the processing performance per time period $H_p = h_p$ of process p in time period t. The number of processed orders with a due date of k time periods at process p in time period t

$$\min\left\{y_{p,k}^{(p-1)}; \max\left\{0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)}\right\}\right\}$$

equals either the number of unprocessed orders $y_{p,k}^{(p-1)}$ immediately before the start of order processing at process p having a due date of k time periods or the residual processing performance remaining after all orders with a due date of $(l < k)$ time periods have already been processed. Consequently, after order processing at process p is completed, the number of unprocessed orders at process p is given by

$$\begin{aligned} y_{p,k}^{(p)} &= y_{p,k}^{(p-1)} - \min\left\{y_{p,k}^{(p-1)}; \max\left\{0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)}\right\}\right\} \\ &= \max\left\{0; y_{p,k}^{(p-1)} - \max\left\{0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)}\right\}\right\} \qquad \forall k \in \mathcal{K}, \end{aligned} \tag{6.9}$$

and the number of unprocessed orders at process $(p+1)$ equals

$$y_{(p+1),k}^{(p)} = y_{(p+1),k}^{(p-1)} + \min\left\{y_{p,k}^{(p-1)}; \max\left\{0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)}\right\}\right\} \qquad \forall k \in \mathcal{K}. \tag{6.10}$$

Sub-step of Due Date Update Due to the transition from time period t to time period $(t+1)$, we have to update the due dates of all unprocessed orders at the end of time period t by reducing their due date by one time period:

$$\begin{aligned} y_{p,k}^{(p_{max}+1)} &= y_{p,(k+1)}^{(p_{max})} \quad \forall p \in \mathcal{P}, \forall k \in \mathcal{K} \setminus \{e_{max}\} \\ y_{p,e_{max}}^{(p_{max}+1)} &= 0 \qquad\qquad\quad \forall p \in \mathcal{P}. \end{aligned} \tag{6.11}$$

Consequently, unprocessed orders having a due date of $(-R)$ time periods at the end of time period t become lost sales.

Sub-step of Order Income The incoming orders at the beginning of time period $(t+1)$ are specified by the realisation of the order income per time period $\mathbf{G} = \mathbf{g}$ at the beginning of time period $(t+1)$. We add the incoming orders to the unprocessed orders at process $p = 1$ since the processes are ordered according to the processing sequence of the considered order type:

$$\begin{aligned} y_{1,k}^{(p_{max}+2)} &= y_{1,k}^{(p_{max}+1)} + g_k \quad && \forall k \in \mathcal{K} \\ y_{p,k}^{(p_{max}+2)} &= y_{p,k}^{(p_{max}+1)} && \forall p \in \mathcal{P} \setminus \{1\}, \forall k \in \mathcal{K}. \end{aligned} \tag{6.12}$$

Transition Probability The transition probability results from the product of the probabilities of the aforementioned sub-steps since we assume that they are independent of each other. The sub-step of order processing depends on the realisation of the processing performance per time period $\mathbf{H} = \mathbf{h}$ of the order fulfilment system in time period t. Thus, the probabilities $P^{(p)}$, $p \in \mathcal{P}$, are given as follows

$$P^{(p)} = P(H_p = h_p) \qquad \forall p \in \mathcal{P}.$$

The sub-step of due date update is deterministic,

$$P^{(p_{max}+1)} = 1,$$

whereas the sub-step of order income depends on the realisation of the order income per time period $\mathbf{G} = \mathbf{g}$ at the beginning of time period $(t+1)$:

$$P^{(p_{max}+2)} = P(\mathbf{G} = \mathbf{g}).$$

By combining these probabilities, we define the transition probability of the state transition from an arbitrary state $\mathbf{X}^t = \mathbf{x}$ at the beginning of time period t to state $\mathbf{X}^{t+1} = \mathbf{z}$ at the beginning of time period $(t+1)$ as follows

$$P(\mathbf{X}^{t+1} = \mathbf{z} \mid \mathbf{X}^t = \mathbf{x}) = \sum_{(\mathbf{g},\mathbf{h}) \in \mathcal{I}(\mathbf{x},\mathbf{z})} \left[\prod_{p \in \mathcal{P}} P(H_p = h_p) \right] \cdot P(\mathbf{G} = \mathbf{g}) \qquad \forall \mathbf{x}, \mathbf{z} \in \mathcal{X}, \tag{6.13}$$

whereby set $\mathcal{I}(\mathbf{x}, \mathbf{z})$ contains all tuples $(\mathbf{g}, \mathbf{h}) \in \mathcal{G} \times \mathcal{H}$ resulting in a state transition from $\mathbf{X}^t = \mathbf{x}$ to $\mathbf{X}^{t+1} = \mathbf{z}$.

6.2.3 State Space

The state space $\mathcal{X}$ of the Markov chain is non-negative as it is impossible to observe a negative number of orders. The upper bound of the state space

$$\mathbf{O} = \begin{pmatrix} O_{1,-R} & \cdots & O_{1,e_{max}} \\ \vdots & \ddots & \vdots \\ O_{p_{max},-R} & \cdots & O_{p_{max},e_{max}} \end{pmatrix} \tag{6.14}$$

results from the structure of the state transition. We observe the maximum possible number of unprocessed orders $O_{p,k}$ at process $p \in \mathcal{P}$ with a due date of $k \in \mathcal{K}$ time periods in any time period if the following conditions hold:

- Maximum possible number of incoming orders at process p with a due date of k time periods,
- Minimum possible residual processing performance at process p for orders with a due date of k time periods, and
- Maximum possible number of orders remaining from previous time periods at process p with a due date of k time periods.

The maximum possible number of incoming orders at process $p = 1$ equals the maximum possible number of incoming orders per time period a_{max} for non-negative due dates and zero for negative due dates. At all subsequent processes $p \in \mathcal{P} \setminus \{1\}$, it corresponds to the maximum possible processing performance per time period $h_{(p-1),max}$ of the previous processing step $(p-1)$. The minimum possible residual processing performance at process p equals the minimum possible processing performance per time period $h_{p,min}$ of process p for a due date of $(-R)$ time periods. For due dates of $(k > -R)$ time periods, it is zero. The maximum possible number of orders remaining from previous time periods at process p is zero if the due date corresponds to maximum possible lead time of e_{max} time periods. For due dates of $(k < e_{max})$ time periods, it corresponds to the maximum possible number of unprocessed orders with a due date of $(k+1)$ time periods.

Thus, the components $O_{p,k}$ of the upper bound $\mathbf{O}$ are calculated as follows

$$\begin{aligned} O_{1,k} &= (e_{max} + 1) \cdot a_{max} && \forall k \in \{-R, \ldots, -1\} \\ O_{1,k} &= (e_{max} - k + 1) \cdot a_{max} && \forall k \in \{0, \ldots, e_{max}\} \\ O_{p,k} &= (e_{max} + |k|) \cdot h_{(p-1),max} && \forall p \in \mathcal{P} \setminus \{1\}, \forall k \in \{-R, \ldots, -1\} \\ O_{p,k} &= (e_{max} - k + 1) \cdot h_{(p-1),max} && \forall p \in \mathcal{P} \setminus \{1\}, \forall k \in \{0, \ldots, e_{max}\}\,. \end{aligned} \tag{6.15}$$

6.2.4 Limiting Distribution

The Markov chain has a finite state space (see Section 6.2.3), and it is possible to reach every state of the Markov chain from every other state either by a single-step transition or by an indirect transition via a finite number of other states. In the case of unreachable states, these states are excluded from the computations (see Section 10.1.1). We then find an irreducible subset of the state space. We assume that there is a tuple $(\mathbf{g}, \mathbf{h}) \in \mathcal{G} \times \mathcal{H}$ that meets the following conditions:

- The processing performance per time period is the same at every process $p \in \mathcal{P}$:
$$h_p = h \qquad \forall p \in \mathcal{P}.$$
- The total number of incoming orders per time period equals the processing performance per time period:
$$\sum_{k \in \mathcal{K}} g_k = h.$$

Thus, the irreducible subset of the state space contains state $\mathbf{x}$ with

$$\begin{aligned} x_{1,k} &= g_k && \forall k \in \mathcal{K} \\ x_{p,k} &= 0 && \forall p \in \mathcal{P} \setminus \{1\}, \forall k \in \mathcal{K} \\ \sum_{k \in \mathcal{K}} x_{1,k} &= h, \end{aligned}$$

for which

$$P(\mathbf{X}^{t+1} = \mathbf{x} \mid \mathbf{X}^t = \mathbf{x}) = \left[\prod_{p \in \mathcal{P}} P(H_p = h) \right] \cdot P(\mathbf{G} = \mathbf{g}) > 0$$

holds. Consequently, state $\mathbf{x}$ as well as all other states of the irreducible subset of the state space are aperiodic. In conclusion, the limiting distribution $\tilde{\pi}$ of the Markov chain exists, and we calculate the limiting distribution $\tilde{\pi}$ of a given initial state as described in Section 5.2.2.

6.3 Performance Measures of the Order Fulfilment System

In the following, we derive several stochastic and deterministic, system- and customer-related performance measures of the order fulfilment system from the limiting distribution of the Markov chain. Table 6.1 provides an overview of the calculated performance measures of the order fulfilment system. The probability of being in state $\mathbf{X} = \mathbf{x}$ after a sufficiently long time, in steady-state, is denoted by $P(\mathbf{X} = \mathbf{x}) := \tilde{\pi}_{\mathbf{x}}$.

6.3.1 Number of Unprocessed Orders

For each process $p \in \mathcal{P}$, we differentiate between

- the number of unprocessed orders at the beginning of a time period (performance measure Q_p, $p \in \mathcal{P}$), and
- the number of unprocessed orders immediately before the start of order processing at process p (performance measure $\tilde{Q}_p$, $p \in \mathcal{P}$).

Since order processing at the different processing steps of the order fulfilment system is decoupled in time, the values of the performances measures Q_p and $\tilde{Q}_p$ only equal for the first process. Q_p is a suitable performance measure to quantify the overall order backlog in the order fulfilment system at any point in time, and $\tilde{Q}_p$ is a meaningful performance measure for buffer sizing at the processing steps $p \in \mathcal{P}$ of the system.

Table 6.1: Performance measures of the order fulfilment system.

Category	Name	Notation
System-related performance measures	Number of unprocessed orders at process p at the beginning of the time period	Q_p
	Number of unprocessed orders in the system at the beginning of the time period	Q
	Number of unprocessed orders at process p immediately before the start of order processing at process p	$\tilde{Q}_p$
	Number of lost sales per time period at process p	S_p
	Total number of lost sales per time period	S
	Performance balance of process p	W_p
	Order backlog-related utilisation of process p	U_p
	Order income-related utilisation	$\tilde{U}$
Customer-related performance measures	Processed orders per time period	$\mathbf{M}$
	Number of processed orders per time period at process p	F_p
	Number of completed orders per time period	F
	Time difference to order deadline of a completed order	D
	Backlog duration of a completed order	$D^{backlog}$
	Time buffer of a completed order	D^{buffer}
	α-service level	SL_α
	β-service level	SL_β
	γ-service level	SL_γ

The *number of unprocessed orders Q_p at process p at the beginning of a time period* is the sum of the number of unprocessed orders $x_{p,k}$ at process p with a due date of k time periods for all due dates $k \in \mathcal{K}$. Its range $\mathcal{Q}_p$ is limited downwards by zero and upwards by the sum of the upper limit of the state space $O_{p,k}$ at process p with a due date of k time periods for all due dates $k \in \mathcal{K}$:

$$\mathcal{Q}_p = \left\{ 0, \ldots, \left(\sum_{k \in \mathcal{K}} O_{p,k} \right) \right\}. \tag{6.16}$$

The probability distribution of Q_p, $p \in \mathcal{P}$, is calculated as follows

$$\begin{aligned} P(Q_p = q_p) &= \sum_{\mathbf{x} \in \mathcal{I}_{\mathcal{Q}}(p,q_p)} P(\mathbf{X} = \mathbf{x}) \qquad \forall q_p \in \mathcal{Q}_p \\ \mathcal{I}_{\mathcal{Q}}(p, q_p) &= \left\{ \mathbf{x} \in \mathcal{X} \mid \sum_{k \in \mathcal{K}} x_{p,k} = q_p \right\}. \end{aligned} \tag{6.17}$$

Furthermore, the *number of unprocessed orders Q in the order fulfilment system at the beginning of a time period* is the sum of the number of unprocessed orders $x_{p,k}$ at process p with a due date of k time periods for all processes $p \in \mathcal{P}$ and all due dates $k \in \mathcal{K}$. Its range $\mathcal{Q}$ is limited downwards by zero and upwards by the sum of the upper limit of the state space $O_{p,k}$ at process p with a due date of k time periods for all processes $p \in \mathcal{P}$ and all due dates $k \in \mathcal{K}$:

$$\mathcal{Q} = \left\{ 0, \ldots, \left(\sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} O_{p,k} \right) \right\}. \tag{6.18}$$

The probability distribution of Q is calculated as follows

$$\begin{aligned} P(Q = q) &= \sum_{\mathbf{x} \in \mathcal{I}_{\mathcal{Q}}(q)} P(\mathbf{X} = \mathbf{x}) \qquad \forall q \in \mathcal{Q} \\ \mathcal{I}_{\mathcal{Q}}(q) &= \left\{ \mathbf{x} \in \mathcal{X} \mid \sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{p,k} = q \right\}. \end{aligned} \tag{6.19}$$

The *number of unprocessed orders $\tilde{Q}_p$ at process p immediately before the start of order processing at process p* corresponds to the sum of unprocessed orders $y_{p,k}^{(p-1)}$ at process p with a due date of k time periods immediately before the start of order processing at process p for all due dates $k \in \mathcal{K}$. Its range $\tilde{\mathcal{Q}}_p$ is limited downwards by zero. At process $p = 1$, the upper bound of $\tilde{\mathcal{Q}}_1$ is given by the sum of the upper limit of the state state $O_{1,k}$ at process $p = 1$ with a due date of k time periods for all due dates $k \in \mathcal{K}$. Otherwise, the upper bound of $\tilde{\mathcal{Q}}_p$ additionally incorporates the maximum possible processing performance per time period $h_{(p-1),max}$ of the previous processing step $(p-1)$:

$$\tilde{\mathcal{Q}}_p = \begin{cases} \left\{ 0, \ldots, \left(\sum_{k \in \mathcal{K}} O_{p,k} \right) \right\} & p = 1 \\ \left\{ 0, \ldots, \left(\sum_{k \in \mathcal{K}} O_{p,k} + h_{(p-1),max} \right) \right\} & p \in \mathcal{P} \setminus \{1\}. \end{cases} \tag{6.20}$$

The probability distribution of $\tilde{Q}_p$, $p \in \mathcal{P}$, is calculated as follows

$$\begin{aligned} P(\tilde{Q}_p = \tilde{q}_p) &= \sum_{(\mathbf{x},\mathbf{h}) \in \mathcal{I}_{\tilde{\mathcal{Q}}}(p,\tilde{q}_p)} P(\mathbf{X} = \mathbf{x}) \cdot P(\mathbf{H} = \mathbf{h}) \qquad \forall \tilde{q}_p \in \tilde{\mathcal{Q}}_p \\ \mathcal{I}_{\tilde{\mathcal{Q}}}(p,\tilde{q}_p) &= \left\{ (\mathbf{x},\mathbf{h}) \in \mathcal{X} \times \mathcal{H} \mid \sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)} = \tilde{q}_p \right\}. \end{aligned} \tag{6.21}$$

6.3.2 Number of Lost Sales

Lost sales occur in the order fulfilment system since the set of due dates $\mathcal{K}$ is limited downwards by the maximum backlog duration R. Unprocessed orders at process p with a due date of $(-R)$ time periods become lost sales in any time period t if their number exceeds the processing performance per time period of process p in time period t. Thus, some of these orders are still unprocessed at the end of time period t and their due dates fall below the minimum possible due date when the due dates are updated at the end of time period t. These unprocessed orders are removed from the order backlog of the order fulfilment system without being processed. The performance measures S_p, $p \in \mathcal{P}$, and S quantify the number of lost sales per time period at process p and the total number of lost sales per time period in the order fulfilment system, respectively.

The *number of lost sales per time period S_p at process p* corresponds to the number of unprocessed orders $y_{p,-R}^{(p)}$ with a due date of $(-R)$ time periods remaining after order processing at process p is completed. Its range $\mathcal{S}_p$ is limited by zero and the upper limit of the state space $O_{p,-R}$ at process p with a due date of $(-R)$ time periods:

$$\mathcal{S}_p = \{0, \ldots, O_{p,-R}\} \tag{6.22}$$

The probability distribution of S_p, $p \in \mathcal{P}$, is calculated as follows

$$\begin{aligned} P(S_p = s_p) &= \sum_{(\mathbf{x},\mathbf{h}) \in \mathcal{I}_{\mathcal{S}}(p,s_p)} P(\mathbf{X} = \mathbf{x}) \cdot P(\mathbf{H} = \mathbf{h}) \quad \forall s_p \in \mathcal{S}_p \\ \mathcal{I}_{\mathcal{S}}(p,s_p) &= \left\{ (\mathbf{x},\mathbf{h}) \in \mathcal{X} \times \mathcal{H} \mid y_{p,-R}^{(p)} = s_p \right\}. \end{aligned} \tag{6.23}$$

The *total number of lost sales per time period* S in the order fulfilment system is the sum of the number of unprocessed orders $y_{p,-R}^{(p)}$ with a due date of $(-R)$ time periods remaining after order processing is completed for all processes $p \in \mathcal{P}$. Its range $\mathcal{S}$ is limited downwards by zero and upwards by the sum of the upper limit of the state space $O_{p,-R}$ at process p with a due date of $(-R)$ time periods for all processes $p \in \mathcal{P}$:

$$\mathcal{S} = \left\{ 0, \ldots, \sum_{p \in \mathcal{P}} O_{p,-R} \right\} \tag{6.24}$$

The probability distribution of S is calculated as follows

$$\begin{aligned} P(S = s) &= \sum_{(\mathbf{x},\mathbf{h}) \in \mathcal{I}_{\mathcal{S}}(s)} P(\mathbf{X} = \mathbf{x}) \cdot P(\mathbf{H} = \mathbf{h}) \qquad \forall s \in \mathcal{S} \\ \mathcal{I}_{\mathcal{S}}(s) &= \left\{ (\mathbf{x}, \mathbf{h}) \in \mathcal{X} \times \mathcal{H} \mid \sum_{p \in \mathcal{P}} y_{p,-R}^{(p)} = s \right\}. \end{aligned} \tag{6.25}$$

6.3.3 Performance Balance

The *performance balance* W_p *of process* $p \in \mathcal{P}$ specifies the difference between the processing performance per time period of process p and the number of unprocessed orders immediately before the start of order processing at process p. It is measured in number of unprocessed orders, and it is a suitable performance measure to evaluate the workload at process p. A positive performance balance at process $p \in \mathcal{P}$ indicates that there is some idle time at process $p \in \mathcal{P}$ since the processing performance provided per time period of process p exceeds the number of unprocessed orders per time period immediately before the start of order processing at process p. Otherwise, in the case of a negative performance balance, the provided processing performance at process $p \in \mathcal{P}$ is fully utilised since the number of unprocessed orders per time period immediately before the start of order processing at process p exceeds the processing performance per time period of process p.

We observe a performance balance of w_p orders at process p if the processing performance per time period h_p of process p differs by w_p orders from the sum of

the number of unprocessed orders $y_{p,k}^{(p-1)}$ at process p with a due date of k time periods immediately before the start of order processing at process p for all due dates $k \in \mathcal{K}$. The range $\mathcal{W}_p$ of the performance balance of process p is limited downwards by the negated maximum possible number of unprocessed orders at process p immediately before the start of order processing at process p. Its upper bound is given by the maximum possible processing performance per time period of process p. Hence, the range $\mathcal{W}_p$ is defined as follows

$$\mathcal{W}_p = \begin{cases} \left\{-\left(\sum_{k\in\mathcal{K}} O_{p,k}\right), \ldots, h_{p,max}\right\} & p = 1 \\ \left\{-\left(\sum_{k\in\mathcal{K}} O_{p,k} + h_{(p-1),max}\right), \ldots, h_{p,max}\right\} & p \in \mathcal{P} \setminus \{1\}. \end{cases} \tag{6.26}$$

The probability distribution of W_p, $p \in \mathcal{P}$, is computed as follows

$$\begin{aligned} P(W_p = w_p) &= \sum_{(\mathbf{x},\mathbf{h})\in\mathcal{I}_{\mathcal{W}}(p,w_p)} P(\mathbf{X} = \mathbf{x}) \cdot P(\mathbf{H} = \mathbf{h}) \qquad \forall w_p \in \mathcal{W}_p \\ \mathcal{I}_{\mathcal{W}}(p, w_p) &= \left\{ (\mathbf{x}, \mathbf{h}) \in \mathcal{X} \times \mathcal{H} \mid h_p - \sum_{k\in\mathcal{K}} y_{p,k}^{(p-1)} = w_p \right\}. \end{aligned} \tag{6.27}$$

6.3.4 Utilisation

The *order backlog-related utilisation U_p of process p* describes the relation between the processing performance per time period and the number of unprocessed orders immediately before the start of order processing at process p. It considers the ratio of the number of unprocessed orders immediately before the start of order processing at process p to the processing performance per time period of process p. Its range is limited to the interval [0,1]. We calculate the order backlog-related utilisation of process p as follows

$$U_p = \sum_{(\mathbf{x},\mathbf{h})\in\mathcal{X}\times\mathcal{H}} \min\left\{1; \frac{\sum_{k\in\mathcal{K}} y_{p,k}^{(p-1)}}{h_p}\right\} \cdot P(\mathbf{X} = \mathbf{x}) \cdot P(\mathbf{H} = \mathbf{h}). \tag{6.28}$$

6.3.5 Number of Processed Orders

The number of processed orders per time period is a suitable performance measure to quantify the throughput of any process $p \in \mathcal{P}$ (performance measure F_p) as well as the throughput of the order fulfilment system (performance measure F).

We initially calculate the *processed orders per time period*

$$\mathbf{M} = \begin{pmatrix} M_{1,-R} & \dots & M_{1,e_{max}} \\ \vdots & \ddots & \vdots \\ M_{p_{max},-R} & \dots & M_{p_{max},e_{max}} \end{pmatrix}, \tag{6.29}$$

whereby $M_{p,k}$ specifies the number of processed orders per time period at process p with a due date of k time periods at their time of processing. Its range $\mathcal{M}$ is defined based on the following conditions:

- The range of every matrix component $M_{p,k}$, $p \in \mathcal{P}$, $k \in \mathcal{K}$, is limited downwards by zero and upwards by the maximum possible processing performance per time period $h_{p,max}$ of process p.
- The total number of processed orders per time period $(\sum_{k\in\mathcal{K}} m_{p,k})$ at process p corresponds to a realisation of the processing performance per time period H_p of process p.

$$\mathcal{M} = \Bigg\{ \mathbf{m} \in \mathbb{N}_0^{p_{max}} \times \mathbb{N}_0^{(R+e_{max}+1)} \mid \begin{aligned} & m_{p,k} \in \{0, \dots, h_{p,max}\} && \forall p \in \mathcal{P}, \forall k \in \mathcal{K} \\ & \wedge \sum_{k\in\mathcal{K}} m_{p,k} \in \mathcal{H}_p && \forall p \in \mathcal{P} \Bigg\}. \end{aligned} \tag{6.30}$$

The number of processed orders per time period at process p with a due date of k time periods at their time of processing

$$\min \left\{ y_{p,k}^{(p-1)}; \max \left\{ 0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)} \right\} \right\}$$

equals either the number of unprocessed orders $y_{p,k}^{(p-1)}$ at process p with a due date of k time periods immediately before the start of order processing at process p or the residual processing performance of process p remaining after all orders with a due date of $(l < k)$ time periods have already been processed.

The probability distribution of $\mathbf{M}$ is calculated as follows

$$\begin{aligned} P(\mathbf{M}=\mathbf{m}) &= \sum_{(\mathbf{x},\mathbf{h})\in\mathcal{I}_{\mathcal{M}}(\mathbf{m})} P(\mathbf{X}=\mathbf{x})\cdot P(\mathbf{H}=\mathbf{h}) \qquad \forall \mathbf{m}\in\mathcal{M} \\ \mathcal{I}_{\mathcal{M}}(\mathbf{m}) &= \Bigg\{ (\mathbf{x},\mathbf{h})\in\mathcal{X}\times\mathcal{H} \mid \\ &\qquad \min\left\{ y_{p,k}^{(p-1)}; \max\left\{ 0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)} \right\} \right\} = m_{p,k} \\ &\qquad\qquad \forall p\in\mathcal{P}, \forall k\in\mathcal{K} \Bigg\}. \end{aligned} \tag{6.31}$$

The *number of processed orders per time period F_p at process p* corresponds to the sum of the number of processed orders $m_{p,k}$ at process p with a due date of k time periods at their time of processing for all due dates $k \in \mathcal{K}$. Its range $\mathcal{F}_p$ is limited downwards by zero and upwards by the maximum possible processing performance per time period $h_{p,max}$ of process p:

$$\mathcal{F}_p = \{0, \ldots, h_{p,max}\}. \tag{6.32}$$

The probability distribution of F_p, $p \in \mathcal{P}$, is computed as follows

$$\begin{aligned} P(F_p=f_p) &= \sum_{\mathbf{m}\in\mathcal{I}_{\mathcal{F}}(p,f_p)} P(\mathbf{M}=\mathbf{m}) \qquad \forall f_p\in\mathcal{F}_p \\ \mathcal{I}_{\mathcal{F}}(p,f_p) &= \left\{ \mathbf{m}\in\mathcal{M} \mid \sum_{k\in\mathcal{K}} m_{p,k} = f_p \right\}. \end{aligned} \tag{6.33}$$

The *number of completed orders per time period* F corresponds to the number of processed orders per time period at process p_{max} of the order fulfilment system. Hence, range and probability distribution of F correspond to the ones of $F_{p_{max}}$:

$$\mathcal{F} = \mathcal{F}_{p_{max}} \tag{6.34}$$

$$P(F = f) = P\left(F_{p_{max}} = f\right) \qquad \forall f \in \mathcal{F}. \tag{6.35}$$

6.3.6 Time Difference to Order Deadline

The *time difference to order deadline* D specifies the time difference between the deadline and the actual time of completion of an order. A negative time difference to order deadline indicates that the order is completed after its deadline, whereas a positive time difference to order deadline indicates that the order is completed before reaching its deadline. The range $\mathcal{D}$ of the time difference to order deadline corresponds to the set of due dates:

$$\mathcal{D} = \mathcal{K}. \tag{6.36}$$

The probability $P(D = d)$ that a completed order has a time difference of d time periods to its deadline at its time of completion is proportional to the weighted sum of the number of processed orders per time period $m_{p_{max},d}$ at process p_{max} with a due date of d time periods at their time of completion for all realisations $\mathbf{m} \in \mathcal{M}$. By normalising this weighted sum, we calculate the probability distribution of D as follows

$$P(D = d) = \frac{\sum_{\mathbf{m} \in \mathcal{M}} m_{p_{max},d} \cdot P(\mathbf{M} = \mathbf{m})}{\sum_{\mathbf{m} \in \mathcal{M}} \left(\sum_{k \in \mathcal{K}} m_{p_{max},k}\right) \cdot P(\mathbf{M} = \mathbf{m})} \qquad \forall d \in \mathcal{D}. \tag{6.37}$$

Based on the time difference to order deadline D, we specify the performance measures backlog duration $D^{backlog}$ and time buffer D^{buffer} of a completed order in the following.

The *backlog duration of a completed order* $D^{backlog}$ equals the number of time periods by which its time of completion exceeds its deadline. Its range $\mathcal{D}^{backlog}$ is limited downwards by one and upwards by the maximum backlog duration R:

$$\mathcal{D}^{backlog} = \{1, \ldots, R\}. \tag{6.38}$$

The probability $P(D^{backlog} = d)$ that a completed order has a backlog duration of d time periods at its time of completion is proportional to the probability $P(D = -d)$ that a completed order has a time difference to deadline of $(-d)$ time periods. By normalising the probabilities $P(D = -d)$, $d \in \mathcal{D}^{backlog}$, the probability distribution of $D^{backlog}$ is computed as follows

$$P(D^{backlog} = d) = \frac{P(D = -d)}{\sum_{k=-R}^{-1} P(D = k)} \qquad \forall d \in \mathcal{D}^{backlog}. \tag{6.39}$$

The *time buffer of a completed order* D^{buffer} corresponds to the number of time periods that remain at its time of completion until reaching its deadline. Its range $\mathcal{D}^{buffer}$ corresponds to the subset of non-negative due dates:

$$\mathcal{D}^{buffer} = \{0, \ldots, e_{max}\}. \tag{6.40}$$

The probability $P(D^{buffer} = d)$ that a completed order has a time buffer of d time periods at its time of completion is proportional to the probability $P(D = d)$ that a completed order has a time difference to deadline of d time periods. By normalising the probabilities $P(D = d)$, $d \in \mathcal{D}^{buffer}$, the probability distribution of D^{buffer} is computed as follows

$$P(D^{buffer} = d) = \frac{P(D = d)}{\sum_{k=0}^{e_{max}} P(D = k)} \qquad \forall d \in \mathcal{D}^{buffer}. \tag{6.41}$$

6.3.7 Service Level

The service level of an order fulfilment system specifies the proportion of on-time completed orders on the total number of orders. Its range is limited to the interval [0,1]. Regarding the level of detail, we differentiate three types of service level:

- α-service level SL_α specifies the probability that all orders are completed on time;
- β-service level SL_β measures the proportion of on-time completed orders;
- γ-service level SL_γ considers the proportion of backorders as well as their backlog duration.

We define α-*service level* SL_α as the probability that none of the completed orders per time period has a backlog duration. Thus, we consider the probability that the sum of processed orders per time period $m_{p_{max},k}$ at process p_{max} with a negative due date $k \in \{-R, \dots, -1\}$ at its time of completion equals zero:

$$\begin{aligned} SL_\alpha &= \sum_{\mathbf{m} \in \mathcal{I}_\alpha} P(\mathbf{M} = \mathbf{m}) \\ \mathcal{I}_\alpha &= \left\{ \mathbf{m} \in \mathcal{M} \mid \sum_{k=-R}^{-1} m_{p_{max},k} = 0 \right\}. \end{aligned} \tag{6.42}$$

β-*service level* SL_β is the proportion of on-time completed orders on the total number of outgoing orders. The number of on-time completed orders per time period corresponds to the number of processed orders per time period at process p_{max} with a non-negative due date $k \in \{0, \dots, e_{max}\}$. The total number of outgoing orders per time period is the sum of the total number of processed orders per time period at process p_{max} and the total number of lost sales per time period. We calculate SL_β as follows

$$\begin{aligned} SL_\beta &= \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{h} \in \mathcal{H}} \frac{\Phi_\beta(\mathbf{x}, \mathbf{h})}{\Psi_\beta(\mathbf{x}, \mathbf{h})} \cdot P(\mathbf{X} = \mathbf{x}) \cdot P(\mathbf{H} = \mathbf{h}) \\ \Phi_\beta(\mathbf{x}, \mathbf{h}) &= \sum_{k=0}^{e_{max}} \min \left\{ y_{p_{max},k}^{(p_{max}-1)}; \max \left\{ 0; h_{p_{max}} - \sum_{l=-R}^{k-1} y_{p_{max},l}^{(p_{max}-1)} \right\} \right\} \\ \Psi_\beta(\mathbf{x}, \mathbf{h}) &= \sum_{k \in \mathcal{K}} \min \left\{ y_{p_{max},k}^{(p_{max}-1)}; \max \left\{ 0; h_{p_{max}} - \sum_{l=-R}^{k-1} y_{p_{max},l}^{(p_{max}-1)} \right\} \right\} \\ &\quad + \sum_{p \in \mathcal{P}} y_{p,-R}^{(p)}. \end{aligned} \tag{6.43}$$

We define γ-*service level* SL_γ as complementary of the proportion of completed orders having a backlog duration and the total number of lost sales on the total number of outgoing orders, whereby each component of the formula is weighted by its corresponding backlog duration. The number of completed orders per time period having a backlog duration corresponds to the number of processed orders per time period at process p_{max} with a negative due date $k \in \{-R, \dots, -1\}$. It is weighted by its backlog duration of $|k|$ time periods. Furthermore, we define the

weight of the total number of lost sales per time period by a backlog duration of $(R+1)$ time periods. The total number of outgoing orders per time period consists of the total number of processed orders at process p_{max}, which is weighted by the maximum backlog duration of R time periods, and the total number of lost sales per time period, weighted by a backlog duration of $(R+1)$ time periods. We calculate SL_γ as follows

$$\begin{aligned} SL_\gamma &= 1 - \left(\sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{h}\in\mathcal{H}} \frac{\Phi_\gamma(\mathbf{x},\mathbf{h})}{\Psi_\gamma(\mathbf{x},\mathbf{h})} \cdot P(\mathbf{X}=\mathbf{x}) \cdot P(\mathbf{H}=\mathbf{h}) \right) \\ \Phi_\gamma(\mathbf{x},\mathbf{h}) &= \sum_{k=-R}^{-1} |k| \cdot \min \left\{ y_{p_{max},k}^{(p_{max}-1)}; \max \left\{ 0; h_{p_{max}} - \sum_{l=-R}^{k-1} y_{p_{max},l}^{(p_{max}-1)} \right\} \right\} \\ &\quad + (R+1) \cdot \sum_{p\in\mathcal{P}} y_{p,-R}^{(p)} \\ \Psi_\gamma(\mathbf{x},\mathbf{h}) &= R \cdot \sum_{k\in\mathcal{K}} \min \left\{ y_{p_{max},k}^{(p_{max}-1)}; \max \left\{ 0; h_{p_{max}} - \sum_{l=-R}^{k-1} y_{p_{max},l}^{(p_{max}-1)} \right\} \right\} \\ &\quad + (R+1) \cdot \sum_{p\in\mathcal{P}} y_{p,-R}^{(p)}. \end{aligned} \tag{6.44}$$

To verify the implementation of the exact model for performance analysis, we conduct a model comparison between the exact model and a simulation model. The simulation model depicts system behaviour of a multi-stage, stochastic order fulfilment system with levelled order release and customer-required deadlines, analogous to the Markov chain. Based on the simulation results, we calculate the same performance measures of the order fulfilment system as in the exact model (see Table 6.1). A detailed description of the simulation model is provided in Appendix B. The results of this model comparison confirm that the exact model is implemented correctly. Details on the design and the results of the model comparison are given in Appendix D.

6.4 Memory and Computation Time Requirements

Computing the exact model for performance analysis consists of calculating (1) the state space, (2) the transition matrix, (3) the limiting distribution of the Markov chain, and (4) the performance measures of the order fulfilment system. The memory and computation time requirements of these steps are predominantly driven by the size of the state space of the Markov chain. As the state space increases, the transition matrix becomes larger. Thus, larger sets of linear equations have to be solved to obtain the limiting distribution. Finally, calculating the performance measures of the order fulfilment system requires more memory and computation time since they are derived from the limiting distribution.

The size of the state space $|\mathcal{X}|$ of the Markov chain is derived from the lower and upper limit of the state space $\mathcal{X}$ (see Section 6.2.3) as follows

$$
\begin{aligned}
|\mathcal{X}| &= \prod_{p\in\mathcal{P}} \prod_{k\in\mathcal{K}} (O_{p,k}+1) \\
|\mathcal{X}| &= \Big[(e_{max}+1)\cdot a_{max}+1\Big]^{R} \cdot \prod_{k=0}^{e_{max}} \Big((e_{max}-k+1)\cdot a_{max}+1\Big) \\
&\quad \cdot \prod_{p=2}^{p_{max}} \Bigg[\prod_{k=-R}^{-1} \Big((e_{max}+|k|)\cdot h_{(p-1),max}+1\Big) \\
&\qquad \cdot \prod_{k=0}^{e_{max}} \Big((e_{max}-k+1)\cdot h_{(p-1),max}+1\Big) \Bigg].
\end{aligned}
\tag{6.45}
$$

An upper limit of the size of state space is obtained by the following estimations

$$
\begin{aligned}
|\mathcal{X}| \leq & \Big[(e_{max}+1)\cdot a_{max}+1\Big]^{(R+e_{max}+1)} \\
& \cdot \prod_{p=2}^{p_{max}} \Bigg[\Big((e_{max}+R)\cdot h_{(p-1),max}+1\Big)^{R} \\
& \qquad \cdot \Big((e_{max}+1)\cdot h_{(p-1),max}+1\Big)^{(e_{max}+1)} \Bigg]
\end{aligned}
\tag{6.46}
$$

$$|\mathcal{X}| \leq \Big[(e_{max}+1)\cdot a_{max}+1\Big]^{(R+e_{max}+1)} \cdot \Big[(e_{max}+R)\cdot h_{max}+1\Big]^{R\cdot(p_{max}-1)} \cdot \Big[(e_{max}+1)\cdot h_{max}+1\Big]^{(e_{max}+1)(p_{max}-1)} \tag{6.47}$$

with

$$h_{max} = \max_{p=1,\ldots,(p_{max}-1)} \{h_{p,max}\}\,.$$

From equation (6.47), we recognize that the maximum backlog duration R, the number of processes p_{max}, and the maximum possible values of the number of incoming orders per time period A, the lead time E, and the processing performances per time period H_p of the processes $p \in \mathcal{P}$ determine the size of the state space of the Markov chain. The maximum backlog duration R, the maximum possible lead time e_{max}, and the number of processes p_{max} have a major impact on the size of the state space since they determine the exponents in equation (6.47).

Apart from the size of the state space, the computational effort of calculating the transition matrix depends on the discretisation of the stochastic parameters. The *discretisation* of a stochastic parameter quantifies the number of its possible realisations. It results from the lower and upper limit of the range of the stochastic parameter. The higher the discretisation of the stochastic parameters $\mathbf{G}$ and $\mathbf{H}$ of the Markov chain, which results from the discretisation of the stochastic parameters A, E, and L_p, $p \in \mathcal{P}$, of the order fulfilment system, the more tuples $(\mathbf{g}, \mathbf{h})$ of possible realisations exist. Each of these tuples has to be considered when calculating the entries of the transition matrix (see equation (6.13)).

6.5 Chapter Conclusion

In this chapter, we introduced an exact analytical model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines by

- modelling system behaviour of such order fulfilment systems according to the exact modelling approach (see Figure 5.1) as a discrete-time Markov chain, and
- calculating multiple stochastic and deterministic, system- and customer-related performance measures of order fulfilment systems exactly based on the limiting distribution of the Markov chain.

Based on the calculated performance measures (see Table 6.1), such as system throughput, service level, utilisation, and backlog duration and time buffer of a completed order, the developed model enables an exact and comprehensive performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. Consequently, the exact model for performance analysis provides an answer to the second research question of the thesis:

> **How can we determine the performance of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines?**

7 Simplified Model for Performance Analysis

This chapter aims at introducing a simplified model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. Section 7.1 provides several preliminary remarks about the simplified model for performance analysis. We introduce the discrete-time Markov chain in Section 7.2 and calculate multiple performance measures of the order fulfilment system in Section 7.3. Section 7.4 discusses the memory and computation time requirements of the provided model and Section 7.5 summarises the results of this chapter.

7.1 Preliminaries

The simplified model for performance analysis is based on the simplified modelling approach. It considers a multi-stage, stochastic order fulfilment system with a finite set of processes $\mathcal{P}$, a unique order type, and levelled order release considering a finite set of due dates $\mathcal{K}$. The processing steps $p \in \mathcal{P}$ are combined to one aggregated process (see Figure 5.2). The order type and each process is specified by the parameters introduced in Sections 3.2.2 and 3.2.3.

We observe the order fulfilment system at discrete-time points in time $t \in \mathbb{N}_0$ that are integer multiples of a constant time period. The time period corresponds to the scheduling interval of operational planning of the levelling concept (see Section 4.3). It commonly has a length of one day or one shift. In each time period, the order fulfilment system is affected by two stochastic processes:

- Order processing specified by the aggregated processing performance per time period;
- Order income specified by the order income per time period.

7.1.1 Aggregated Processing Performance per Time Period

The *aggregated processing performance per time period* of the order fulfilment system specifies the number of orders that can be completely processed in the order fulfilment system within one time period. We calculate the aggregated processing performance per time period $\tilde{H}$ as the minimum of the processing performances per time period of the processes $p \in \mathcal{P}$:

$$\tilde{H} = \min_{p \in \mathcal{P}} \{H_p\} . \tag{7.1}$$

Hence, its range $\tilde{\mathcal{H}}$ results from the smallest value of the minimum possible processing performances per time period $h_{p,min}$ of the processes $p \in \mathcal{P}$ and the smallest value of the maximum possible processing performances per time period $h_{p,max}$ of the processes $p \in \mathcal{P}$ as follows

$$\tilde{\mathcal{H}} = \left\{ \min_{p \in \mathcal{P}} \{h_{p,min}\}, \ldots, \min_{p \in \mathcal{P}} \{h_{p,max}\} \right\} . \tag{7.2}$$

Assuming that the processing performance per time period of the processes $p \in \mathcal{P}$ are independent of each other, the probability distribution of $\tilde{H}$ is given by

$$\begin{aligned} P(\tilde{H} = \tilde{h}) &= \sum_{\mathbf{h} \in \mathcal{I}_{\tilde{\mathcal{H}}}(\tilde{h})} \left[\prod_{p \in \mathcal{P}} P(H_p = h_p) \right] \quad \forall \tilde{h} \in \tilde{\mathcal{H}} \\ \mathcal{I}_{\tilde{\mathcal{H}}}(\tilde{h}) &= \left\{ \mathbf{h} \in \mathcal{H} \mid \min_{p \in \mathcal{P}} \{h_p\} = \tilde{h} \right\} . \end{aligned} \tag{7.3}$$

7.1.2 Order Income per Time Period

The order income per time period is specified by the random vector $\mathbf{G}$, analogous to the exact model (see Section 6.1.2). The range and the probability distribution of $\mathbf{G}$ are given by equations (6.5) and (6.6), respectively.

7.1.3 Stable Order Fulfilment System

We assume the order fulfilment system to be stable. An order fulfilment system is stable, if its order income-related utilisation $\tilde{U}$ is smaller than one. In the simplified model, the *order income-related utilisation* $\tilde{U}$ is calculated as the ratio of the expected value of the number of incoming orders per time period $E(A)$ to the expected value of the aggregated processing performance per time period $E(\tilde{H})$:

$$\tilde{U} = \frac{E(A)}{E(\tilde{H})}. \tag{7.4}$$

7.2 Discrete-time Markov Chain

In the following, we introduce the discrete-time Markov chain modelling system behaviour of a multi-stage, stochastic order fulfilment system with levelled order release and customer-required deadlines according to the simplified modelling approach.

7.2.1 System State

The system state $\mathbf{X}$ of the Markov chain depicts the number of unprocessed orders in the order fulfilment system at the beginning of any time period. Since order fulfilment systems are modelled as single-stage systems in the simplified model, we neglect the process index p. Hence, the definition of system state simplifies to

$$\mathbf{X} = \begin{pmatrix} X_{-R} & \dots & X_{e_{max}} \end{pmatrix}, \tag{7.5}$$

whereby X_k, $k \in \mathcal{K}$, specifies the number of unprocessed orders with a due date of k time periods at the beginning of the time period.

7.2.2 State Transition

The state transition from an arbitrary state $\mathbf{X}^t = \mathbf{x}$ at the beginning of time period t to state $\mathbf{X}^{t+1} = \mathbf{z}$ at the beginning of time period $(t+1)$ consists of the same sub-steps as the state transition of the exact model (see Section 6.2.2). However, the number of sub-steps reduces to three since only order processing at one process, namely the aggregated process, has to be modelled.

Due to these simplifications, state transition and transition probability can be expressed in a closed formula as follows

$$
\begin{aligned}
P(\mathbf{X}^{t+1} = \mathbf{z} \mid \mathbf{X}^t = \mathbf{x}) = & \sum_{(\mathbf{g},\tilde{h}) \in \mathcal{I}(\mathbf{x},\mathbf{z})} P(\mathbf{G} = \mathbf{g}) \cdot P(\tilde{H} = \tilde{h}) \\
\mathcal{I}(\mathbf{x},\mathbf{z}) = & \Bigg\{ (\mathbf{g},\tilde{h}) \in \mathcal{G} \times \tilde{\mathcal{H}} \mid \\
& g_k + \max\left\{0; x_{k+1} - \max\left\{0; \tilde{h} - \sum_{l=-R}^{k} x_l\right\}\right\} = z_k \\
& \qquad\qquad \forall k \in \mathcal{K} \setminus \{e_{max}\} \\
& \wedge \; g_{e_{max}} = z_{e_{max}} \Bigg\} \qquad \forall \mathbf{x}, \mathbf{z} \in \mathcal{X}.
\end{aligned}
\tag{7.6}
$$

The number of unprocessed orders z_k with a due date of k time periods at the beginning of time period $(t+1)$ is the sum of the number of incoming orders per time period g_k with a lead time of k time periods at the beginning of time period $(t+1)$ and the number of unprocessed orders with a due date of $(k+1)$ time periods at the end of time period t

$$
\max\left\{0; x_{k+1} - \max\left\{0; \tilde{h} - \sum_{l=-R}^{k} x_l\right\}\right\}.
$$

The number of unprocessed orders with a due date of $(k+1)$ time periods at the end of time period t is either zero or it corresponds to the difference of the

number of unprocessed orders x_{k+1} with a due date of $(k+1)$ time periods at the beginning of time period t and the residual processing performance of time period t remaining after all orders with a due date of $(l < k+1)$ time periods have already been processed. The number of unprocessed orders $z_{e_{max}}$ with a due date of e_{max} time periods at the beginning of time period $(t+1)$ equals the number of incoming orders per time period $g_{e_{max}}$ with a lead time of e_{max} time periods at the beginning of time period $(t+1)$.

7.2.3 State Space

We derive the state space $\mathcal{X}$ of the Markov chain based on the same considerations as in the exact model (see Section 6.2.3). The state space $\mathcal{X}$ is non-negative, and its upper bound

$$\mathbf{O} = \begin{pmatrix} O_{-R} & \dots & O_{e_{max}} \end{pmatrix} \tag{7.7}$$

is defined as follows

$$\begin{aligned} O_k &= (e_{max}+1) \cdot a_{max} && \forall k \in \{-R, \dots, -1\} \\ O_k &= (e_{max}-k+1) \cdot a_{max} && \forall k \in \{0, \dots, e_{max}\}\,. \end{aligned} \tag{7.8}$$

7.2.4 Limiting Distribution

The Markov chain has a finite state space (see Section 7.2.3), and it is possible to reach every state of the Markov chain from every other state either by a single-step transition or by an indirect transition via a finite number of other states. In the case of unreachable states, these states are excluded from the computations (see Section 10.1.1). We then find an irreducible subset of the state space. We assume that there is a tuple $(\mathbf{g}, \tilde{h}) \in \mathcal{G} \times \tilde{\mathcal{H}}$ for which the total number of incoming orders per time period equals the processing performance per time period:

$$\sum_{k \in \mathcal{K}} g_k = \tilde{h}.$$

Thus, the irreducible subset of the state space contains state $\mathbf{x}$ with

$$x_k = g_k \quad \forall k \in \mathcal{K}$$
$$\sum_{k \in \mathcal{K}} x_k = \tilde{h},$$

for which

$$P(\mathbf{X}^{t+1} = \mathbf{x} \mid \mathbf{X}^t = \mathbf{x}) = P(\tilde{H} = \tilde{h}) \cdot P(\mathbf{G} = \mathbf{g}) > 0$$

holds. Consequently, state x as well as all other states of the irreducible subset of the state space are aperiodic. In conclusion, the limiting distribution $\tilde{\pi}$ of the Markov chain exists, and we calculate the limiting distribution $\tilde{\pi}$ of a given initial state as described in Section 5.2.2.

7.3 Performance Measures of the Order Fulfilment System

In the simplified model, we calculate the same performance measures of the order fulfilment system as in the exact model (see Table 6.1) using the formulas introduced in Section 6.3. However, the subsequent adjustments are necessary:

- Replace the limiting distribution of the exact model by the one of the simplified model, and
- Replace the processing performance per time period $\mathbf{H}$ by the aggregated processing performance per time period $\tilde{H}$.

In the exact model, we subdivide the number of unprocessed orders into two key figures regarding their time of consideration (see Section 6.3.1). This differentiation is not meaningful in the simplified model since in single-stage order fulfilment systems, the point in time immediately before the start of order processing at a process corresponds to the beginning of the time period. Furthermore, the differentiation between process- and system-specific key figures regarding the

number of unprocessed orders, the number of lost sales, and the number of processed orders is no longer reasonable in the simplified model.

7.4 Memory and Computation Time Requirements

Memory and computation time requirements of the simplified model for performance analysis depend on the same factors as the ones of the exact model for performance analysis (see Section 6.4): The size of the state space of the Markov chain and the discretisation of the stochastic parameters.

In the simplified model, the size of the state space $|\mathcal{X}|$ of the Markov chain is derived from the lower and upper limits of the state space $\mathcal{X}$ (see Section 7.2.3) as follows

$$\begin{aligned} |\mathcal{X}| &= \prod_{k \in \mathcal{K}} (O_k + 1) \\ |\mathcal{X}| &= \Big[(e_{max} + 1) \cdot a_{max} + 1\Big]^R \cdot \prod_{k=0}^{e_{max}} \Big((e_{max} - k + 1) \cdot a_{max} + 1\Big). \end{aligned} \tag{7.9}$$

An upper limit of the size of state space is obtained by the following estimation

$$|\mathcal{X}| \leq \Big[(e_{max} + 1) \cdot a_{max} + 1\Big]^{(R + e_{max} + 1)}. \tag{7.10}$$

From equation (7.10), we recognize that the maximum backlog duration R and the maximum possible values of the number of incoming order per time period A and the lead time E determine the size of the state space of the Markov chain. The maximum backlog duration R and the maximum possible lead time e_{max} have a major impact on the size of the state space since they determine the exponent in equation (7.10).

7.5 Chapter Conclusion

In this chapter, we introduced a simplified analytical model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines by

- modelling system behaviour of such order fulfilment systems according to the simplified modelling approach (see Figure 5.2) as a discrete-time Markov chain, and
- calculating multiple stochastic and deterministic, system- and customer-related performance measures of order fulfilment system based on the limiting distribution of the Markov chain.

The simplified model can be seen as a special case of the exact model introduced in Chapter 6 since it ensues from the exact model when modelling is restricted to single-stage order fulfilment systems. In Chapter 8, we will evaluate and compare both models for performance analysis regarding modelling accuracy, accuracy of performance analysis, and memory and computation time requirements.

8 Evaluation of Models for Performance Analysis

The exact and the simplified model for performance analysis differ regarding the modelling of partially processed orders that are transmitted between the processing steps of a multi-stage order fulfilment system. The exact model provides an exact recording of partially processed orders, whereas partially processed orders are neglected in the simplified model. This chapter aims at evaluating and comparing the exact and the simplified model for performance analysis regarding modelling accuracy (see Section 8.1), accuracy of performance analysis (see Section 8.2), and memory and computation time requirements (see Section 8.3). By summarising the results of this chapter, Section 8.4 provides recommendations when to use which model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required deadlines.

8.1 Modelling Accuracy

The main difference between the exact and the simplified model concerns the modelling of partially processed orders that are transmitted between the processing steps of a multi-stage order fulfilment system. The exact model considers partially processed orders by exactly recording the unprocessed orders per process (see Figure 5.1). In contrast, partially processed orders are neglected in the simplified model (see Figure 5.2) since an order is either completely unprocessed or completely processed. In the following, we initially derive two hypotheses on the impact of the modelling of partially processed orders on the modelling

accuracy of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required deadlines (see Section 8.1.1). Subsequently, we prove that these hypotheses hold (see Section 8.1.2).

8.1.1 Hypotheses

In order fulfilment systems with low utilisation, the expected value of the number of incoming orders per time period is significantly smaller than the expected value of the aggregated processing performance per time period (see equation (7.4)). The probability that the number of incoming orders per time period exceeds the minimum of the processing performances per time period of the processes $p \in \mathcal{P}$ in any time period is close to zero:

$$\begin{aligned}
\frac{E(A)}{E(\tilde{H})} << 1 &\Leftrightarrow E(A) << E(\tilde{H}) \\
&\Leftrightarrow \sum_{a \in \mathcal{A}} a \cdot P(A = a) << \sum_{\tilde{h} \in \tilde{\mathcal{H}}} \tilde{h} \cdot P(\tilde{H} = \tilde{h}) \\
&\Leftrightarrow \sum_{\mathbf{g} \in \mathcal{G}} \left(\sum_{k \in \mathcal{K}} g_k \right) P(\mathbf{G} = \mathbf{g}) << \sum_{\mathbf{h} \in \mathcal{H}} \min_{p \in \mathcal{P}} \{h_p\} \cdot P(\mathbf{H} = \mathbf{h}) \\
&\Rightarrow \sum_{(\mathbf{g}, \mathbf{h}) \in \mathcal{G} \times \mathcal{H}} P\left(\sum_{k \in \mathcal{K}} g_k > \min_{p \in \mathcal{P}} \{h_p\} \right) \rightarrow 0.
\end{aligned} \tag{8.1}$$

Consequently, the majority of the incoming orders per time period is processed immediately after their time of arrival without being buffered at any processing step, and the probability that partially processed orders occur at any processing step is close to zero.

In contrast, in order fulfilment systems with high utilisation, the expected value of the number of incoming orders per time period is approximately as high as the expected value of the aggregated processing performance per time period. There is a significant probability that the number of incoming orders per time

period exceeds the minimum of the processing performances per time period of the processes $p \in \mathcal{P}$ in any time period:

$$\begin{aligned}\frac{E(A)}{E(\tilde{H})} \to 1 \Leftrightarrow E(A) &\approx E(\tilde{H}) \\ \Leftrightarrow \sum_{a \in \mathcal{A}} a \cdot P(A = a) &\approx \sum_{\tilde{h} \in \tilde{\mathcal{H}}} \tilde{h} \cdot P(\tilde{H} = \tilde{h}) \\ \Leftrightarrow \sum_{\mathbf{g} \in \mathcal{G}} \left(\sum_{k \in \mathcal{K}} g_k \right) P(\mathbf{G} = \mathbf{g}) &\approx \sum_{\mathbf{h} \in \mathcal{H}} \min_{p \in \mathcal{P}} \{h_p\} \cdot P(\mathbf{H} = \mathbf{h}) \\ \Rightarrow \sum_{(\mathbf{g},\mathbf{h}) \in \mathcal{G} \times \mathcal{H}} P &\left(\sum_{k \in \mathcal{K}} g_k > \min_{p \in \mathcal{P}} \{h_p\} \right) >> 0.\end{aligned} \tag{8.2}$$

Consequently, there is a significant probability that temporary buffers occur at any processing step of the multi-stage system. Buffers in a multi-stage system contain partially processed orders. In conclusion, we state that partially processed orders predominantly occur in systems with high utilisation. Hence, the focus for investigating the impact of the modelling of partially processed orders is on order fulfilment systems with high utilisation in the following.

In order fulfilment systems with high utilisation, the processing performance per time period of the system is the limiting factor of system throughput (see equation (8.2)). Regarding the simplified model, this means that system throughput depends on the aggregated processing performance per time period, and thus on the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ (see equation (7.1)). In the exact model, the relationship is more complicated since there is no parameter describing the processing performance per time period of the whole system. Instead, system throughput of systems with high utilisation depends on

- the processing performance per time period H_p of processes $p \in \mathcal{P}$,
- the number of partially processed orders that are buffered at the processes $p \in \mathcal{P}$, and
- the relation between the current values of the aforementioned parameters.

To illustrate these dependencies, we consider the processing step p_{max} of a multi-stage system in any time period in detail, without loss of generality: If the processing performance per time period $h_{p_{max}}$ of process p_{max} is at most as high as the processing performance per time period $h_{(p_{max}-1)}$ of process $(p_{max}-1)$, $(h_{p_{max}} \leq h_{p_{max}-1})$, the number of completed orders equals the processing performance per time period $h_{p_{max}}$ of process p_{max} independent of the number of partially processed orders at process p_{max}. Otherwise, if the processing performance per time period $h_{p_{max}}$ of process p_{max} exceeds the processing performance per time period $h_{(p_{max}-1)}$ of process $(p_{max}-1)$, $(h_{p_{max}} > h_{p_{max}-1})$, the number of completed orders equals the processing performance per time period $h_{(p_{max}-1)}$ of process $(p_{max}-1)$ if no partially processed orders are buffered at process p_{max}. However, it depends on the processing performance per time period $h_{p_{max}}$ of process p_{max} if there is a sufficiently high number of partially processed orders buffered at process p_{max}. The underlying precondition is that temporary buffers of partially processed orders can occur at any processing step of the system. Referring to common bottleneck definitions in literature (e.g. Lawrence and Buss (1994), Lai et al. (2021)), we call them systems with a shifting bottleneck.

A system with a finite set of processes $\mathcal{P}$ and a processing performance per time period $\mathbf{H}$ is said to be a *system with a shifting bottleneck* if the processing performance per time period $\mathbf{H}$ meets the following conditions:

$$\exists\, \mathbf{h} \in \mathcal{H} : h_i > h_j \qquad i,j \in \mathcal{P}, i \neq j \tag{8.3}$$

$$\exists\, \mathbf{h} \in \mathcal{H} : h_i \leq h_j \qquad i,j \in \mathcal{P}, i \neq j. \tag{8.4}$$

In contrast, it is said to be a *system with a static bottleneck* if there is a process $p^* \in \mathcal{P}$, called static bottleneck, for which the following condition holds:

$$\forall\, \mathbf{h} \in \mathcal{H} : h_{p^*} < h_j \qquad j \in \mathcal{P}, j \neq p^*. \tag{8.5}$$

In the special case of systems with a static bottleneck, buffers of partially processed orders only occur at the bottleneck and at processes upstream the bottleneck. In this case, the number of completed orders per time period does not depend on the three factors mentioned above, but it only depends on the processing performance per time period of the bottleneck. However, the processing performance per time

period of the bottleneck equals the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ (see equation (8.5)), and thus we observe the same relation as in the simplified model. In conclusion, we state the following: In systems with high utilisation and a static bottleneck, system throughput only depends on the processing performance per time period of the bottleneck, and modelling of partially processed orders can be neglected. Otherwise, in systems with high utilisation and a shifting bottleneck, system throughput additionally depends on the temporary number and location of partially processed orders. These findings lead to the following hypothesis:

Hypothesis 1. *For performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck, there is a loss of modelling accuracy when using the simplified model instead of the exact model.*

As mentioned before, in the simplified model, system throughput of systems with high utilisation is determined by the aggregated processing performance per time period, which corresponds to the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ (see equation (7.1)). In contrast, in the exact model, system throughput of systems with high utilisation and a shifting bottleneck is determined by the processing performance per time period of process p_{max}, when assuming a sufficiently high number of partially processed orders buffered at process p_{max}. There are realisations $\mathbf{h} \in \mathcal{H}$ for which the processing performance per time period of process p_{max} is the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ (see equation (8.4)), but there are also realisations $\mathbf{h} \in \mathcal{H}$ for which this condition does not hold (see equation (8.3)). These findings lead to the following hypothesis:

Hypothesis 2. *In systems with high utilisation and a shifting bottleneck, the simplified model underestimates the exact value of the expected value of the system throughput.*

8.1.2 Mathematical Proof

In the following, we prove hypotheses 1 and 2. For this, some preliminary remarks on the characteristics of order processing at any process $p \in \mathcal{P}$ (see Section 8.1.2.1) and the lower bound of the total number of unprocessed orders at process $p \in \mathcal{P}$ immediately before the start of order processing at process p (see Section 8.1.2.2) are required. Subsequently, we derive a formula of system throughput for each system configuration – system with a static and a shifting bottleneck – in each modelling approach – exact and simplified model – under the precondition of high system utilisation (see equation (8.2)) (see Sections 8.1.2.3-8.1.2.5). By comparing these formulas in Section 8.1.2.6, we finally prove that hypotheses 1 and 2 hold.

8.1.2.1 Preliminaries on Order Processing at Process $p \in \mathcal{P}$

Order processing at process p in time period t depends on the realisation $H_p = h_p$ of the processing performance per time period of process p in time period t. Following the principles of levelled order release, order processing at process p starts with orders having a due date of $(-R)$ time periods. Thus, in time period t, a processing performance of h_p orders is available for processing orders with a due date of $(-R)$ time periods. In contrast, for processing orders with a due date of $(k > -R)$ time periods, only the so-called residual processing performance h_p^{res} is available that corresponds to the processing performance remaining after all orders with a due date of $(l < k)$ time periods have already been processed.

The *residual processing performance* h_p^{res} *of process* p *in time period* t can be written as a function of due date $k \in \mathcal{K}$

$$h_p^{res}(k) = \max\left\{0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)}\right\}, \tag{8.6}$$

whereby $h_p^{res}(k)$ specifies the processing performance remaining for processing orders with a due date of k time periods at process p in time period t. Due to the principles of levelled order release, $h_p^{res}(k)$ is monotonously decreasing in k:

$$\begin{aligned} h_p^{res}(-R) &= \max\{0; h_p\} = h_p \\ h_p^{res}(-R+1) &= \max\left\{0; h_p - y_{p,-R}^{(p-1)}\right\} \\ &\vdots \\ h_p^{res}(e_{max}) &= \max\left\{0; h_p - \sum_{l=-R}^{e_{max}-1} y_{p,l}^{(p-1)}\right\} \end{aligned}$$

We define the *critical due date k_p^* of process p in time period t* as the shortest due date whose orders are not processed in time period t:

$$\begin{aligned} k_p^* &= \begin{cases} \min\left\{k \in \mathcal{K} \mid h_p^{res}(k) = 0\right\} & \exists k \in \mathcal{K} : h_p^{res}(k) = 0 \\ \infty & \text{otherwise} \end{cases} \\ k_p^* &= \begin{cases} \min\left\{k \in \mathcal{K} \mid \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)} \geq h_p\right\} & \exists k \in \mathcal{K} : h_p^{res}(k) = 0 \\ \infty & \text{otherwise.} \end{cases} \end{aligned} \tag{8.7}$$

The critical due date k_p^* of process p in time period t can be interpreted as follows: No order with a due date of $(k \geq k_p^*)$ time periods is processed at process p in time period t. In contrast, unprocessed orders with a due date of $(k_p^* - 1)$ time periods are either partially or completely processed at process p in time period t, and all unprocessed orders having a due date of $(k \leq k_p^* - 2)$ time periods are completely processed at process p in time period t.

The number of processed orders per time period $m_{p,k}$ at process p with a due date of k time periods at their time of processing in time period t depends on number of unprocessed orders at process p with a due date of k time periods

immediately before the start of order processing at process p and the residual processing performance (see equation (6.31)):

$$\begin{aligned} m_{p,k} &= \min\left\{ y_{p,k}^{(p-1)}; \max\left\{ 0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)} \right\} \right\} \\ m_{p,k} &\overset{(8.6)}{=} \min\left\{ y_{p,k}^{(p-1)}; h_p^{res}(k) \right\} \\ m_{p,k} &\overset{(8.7)}{=} \begin{cases} y_{p,k}^{(p-1)} & k \in \mathcal{K}, k \leq k_p^* - 2 \\ h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)} & k \in \mathcal{K}, k = k_p^* - 1 \\ 0 & k \in \mathcal{K}, k \geq k_p^*. \end{cases} \end{aligned} \tag{8.8}$$

Consequently, the total number of processed orders per time period f_p at process p in time period t (see equation (6.33)) can be written as follows

$$\begin{aligned} f_p &= \sum_{k \in \mathcal{K}} m_{p,k} \\ f_p &\overset{(8.8)}{=} \sum_{k=-R}^{k_p^*-2} y_{p,k}^{(p-1)} + \left(h_p - \sum_{l=-R}^{k_p^*-2} y_{p,l}^{(p-1)} \right) + \sum_{k=k_p^*}^{e_{max}} 0 \\ f_p &\overset{(8.7)}{=} \begin{cases} \sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)} & k_p^* = \infty \\ h_p & k_p^* \in \mathcal{K}. \end{cases} \end{aligned} \tag{8.9}$$

8.1.2.2 Preliminaries on Lower Bound of Total Number of Unprocessed Orders at Process $p \in \mathcal{P}$ Immediately Before the Start of Order Processing

Using the notation of the state transition of the Markov chain (see Section 6.2.2), $\mathbf{y}_p^{(p-1)}$, $p \in \mathcal{P}$, denotes the number of unprocessed orders at process $p \in \mathcal{P}$ immediately before the start of order processing at process p in any time period t.

The total number of unprocessed orders $\left(\sum_{k \in \mathcal{K}} y_{1,k}^{(0)} \right)$ at process $p = 1$ immediately before the start of order processing at this process corresponds to the total number of unprocessed orders at process $p = 1$ at the beginning of time period t. The last sub-step of every state transition specifies the order income per time

period $\mathbf{G} = \mathbf{g}$ of the following time period (see Section 6.2.2). Thus, the total number of unprocessed orders at process $p = 1$ immediately before the start of order processing at this process is at least as high as the total number of incoming orders at the beginning of time period t:

$$\sum_{k \in \mathcal{K}} y_{1,k}^{(0)} \geq \sum_{k \in \mathcal{K}} g_k. \tag{8.10}$$

The total number of unprocessed orders $\left(\sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)}\right)$ at process $p \in \mathcal{P} \setminus \{1\}$ immediately before the start of order processing at this process in time period t is the sum of the number of unprocessed orders at process p at the end of the previous time period $(t-1)$ and the number of processed orders at the previous processing step $(p-1)$ in time period t (see equation (6.10))

$$\begin{aligned}
\sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)} &= \sum_{k \in \mathcal{K}} y_{p,k}^{(p-2)} \\
&\quad + \sum_{k \in \mathcal{K}} \min \left\{ y_{(p-1),k}^{(p-2)}; \max \left\{ 0; h_{p-1} - \sum_{l=-R}^{k-1} y_{(p-1),l}^{(p-2)} \right\} \right\} \qquad (8.11) \\
\sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)} &\overset{(8.9)}{=} \begin{cases} \sum_{k \in \mathcal{K}} y_{p,k}^{(p-2)} + \sum_{k \in \mathcal{K}} y_{(p-1),k}^{(p-2)} & k_{p-1}^* = \infty \\ \sum_{k \in \mathcal{K}} y_{p,k}^{(p-2)} + h_{p-1} & k_{p-1}^* \in \mathcal{K}, \end{cases}
\end{aligned}$$

whereby $\mathbf{H} = \mathbf{h}$ specifies the processing performance per time period in time period t. Based on equations (8.10) and (8.11), we can derive a lower bound for the total number of unprocessed orders at any process $p \in \mathcal{P} \setminus \{1\}$ immediately before the start of order processing at this process in time period t. For process $p = 2$, the lower bound is derived as follows

$$\begin{aligned}
\sum_{k \in \mathcal{K}} y_{2,k}^{(1)} &= \begin{cases} \sum_{k \in \mathcal{K}} y_{2,k}^{(0)} + \sum_{k \in \mathcal{K}} y_{1,k}^{(0)} & k_1^* = \infty \\ \sum_{k \in \mathcal{K}} y_{2,k}^{(0)} + h_1 & k_1^* \in \mathcal{K} \end{cases} \\
&\overset{(8.10)}{\Rightarrow} \sum_{k \in \mathcal{K}} y_{2,k}^{(1)} \geq \begin{cases} \sum_{k \in \mathcal{K}} g_k & k_1^* = \infty \\ h_1 & k_1^* \in \mathcal{K}. \end{cases}
\end{aligned} \tag{8.12}$$

For process $p = 3$, the lower bound is derived as follows

$$\begin{aligned} \sum_{k\in\mathcal{K}} y_{3,k}^{(2)} &= \begin{cases} \sum_{k\in\mathcal{K}} y_{3,k}^{(1)} + \sum_{k\in\mathcal{K}} y_{2,k}^{(1)} & k_2^* = \infty \\ \sum_{k\in\mathcal{K}} y_{3,k}^{(1)} + h_2 & k_2^* \in \mathcal{K} \end{cases} \\ \overset{(8.12)}{\Rightarrow} \sum_{k\in\mathcal{K}} y_{3,k}^{(2)} &\geq \begin{cases} \sum_{k\in\mathcal{K}} g_k & k_1^* = \infty, k_2^* = \infty \\ h_1 & k_1^* \in \mathcal{K}, k_2^* = \infty \\ h_2 & k_2^* \in \mathcal{K}. \end{cases} \end{aligned} \tag{8.13}$$

Consequently, the lower bound of any process $p \in \mathcal{P} \setminus \{1\}$ is defined as follows

$$\sum_{k\in\mathcal{K}} y_{p,k}^{(p-1)} \geq \begin{cases} \sum_{k\in\mathcal{K}} g_k & k_1^*, \ldots, k_{p-1}^* = \infty \\ h_j & k_j^* \in \mathcal{K}, k_{j+1}^*, \ldots, k_{p-1}^* = \infty, \\ & j \in \{1, \ldots, (p-1)\}. \end{cases} \tag{8.14}$$

8.1.2.3 System Throughput in Systems With a Static Bottleneck in the Exact Model

Under the precondition of high utilisation (see equation (8.2)), in systems with a static bottleneck p^* (see equation (8.5)), the lower bound of the total number of unprocessed orders at the bottleneck p^* immediately before the start of order processing at process p^* (see equation (8.14)) simplifies to

$$\sum_{k\in\mathcal{K}} y_{p^*,k}^{(p^*-1)} > h_{p^*} \qquad \forall \mathbf{h} \in \mathcal{H}. \tag{8.15}$$

Based on this, we conclude that the critical due date $k_{p^*}^*$ of the bottleneck p^* satisfies $k_{p^*}^* \in \mathcal{K}$ and that the throughput f_{p^*} of the bottleneck p^* is given by its processing performance per time period h_{p^*} for all realisations $\mathbf{h} \in \mathcal{H}$:

$$\begin{aligned} \sum_{k\in\mathcal{K}} y_{p^*,k}^{(p^*-1)} > h_{p^*} \quad \forall \mathbf{h} \in \mathcal{H} &\overset{(8.7)}{\Rightarrow} k_{p^*}^* \in \mathcal{K} \qquad \forall \mathbf{h} \in \mathcal{H} \\ &\overset{(8.9)}{\Rightarrow} f_{p^*} = h_{p^*} \quad \forall \mathbf{h} \in \mathcal{H}. \end{aligned} \tag{8.16}$$

If the last processing step p_{max} is the bottleneck of the system, system throughput directly results from equation (8.16) as follows

$$f \overset{(6.35)}{=} f_{p_{max}} = f_{p^*} \overset{(8.16)}{=} h_{p^*} \qquad \forall \mathbf{h} \in \mathcal{H}. \tag{8.17}$$

Otherwise, if any other processing step $p \in \mathcal{P} \setminus \{p_{max}\}$ represents the bottleneck p^* of the order fulfilment system, we additionally consider order processing at the subsequent processing step $(p^* + 1)$ of the bottleneck p^*. Since the throughput of the bottleneck p^* corresponds to its processing performance per time period h_{p^*} for all realisations $\mathbf{h} \in \mathcal{H}$ (see equation (8.16)), and the processing performance per time period $h_{(p^*+1)}$ of process $(p^* + 1)$ is higher than the one of the bottleneck p^* for all realisations $\mathbf{h} \in \mathcal{H}$ (see equation (8.5)), there never remain any unprocessed order at process $(p^* + 1)$ at the end of any time period. The total number of unprocessed orders at process $(p^* + 1)$ immediately before the start of order processing at process $(p^* + 1)$ (see equation (8.11)) simplifies to

$$\sum_{k \in \mathcal{K}} y^{(p^*)}_{(p^*+1),k} = h_{p^*} \qquad \forall \mathbf{h} \in \mathcal{H}. \tag{8.18}$$

Based on this, we conclude that the throughput $f_{(p^*+1)}$ of process $(p^* + 1)$ is given by the processing performance per time period h_{p^*} of the bottleneck p^* for all realisations $\mathbf{h} \in \mathcal{H}$:

$$\begin{aligned}
\sum_{k \in \mathcal{K}} y^{(p^*)}_{(p^*+1),k} = h_{p^*} \quad \forall \mathbf{h} \in \mathcal{H} &\overset{(8.5)}{\Rightarrow} \sum_{k \in \mathcal{K}} y^{(p^*)}_{(p^*+1),k} < h_{(p^*+1)} \quad \forall \mathbf{h} \in \mathcal{H} \\
&\overset{(8.7)}{\Rightarrow} k^*_{(p^*+1)} = \infty \qquad \forall \mathbf{h} \in \mathcal{H} \\
&\overset{(8.9)}{\Rightarrow} f_{(p^*+1)} = \sum_{k \in \mathcal{K}} y^{(p^*)}_{(p^*+1),k} \quad \forall \mathbf{h} \in \mathcal{H} \\
&\overset{(8.18)}{\Rightarrow} f_{(p^*+1)} = h_{p^*} \qquad \forall \mathbf{h} \in \mathcal{H}.
\end{aligned} \tag{8.19}$$

The same reasoning holds for all subsequent processes $p \in \{(p^* + 2), \ldots, p_{max}\}$, and thus system throughput is given by the processing performance per time period h_{p^*} of the bottleneck p^*

$$f = h_{p^*} \qquad \forall \mathbf{h} \in \mathcal{H}. \tag{8.20}$$

In conclusion, we showed that under the precondition of high utilisation (see equation (8.2)), system throughput f of order fulfilment systems with a static bottleneck p^* in any time period is determined by the processing performance per

time period h_{p^*} of the bottleneck p^* for all realisations $\mathbf{h} \in \mathcal{H}$, independent of the position of the bottleneck $p^* \in \mathcal{P}$

$$f = h_{p^*} \overset{(8.5)}{=} \min_{p \in \mathcal{P}} \{h_p\} \qquad \forall \mathbf{h} \in \mathcal{H}. \tag{8.21}$$

8.1.2.4 System Throughput in Systems With a Shifting Bottleneck in the Exact Model

For the analysis of system throughput in systems with a shifting bottleneck, we focus on the last processing step p_{max} of the multi-stage system in any time period t, whereby $\mathbf{H} = \mathbf{h}$ specifies the processing performance per time period in time period t. The following considerations assume high utilisation (see equation (8.2)).

If the last processing step is the bottleneck of the system in time period t (see equation (8.4)), following the reasoning of equation (8.16), system throughput f in time period t corresponds to the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ (see equation (8.17)).

Otherwise, if the last processing step is not the bottleneck of the system in time period t (see equation (8.3)), the throughput $f_{p_{max}}$ of process p_{max} in time period t either corresponds to the processing performance per time period of process p_{max} in time period t or to the total number of unprocessed orders at process p_{max} immediately before the start of order processing at process p_{max} in time period t, depending on the value of the critical due date $k^*_{p_{max}}$ of process p_{max} in time period t (see equation (8.9)):

- Critical due date of $k^*_{p_{max}} \in \mathcal{K}$

$$f_{p_{max}} = h_{p_{max}} \overset{(8.3)}{\Rightarrow} f_{p_{max}} > \min_{p \in \mathcal{P}} \{h_p\}. \tag{8.22}$$

- Critical due date of $k^*_{p_{max}} = \infty$

$$
\begin{aligned}
& f_{p_{max}} = \sum_{k \in \mathcal{K}} y_{p_{max},k}^{(p_{max}-1)} \\
\overset{(8.14)}{\Rightarrow} \; & f_{p_{max}} \geq \begin{cases} \sum_{k \in \mathcal{K}} g_k & k^*_1, \ldots, k^*_{p_{max}-1} = \infty \\ h_j & k^*_j \in \mathcal{K}, k^*_{j+1}, \ldots, k^*_{p_{max}-1} = \infty, \\ & j \in \{1, \ldots, (p_{max}-1)\} \end{cases} \\
\overset{(8.3),(8.2)}{\Rightarrow} \; & f_{p_{max}} \geq \min_{p \in \mathcal{P}} \{h_p\}.
\end{aligned}
\tag{8.23}
$$

In conclusion, we showed that under the precondition of high utilisation (see equation (8.2)), system throughput f of order fulfilment systems with a shifting bottleneck in any time period depends on the realisation $\mathbf{h} \in \mathcal{H}$ as follows

$$f = \min_{p \in \mathcal{P}} \{h_p\} \qquad \forall \mathbf{h} \in \mathcal{H} : h_{p_{max}} = \min_{p \in \mathcal{P}} \{h_p\} \tag{8.24}$$

$$f > \min_{p \in \mathcal{P}} \{h_p\} \qquad \forall \mathbf{h} \in \mathcal{H} : h_{p_{max}} > \min_{p \in \mathcal{P}} \{h_p\}, k^*_{p_{max}} \in \mathcal{K} \tag{8.25}$$

$$f \geq \min_{p \in \mathcal{P}} \{h_p\} \qquad \forall \mathbf{h} \in \mathcal{H} : h_{p_{max}} > \min_{p \in \mathcal{P}} \{h_p\}, k^*_{p_{max}} = \infty. \tag{8.26}$$

8.1.2.5 System Throughput in the Simplified Model

In the simplified model, system throughput in time period t either corresponds to the aggregated processing performance per time period $\tilde{h}$ in time period t or to the total number of unprocessed orders at the beginning of time period t, depending on the value of the critical due date k^* in time period t (simplification of equation (8.9)):

$$f = \begin{cases} \tilde{h} & k^* \in \mathcal{K} \\ \sum_{k \in \mathcal{K}} y_k^{(0)} & k^* = \infty. \end{cases} \tag{8.27}$$

Under the precondition of high utilisation (see equation (8.2)), we conclude from the lower bound of the total number of unprocessed orders (see equation (8.10))

that system throughput f in any time period is determined by the aggregated processing performance per time period $\tilde{h}$ for all realisations $\tilde{h} \in \tilde{\mathcal{H}}$:

$$\begin{aligned} \sum_{k \in \mathcal{K}} y_k^{(0)} \geq \sum_{k \in \mathcal{K}} g_k &\overset{(8.2)}{\Rightarrow} \sum_{k \in \mathcal{K}} y_k^{(0)} > \tilde{h} && \forall \tilde{h} \in \tilde{\mathcal{H}} \\ &\overset{(8.7)}{\Rightarrow} k^* \in \mathcal{K} && \forall \tilde{h} \in \tilde{\mathcal{H}} \\ &\overset{(8.27)}{\Rightarrow} f = \tilde{h} && \forall \tilde{h} \in \tilde{\mathcal{H}} \\ &\overset{(7.1)}{\Rightarrow} f = \min_{p \in \mathcal{P}} \{h_p\} && \forall \mathbf{h} \in \mathcal{H}. \end{aligned} \tag{8.28}$$

8.1.2.6 Comparison of System Throughput in the Exact and the Simplified Model

Table 8.1 summarises the values of system throughput in time period t for each system configuration – system with a static or a shifting bottleneck – in each modelling approach – exact or simplified model – under the precondition of high system utilisation (see equation (8.2)) as derived in the previous sections.

Table 8.1: Comparison of system throughput f in any time period t in the exact and the simplified model (under the precondition of high system utilisation).

	Exact model	Simplified model
System with a shifting bottleneck	$f = \min_{p \in \mathcal{P}} \{h_p\}$ $\forall \mathbf{h} \in \mathcal{H} : h_{p_{max}} = \min_{p \in \mathcal{P}} \{h_p\}$ $f > \min_{p \in \mathcal{P}} \{h_p\}$ $\forall \mathbf{h} \in \mathcal{H} : h_{p_{max}} > \min_{p \in \mathcal{P}} \{h_p\}, k^*_{p_{max}} \in \mathcal{K}$ $f \geq \min_{p \in \mathcal{P}} \{h_p\}$ $\forall \mathbf{h} \in \mathcal{H} : h_{p_{max}} > \min_{p \in \mathcal{P}} \{h_p\}, k^*_{p_{max}} = \infty$	$f = \min_{p \in \mathcal{P}} \{h_p\}$ $\forall \mathbf{h} \in \mathcal{H}$
System with a static bottleneck	$f = \min_{p \in \mathcal{P}} \{h_p\}$ $\forall \mathbf{h} \in \mathcal{H}$	$f = \min_{p \in \mathcal{P}} \{h_p\}$ $\forall \mathbf{h} \in \mathcal{H}$

We proved that system throughput of order fulfilment systems with a static bottleneck in any time period t equals the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ under the precondition of high utilisation. It is calculated exactly based on the simplified model. There is no loss of modelling accuracy for performance analysis of order fulfilment systems with a static bottleneck when using the simplified model, which does not model partially processed orders, instead of the exact model. Consequently, the modelling of partially processed orders has no impact on the modelling accuracy of order fulfilment systems with a static bottleneck.

Furthermore, we proved that system throughput of order fulfilment systems with a shifting bottleneck in any time period t is at least as high as the minimum of the processing performances per time period of processes $p \in \mathcal{P}$ under the precondition of high utilisation. For systems with high utilisation and a shifting bottleneck, the simplified model only provides a lower bound of the exact system throughput since it approximates system throughput in any time period t by the minimum of the processing performances per time period of processes $p \in \mathcal{P}$. There are modelling inaccuracies for performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck when using the simplified model instead of the exact one. Consequently, the modelling of partially processed orders affects modelling accuracy of order fulfilment systems with high utilisation and a shifting bottleneck. The expected value of the system throughput calculated based on the simplified model is smaller than the exact expected value of the system throughput calculated based on the exact model.

By these findings, hypotheses 1 and 2 are mathematically proven.

8.2 Accuracy of Performance Analysis

For performance analysis of an order fulfilment system, several performance measures (see Table 6.1) are derived from the limiting distribution of the Markov chain in both the exact and the simplified model. Due to the proven modelling

inaccuracies of the simplified model (see Section 8.1), we expect inaccuracies of performance analysis when using the simplified model. In the following, we initially derive two hypotheses on the impact of the modelling inaccuracies of the simplified model on the accuracy of performance analysis (see Section 8.2.1). Subsequently, we verify these hypotheses in a numerical analysis (see Section 8.2.2).

8.2.1 Hypotheses

In Section 8.1, we showed that the simplified model suffers from modelling inaccuracies for performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck indicated by deviations of system throughput in any time period t. Since the customer-related performance measures of the order fulfilment system are derived from the system throughput and the characteristics of completed orders, the results of Section 8.1 lead to the following hypothesis:

Hypothesis 3. *For performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck, the values of customer-related performance measures calculated based on the simplified model deviate from the exact ones calculated based on the exact model.*

Furthermore, we proved in Section 8.1 that in systems with high utilisation and a shifting bottleneck, the expected value of the system throughput calculated based on the simplified model is smaller than the exact one calculated based on the exact model. Thus, fewer orders are completed on time, and the proportion of on-time completed orders on the total number of orders in the simplified model is smaller than the one in the exact model. Based on the definition of the service level (see Section 6.3.7), these findings lead to the following hypothesis:

Hypothesis 4. *For performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck, the simplified model underestimates the exact values of α-, β-, and γ-service level.*

8.2.2 Numerical Results

To verify hypotheses 3 and 4, we conduct a numerical analysis based on samples C and E. Sample C contains 256 two-stage order fulfilment systems, and sample E consists of 353 three-stage order fulfilment systems. The samples differ regarding the ranges of the parameters, especially regarding the discretisation of the number of incoming orders per time period A: Its range is given by $\mathcal{A}^C = \{3, 4, \ldots, 14\}$ in sample C and $\mathcal{A}^E = \{2, 3, \ldots, 7\}$ in sample E. A comprehensive description of the samples is given in Appendix C.2. Sample C consists of 167 systems with a shifting bottleneck (65% of sample size) and 89 systems with a static bottleneck. Sample E consists of 334 systems with a shifting bottleneck (95% of sample size) and 19 systems with a static bottleneck.

The numerical analysis focuses on α- and β-service level for simplicity reasons. As evaluation criteria, we calculate the absolute and the relative deviation of the values of α- and β-service level calculated based on the simplified model from the ones calculated based on the exact model, respectively.

8.2.2.1 Order Fulfilment Systems With a Static Bottleneck

Tables 8.2 and 8.3 present the absolute and the relative deviations of α- and β-service level between the simplified and the exact model for order fulfilment systems with a static bottleneck in samples C and E, respectively. Each sample is subdivided into multiple classes according to the utilisation of the order fulfilment system (see equation (6.7)). The tables give the minimum, the maximum, and the average values of the absolute and relative deviations for each class of utilisation as well as for the whole sample.

The absolute deviation of α-service level is between -5.17E-05 and 8.53E-05 in sample C and between -1.92E-05 and 2.56E-05 in sample E. We observe absolute deviations of β-service level between -3.14E-05 and 4.88E-05 in sample C and between -9.66E-06 and 2.05E-05 in sample E. Thus, the absolute values of the absolute deviations of α- and β-service level are smaller than 8.6E-05 in both

Table 8.2: Deviation of α-service level between the simplified and the exact model for order fulfilment systems with a static bottleneck.

		Absolute deviation[1]			Relative deviation [%][2]			
	Utilisation	Min	Max	Average	Min	Max	Average	No.
Sample C	[0.3,0.4)	-1.13E-06	-5.90E-11	-5.65E-07	-0.0001	0.0000	-0.0001	2
	[0.4,0.5)	0.00E+00	2.22E-16	1.11E-16	0.0000	0.0000	0.0000	3
	[0.5,0.6)	-5.55E-16	3.83E-05	4.74E-06	0.0000	0.0038	0.0005	16
	[0.6,0.7)	-1.99E-14	2.26E-05	2.21E-06	0.0000	0.0023	0.0002	14
	[0.7,0.8)	-3.19E-05	4.87E-05	1.03E-06	-0.0032	0.0049	0.0001	17
	[0.8,0.9)	-3.80E-05	8.53E-05	7.04E-06	-0.0038	0.0087	0.0007	23
	[0.9,1.0)	-5.17E-05	4.59E-05	3.35E-06	-0.0052	0.0046	0.0002	14
	[0.3,1.0)	-5.17E-05	8.53E-05	3.73E-06	-0.0052	0.0087	0.0004	89
Sample E	[0.5,0.6)	-2.22E-16	-1.11E-16	-1.67E-16	0.0000	0.0000	0.0000	2
	[0.6,0.7)	-7.77E-16	2.56E-05	6.40E-06	0.0000	0.0026	0.0006	4
	[0.7,0.8)	-7.29E-09	-3.33E-16	-3.64E-09	0.0000	0.0000	0.0000	2
	[0.8,0.9)	-3.22E-08	0.00E+00	-1.07E-08	0.0000	0.0000	0.0000	3
	[0.9,1.0)	-1.92E-05	1.58E-05	3.80E-07	-0.0025	0.0019	0.0001	8
	[0.5,1.0)	-1.92E-05	2.56E-05	1.50E-06	-0.0025	0.0026	0.0002	19

[1] Difference between the value of α-service level calculated based on the simplified model and the one calculated based on the exact model.
[2] Difference between the value of α-service level calculated based on the simplified model and the one calculated based on the exact model, divided by the value calculated based on the exact model.

samples. We state these deviations to be negligible. They result from numerical errors during the calculations. By comparing the magnitude of the absolute and relative deviations of α- and β-service level of different classes of utilisation in each sample, we note that the utilisation of the order fulfilment system does not have any systematic impact on the absolute and relative deviations of neither α- nor β-service level.

In conclusion, these results indicate that the chosen model for performance analysis – exact or simplified model – has a negligible impact on the values of α- and β-service level in order fulfilment systems with a static bottleneck.

Table 8.3: Deviation of β-service level between the simplified and the exact model for order fulfilment systems with a static bottleneck.

		Absolute deviation[1]			Relative deviation [%][2]			
	Utilisation	Min	Max	Average	Min	Max	Average	No.
Sample C	[0.3,0.4)	-2.27E-07	-5.68E-11	-1.14E-07	0.0000	0.0000	0.0000	2
	[0.4,0.5)	-1.11E-16	3.33E-16	1.48E-16	0.0000	0.0000	0.0000	3
	[0.5,0.6)	-3.33E-16	2.74E-05	3.23E-06	0.0000	0.0027	0.0003	16
	[0.6,0.7)	-6.73E-07	1.66E-05	1.54E-06	-0.0001	0.0017	0.0002	14
	[0.7,0.8)	-2.80E-05	3.93E-05	9.29E-07	-0.0028	0.0039	0.0001	17
	[0.8,0.9)	-3.14E-05	4.88E-05	3.80E-06	-0.0031	0.0049	0.0004	23
	[0.9,1.0)	-2.84E-05	4.29E-05	2.64E-06	-0.0028	0.0049	0.0003	14
	[0.3,1.0)	-3.14E-05	4.88E-05	2.39E-06	-0.0031	0.0049	0.0002	89
Sample E	[0.5,0.6)	-5.55E-16	-3.33E-16	-4.44E-16	0.0000	0.0000	0.0000	2
	[0.6,0.7)	-1.11E-15	2.05E-05	5.12E-06	0.0000	0.0020	0.0005	4
	[0.7,0.8)	-4.02E-09	-3.33E-16	-2.01E-09	0.0000	0.0000	0.0000	2
	[0.8,0.9)	-1.36E-08	-1.11E-16	-4.52E-09	0.0000	0.0000	0.0000	3
	[0.9,1.0)	-9.66E-06	7.89E-06	1.22E-06	-0.0011	0.0010	0.0001	8
	[0.5,1.0)	-9.66E-06	2.05E-05	1.59E-06	-0.0011	0.0020	0.0002	19

[1] Difference between the value of β-service level calculated based on the simplified model and the one calculated based on the exact model.
[2] Difference between the value of β-service level calculated based on the simplified model and the one calculated based on the exact model, divided by the value calculated based on the exact model.

8.2.2.2 Order Fulfilment Systems With a Shifting Bottleneck

Tables 8.4 and 8.5 present the absolute and the relative deviations of α- and β-service level between the simplified and the exact model for order fulfilment systems with a shifting bottleneck in samples C and E, respectively. Each sample is subdivided into multiple classes according to the utilisation of the order fulfilment system (see equation (6.7)). The tables give the minimum, the maximum, and the average values of the absolute and relative deviations for each class of utilisation as well as for the whole sample.

Table 8.4: Deviation of α-service level between the simplified and the exact model for order fulfilment systems with a shifting bottleneck.

		Absolute deviation[1]			Relative deviation [%][2]			
	Utilisation	Min	Max	Average	Min	Max	Average	No.
Sample C	[0.2,0.3)	-4.29E-07	3.21E-05	6.30E-06	0.0000	0.0032	0.0006	5
	[0.3,0.4)	-3.37E-06	6.21E-08	-4.61E-07	-0.0003	0.0000	0.0000	15
	[0.4,0.5)	-1.25E-06	4.17E-05	6.00E-06	-0.0001	0.0042	0.0006	24
	[0.5,0.6)	-1.84E-05	4.15E-05	8.47E-06	-0.0018	0.0041	0.0008	23
	[0.6,0.7)	-1.75E-05	2.73E-05	3.39E-06	-0.0018	0.0027	0.0003	31
	[0.7,0.8)	-3.37E-05	5.39E-05	4.91E-06	-0.0034	0.0054	0.0005	22
	[0.8,0.9)	-3.81E-03	7.85E-05	-2.37E-04	-0.4006	0.0079	-0.0245	28
	[0.9,1.0)	-3.12E-02	7.13E-05	-3.57E-03	-3.2642	0.0074	-0.4600	19
	[0.2,1.0)	-3.12E-02	7.85E-05	-4.42E-04	-3.2642	0.0079	-0.0561	167
Sample E	[0.3,0.4)	-1.40E-09	2.29E-06	8.69E-07	0.0000	0.0002	0.0001	3
	[0.4,0.5)	-3.75E-06	5.29E-05	1.01E-05	-0.0004	0.0053	0.0010	14
	[0.5,0.6)	-1.42E-05	6.37E-05	6.33E-06	-0.0014	0.0064	0.0006	40
	[0.6,0.7)	-4.44E-04	6.02E-05	-2.47E-05	-0.0448	0.0060	-0.0025	58
	[0.7,0.8)	-2.87E-03	5.72E-05	-2.41E-04	-0.2935	0.0057	-0.0244	83
	[0.8,0.9)	-1.39E-01	3.67E-05	-9.92E-03	-17.2552	0.0037	-1.1581	89
	[0.9,1.0)	-2.78E-01	2.62E-05	-4.92E-02	-47.5861	0.0026	-6.8910	47
	[0.3,1.0)	-2.78E-01	6.37E-05	-9.63E-03	-47.5861	0.0064	-1.2847	334

[1] Difference between the value of α-service level calculated based on the simplified model and the one calculated based on the exact model.
[2] Difference between the value of α-service level calculated based on the simplified model and the one calculated based on the exact model, divided by the value calculated based on the exact model.

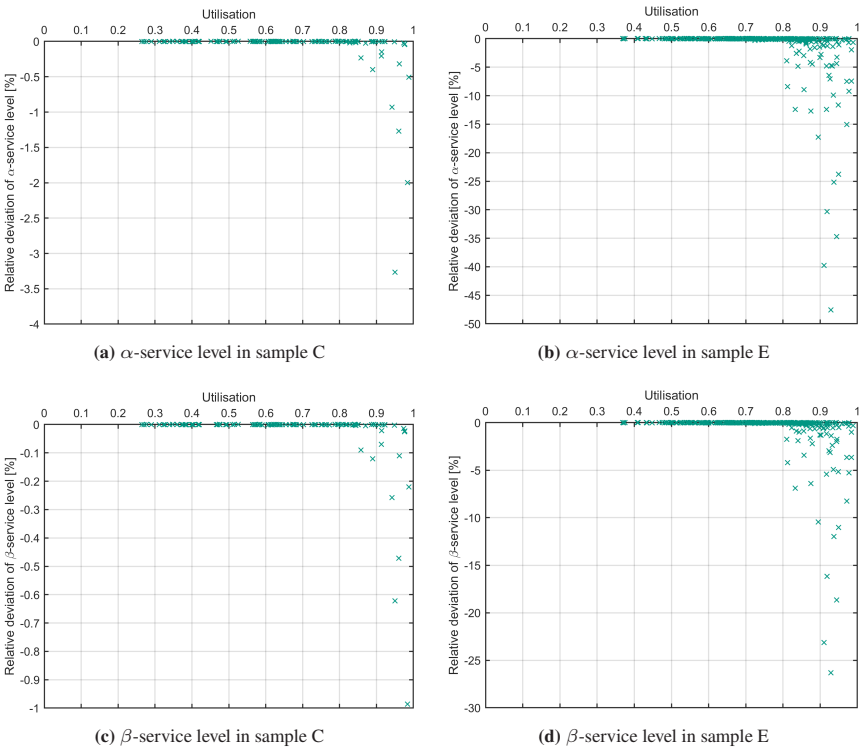
We observe absolute deviations of α-service level between -3.12E-02 and 7.85E-05 in sample C and between -2.78E-01 and 6.37E-05 in sample E. The absolute deviation of β-service level is between -6.69E-03 and 5.99E-05 in sample C and between -2.29E-01 and 5.43E-05 in sample E. These results indicate that for order fulfilment systems with a shifting bottleneck, there are data points in both samples for which the values of α- and β-service level calculated based on the simplified model deviate significantly from the ones calculated based on the exact model.

Table 8.5: Deviation of β-service level between the simplified and the exact model for order fulfilment systems with a shifting bottleneck.

		Absolute deviation[1]			Relative deviation [%][2]			
	Utilisation	Min	Max	Average	Min	Max	Average	No.
Sample C	[0.2,0.3)	-3.23E-08	2.80E-05	5.60E-06	0.0000	0.0028	0.0006	5
	[0.3,0.4)	-2.61E-06	1.25E-08	-3.03E-07	-0.0003	0.0000	0.0000	15
	[0.4,0.5)	-1.11E-06	3.55E-05	4.98E-06	-0.0001	0.0035	0.0005	24
	[0.5,0.6)	-1.19E-05	3.23E-05	6.78E-06	-0.0012	0.0032	0.0007	23
	[0.6,0.7)	-1.35E-05	2.03E-05	2.39E-06	-0.0013	0.0020	0.0002	31
	[0.7,0.8)	-1.09E-05	4.28E-05	4.85E-06	-0.0011	0.0043	0.0005	22
	[0.8,0.9)	-1.19E-03	5.64E-05	-7.28E-05	-0.1205	0.0056	-0.0073	28
	[0.9,1.0)	-6.69E-03	5.99E-05	-1.27E-03	-0.9864	0.0060	-0.1483	19
	[0.2,1.0)	-6.69E-03	5.99E-05	-1.54E-04	-0.9864	0.0060	-0.0178	167
Sample E	[0.3,0.4)	-1.03E-09	2.09E-06	7.21E-07	0.0000	0.0002	0.0001	3
	[0.4,0.5)	-3.36E-06	3.89E-05	7.58E-06	-0.0003	0.0039	0.0008	14
	[0.5,0.6)	-1.19E-05	5.43E-05	4.80E-06	-0.0012	0.0054	0.0005	40
	[0.6,0.7)	-1.44E-04	4.77E-05	-5.70E-06	-0.0145	0.0048	-0.0006	58
	[0.7,0.8)	-9.96E-04	4.36E-05	-7.16E-05	-0.1002	0.0044	-0.0072	83
	[0.8,0.9)	-9.49E-02	2.92E-05	-5.11E-03	-10.4485	0.0029	-0.5413	89
	[0.9,1.0)	-2.29E-01	1.68E-05	-3.11E-02	-26.3286	0.0017	-3.5121	47
	[0.3,1.0)	-2.29E-01	5.43E-05	-5.76E-03	-26.3286	0.0054	-0.6402	334

[1] Difference between the value of β-service level calculated based on the simplified model and the one calculated based on the exact model.
[2] Difference between the value of β-service level calculated based on the simplified model and the one calculated based on the exact model, divided by the value calculated based on the exact model.

Figure 8.1 presents the relative deviations of α- and β-service level between the simplified and the exact model depending on the utilisation of the order fulfilment system for order fulfilment systems with a shifting bottleneck in samples C and E. It indicates that the utilisation of the order fulfilment system has a systematic impact on the relative deviations of α- and β-service level since the absolute value of the relative deviations of α- and β-service level increases as the utilisation converges towards one.



(a) α-service level in sample C

(b) α-service level in sample E

(c) β-service level in sample C

(d) β-service level in sample E

Figure 8.1: Deviation of α- and β-service level between the simplified and the exact model for order fulfilment systems with a shifting bottleneck.

In sample C, for systems with a utilisation smaller than 0.8, the relative deviation is between -0.003% and 0.005% for α-service level and between -0.001% and 0.004% for β-service level. In sample E, for systems with a utilisation smaller than 0.6, we observe a relative deviation of α-service level between -0.001% and 0.006% and of β-service level between -0.001% and 0.005%. Thus, for order fulfilment systems with a utilisation smaller than approximately 0.6, the absolute value of the relative deviation of α- and β-service level is at most 0.005% in both samples. We state these deviations to be negligible and conclude that the chosen model for performance analysis – exact or simplified model – has a negligible

impact on the values of α- and β-service level in order fulfilment systems with a utilisation smaller than approximately 0.6 and with a shifting bottleneck.

In contrast, in sample C, the maximum absolute value of the relative deviation is 3.3% for α-service level and 1% for β-service level for systems with a utilisation of [0.9,1.0). In sample E, the maximum absolute value of the relative deviation is 17.3% for α-service level and 10.4% for β-service level when considering systems with a utilisation of [0.8,0.9). Furthermore, for systems with a utilisation of [0.9,1.0), we observe a maximum absolute value of the relative deviation of 47.6% for α-service level and 26.3% for β-service level. These results indicate that the values of α- and β-service level calculated based on the simplified model can strongly deviate from the ones calculated based on the exact model for systems with a utilisation higher than 0.8 and a shifting bottleneck.

By comparing the magnitude of the absolute and relative deviations of α- and β-service level between sample C and sample E, we observe that the number of processes of the considered order fulfilment system has a systematic impact on the deviations of α- and β-service level between the simplified and the exact model. The maximum absolute value of the relative deviation of α-service level is 3.3% for the two-stage systems of sample C, whereas it is 47.6% for the three-stage systems of sample E. The maximum absolute value of the relative deviation of β-service level is 1% for the two-stage systems of sample C and 26.3% for the three-stage systems of sample E. These results indicate that the absolute values of the deviations of α- and β-service level between the simplified and the exact model increase with an increasing number of processes in the system.

8.2.3 Discussion

For order fulfilment systems with a utilisation smaller than approximately 0.6 or systems with a static bottleneck, the chosen model for performance analysis – exact or simplified model – has a negligible impact on the values of α- and β-service level in both samples. For these systems, the observed absolute deviations have a magnitude of at most 6.4E-05. In contrast, for order fulfilment systems

with a utilisation higher than 0.8 and a shifting bottleneck, there are significant deviations of α- and β-service level between the simplified and the exact model. For instance, for systems with a utilisation of [0.9,1.0), we observe an absolute value of the relative deviation of α-service level of at most 3.3% and on average 0.5% in sample C and of at most 47.6% and on average 6.9% in sample E. In conclusion, these results confirm that the values of customer-related performance measures calculated based on the simplified model deviate from the exact ones calculated based on the exact model for performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck (see hypothesis 3).

The deviations of α- and β-service level observed for order fulfilment systems with a utilisation higher than 0.8 and a shifting bottleneck are negative. Thus, the values of α- and β-service level calculated based on the simplified model are smaller than the ones calculated based on the exact model. These results confirm that the simplified model underestimates the exact values of α- and β-service level for performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck (see hypothesis 4).

Furthermore, for order fulfilment systems with a utilisation higher than 0.6 and a shifting bottleneck, the number of processes of the considered order fulfilment system has a positive impact on the absolute magnitude of the deviations of α- and β-service level between the simplified and the exact model. The modelling inaccuracies of the simplified model result from the fact that partially processed orders are not modelled in the simplified model (see Section 8.1). Partially processed orders occur at the buffers of the processes $p \in \mathcal{P} \setminus \{1\}$ of an order fulfilment system. Consequently, the higher the number of processes, the more partially processed orders can occur in the order fulfilment system, and the higher are the modelling inaccuracies and thus the inaccuracies of performance analysis of the simplified model.

In the context of workload balancing in order fulfilment systems, the focus is neither on systems with low utilisation nor on systems with a static bottleneck: On the one hand, in order fulfilment systems with low utilisation, provided capacity is not used efficiently and thus the order fulfilment system is not competitive in

the long run due to the recently intensified competition (Van Gils et al. 2018; Kundu et al. 2020). On the other hand, in order fulfilment systems with a static bottleneck, the most crucial measure for improvement is not workload balancing but increasing the processing performance at the bottleneck. Consequently, we expect order fulfilment systems with high utilisation and a shifting bottleneck to be the main application field for workload balancing. For performance analysis of these systems, one should prefer the exact model as the simplified model is limited to worst-case analyses of system throughput and service level.

8.3 Memory and Computation Time Requirements

In the simplified model, any multi-stage order fulfilment system is modelled as a single-stage system consisting of one aggregated process (see Figure 5.2). Thus, modelling the corresponding Markov chain is less complex than modelling the one in the exact model. The simplified model can be seen as a special case of the exact model since its Markov chain ensues from the Markov chain of the exact model when modelling is restricted to single-stage order fulfilment systems. Consequently, we expect differences between the simplified and the exact model regarding memory usage and computation time. In the following, we initially derive two hypotheses on the impact of the selected model for performance analysis on memory and computation time requirements (see Section 8.3.1). Subsequently, we verify these hypotheses in a numerical analysis (see Section 8.3.2).

8.3.1 Hypotheses

In the simplified model, any multi-stage order fulfilment system is modelled as a single-stage system by combining the processing steps $p \in \mathcal{P}$ to one aggregated process. The process index p is neglected in the definition of system state (see Section 7.2.1). System state is a $(R + e_{max} + 1)$-dimensional vector in

the simplified model, whereas system state in the exact model corresponds to a $(p_{max} \times (R + e_{max} + 1))$-dimensional matrix (see Section 6.2.1). The size of the state space in the exact model (see equation (6.45)) can be expressed as an integer multiple $N \in \mathbb{N}$ of the size of the state space in the simplified model (see equation (7.9)) with

$$N = \prod_{p=2}^{p_{max}} \left[\prod_{k=-R}^{-1} \Big((e_{max} + |k|) \cdot h_{(p-1),max} + 1 \Big) \cdot \prod_{k=0}^{e_{max}} \Big((e_{max} - k + 1) \cdot h_{(p-1),max} + 1 \Big) \right]. \tag{8.29}$$

Thus, the state space of the Markov chain in the exact model is always larger than the one in the simplified model. As the size of the state space drives the memory requirements to store the state space, the transition matrix, and the limiting distribution of the Markov chain as well as the computational effort to calculate the Markov chain (see Section 6.4), we formulate the following hypotheses regarding memory and computation time requirements:

Hypothesis 5. *Performance analysis of order fulfilment systems based on the exact model requires more memory than performance analysis based on the simplified model.*

Hypothesis 6. *Performance analysis of order fulfilment systems based on the exact model requires more computation time than performance analysis based on the simplified model.*

8.3.2 Numerical Results

To verify hypotheses 5 and 6, we conduct a numerical analysis based on samples C and E. Sample C contains 256 two-stage order fulfilment systems, and sample E consists of 353 three-stage order fulfilment systems. The samples differ regarding the ranges of the parameters, especially regarding the discretisation of the number of incoming orders per time period A: Its range is given by $\mathcal{A}^C = \{3, 4, \ldots, 14\}$

in sample C and $\mathcal{A}^E = \{2, 3, \ldots, 7\}$ in sample E. A comprehensive description of the samples is given in Appendix C.2. As evaluation criterion of the computation time, we measure the time required to calculate the Markov chain and the performance measures. Furthermore, we use the number of reachable states of the Markov chain as an indicator of memory requirements.

Table 8.6 presents the number of reachable states and the computation time of the exact and the simplified model as well as the absolute and relative deviations in the number of reachable states and the computation time between the exact and the simplified model for samples C and E. For each of these criteria, Table 8.6 gives the minimum, the maximum, and the average value.

Table 8.6: Number of reachable states and computation time of the exact and the simplified model.[1,2]

		Number of reachable states			Computation time [s]		
		Min	Max	Average	Min	Max	Average
Sample C	Exact model	11,131	215,155	135,072	12.40	20,418.07	4,890.58
	Simplified model	3,610	8,387	7,568	1.23	4.26	2.63
	Absolute deviation[3]	-207,334	-3,309	-127,504	-20,415	-10	-4,888
	Relative deviation [%][4]	-98.08	-29.73	-91.74	-99.99	-80.69	-99.31
Sample E	Exact model	1,807	133,646	74,503	1.77	7,418.54	1,534.94
	Simplified model	931	1,212	1,101	0.09	0.50	0.15
	Absolute deviation[3]	-132,535	-680	-73,402	-7,418	-1.64	-1,534
	Relative deviation [%][4]	-99.17	-37.63	-98.11	-100.00	-92.83	-99.96

[1] We apply the strategies for runtime and memory optimisation introduced in Chapter 10 to calculate the Markov chain of the exact and the simplified model.
[2] Computations are conducted on a server with a CPU of 64 kernels and 128 threads and a RAM of 128 GB.
[3] Difference between the value of the considered criterion in the simplified model and the one in the exact model.
[4] Difference between the value of the considered criterion in the simplified model and the one in the exact model, divided by the value in the exact model.

In sample C, the number of reachable states is on average 135,072 states in the exact model and on average 7,568 states in the simplified model. In sample E, we observe on average 74,503 reachable states in the exact model and on average 1,101 reachable states in the simplified model. Thus, we observe an average

relative deviation in the number of reachable states between the simplified and the exact model of -91.7% in sample C and -98.1% in sample E. These results confirm that performance analysis based on the exact model requires significantly more memory than performance analysis based on the simplified model (see hypothesis 5).

In sample C, the average computation time is 1.4 hours for the exact model and 2.6 seconds for the simplified model. In sample E, we observe an average computation time of 25.6 minutes for the exact model and 0.2 seconds for the simplified model. In both samples, the average relative deviation in computation time between the simplified and the exact model is -99%. These results confirm that performance analysis based on the exact model requires significantly more computation time than performance analysis based on the simplified model (see hypothesis 6).

Note that it is not meaningful to compare the absolute values of the number of reachable states and the computation time between samples C and E since they depend on the discretisation of the stochastic parameters of the order fulfilment systems, which differs between samples C and E (see Appendix C.2).

8.4 Chapter Conclusion

In this chapter, we evaluated and compared the exact and the simplified model for performance analysis regarding modelling accuracy, accuracy of performance analysis, memory usage, and computational effort.

Regarding memory usage and computational effort, the Markov chain of the simplified model requires significantly less memory and significantly shorter computation times than the corresponding Markov chain of the exact model due to its smaller number of reachable states (average reduction in the number of reachable states by more than 90%). Thus, when memory and computation time are scarce, the simplified model should be preferred over the exact model.

Regarding modelling accuracy, we showed that the simplified model suffers from modelling inaccuracies for performance analysis of order fulfilment systems with a shifting bottleneck and high utilisation: We mathematically proved that for this configuration of order fulfilment systems, the expected value of the system throughput calculated based on the simplified model is smaller than the actual expected value of the system throughput calculated based on the exact model. Furthermore, we numerically showed that these modelling inaccuracies of the simplified model lead to inaccuracies in the values of customer-related performance measures. For order fulfilment systems with high utilisation and a shifting bottleneck, the simplified model results in smaller values of α- and β-service level than the exact model. For instance, for systems with a utilisation of [0.9,1.0), the absolute value of the relative deviation of α-service level is at most 3.3% and on average 0.5% in the sample of two-stage systems, and it is at most 47.6% and on average 6.9% in the sample of three-stage systems. Hence, for order fulfilment systems with high utilisation and a shifting bottleneck, it is only reasonable to use the simplified model for worst-case analyses of system throughput and service level. In contrast, for order fulfilment systems with high utilisation and a static bottleneck, we mathematically proved that the expected value of the system throughput calculated based on the simplified model corresponds to the actual expected value of the system throughput calculated based on the exact model. For order fulfilment systems with low utilisation, we numerically showed that system performance calculated based on the simplified model is the same as the one calculated based on the exact model, apart from negligible numerical errors.

Order fulfilment systems with high utilisation and a shifting bottleneck are predominant in practical applications of workload balancing. For performance analysis of these systems, the exact model outperforms the simplified model in terms of modelling accuracy and accuracy of performance analysis, despite its drawbacks regarding memory and computation time requirements. The simplified model is limited to a worst-case performance analysis for these systems.

9 Formalisation and Solution Algorithms of the Capacity Planning Problem

The analytical models developed in Chapters 6 and 7 enable the performance analysis of any order fulfilment system with a given system configuration. However, the focus of operations managers is not on the performance analysis of a given system configuration but on adapting the current system configuration to guarantee promised performance requirements to their customers. In particular, they have to decide on the capacity that is provided at the processing steps of the order fulfilment system to meet the performance requirements of the customers. Hence, this chapter aims at formalising the capacity planning problem in order fulfilment systems and at providing suitable solution algorithms. Section 9.1 introduces the mathematical formulation of the capacity planning problem, and Section 9.2 provides an overview of the research field of derivative-free and black-box optimisation algorithms. In Section 9.3, we identify *Mesh Adaptive Direct Search* (MADS) and *Surrogate Optimisation Integer* (SO-I) to be suitable solution algorithms for capacity planning in order fulfilment systems. Their procedure and their problem-specific configuration are presented in Sections 9.4 and 9.5, respectively. By summarising the results of this chapter, Section 9.6 provides an answer to the third research question of this thesis.

9.1 Capacity Planning Problem

Capacity planning in multi-stage order fulfilment systems determines the minimum possible capacity that has to be provided at each processing step to meet the performance requirements of the customers, such as a β-service level of 98%, in a multi-stage order fulfilment system with levelled order release. In the following, we introduce the mathematical formulation of the capacity planning problem (see Section 9.1.1) and specify its characteristics (see Section 9.1.2).

9.1.1 Mathematical Formulation

The decision variables of the capacity planning problem are defined by

$$\mathbf{c} = \begin{pmatrix} c_1 & \dots & c_{p_{max}} \end{pmatrix}, \tag{9.1}$$

whereby $c_p \in \mathbb{N}$ specifies the capacity provided at processing step $p \in \mathcal{P}$. The objective function minimises the sum of provided capacity:

$$\min \sum_{p \in \mathcal{P}} c_p.$$

We ensure that the performance $SL(\mathbf{c})$ achieved with the capacity $\mathbf{c}$ is as least as high as the required performance SL^* by formulating the following constraint:

$$SL(\mathbf{c}) \geq SL^*.$$

Performance requirements in order fulfilment are commonly given as service level requirements. However, any other performance measure of the order fulfilment system (see Table 6.1) can be used to specify the performance requirements of the customers. It is even possible to consider multiple different performance requirements in the capacity planning problem. In this case, each performance requirement is modelled as a separate constraint.

Furthermore, we ensure the order fulfilment system to be stable by limiting the order income-related utilisation $\tilde{U}(\mathbf{c})$ of the order fulfilment system with the provided capacity $\mathbf{c}$ to at most one:

$$\tilde{U}(\mathbf{c}) \leq 1.$$

We can derive lower $c_{p,min}$ and upper limits $c_{p,max}$ of the provided capacity c_p, $p \in \mathcal{P}$, based on the ranges of the number of incoming orders per time period A and the processing performances per time unit L_p of the processes $p \in \mathcal{P}$ of the considered order fulfilment system. We obtain the minimum possible value of the required capacity $c_{1,min}$ at process $p = 1$ if the number of incoming orders per time period is at its minimum a_{min}, and the processing performance per time unit of process $p = 1$ is at its maximum $l_{1,max}$. For processes $p \in \mathcal{P} \setminus \{1\}$, the minimum possible value of the required capacity $c_{p,min}$ corresponds to the smallest value of the minimum possible number of incoming orders per time period a_{min} and the minimum possible processing performances per time period $(c_{p',min} \cdot l_{p',min})$ of the previous processing steps $(p' < p)$:

$$c_{p,min} = \begin{cases} \left\lceil \frac{a_{min}}{l_{1,max}} \right\rceil & p = 1 \\ \left\lceil \frac{\min\left\{a_{min}; \min_{p' \in \mathcal{P}, p' < p}\left\{c_{p',min} \cdot l_{p',min}\right\}\right\}}{l_{p,max}} \right\rceil & p \in \mathcal{P} \setminus \{1\} \end{cases} \tag{9.2}$$

In contrast, we obtain the maximum possible value of the required capacity $c_{1,max}$ at process $p = 1$ if the number of incoming orders per time period is at its maximum a_{max}, and the processing performance per time unit of process $p = 1$ is at its minimum $l_{1,min}$. For processes $p \in \mathcal{P} \setminus \{1\}$, the maximum possible value of the required capacity $c_{p,max}$ results from the ratio of the maximum possible processing performance per time period $\left(c_{(p-1),max} \cdot l_{(p-1),max}\right)$ of the previous processing step $(p-1)$ to the minimum possible processing performance per time unit $l_{p,min}$ of process p:

$$c_{p,max} = \begin{cases} \left\lceil \frac{a_{max}}{l_{1,min}} \right\rceil & p = 1 \\ \left\lceil \frac{c_{(p-1),max} \cdot l_{(p-1),max}}{l_{p,min}} \right\rceil & p \in \mathcal{P} \setminus \{1\}. \end{cases} \tag{9.3}$$

Consequently, the domain $\mathcal{C}$ of the decision variables $\mathbf{c}$ is specified as follows

$$\mathcal{C} = \{\mathbf{c} \in \mathbb{N}^{p_{max}} \mid c_{p,min} \leq c_p \leq c_{p,max} \; \forall p \in \mathcal{P}\} \tag{9.4}$$

The resulting mathematical formulation of the capacity planning problem is given as follows

$$\min \sum_{p \in \mathcal{P}} c_p \tag{9.5}$$

$$\text{s.t. } SL(\mathbf{c}) \geq SL^* \tag{9.6}$$

$$\tilde{U}(\mathbf{c}) \leq 1 \tag{9.7}$$

$$\mathbf{c} \in \mathcal{C} \tag{9.8}$$

$$\mathbf{c} \in \mathbb{N}^{p_{max}}. \tag{9.9}$$

9.1.2 Characteristics

The capacity planning problem (9.5)-(9.9) is a constrained integer optimisation model. Its special characteristic is that the relationship between the provided capacity $\mathbf{c}$ and the performance $SL(\mathbf{c})$ that is achieved with this capacity cannot be specified by a mathematical equation describing the performance of the order fulfilment system depending on the provided capacity. Instead, the relationship between the provided capacity and the resulting performance is specified by the analytical model for performance analysis that calculates for any order fulfilment system with given capacity the resulting system performance. This kind of optimisation model is called blackbox optimisation model.

The optimisation model

$$\min \; \Theta(\xi) \tag{9.10}$$

$$\text{s.t. } \Lambda_j(\xi) \leq 0 \qquad \forall j \in \mathcal{J} \tag{9.11}$$

$$\xi \in \Omega \tag{9.12}$$

$$\xi \in \mathbb{R}^n \tag{9.13}$$

with objective function $\Theta(\xi)$, a finite number of constraints $\Lambda_j(\xi)$, $j \in \mathcal{J}$, and a finite variable domain Ω is said to be a *blackbox optimisation model* if the objective function $\Theta(\xi)$ and/or the constraints $\Lambda_j(\xi)$, $j \in \mathcal{J}$, are not given as mathematical equations but as a blackbox (Audet and Hare 2017, p.5).

In the case of the capacity planning problem, constraint (9.6) is given as a blackbox, whereby the analytical model for performance analysis represents the blackbox. In conclusion, the blackbox optimisation model for capacity planning has the following characteristics:

- One linear objective function (see equation (9.5)),
- One blackbox constraint (see equation (9.6)),
- One non-linear constraint (see equation (9.7)),
- Bound constraints (see equation (9.8)), and
- Integer decision variables (see equation (9.9)).

Furthermore, the blackbox constraint is said to be relaxable since it does not need to be satisfied to obtain a meaningful output of the blackbox (Le Digabel and Wild 2015). These characteristics become crucial in Section 9.3 when selecting suitable solution algorithms for the capacity planning problem.

9.2 Derivative-free and Blackbox Optimisation Algorithms

In contrast to classical optimisation models, gradient information of the objective functions and the constraints of blackbox optimisation models are not or only partially available. Thus, common gradient-based solution approaches of optimisation models cannot be applied to solve blackbox optimisation models, such as the capacity planning problem. The research field of derivative-free and blackbox

optimisation studies optimisation algorithms that do not require derivative information. Thus, these optimisation algorithms are applicable to solve the capacity planning problem.

9.2.1 Definition

Derivative-free optimisation is defined as the mathematical study of optimisation algorithms that do not use gradient information. In contrast, *blackbox optimisation* focuses on the design and analysis of algorithms that assume the objective function and/or the constraints to be given as blackboxes. Blackbox optimisation usually does not make any assumptions regarding continuity, differentiability, or smoothness of the outputs. It includes research on heuristic methods or ad hoc methods for solving blackbox optimisation models. In contrast, derivative-free optimisation focuses on methods that can be mathematically analysed to prove convergence and/or to provide a stopping criterion (Audet and Hare 2017, p.5f.).

Typical application fields of derivative-free and blackbox optimisation are optimisation models whose functions are provided by a computer simulation that cannot be easily subjected to differentiation, optimisation models that involve conducting laboratory experiments that cannot be described by explicit mathematical equations, and optimisation models including noisy functions whose gradients are highly unreliable (Audet and Hare 2017, p.11f.).

If gradient information of the objective function and the constraints are available, reliable, and obtainable at reasonable cost, derivative-free and blackbox optimisation algorithms rarely outperform standard gradient-based optimisation algorithms. Thus, in this case, gradient-based algorithms should always be preferred (Audet and Hare 2017, p.6).

9.2.2 Classification

There are several criteria to classify the field of derivative-free and blackbox optimisation algorithms. Rios and Sahinidis (2013) provide a review on derivative-free algorithms for blackbox optimisation models with bound constraints and classify the algorithms into direct versus model-based algorithms, local versus global algorithms, and stochastic versus deterministic algorithms. The classification into direct and model-based methods is the most common one in the literature.

9.2.2.1 Direct Search Methods

There is no exact definition of the term *direct search* in the literature. It was first mentioned by Hooke and Jeeves (1961), who defined direct search as "sequential examination of trial solutions involving comparison of each trial solution with the best [solution] obtained up to that time together with a strategy for determining what the next trial solution will be." A direct search method is an iterative method that only uses comparisons of function values to determine trial points and that does not attempt to develop or use approximate gradients (Audet 2014; Larson et al. 2019, p.293). The pioneers among the direct search methods are the *Coordinate Search* (Fermi 1952), the *Nelder-Mead algorithm* (Nelder and Mead 1965), and the *Hooke and Jeeves algorithm* (Hooke and Jeeves 1961).

9.2.2.2 Model-based Derivative-Free Methods

Model-based derivative-free methods iteratively construct a surrogate model to approximate the blackbox model (Sóbester et al. 2012). The objective function and the constraints of a surrogate model are reasonably accurate representations of the ones of the blackbox model at least locally, but the surrogate model is much faster to evaluate than the blackbox model. Note that the surrogate model of the blackbox optimisation model (9.10)-(9.13) consists of $(|\mathcal{J}| + 1)$ surrogate functions, one for the objective function and one for every constraint $j \in \mathcal{J}$.

In each iteration, the surrogate model is used to determine new candidate points (Koziel et al. 2011, p.34f.), (Audet and Hare 2017, p.236).

The generic procedure of a model-based derivative-free method consists of the following phases (Vu et al. 2017):

1. Design phase,
2. Model phase, and
3. Search phase.

In the *design phase*, a set of starting points is created, and the blackbox model is evaluated at these points. The Latin hypercube design is a commonly used approach to create this set. The purpose of the design phase is to create a set of points uniformly spread over the domain Ω of the blackbox model to get a first, global picture of its behaviour. In the *model phase*, a surrogate model of the blackbox model is constructed based on the already evaluated points. The most commonly used surrogate functions to approximate the blackbox model are polynomials, radial basis functions, and kriging models. It can be reasonable to consider multiple surrogate models in the model phase to prevent the user from choosing a poorly fitted model (Viana et al. 2013). Subsequently, in the *search phase*, a new candidate point is selected based on the surrogate model, the blackbox model is evaluated at this point, and this point is added to the set of already evaluated points. The selection of a new candidate point is the most crucial step in this phase. There is always the trade-off between exploitation – select a point around the current best point – and exploration – select a point that is far from all previously considered points to ensure that no global solution is overlooked. The model and the search phase are repeated until a predefined stopping criterion, such as a maximum number of evaluations of the blackbox model, is met (Vu et al. 2017).

Koziel et al. (2011) and Vu et al. (2017) provide comprehensive reviews of model-based derivative-free algorithms for blackbox optimisation models.

9.3 Selection of Solution Algorithms for Capacity Planning

To select suitable algorithms for the capacity planning problem, we refer to the following criteria:

- The algorithm meets the characteristics of the capacity planning problem (see Section 9.1.2). Thus, we require an algorithm for constrained integer blackbox optimisation models.
- The convergence of the algorithm is mathematically proven.
- The implementation of the algorithm is open source.

We select one direct search method and one model-based derivative-free method to investigate a representative of each category of blackbox optimisation algorithms regarding its performance in solving the capacity planning problem.

The focus of derivative-free and blackbox optimisation algorithms in the literature is on unconstrained problems on the one hand and optimisation models with continuous decision variables on the other hand. However, derivative-free algorithms solving constrained integer blackbox optimisation models have barely been studied in the literature so far (Müller et al. 2014).

Regarding the class of direct search methods, we prefer the *Mesh Adaptive Direct Search* (MADS) (Audet and Dennis Jr 2006) over the *Generalised Pattern Search* (Torczon 1997), and the *Coordinate Search* (Fermi 1952) since the MADS algorithm is the only method that considers general constraints. Furthermore, integer variables can be easily handled by the MADS algorithm since its mesh imposes a discrete structure on the variable domain. The MADS algorithm is a local optimisation method whose convergence to a local optimum is mathematically proven (Audet 2014). The implementation of the MADS algorithm is provided in the open-source software tool NOMAD (Abramson et al. 2018).

Regarding the class of model-based derivative-free methods, we select the *Surrogate Optimisation Integer* (SO-I) algorithm (Müller et al. 2014), which is an asymptotically complete model-based algorithm for integer optimisation models with computationally expensive blackbox objective functions and constraints. The implementation of the SO-I algorithm is available as MATLAB surrogate model toolbox MATSuMoTo (Müller 2014).

9.4 Capacity Planning using Mesh Adaptive Direct Search (MADS)

The direct search method *Mesh Adaptive Direct Search* (MADS) introduced by Audet and Dennis Jr (2006) is an iterative procedure to identify a local optimum of the blackbox optimisation model by evaluating the blackbox model at selected points, starting at a given initial point. In each iteration, new points are selected based on different criteria: Search step and poll step. The main characteristic of the MADS algorithm is that every trial point has to be part of the mesh. The mesh is a discretisation of the variable domain Ω, and the mesh size parameter controls its fineness (Audet and Hare 2017, p.136-140).

In the following, we initially provide a high-level description of the MADS algorithm for solving unconstrained blackbox optimisation models with bound constraints (see Section 9.4.1). Subsequently, we present several selected extensions of the MADS algorithm, such as constraint handling, that are relevant for solving the capacity planning problem (see Section 9.4.2). Finally, we introduce the problem-specific configuration of the MADS algorithm for capacity planning in order fulfilment systems (see Section 9.4.3).

9.4.1 General Procedure

We only provide a high-level description of the procedure and the main characteristics of the MADS algorithm for solving unconstrained blackbox optimisation models with bound constraints. A detailed description can be found in Audet and Hare (2017, p.136-140).

Each iteration of the MADS algorithm consists of the following steps:

1. Search step,
2. Poll step,
3. Parameter update, and
4. Check of stopping criterion.

In the *search step*, any method can be used to select a finite number of trial points. The focus of the search step is on exploring the variable domain Ω to escape from local optima and seek a global optimum. Line search, surrogate models, or heuristic methods, such as Tabu search, variable neighbourhood search, and simulated annealing, are commonly used methods (see Section 9.4.2.4). The set of trial points is evaluated either opportunistically or completely. In the case of an opportunistic evaluation, the evaluation of trial points stops as soon as a new incumbent solution is found. Otherwise, in the case of a complete evaluation, all trial points are evaluated, and the best trial point is selected as new incumbent solution, only if it outperforms the current incumbent solution.

In the *poll step*, a finite number of trial points around the current incumbent solution is selected based on given polling conditions:

- The distance between the incumbent solution and every trial point generated in the poll step is limited by the poll size parameter.
- The directions used to construct the poll set form a positive spanning set.

The MADS algorithm provides different strategies to generate the polling directions (see Section 9.4.2.2). The set of trial points is evaluated using either the opportunistic or the complete strategy, analogous to the search step. The poll step ensures global convergence towards a local optimum.

In the *step of parameter update*, mesh and poll size parameter are updated depending on whether the current iteration was successful – a new incumbent solution was found – or unsuccessful – no new incumbent solution was found. The mesh size parameter converges much faster towards zero than the poll size parameter does. Hence, trial points in the search and the poll step can be chosen on a finer mesh than the mesh defined by the poll size parameter.

At the end of each iteration, the *stopping criterion* is checked. Either the total number of blackbox model evaluations, the computation time, or a threshold of the mesh size parameter are used to specify the stopping criterion. If the stopping criterion is met, the algorithm terminates. Otherwise, a new iteration is conducted.

The choice of the initial point is a crucial issue since the number of required iterations significantly reduces when choosing a good initial point. Either the initial point is chosen based on practitioners' professional intuition, or a small number of blackbox model evaluations is conducted to explore the behaviour of the blackbox model in the variable domain and to determine the initial point based on this sample.

9.4.2 Extensions

In the following, we consider several extensions of the generic procedure of the MADS algorithm presented in Section 9.4.1. However, we limit ourselves to the extensions that are relevant for solving the capacity planning problem by the MADS algorithm.

9.4.2.1 Constraint Handling

The MADS algorithm provides multiple approaches for constraint handling depending on the fact whether the constraints are relaxable or unrelaxable. A relaxable constraint does not need to be satisfied to obtain a meaningful output of the blackbox. In contrast, a blackbox model with an unrelaxable constraint only provides a meaningful output if the unrelaxable constraint is satisfied (Le Digabel and Wild 2015).

The *extreme barrier approach* considers unrelaxable constraints. It sets the objective value to infinity for every infeasible point. Thus, the constrained blackbox optimisation model is treated as an unconstrained one, and any feasible point is preferred over any infeasible one (Audet and Dennis Jr 2006).

The *progressive barrier approach* uses the following non-negative constraint violation function

$$\Gamma(\xi) = \sum_{j \in \mathcal{J}} \left(\max\left\{0; \Lambda_j(\xi)\right\}\right)^2 \tag{9.14}$$

to handle relaxable constraints. It places a threshold on the constraint violation it allows. All infeasible points that exceed this threshold are rejected, whereby the value of the threshold is progressively tightened with an increasing number of conducted iterations. The MADS algorithm with progressive barrier approach differentiates between feasible and infeasible incumbent solutions. In the poll step, polling is done around some points of the set of feasible incumbent solutions and the set of infeasible incumbent solutions (Audet and Dennis Jr 2009).

The *progressive to extreme barrier approach* is a hybrid method of progressive and extreme barrier approach. Relaxable constraints are initially treated by the progressive barrier approach. If polling around an infeasible incumbent solution generates a new infeasible incumbent that satisfies a constraint violated by the current infeasible incumbent, the corresponding constraint is treated by the extreme barrier approach in all future iterations. Consequently, all relaxable constraints are treated by the extreme barrier approach after finitely many iterations (Audet et al. 2010).

9.4.2.2 Strategies to Generate Poll Directions

The MADS algorithm provides several strategies to generate the directions used to construct trial points in the poll step. The precondition is that the directions form a positive spanning set.

GPS is the most straightforward strategy to generate poll directions. Following the approach of the *Generalised Pattern Search* (Torczon 1997), GPS uses the coordinate directions as poll directions.

LTMADS is a random procedure to generate poll directions. The main drawbacks are that the runs of the MADS algorithm are not reproducible due to the randomly generated directions and that the generated directions are not necessarily orthogonal, which possibly leads to large angles between directions. Audet and Dennis Jr (2006) provide a detailed description of LTMADS.

In contrast, *ORTHOMADS* is a deterministic procedure to generate poll directions that provides an orthogonal positive spanning set of polling directions. Thus, the directions cover the surface of the unit sphere more densely and evenly than the ones of LTMADS. The procedure of ORTHOMADS is presented in detail in Abramson et al. (2009).

9.4.2.3 Surrogate Management Framework

If it is computationally expensive to evaluate the blackbox model at a certain point, it can be reasonable to use surrogate models. An overview of commonly used surrogate functions is given in Table 9.1. The surrogate management framework specifies how surrogate models can be integrated into direct search methods, especially into the MADS algorithm (Audet 2014).

On the one hand, surrogate models can be applied to conduct a model-based search in the search step. For this, a surrogate model is constructed based on the set of already evaluated points. The surrogate model is optimised to generate a

Table 9.1: Commonly used surrogate functions in derivative-free and blackbox optimisation (Conn and Le Digabel 2013; Vu et al. 2017; Audet et al. 2018; Bhosekar and Ierapetritou 2018).

Quadratic model	$\Sigma(\xi) = \omega_0 + \omega_1 \cdot \xi + \omega_2 \cdot \xi^2$
Polynomial response surface	$\Sigma(\xi) = \sum_{j=1}^{n_F} \omega_j \cdot \chi_j(\xi)$
Kriging model	$\Sigma(\xi) = \sum_{j=1}^{n_F} \omega_j \cdot \psi_j(\xi) + \epsilon(\xi)$
Radial basis function	$\Sigma(\xi) = \sum_{i=1}^{n_S} \omega_i \cdot \phi(\delta(\xi, \xi_i))$
Kernel smoothing model	$\Sigma(\xi) = \frac{\sum_{i=1}^{n_S} \phi(\delta(\xi,\xi_i)) \cdot \Theta(\xi_i)}{\sum_{i=1}^{n_S} \phi(\delta(\xi,\xi_i))}$

Notation

$\Sigma(\xi)$	Surrogate function
$\chi(\xi)$	Polynomial basis function of polynomial response surface
$\psi(\xi)$	Basis function of Kriging model that defines the trend of the mean prediction
$\epsilon(\xi)$	Random error of Kriging model that is normally distributed with zero mean
$\phi(\xi)$	Kernel function • Linear: $\phi(\xi) = \xi$ • Cubic: $\phi(\xi) = \xi^3$ • Thin plate spline: $\phi(\xi) = \xi^2 \cdot ln(\xi)$ • Gaussian: $\phi(\xi) = e^{-\frac{\xi^2}{2\sigma^2}}$, whereby σ denotes the shape coefficient • Multi-quadric: $\phi(\xi) = \sqrt{\xi^2 + \sigma^2}$, whereby σ denotes the shape coefficient • Inverse multi-quadric: $\phi(\xi) = \frac{1}{\sqrt{\xi^2+\sigma^2}}$, whereby σ denotes the shape coefficient
$\delta(\xi, \xi_i)$	Distance function, for instance l_1-, l_2- and l_∞-norm
ω	Weight
n_F	Number of independent basis functions
n_S	Sample size of sample used to construct the surrogate function
ξ_i	i^{th} sample point
$\Theta(\xi_i)$	True function value of the i^{th} sample point

finite set of trial points, and the blackbox model is evaluated at these trial points using the opportunistic strategy (Audet and Hare 2017, p.238).

On the other hand, surrogate models can be used to order a given set of trial points either in the search or the poll step. For this, a surrogate model is constructed based on the set of already evaluated points. The surrogate model is evaluated at every trial point, and the trial points are sorted according to their value of the surrogate model, such that the most promising trial point is the first point of the ordered set. Finally, the blackbox model is evaluated at the ordered trial points using the opportunistic strategy (Audet and Hare 2017, p.238).

9.4.2.4 Methods to Use in the Search Step

There is a plethora of suitable methods to create trial points in the search step. It is even possible to use several different methods in the search step. In the following, we focus on the methods applied to solve the capacity planning problem by the MADS algorithm.

As already described in Section 9.4.2.3, a model-based method can be used in the search step to create a set of trial points based on a surrogate model.

Speculative search represents a further suitable method for the search step. If the previous iteration of the MADS algorithm succeeded in finding an improved incumbent solution, the speculative search in the search step of the current iteration executes a simple line search along the previously successful polling direction (Audet and Dennis Jr 2006).

Moreover, it is possible to use a problem-specific adaption of the *Nelder-Mead algorithm* in the search step. Each iteration of the adapted Nelder-Mead algorithm starts with a set of affinely independent points defining a simplex. These simplex points are ordered based on the values of objective function and constraints from the best to the worst. In each iteration, the worst point of the simplex is replaced by a new point. Audet and Tribes (2018) give a detailed description of the adapted Nelder-Mead algorithm.

9.4.3 Problem-specific Configuration

As already indicated by the discussed extensions in Section 9.4.2, the MADS algorithm provides a plethora of algorithmic parameters enabling its problem-specific configuration. The user manual of the corresponding software tool NOMAD (Audet et al. 2009) provides a comprehensive overview of all algorithmic parameters.

Regarding a problem-specific configuration of the MADS algorithm for the capacity planning problem, we determine the most important algorithmic parameters

based on the results of numerical studies provided in the literature: First, we choose the progressive barrier approach (see Section 9.4.2.1) to handle the blackbox constraint of the capacity planning problem since it is a relaxable constraint (see Section 9.1.2). Furthermore, numerical studies show that the progressive barrier approach is the most reliable strategy for constraint handling (Audet and Dennis Jr 2009; Audet et al. 2010). Second, regarding the strategy to generate the poll directions (see Section 9.4.2.2), numerical studies show that ORTHOMADS outperforms GPS and that solutions of ORTHOMADS are not significantly bested by LTMADS in any scenario, especially in the case of computationally expensive blackbox models, when only one run is conducted (Abramson et al. 2009; Audet et al. 2010). Since the blackbox of the capacity planning problem is computationally expensive, we prefer ORTHOMADS over LTMADS.

We use the speculative search in the search step (see Section 9.4.2.4) and we will investigate the added value of an additional use of the model-based method and the Nelder-Mead algorithm in a numerical analysis on fine-tuning of the MADS algorithm for solving the capacity planning problem in Chapter 11. Furthermore, we will investigate different approaches to determine the initial point of the MADS algorithm in this numerical analysis. The stopping criterion of the MADS algorithm is defined by a threshold on the mesh size parameter. We choose a tolerance of one since this corresponds to the natural threshold of the mesh size parameter concerning integer blackbox optimisation models.

The blackbox optimisation model for capacity planning (9.5)-(9.9) has the following characteristics:

- The objective function (9.5) is continuous and strictly differentiable at any point $\mathbf{c} \in \mathbb{N}^{p_{max}}$.
- The variable domain $\mathcal{C}$ is finite (see equation (9.4)).
- For every point within the variable domain $\mathcal{C}$, the value of the objective function and the value of $\tilde{U}(\mathbf{c})$ are finite, and the value of $SL(\mathbf{c})$ is in the range [0,1] due to the domain of the service level (see Section 6.3.7).

Based on these characteristics, applying the MADS algorithm with the progressive barrier approach, the constraint violation function (9.14), and ORTHOMADS to the capacity planning problem generates a convergent refining subsequence of incumbent solutions. The limit of the convergent refining subsequence is a Clarke KKT stationary point for the optimisation of the capacity planning problem. Audet et al. (2010) provide details on the convergence analysis of the MADS algorithm with progressive barrier approach and ORTHOMADS.

9.5 Capacity Planning using Surrogate Optimisation Integer (SO-I)

The *Surrogate Optimisation Integer* (SO-I) algorithm introduced by Müller et al. (2014) is a model-based derivative-free method for solving constrained, integer blackbox optimisation models. In the following, we provide a high-level description of the SO-I algorithm (see Section 9.5.1) and introduce the problem-specific configuration of the SO-I algorithm for capacity planning in order fulfilment systems (see Section 9.5.2).

9.5.1 General Procedure

The SO-I algorithm consists of two separate optimisation phases. The first optimisation phase aims at finding a first feasible point if the initial set of points does not contain any feasible point. For this, a model-based derivative-free method (see below) is used to minimise the constraint violation function

$$\Gamma(\xi) = \sum_{j \in \mathcal{J}} \max\{0; \Lambda_j(\xi)\} \tag{9.15}$$

with respect to the bound and integrality constraints of the blackbox optimisation model. The second optimisation phase aims at identifying the minimum of the

constrained blackbox optimisation model. For this, a model-based derivative-free method (see below) is used to minimise the penalty-augmented objective function

$$\Theta_p(\xi) = \begin{cases} \Theta_{max} + \rho \cdot \sum_{j \in \mathcal{J}} \left(\max\{0; \Lambda_j(\xi)\}\right)^2 & \text{if } \xi \text{ is infeasible} \\ \Theta(\xi) & \text{if } \xi \text{ is feasible,} \end{cases} \tag{9.16}$$

with respect to the bound and integrality constraints of the blackbox optimisation model. Θ_{max} denotes the current worst feasible objective function value, and ρ denotes the penalty factor.

The model-based derivative-free method used in both optimisation phases follows the generic procedure of model-based derivative-free methods as described in Section 9.2.2.2: In the design phase, a Latin hypercube design is used to create the initial set of points. A cubic radial basis function with l_2-norm and a linear polynomial tail is used as surrogate function. In the search phase, a finite set of trial points is created by

- uniformly selecting integer points in the variable domain, and
- perturbing the best feasible point found so far.

Based on a score, the new candidate point is selected from the set of trial points. The score is calculated as the weighted sum of the predicted objective function value on the one hand and a distance criterion on the other hand. The predicted objective function value of a trial point corresponds to the value of the surrogate model at this point, scaled to the range [0,1]. The distance criterion calculates the distance of a trial point to the set of already evaluated points, whereby the result is also scaled to the range [0,1]. The iterative procedure terminates if either a local minimum is found, or the number of blackbox model evaluations equals the predefined maximum number of blackbox model evaluations (Müller et al. 2014).

The SO-I algorithm is asymptotically complete. Under the assumption of indefinitely long runtime and exact calculations, the SO-I algorithm will find the global minimum of the blackbox optimisation model with probability one (Müller 2014).

9.5.2 Problem-specific Configuration

The SO-I algorithm, as described in Section 9.5.1, does not contain any degree of freedom enabling a problem-specific configuration of the algorithm with respect to the capacity planning problem. However, the SO-I implementation in the MATLAB-toolbox MATSuMoTo provides several algorithmic parameters to configure the SO-I algorithm. We will investigate some of them, such as the selected surrogate function and the selected sampling strategy in the search step, in a numerical analysis in Chapter 11 in order to fine-tune the SO-I algorithm for solving the capacity planning problem.

9.6 Chapter Conclusion

In this chapter, we formulated the decision problem of capacity planning in multi-stage order fulfilment systems with performance requirements as a mathematical optimisation model and provided two solution algorithms.

The capacity planning problem is a blackbox optimisation problem since the relationship between the provided capacity and the performance that is achieved with this capacity cannot be specified by a mathematical equation, but it is given by the analytical model for performance analysis. The analytical model for performance analysis that calculates for any order fulfilment system with given capacity the resulting system performance represents the blackbox of the capacity planning problem. The research field of derivative-free and blackbox optimisation provides solution algorithms for blackbox optimisation models that do not require derivative information. We selected the direct search method *Mesh Adaptive Direct Search* (MADS) and the model-based derivative-free method *Surrogate Optimisation Integer* (SO-I) as suitable solution algorithms for capacity planning since they meet the characteristics of the capacity planning problem, their convergence is mathematically proven, and their implementation is open source.

The problem-specific configurations of the MADS algorithm and the SO-I algorithm enable a target-oriented determination of the minimum required, process-specific capacity to meet any performance requirement of the customers that is specified based on one or multiple performance measures of the order fulfilment system (see Table 6.1). Thus, the MADS algorithm and the SO-I algorithm provide an answer to the third research question of the thesis:

> **How can we determine the capacity required to meet specific performance requirements in multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines?**

10 Runtime and Memory Optimisation of the Markov Chain

When modelling system behaviour of a real-life order fulfilment system as a discrete-time Markov chain, memory and computation time requirements become challenging issues. This chapter aims at evaluating multiple approaches for runtime and memory optimisation of a Markov chain. In Section 10.1, we introduce multiple strategies to optimise memory and computation time requirements of a Markov chain. The optimisation potentials of these strategies regarding the models provided in this thesis are analysed in a numerical study in Section 10.2. Section 10.3 summarises the results of this chapter.

10.1 Strategies for Runtime and Memory Optimisation

Memory and computation time requirements of a Markov chain are driven by the size of its state space and the discretisation of its stochastic parameters (see Section 6.4). Since the size of the state space of the Markov chain quickly increases when modelling the system behaviour of larger order fulfilment systems (see equation (6.45)), memory usage and computational effort are often challenging issues. In the following, we propose multiple strategies to optimise runtime and memory usage of a Markov chain.

10.1.1 Limitation of State Space

The state space of the Markov chain results from its lower and upper limit (see Section 6.2.3), and its size is calculated by equation (6.45). However, depending on the parameterisation of the order fulfilment system and the discretisation of its stochastic parameters, this set of states contains multiple unreachable states. Unreachable states are not relevant in the subsequent calculation steps of the Markov chain. Consequently, these states are removed from the state space, and the subsequent calculation steps are limited to the set of reachable states.

The *classical procedure* to determine the set of reachable states of the Markov chain initially calculates all theoretically reachable states based on the lower and upper limit of the state space as well as the corresponding transition matrix. Unreachable states are then identified based on the transition matrix. State $i \in \mathcal{X}$ is said to be an *unreachable state* if the i^{th} column of the transition matrix is a zero-vector. Finally, the identified unreachable states are removed from the state space and the transition matrix to obtain the set of reachable states and the corresponding transition matrix. The main drawback of this procedure is its inefficient use of runtime and memory: State space and transition matrix are first calculated and stored for all theoretically reachable states. Subsequently, unreachable states are removed.

The *alternative procedure*, which we propose to determine the set of reachable states of the Markov chain, iteratively identifies all states that are reachable from a given initial state of the Markov chain, either by a single-step transition or by an indirect transition via a finite number of other states. In this way, the alternative procedure determines the set of reachable states in a runtime- and memory-efficient manner since the state space and the transition matrix do not contain any unreachable state at any intermediate step of the calculation procedure.

For irreducible Markov chains, the alternative procedure does not provide any advantage compared to the classical procedure since the Markov chain consists of a unique communicating class. However, for reducible Markov chains, the alternative procedure does not determine the whole set of reachable states of the

Markov chain but only the relevant subset of reachable states that contains all states that are reachable from the initial state. Thus, by using the alternative procedure, computation time and memory usage are reduced.

Based on the alternative procedure, it is impossible to calculate the matrix of limiting distributions of a reducible Markov chain. In contrast, we can only calculate the limiting distribution of the initial state. Hence, from a mathematical point of view, the alternative procedure is insufficient since it does not enable a complete description of the asymptotic behaviour of the Markov chain. However, from a practical perspective, it is sufficient to know the limiting distribution of the initial state of the Markov chain since the initial state describes the current state of the considered system.

10.1.2 Sparse Storage Schemes

Transition matrices of Markov chains are usually large-sized but sparse matrices. A *sparse matrix* is a matrix with very few nonzero elements. The key idea of sparse storage schemes is to neglect the zero elements and to only store nonzero matrix entries. At the same time, common matrix operations still have to be possible (Saad 2003, p.73).

The *compressed sparse row* (CSR) format is the most popular sparse storage scheme. It uses three arrays E_v, E_c, and E_r to store the sparse matrix $\mathbf{E}$:

- Array E_v contains the real values $e_{i,j}$ of matrix $\mathbf{E}$ stored row by row, from the first to the last row.
- Array E_c contains the column indices of the matrix entries $e_{i,j}$ as stored in array E_v.
- Array E_r contains the pointers to the beginning of each row in array E_v and E_c. The i^{th} entry of array E_r is the position where the i^{th} row starts in the arrays E_v and E_c.

There are multiple variations of the CSR format, such as the *compressed sparse column* (CSC) format, which stores columns instead of rows (Saad 2003, p.89f.).

10.1.3 Indirect Solution Methods for Linear Systems

The limiting distribution of the Markov chain is obtained by solving a set of linear equations based on the transition matrix of the Markov chain. Academic literature provides a plethora of methods to solve sets of linear equations. These methods are classified into direct and indirect solution methods. *Direct methods* modify the parameter matrix and use a fixed number of operations to exactly solve the set of linear equations. Common examples are the Gaussian elimination and Grassmann's algorithm (Bolch 2006, p.118f.). On the contrary, *indirect methods* iteratively calculate estimates starting with an initial estimate of the unknown solution. The sequence of estimates is expected to converge towards the solution eventually. The iterative procedure terminates if the estimates are sufficiently close to the exact solution. Hence, indirect methods only provide an approximation of the exact solution (Bolch 2006, p.132). Table 10.1 provides a comparison of direct and indirect solution methods for linear systems. We point out some differences in detail in the following.

Table 10.1: Comparison of direct and indirect solution methods for linear systems (Stewart 1994, p.61f.), (Bolch 2006, p.104,118f.,132).

	Direct Methods	Indirect Methods
Result quality	Exact	Approximate
Convergence rate	No issue	Dependent on various factors
Use of sparse storage schemes	Difficult	Easily possible
Round-off errors	Accumulation	No issue
Computation time	Fixed number of operations independent of the values of parameter matrix	Number of required iterations depends on various factors

Direct methods modify the entries of the parameter matrix, so that original zero entries can become nonzero entries. Consequently, round-off errors occur, and it is difficult to use sparse storage schemes. In contrast, indirect methods only use matrix multiplications that do not alter the parameter matrix: Original zero entries stay zero entries, and the sparsity of parameter matrix is preserved. Hence, there are no round-off errors, and sparse storage schemes are applicable. The convergence rate and the computation time of indirect methods depend on multiple factors, such as the quality of initial estimate, the required solution accuracy, the values of the parameter matrix, the chosen method, and the use of a preconditioning technique (Stewart 1994, p.61f.), (Bolch 2006, p.104,118f.,132). Preconditioning techniques are used to improve the efficiency and robustness of indirect methods. A *preconditioner* transforms the original set of linear equations into a set of linear equations that has the same solution but that is likely to be easier solved by an indirect method (Saad 2003, p.261f.). Saad (2003, p.283-345) provides a detailed overview of standard preconditioning techniques. In general, indirect methods are more efficient in space and time than direct methods. Hence, indirect methods are preferred over direct methods when solving large sets of linear systems (Bolch 2006, p.104).

In this thesis, we use the indirect method *Generalized Minimal Residual Method* (GMRES) that is a popular method for solving large sets of linear systems. GMRES is an iterative Krylov subspace method to solve a nonsymmetric set of linear equations that minimises the norm of the residual vector over a Krylov subspace in each iteration (Saad and Schultz 1986). The procedure of GMRES is presented in detail in Stewart (1994, p.190-206) and Saad (2003, p.164-184).

10.1.4 Parallel Computing

Parallel computing speeds up the calculation of a Markov chain since multiple calculation steps of the Markov chain can be conducted in parallel: First, when calculating the state space using the alternative procedure (see Section 10.1.1), we can determine the states reachable from a given state in a single-step transition in

parallel for different states. Second, when calculating the transition matrix, it is possible to calculate each row in parallel, and for a given row, we can calculate all entries of this row in parallel. Finally, when calculating the limiting distribution, the matrix-vector operations, which are used to solve the set of linear equations, can be executed in parallel.

For the calculations of this thesis, we use the available processors of the CPU to conduct the aforementioned calculation steps in parallel. Graphics processing units (GPUs) provide further potentials to reduce computation time and memory usage. GPUs are widely used in high-performance computing, and they provide high computing power and large memory bandwidth. However, new programming challenges occur, such as adapting the parallel algorithms to the architecture of parallel computers with GPUs (Khodja et al. 2014). The calculation of Markov chains using GPUs is beyond the scope of this thesis.

10.2 Evaluation of Optimisation Potentials

In the following, we evaluate the potentials of the proposed strategies for runtime and memory optimisation regarding the exact model for performance analysis in a numerical analysis based on samples B and D. Sample B contains 133 single-stage order fulfilment systems, and sample D consists of 256 two-stage order fulfilment systems. The samples differ regarding the ranges of the parameters, especially regarding the discretisation of the number of incoming orders per time period A: Its range is given by $\mathcal{A}^B = \{3, 4, \ldots, 15\}$ in sample B and $\mathcal{A}^D = \{2, 3, \ldots, 6\}$ in sample D. A comprehensive description of the samples is given in Appendix C.2.

10.2.1 Limitation of State Space

The alternative procedure, proposed in Section 10.1.1 to calculate the set of reachable states of a Markov chain based on a given initial state, directly determines the set of reachable states. In contrast, the classical procedure initially calculates

the set of all theoretically reachable states and subsequently reduces this set to the set of reachable states. To show and to quantify the benefit of the alternative procedure, we compare the number of theoretically reachable states, which is calculated based on equation (6.45), with the number of reachable states. Note that the initial state of each order fulfilment system is randomly selected from its set of theoretically reachable states.

Table 10.2 presents the number of theoretically reachable states and the number of reachable states, as well as the absolute and relative deviations between these two metrics for samples B and D. For each of these criteria, Table 10.2 gives the minimum, the maximum, and the average value.

Table 10.2: Number of theoretically reachable states, number of reachable states, and potentials of state space limitation when using the alternative procedure.

	Criterion	Min	Max	Average
Sample B	Number of theoretically reachable states	8,125	15,376	14,115
	Number of reachable states	156	9,893	8,040
	Absolute deviation[1]	-12,251	-3,419	-6,075
	Relative deviation [%][2]	-35.66	-98.08	-43.57
Sample D	Number of theoretically reachable states	692,055	8,471,463	4,534,143
	Number of reachable states	1,401	9,625	6,859
	Absolute deviation[1]	-8,465,731	-685,779	-4,527,284
	Relative deviation [%][2]	-99.09	-99.93	-99.79

[1] Difference between the number of reachable states and the number of theoretically reachable states.
[2] Difference between the number of reachable states and the number of theoretically reachable states, divided by the number of theoretically reachable states.

For the single-stage order fulfilment systems of sample B, the number of theoretically reachable states is between 8,125 and 15,376 states, whereas the number of reachable states is between 156 and 9,893 states. For the two-stage order fulfilment systems of sample D, the number of theoretically reachable states is between 692,055 and 8,471,463 states. In contrast, the number of reachable states is between 1,401 and 9,625 states. Thus, we observe an average reduction in the

number of calculated states of 43.6% in sample B and 99.8% in sample D. These results show that the number of states calculated by the alternative procedure is significantly smaller than the one calculated by the classical procedure. Consequently, we recommend using the alternative procedure instead of the classical procedure to calculate the set of reachable states of the Markov chain in order to reduce memory usage and computational effort.

By comparing the magnitude of the number of theoretically reachable states between samples B and D, we observe that the number of processes of the considered order fulfilment system has a significant impact on the number of theoretically reachable states. The average number of theoretically reachable states is 14,115 states for the single-stage systems of sample B, whereas it is 4,534,143 states for the two-stage systems of sample D. The number of theoretically reachable states results from the lower and upper limit of the state space (see equation (6.45)). Since the number of processes p_{max} determines the size of the exponent in equation (6.45), it has a major impact on the number of theoretically reachable states (see Section 6.4). In contrast, its impact on the number of reachable states is significantly smaller. Consequently, we expect that the potential of the alternative procedure in optimising memory and computation time requirements increases with an increasing number of processes of the considered order fulfilment system.

Note that it is not meaningful to compare the absolute values of the number of reachable states between samples B and D since these depend on the discretisation of the stochastic parameters of the order fulfilment systems, which differs between samples B and D (see Appendix C.2).

10.2.2 Indirect Solution Methods for Linear Systems

In Section 10.1.3, we proposed using an indirect solution method for linear systems to reduce the computational effort of the Markov chain. To quantify the savings in computation time resulting from using an indirect solution method instead of a direct method for solving linear systems, we compare the computation time

required to calculate the exact model for performance analysis when using the Gaussian elimination with the one when using GMRES.

Table 10.3 presents the minimum, the maximum, and the average value of the computation time in samples B and D for the following specifications of the Markov chain: Use of the Gaussian elimination and sequential computing, use of GMRES and sequential computing, and use of GMRES and parallel computing. Furthermore, Table 10.3 gives the minimum, the maximum, and the average value of the reduction in computation time when using GMRES instead of the Gaussian elimination and when computing the Markov chain in parallel instead of sequential computation.

Table 10.3: Computation time and savings of computational effort when using GMRES and parallel computing.[1]

	Criterion	Specification of Markov chain	Min	Max	Average
Sample B	Computation time [s]	Gauss, sequential computing	0.04	432.05	219.68
		GMRES, sequential computing	0.03	308.44	121.08
		GMRES, parallel computing	0.05	6.47	3.51
	Reduction of computation time [%]	Use of GMRES[2]	-4.38	69.69	46.11
		Use of parallel computing[3]	-64.95	98.50	94.55
Sample D	Computation time [s]	Gauss, sequential computing	2.00	428.45	124.86
		GMRES, sequential computing	1.69	286.00	64.15
		GMRES, parallel computing	0.27	4.65	2.59
	Reduction of computation time [%]	Use of GMRES[2]	15.51	68.00	49.47
		Use of parallel computing[3]	83.89	98.37	95.36

[1] Computations are conducted on a server with a CPU of 64 kernels and 128 threads and a RAM of 128 GB.
[2] Relative deviation of the computation time of the exact model that uses GMRES and that is computed in sequential manner from the computation time of the exact model that uses the Gaussian elimination and that is computed in sequential manner.
[3] Relative deviation of the computation time of the exact model that uses GMRES and that is computed in parallel from the computation time of the exact model that uses GMRES and that is computed in sequential manner.

In sample B, when using the Gaussian elimination, computation time is at least 0.04 seconds, at most 432 seconds, and on average 220 seconds. In contrast, when using GMRES, computation time is between 0.03 and 308 seconds, and it is 121

seconds on average. Thus, by using GMRES instead of the Gaussian elimination, we reduce the computation time of the exact model by on average 46%. Sample B contains one data point for which computation time increases by 4% when using GMRES instead of the Gaussian elimination. In sample D, when using the Gaussian elimination, computation time is between 2 and 428 seconds, and it is 125 seconds on average. In contrast, when using GMRES, computation time is at least 1.7 seconds, at most 286 seconds, and on average 64 seconds. Hence, we observe an average reduction in the computation time of the exact model by 49% when using GMRES instead of the Gaussian elimination. These results confirm that using the indirect method GMRES to solve linear systems provides significant savings in computation time compared to using the Gaussian elimination. Note that it is not meaningful to compare the absolute values of the computation time between samples B and D since the computation time depends on the ranges of the parameters of the order fulfilment systems, which differ between samples B and D (see Appendix C.2).

Indirect methods, such as GMRES, suffer from a lack of result accuracy, whereas direct methods are said to calculate exact results. However, direct methods suffer from round-off errors (see Table 10.1). For this reason, we evaluate the result accuracy by comparing the values of selected performance measures – α-service level, β-service level, expected number of unprocessed orders, and expected number of completed orders – obtained when using GMRES with the ones obtained when using the Gaussian elimination. Table 10.4 presents the absolute deviations of the selected performance measures when using GMRES instead of the Gaussian elimination in samples B and D. For each performance measure, Table 10.4 gives the minimum, the maximum, and the average value, as well as the 2.5%- and the 97.5%-quantile of the absolute deviations.

The absolute deviation of α-service level is between -1.77E-04 and 1.2E-03 in sample B and between -2.43E-04 and 1.11E-04 in sample D. The corresponding 95%-confidence interval is given by [-7.26E-05, 4.06E-05] in sample B and [-1.04E-04, 3.57E-05] in sample D. The absolute deviations of β-service level, the expected number of unprocessed orders, and the expected number of completed orders are of comparable magnitude. Note that the calculated deviations only

Table 10.4: Deviation of selected performance measures when using GMRES instead of the Gaussian elimination.

		Absolute deviation[1]				
	Criterion	Min	Max	Average	$Q_{0.025}$	$Q_{0.975}$
Sample B	SL_α	-1.77E-04	1.20E-03	2.66E-06	-7.26E-05	4.06E-05
	SL_β	-1.89E-04	8.21E-04	1.23E-06	-6.71E-05	3.30E-05
	$E(Q)$	-1.99E-02	2.23E-03	-8.30E-06	-6.34E-04	1.45E-03
	$E(F)$	-3.34E-04	2.37E-04	2.12E-05	-3.12E-05	1.17E-04
Sample D	SL_α	-2.43E-04	1.11E-04	-1.06E-05	-1.04E-04	3.57E-05
	SL_β	-1.36E-04	8.52E-05	-8.22E-06	-7.78E-05	2.35E-05
	$E(Q)$	-6.83E-04	1.74E-03	9.51E-05	-1.30E-04	7.28E-04
	$E(F)$	-3.26E-05	1.48E-04	1.30E-05	-1.15E-05	7.49E-05

[1] Difference between the value of the considered performance measure resulting when using GMRES and the one resulting when using the Gaussian elimination.

quantify the deviations obtained when using two different methods to solve linear systems that both suffer from inaccuracies. The actual values of the performance measures are still unknown. In conclusion, we state the observed deviations to be negligible for all considered performance measures, and we recommend using GMRES instead of the Gaussian elimination to speed up the calculation of the Markov chain.

10.2.3 Parallel Computing

In Section 10.1.4, we pointed out some calculation steps of the Markov chain that can be conducted in parallel. To quantify the savings in computation time of parallel computing, we compare the computation time required to calculate the exact model for performance analysis in sequential manner with the one when calculating the exact model in parallel.

In sample B, computation time when using parallel computing is between 0.05 and 6.5 seconds, and it is 3.5 seconds on average. Compared to the computational effort of sequential computing, we observe an average reduction in computation time of

95%. For one data point in sample B, we observe a higher computation time for parallel computing (0.05 seconds) than for sequential computing (0.03 seconds). In sample D, computation time when using parallel computing is between 0.3 and 4.7 seconds, and it is 2.6 seconds on average. In comparison to the computational effort of sequential computing, we observe an average reduction in computation time of 95% (see Table 10.3). The observed savings in computation time when calculating the Markov chain in parallel confirm the potential of parallel computing for runtime optimisation of the Markov chain.

The outlier in sample B, for which the computation time of parallel computing is higher than the one of sequential computing, indicates that parallel computing is only beneficial for sufficiently large problem instances. For small problem instances, sequential computing can be quicker than parallel computing since the time required for starting and accumulating the threads in parallel computing exceeds the time saved by parallel computing. In conclusion, we recommend using parallel computing to speed up the calculation of the Markov chain.

10.3 Chapter Conclusion

Runtime optimisation and efficient memory usage become challenging issues when modelling system behaviour of real-life order fulfilment systems as a discrete-time Markov chain since the size of the state space of the Markov chain quickly increases. For this reason, we proposed and analysed different strategies for runtime and memory optimisation of the Markov chain in this chapter.

By the proposed alternative procedure to calculate the set of reachable states of a Markov chain based on a given initial state, we achieve an average reduction in number of calculated states by 43.6% for single-stage order fulfilment systems and 99.8% for two-stage order fulfilment systems. A further strategy to reduce memory usage of the Markov chain is to use a sparse storage scheme to store the transition matrix of the Markov chain since transition matrices are usually sparse.

To reduce the computation time of the Markov chain, we recommend using the indirect method GMRES to solve linear systems and computing the Markov chain in parallel. By using GMRES instead of the Gaussian elimination, we achieve an average reduction in computation time by 46% for single-stage order fulfilment systems and 49% for two-stage order fulfilment systems. An analysis regarding the result accuracy when using GMRES indicates that the absolute deviations of the values of the performance measures are negligible. By calculating the Markov chain in parallel, we achieve an average reduction in computation time of 95% compared to a sequential computation. Parallel computing is limited to the available processors of the CPU. The additional use of GPUs is beyond the scope of this thesis. However, we expect the calculation of the Markov chain to further speed up when using GPUs.

11 Evaluation of Capacity Planning Algorithms

This chapter aims at fine-tuning and evaluating the solution algorithms proposed in Chapter 9 for capacity planning in order fulfilment systems: *Mesh Adaptive Direct Search* (MADS) and *Surrogate Optimisation Integer* (SO-I). In Section 11.1, we specify the evaluation criteria. Both the MADS algorithm and the SO-I algorithm provide multiple algorithmic parameters for a problem-specific configuration of the algorithm. In Sections 11.2 and 11.3, we evaluate and compare several parameter settings of both algorithms to identify the most suitable parameter setting for the capacity planning problem, respectively. In Section 11.4, we evaluate and compare the overall performance of the algorithms. Based on the results of this chapter, Section 11.5 provides recommendations which algorithm to use for capacity planning in order fulfilment systems.

11.1 Evaluation Criteria

We use solution quality and runtime efficiency as evaluation criteria for the numerical analysis and comparison of both different parameter settings of an algorithm and different algorithms based on a given sample of data points in the subsequent sections.

11.1.1 Solution Quality

We evaluate the solution quality of an algorithm based on the objective function value of its solution (see equation (9.5)). Since the true minimum objective function value is unknown, we use the smallest objective function value of the solutions of all considered algorithms as an estimate of the minimum objective function value. To quantify the solution quality of an algorithm, we calculate the following key figures:

- The *share of suboptimal data points* corresponds to the proportion of data points for which the objective function value of the solution of the algorithm deviates from the minimum objective function value.
- The *average deviation from optimum* is the average value of the absolute deviations of the objective function value of the solution of the algorithm from the minimum objective function value for all suboptimal data points.

11.1.2 Runtime Efficiency

We evaluate the runtime efficiency of an algorithm based on the number of calculated blackbox instances since in the case of capacity planning, the calculation of a blackbox instance, which corresponds to the calculation of the analytical model for performance analysis, is the most computationally expensive step of the algorithm. To quantify the runtime efficiency of an algorithm, we calculate the following key figures:

- The *average number of blackbox instances* is the average value of the number of calculated blackbox instances per data point for all data points. The number of calculated blackbox instances per data point refers to the number of blackbox instances the algorithm has to calculate per data point until a solution is found.
- The *average rank of runtime efficiency* is the average rank of the algorithm regarding the number of calculated blackbox instances per data point. We

initially calculate the rank of each algorithm regarding the absolute number of calculated blackbox instances per data point. Subsequently, we obtain the average rank of runtime efficiency by calculating the average value of the rank values for all data points.

The average rank of runtime efficiency is a meaningful additional key figure since it abstracts from the differences in the absolute number of calculated blackbox instances between the data points of the considered sample. The smaller the average number of blackbox instances and the smaller the average rank of runtime efficiency of an algorithm, the higher the runtime efficiency of this algorithm.

11.2 Fine-Tuning of the MADS Algorithm

The MADS algorithm provides a plethora of algorithmic parameters enabling a problem-specific configuration of the algorithm. We can determine some of them based on the results of numerical studies provided in the literature (see Section 9.4.3). However, further algorithmic parameters require a closer examination in a problem-specific numerical analysis. In the following, we initially determine the algorithmic parameters to investigate in the fine-tuning of the MADS algorithm (see Section 11.2.1). Subsequently, we evaluate and compare the resulting parameter settings to identify the most suitable parameter setting of the MADS algorithm for the capacity planning problem (see Section 11.2.2).

11.2.1 Investigated Algorithmic Parameters

The fine-tuning of the MADS algorithm focuses on different approaches to choose the initial point and different methods to use in the search step. Regarding the choice of the initial point, we either calculate the initial point based on equation (4.1) proposed in the step of system parametrisation of the levelling concept (see Section 4.3.1), or we use the best point of a Latin hypercube design whose sample size corresponds to 25% of the total number of points within the

variable domain $\mathcal{C}$ of the capacity planning problem. Furthermore, we investigate using a surrogate model both as an additional method in the search step and to order the set of trial points in the search and the poll step (see Section 9.4.2.3). We consider five different surrogate functions: A quadratic model, a polynomial response surface, a radial basis function, a kernel smoothing model, and a Kriging model. Moreover, we investigate using the Nelder-Mead algorithm as an additional method in the search step.

Table 11.1 provides an overview of the values of the algorithmic parameters to investigate in the fine-tuning of the MADS algorithm. In total, we consider 24 different parameter settings of the MADS algorithm.

Table 11.1: Algorithmic parameters and their values investigated in the fine-tuning of the MADS algorithm.

	Value of algorithmic parameter	
Algorithmic parameter	Notation	Description
Choice of initial point	SP	Initial point is calculated based on equation (4.1)
	LHD	Initial point corresponds to best point of a Latin hypercube design with a sample size of $0.25 \cdot \lvert\mathcal{C}\rvert$
Use of surrogate model	SMn	No surrogate model is used
	SMy-QM	Use of a quadratic model
	SMy-PRS	Use of a polynomial response surface of degree two
	SMy-RBF	Use of a radial basis function with Gaussian kernel and l_1-norm
	SMy-KSM	Use of a kernel smoothing model with Gaussian kernel and l_1-norm
	SMy-KM	Use of a Kriging model with l_1-norm
Use of Nelder-Mead algorithm	NMn	No use of Nelder-Mead algorithm
	NMy	Use of Nelder-Mead algorithm

11.2.2 Numerical Results

We conduct a numerical analysis based on sample G to evaluate and compare the parameter settings of the MADS algorithm shown in Table 11.1 regarding solution quality and runtime efficiency. Sample G consists of 200 two-stage order fulfilment systems. A comprehensive description of sample G is given in Appendix C.3. The performance requirement of the capacity planning problem is specified by a predefined, individual value of α-service level. The results of the numerical analysis are given in Table 11.2.

For parameter settings SP/SMn/NMn and SP/SMy-QM/NMy, we obtain one unsolvable data point, respectively. Apart from this, the considered parameter settings do not differ regarding solution quality. Independent of the considered parameter setting, the MADS algorithm determines the optimal solution for all data points of sample G.

However, we observe differences regarding runtime efficiency between the considered parameter settings of the MADS algorithm. The average number of blackbox instances is between 20.19 and 24.03 for parameter settings using SP, whereas it is between 30.23 and 31.94 for parameter settings using LHD. The average rank of runtime efficiency is between 4.33 and 9.56 for parameter settings using SP and between 15.69 and 17.69 for parameter settings using LHD. Thus, using a Latin hypercube design to determine the initial point has a negative impact on the runtime efficiency. In contrast, regarding the use of Nelder-Mead algorithm and the use of a surrogate model, we do not observe any systematic impact on the runtime efficiency. We observe SP/SMy-QM/NMn (20.19), SP/SMn/NMn (20.61), and SP/SMy-QM/NMy (20.76) to be the best parameter settings regarding the average number of blackbox instances. SP/SMy-PRS/NMn (4.33), SP/SMy-KM/NMn (4.34), and SP/SMy-QM/NMn (4.62) are the parameter settings with the smallest average rank of runtime efficiency. Furthermore, LHD/SMy-KM/NMn (31.71; 17.12), LHD/SMy-RBF/NMn (31.7; 17.22), and LHD/SMy-PRS/NMn (31.94; 17.69) are the parameter settings with the highest average number of blackbox instances and the highest average rank of runtime efficiency, respectively.

Table 11.2: Solution quality and runtime efficiency of different parameter settings of the MADS algorithm based on sample G.

Parameter setting			Solution quality		Runtime efficiency	
			Share of sub-optimal data points [%]	Average deviation from optimum	Average number of blackbox instances	Average rank of runtime efficiency
LHD	SMn	NMn	0.0	0.0	30.55	16.28
LHD	SMn	NMy	0.0	0.0	31.33	17.08
LHD	SMy-KM	NMn	0.0	0.0	31.71	17.11
LHD	SMy-KM	NMy	0.0	0.0	30.94	16.51
LHD	SMy-KSM	NMn	0.0	0.0	31.43	16.87
LHD	SMy-KSM	NMy	0.0	0.0	30.74	16.12
LHD	SMy-PRS	NMn	0.0	0.0	31.94	17.69
LHD	SMy-PRS	NMy	0.0	0.0	30.93	16.75
LHD	SMy-QM	NMn	0.0	0.0	30.33	15.69
LHD	SMy-QM	NMy	0.0	0.0	30.62	16.26
LHD	SMy-RBF	NMn	0.0	0.0	31.70	17.22
LHD	SMy-RBF	NMy	0.0	0.0	30.83	16.32
SP	SMn	NMn	0.0	0.0	20.61	4.84
SP	SMn	NMy	0.0	0.0	20.95	5.54
SP	SMy-KM	NMn	0.0	0.0	20.94	4.34
SP	SMy-KM	NMy	0.0	0.0	23.23	8.49
SP	SMy-KSM	NMn	0.0	0.0	21.60	5.16
SP	SMy-KSM	NMy	0.0	0.0	23.78	9.34
SP	SMy-PRS	NMn	0.0	0.0	21.16	4.33
SP	SMy-PRS	NMy	0.0	0.0	23.56	8.83
SP	SMy-QM	NMn	0.0	0.0	20.19	4.62
SP	SMy-QM	NMy	0.0	0.0	20.76	5.35
SP	SMy-RBF	NMn	0.0	0.0	21.53	4.98
SP	SMy-RBF	NMy	0.0	0.0	24.03	9.56

These results indicate that concerning the capacity planning problem, parameter settings using SP should be preferred over the ones using LHD. In conclusion, we recommend the parameter setting SP/SMy-QM/NMn of the MADS algorithm for solving the capacity planning problem in order fulfilment systems since the optimal solution is found for all data points of sample G when using this parameter

setting. Furthermore, this parameter setting is the only parameter setting that belongs to the three best parameter settings regarding both evaluation criteria of runtime efficiency.

11.3 Fine-Tuning of the SO-I Algorithm

The implementation of the SO-I algorithm in the MATLAB-toolbox MATSuMoTo provides some algorithmic parameters for a problem-specific configuration of the SO-I algorithm. Compared to the configuration options of the MADS algorithm, the ones of the SO-I algorithm are rather limited. In the following, we initially determine the algorithmic parameters to investigate in the fine-tuning of the SO-I algorithm (see Section 11.3.1). Subsequently, we evaluate and compare the resulting parameter settings to identify the most suitable parameter setting of the SO-I algorithm for the capacity planning problem (see Section 11.3.2).

11.3.1 Investigated Algorithmic Parameters

The SO-I algorithm uses a Latin hypercube design to create the initial set of points in the design phase. We determine its sample size by 25% of the total number of points within the variable domain $\mathcal{C}$ of the capacity planning problem. A predefined maximum number of blackbox model evaluations is one of the stopping criteria of the SO-I algorithm. We determine the maximum number of blackbox evaluations by the total number of points within the variable domain $\mathcal{C}$ of the capacity planning problem.

The fine-tuning of the SO-I algorithm focuses on different sampling strategies and different surrogate functions. Regarding the sampling strategy that determines the finite number of trial points in the search phase, we either perturb the best feasible point found so far and uniformly select integer points in the variable domain, or we select the local minimum of the surrogate model as a new candidate point.

Regarding the surrogate function, we use either a cubic radial basis function or a quadratic polynomial response surface.

Table 11.3 provides an overview of the values of the algorithmic parameters to investigate in the problem-specific fine-tuning of the SO-I algorithm. In total, we consider four different parameter settings of the SO-I algorithm.

Table 11.3: Algorithmic parameters and their values investigated in the fine-tuning of the SO-I algorithm.

	Value of algorithmic parameter	
Algorithmic parameter	Notation	Description
Sampling strategy	CANDglob	Perturb the best feasible point found so far and uniformly select integer points in the variable domain
	SurfMin	Select the local minimum of the surrogate model as new candidate point
Surrogate function	RBF	Cubic radial basis function with l_2-norm
	PRS	Quadratic polynomial response surface

11.3.2 Numerical Results

We conduct a numerical analysis based on sample G (see Appendix C.3) to evaluate and compare the parameter settings of the SO-I algorithm shown in Table 11.3 regarding solution quality and runtime efficiency. The results of the numerical analysis are given in Table 11.4.

We obtain the optimal solution for all data points of sample G when using the sampling strategy CANDglob, independent of the selected surrogate function. In contrast, when using the sampling strategy SurfMin, we obtain a suboptimal solution for some data points of sample G. The share of suboptimal data points is 24% for parameter setting SurfMin/PRS and 1.5% for parameter setting SurfMin/RBF.

Table 11.4: Solution quality and runtime efficiency of different parameter settings of the SO-I algorithm based on sample G.

Parameter setting		Solution quality		Runtime efficiency	
		Share of suboptimal data points [%]	Average deviation from optimum	Average number of blackbox instances	Average rank of runtime efficiency
CANDglob	PRS	0.0	0.0	62.74	2.49
CANDglob	RBF	0.0	0.0	58.36	1.78
SurfMin	PRS	24.0	1.0	58.32	1.98
SurfMin	RBF	1.5	1.0	61.42	1.85

For both parameter settings, the corresponding average deviation from optimum equals one. These results indicate that using the sampling strategy SurfMin has a negative impact on the solution quality of the SO-I algorithm when solving the capacity planning problem.

Regarding runtime efficiency, we observe SurfMin/PRS (58.32) and CANDglob/RBF (58.36) to be the best parameter settings regarding the average number of blackbox instances. CANDglob/RBF (1.78) and SurfMin/RBF (1.85) are the parameter settings with the smallest average rank of runtime efficiency.

In conclusion, we recommend not to use parameter settings SurfMin/PRS and SurfMin/RBF of the SO-I algorithm due to the occurrence of suboptimal data points. Instead, we state parameter setting CANDglob/RBF to be the most suitable parameter setting of the SO-I algorithm for capacity planning since it outperforms parameter setting CANDglob/PRS in both evaluation criteria of runtime efficiency.

11.4 Comparison of the MADS Algorithm and the SO-I Algorithm

In the following, we evaluate and compare the overall performance of the MADS algorithm and the SO-I algorithm to solve the capacity planning problem in order fulfilment systems in a numerical analysis based on sample G (see Appendix C.3). For this, we focus on the most suitable parameter setting of both algorithms for solving the capacity planning problem, respectively: Parameter setting SP/SMy-QM/NMn of the MADS algorithm and parameter setting CANDglob/RBF of the SO-I algorithm. We use the complete enumeration as a benchmark for the algorithms. The complete enumeration identifies the optimal solution of the capacity planning problem by investigating every point in the variable domain $\mathcal{C}$.

We concentrate on a comparison regarding runtime efficiency since the results of Sections 11.2 and 11.3 show that the MADS algorithm and the SO-I algorithm do not differ regarding solution quality. The selected parameter settings of both algorithms identify the optimal solution for every data point of sample G.

Table 11.5 presents the minimum, the maximum, and the average value of the number of calculated blackbox instances per data point of the MADS algorithm, the SO-I algorithm, and the complete enumeration in sample G. Furthermore, Table 11.5 gives the minimum, the maximum, and the average value of the absolute and the relative reduction in the number of calculated blackbox instances when using the MADS algorithm compared to the complete enumeration, the SO-I algorithm compared to the complete enumeration, and the MADS algorithm compared to the SO-I algorithm.

The number of calculated blackbox instances per data point varies between 9 and 42 blackbox instances when using the MADS algorithm and between 49 and 72 blackbox instances when using the SO-I algorithm. The average number of calculated blackbox instances per data point is 20.19 for the MADS algorithm and 58.36 for the SO-I algorithm. The complete enumeration calculates 72 blackbox instances for every data point since the variable domain $\mathcal{C}$ of the capacity planning problem is given by $\mathcal{C} = \{1, 2, \ldots, 6\} \times \{1, 2, \ldots, 12\}$ (see equation (9.4)) for

Table 11.5: Comparison of the MADS algorithm and the SO-I algorithm with complete enumeration regarding runtime efficiency based on sample G.

Criterion	Algorithm	Min	Max	Average
Number of calculated blackbox instances per data point	MADS algorithm	9	42	20.19
	SO-I algorithm	49	72	58.36
	Complete enumeration	72	72	72.00
Absolute reduction in number of calculated blackbox instances	MADS compared to complete enumeration	63	30	51.82
	SO-I compared to complete enumeration	23	0	13.65
	MADS compared to SO-I	63	15	38.17
Relative reduction in number of calculated blackbox instances [%]	MADS compared to complete enumeration	87.50	41.67	71.97
	SO-I compared to complete enumeration	31.94	0.00	18.95
	MADS compared to SO-I	87.50	26.32	65.04

every data point of sample G. Thus, the total number of points within the variable domain is 72 points.

By using the MADS algorithm instead of complete enumeration, we observe an average reduction in the number of calculated blackbox instances per data point by 51.82 blackbox instances. The corresponding relative reduction is between 42% and 88%, and it is 72% on average. By using the SO-I algorithm, the number of calculated blackbox instances per data point reduces by on average 13.65 blackbox instances compared to complete enumeration. The corresponding relative reduction is on average 19% and at most 32%. For 12% of the data points of sample G, the number of calculated blackbox instances per data point of the SO-I algorithm corresponds to the total number of points within the variable domain. Thus, for these data points, the SO-I algorithm provides no benefit in runtime efficiency compared to complete enumeration. Furthermore, we observe an average reduction in the number of calculated blackbox instances per data point by 38.2 blackbox instances when using the MADS algorithm instead the SO-I algorithm. The corresponding relative reduction is between 26% and 88%.

In conclusion, these results indicate that the MADS algorithm has a remarkable higher runtime efficiency compared to the SO-I algorithm and complete enumeration. Consequently, we recommend using the MADS algorithm to solve the capacity planning problem in order fulfilment systems.

11.5 Chapter Conclusion

In this chapter, we investigated multiple settings of algorithmic parameters of the MADS algorithm and the SO-I algorithm regarding solution quality and runtime efficiency in a numerical analysis to identify the most suitable parameter setting of each algorithm for capacity planning in order fulfilment systems. Furthermore, we evaluated and compared the overall performance of the MADS algorithm and the SO-I algorithm regarding solution quality and runtime efficiency.

We identify the parameter settings SP/SMy-QM/NMn and CANDglob/RBF to be the most suitable parameter settings of the MADS algorithm and the SO-I algorithm for solving the capacity planning problem, respectively. The MADS algorithm and the SO-I algorithm do not differ regarding solution quality since their most suitable parameter setting finds the optimal solution for all considered data points. However, the MADS algorithm has a remarkable higher runtime efficiency than the SO-I algorithm (average reduction in the number of calculated blackbox instances per data point by 65%) and complete enumeration (average reduction in the number of calculated blackbox instances per data point by 72%). Consequently, we recommend using the parameter setting SP/SMy-QM/NMn of the MADS algorithm for solving the capacity planning problem.

12 Evaluation of the Strategy of Levelled Order Release

This chapter aims at evaluating the *Strategy of Levelled Order Release* in multi-stage, stochastic order fulfilment systems with customer-required order deadlines by comparing its performance with the one of an alternative strategy. A suitable alternative strategy is selected in Section 12.1. In Section 12.2, we derive several hypotheses on the expected behaviour of the strategies to compare. The models developed in this thesis enable an evaluation of the *Strategy of Levelled Order Release* regarding resulting system performance (see Section 12.3) as well as regarding required capacity (see Section 12.4). Section 12.5 summarises the results of this chapter.

12.1 Alternative Strategies

The main characteristics of the *Strategy of Levelled Order Release* are (1) that a fixed capacity is reserved for order processing in each time period, and (2) that the orders are processed according to ascending due dates in each time period (see Chapter 4). Thus, the *Strategy of Levelled Order Release* comprises two different planning problems: (1) Capacity planning and (2) order dispatching. It is difficult to find suitable alternative strategies to evaluate the *Strategy of Levelled Order Release* since related research fields only focus on one of these planning problems.

Furthermore, to ensure comparability, the alternative strategies used to evaluate the *Strategy of Levelled Order Release* have to rely on the same amount of available information as the *Strategy of Levelled Order Release*:

- Order income per time period $\mathbf{G}$ and processing performance per time period $\mathbf{H}$ are not known in advance. Instead, they are given as stochastic parameters with known probability distributions.
- Order income per time period $\mathbf{G}$ and processing performance per time period $\mathbf{H}$ are independent of each other.
- The order backlog of the current time period and the characteristics of these orders are known.

In the following, we investigate the research fields of dispatching (see Section 12.1.1), scheduling (see Section 12.1.2), and flexible capacity adaption (see Section 12.1.3) to identify suitable alternative strategies for evaluating the *Strategy of Levelled Order Release*. The selected strategies are specified in Section 12.1.4.

12.1.1 Dispatching Policies

A *dispatching policy* specifies a rule used to select the next order to be processed from a set of orders awaiting service (Blackstone et al. 1982). Blackstone et al. (1982), Pinedo (2009, p.442-444), and Baumann (2019, p.22-24) provide comprehensive reviews on dispatching policies in production systems.

Due to the limited focus of dispatching policies, the comparison of the *Strategy of Levelled Order Release* with any dispatching policy is limited to the planning problem of order dispatching. A comparison with respect to capacity planning is not possible. Instead, any dispatching policy selected to evaluate the *Strategy of Levelled Order Release* has to be combined with the capacity planning approach of the *Strategy of Levelled Order Release* in order to ensure comparability.

Dispatching policies are classified into the following categories:

1. Processing-related dispatching policies,
2. Due date-related dispatching policies, and
3. Dispatching policies based on other characteristics.

Processing-related dispatching policies select the next order to be processed based on either the number of processing steps or the processing time. These policies are further differentiated regarding the considered metric – total sum or remaining sum – and the used sort order – ascending or descending. Common examples of this category are *Smallest number of remaining operations*, *Highest number of total operations*, *Shortest remaining processing time*, and *Longest total processing time*. The category of due-date related dispatching policies contains the policies *Earliest due date* and *Minimum slack time*. The slack time of an order specifies the time period that remains until reaching its due date and that is not required for its processing. The third category summarises further dispatching policies that use other characteristics to select the next order to be processed. Common examples of this category are *Random service*, *First come first serve*, and *Greatest dollar first* (Baumann 2019, p.22-24).

Processing-related dispatching policies implicitly assume each order to have a priori known, deterministic, and individual processing times and an individual processing sequence. In contrast, in order fulfilment systems, the processing performance per time period is stochastic, and it is specified by an order type- and processing step-specific probability distribution (see Section 3.2.3). Furthermore, all orders of a particular order type have the same processing sequence (see Section 3.2.2). Thus, orders of the same order type differ neither regarding processing time nor regarding processing sequence in order fulfilment systems. Since we investigate different order types separately in this thesis (see Section 5.3.1), both the processing time and the number of processing steps are no suitable criteria to select the next order to be processed in the order fulfilment system. Consequently, processing-related dispatching policies are unsuitable alternative strategies to evaluate the *Strategy of Levelled Order Release*.

The due date-related dispatching policy *Earliest due date* selects the next order to be processed based on its due date, analogous to the *Strategy of Levelled Order Release*. The dispatching policy *Minimum slack time* corresponds to *Earliest due date* in the context of order fulfilment due to the lack of order-specific processing times and an order-specific processing sequence. Consequently, both dispatching

policies are unsuitable alternative strategies to evaluate the *Strategy of Levelled Order Release*.

Regarding the third category of dispatching policies, *Greatest dollar first* is an unsuitable alternative strategy since we do not model revenue and cost aspects of order fulfilment systems. However, *Random service* and *First come first serve* are suitable alternative strategies to evaluate the *Strategy of Levelled Order Release*.

12.1.2 Scheduling

Scheduling is a decision-making process used in many manufacturing and service industries that allocates resources, such as machines and workers, to orders over a given planning horizon to optimise one or multiple objectives (Pinedo 2016, p.1). Scheduling is conducted based on the following information: A given finite set of orders, a given finite set of resources, the processing time of each order at each resource, and the release date, the due date, and the importance factor of each order (Pinedo 2016, p.13f.). In deterministic scheduling, processing times, release times, and due dates are deterministic, whereas, in stochastic scheduling, these parameters are stochastic, and scheduling is conducted based on the characteristics of their probability distributions (Pinedo 2016, p.246).

The comparison of the *Strategy of Levelled Order Release* with any scheduling problem is limited to the planning problem of order dispatching. A comparison with respect to capacity planning is not possible. Instead, any scheduling problem selected to evaluate the *Strategy of Levelled Order Release* has to be combined with the capacity planning approach of the *Strategy of Levelled Order Release* in order to ensure comparability.

In stochastic scheduling, processing time, release date, and due date of each order are defined by individual probability distributions, respectively. In contrast, in order fulfilment systems, the processing performance per time period is specified by an order type- and processing step-specific probability distribution (see Section 3.2.3), and the lead time is defined by an order type-specific probability

distribution (see Section 3.2.2). Thus, orders of the same order type differ neither regarding processing times nor regarding lead time. Since we investigate different order types separately in this thesis (see Section 5.3.1), both processing time and lead time are unsuitable criteria to schedule the orders in an order fulfilment system. Consequently, stochastic scheduling is an unsuitable alternative strategy to evaluate the *Strategy of Levelled Order Release*.

12.1.3 Strategy of Flexible Capacity Adaption

The *Strategy of Levelled Order Release* is characterised by a constant capacity per time period. In contrast, the main idea of the *Strategy of Flexible Capacity Adaption* is to match provided capacity as precisely as possible to demanded capacity resulting from the current system workload by hiring and laying-off workers (Chen et al. 2009).

There are different possible configurations of the *Strategy of Flexible Capacity Adaption* regarding the degree of capacity flexibility. In the case of an instantaneous capacity adaption, capacity levels are adapted at the beginning of each time period when order income and processing performance of the current time period are already known. Hence, provided capacity always fits exactly to demanded capacity. However, an instantaneous capacity adaption does not meet the assumptions of the *Strategy of Levelled Order Release* that order income and processing performance per time period are unknown in advance. Thus, it is an unsuitable alternative strategy to evaluate the *Strategy of Levelled Order Release*. Furthermore, periodic capacity adaptions are more realistic than instantaneous capacity adaptions for several reasons: First, working time decisions (working over/under time) are often taken periodically to abide to labour regulations and to accomplish the timely communication of these decisions to the relevant workers. Second, deployment of temporary workers may be restricted to specific times, such as the start of a day or the start of a week (Buyukkaramikli et al. 2013).

Independent of the chosen degree of flexibility, the *Strategy of Flexible Capacity Adaption* concentrates on capacity planning. The second planning problem of

the *Strategy of Levelled Order Release* on order dispatching is not considered. Furthermore, the *Strategy of Flexible Capacity Adaption* leads to additional operational costs for hiring and laying off permanent and temporary workers and for other organisational efforts of capacity adaption. Cost aspects are neglected in our models of order fulfilment systems, but the evaluation of the *Strategy of Levelled Order Release* is restricted to the performance measures of an order fulfilment system (see Table 6.1). Consequently, in the context of this thesis, the *Strategy of Flexible Capacity Adaption* is an unsuitable alternative strategy since its flexibility costs cannot be incorporated into the comparison.

12.1.4 Selected Alternative Strategy

Although none of the related research fields provides any planning approach that incorporates both capacity planning and order dispatching, we identify the dispatching strategies *Random Service* and *First come first serve* as suitable alternative strategies for evaluating the *Strategy of Levelled Order Release*. *Random Service* randomly selects the next order to be processed. *First come first serve* selects the next order to be processed based on the arrival times of the orders.

In the following, we focus on *First come first serve (FCFS)* to evaluate the *Strategy of Levelled Order Release (LOR)* since *FCFS* is widely used in production and logistics systems. Note that this comparison is limited to the planning problem of order dispatching. *FCFS* is combined with the capacity planning approach of *LOR* in order to ensure the comparability of both strategies.

12.2 Hypotheses

The models developed in this thesis provide two possibilities to compare *LOR* and *FCFS*: When comparing the strategies regarding the resulting system performance, we investigate the impact of the selected strategy on the system performance, measured by at least one performance measure (see Table 6.1), for

order fulfilment systems with a given capacity. Otherwise, when comparing the strategies regarding the required capacity, we analyse the impact of the selected strategy on the total required capacity in order fulfilment systems with predefined performance requirements.

In order fulfilment systems with low utilisation (see equation (8.1)), the order income per time period is the limiting factor of system throughput. The majority of incoming orders per time period is processed immediately after their time of arrival without being buffered at any processing step since there is enough idle capacity in systems with low utilisation. If orders are predominantly processed immediately after their time of arrival, their processing sequence has no impact on system throughput and the characteristics of the completed orders. These findings lead to the following hypothesis:

Hypothesis 7. *In order fulfilment systems with low utilisation, system performance is independent of the strategy that is used to select the next order to be processed.*

In contrast, in order fulfilment systems with high utilisation (see equation (8.2)), the processing performance per time period is the limiting factor of system throughput. A significant proportion of the incoming orders per time period is not processed immediately after their time of arrival, but the orders are buffered at some processing step of the multi-stage order fulfilment system before being processed. If it is impossible to process all incoming orders immediately after their time of arrival, a strategy is required to select the next order to be processed from the set of buffered orders. Thus, this strategy has an impact on system throughput and the characteristics of the completed orders.

FCFS does not consider the due dates of the orders, but the next order to be processed is selected based on its time of arrival. In contrast, *LOR* selects the order with the shortest due date as the next order to be processed. Hence, *LOR* tries to systematically avoid the occurrence of backorders. Its proportion of on-time processed orders is expected to be higher than the one of *FCFS*. α-service level measures the probability that none of the completed orders has a backlog duration,

and β-service level quantifies the proportion of on-time completed orders on the total number of outgoing orders (see Section 6.3.7). Hence, we formulate the following hypothesis:

Hypothesis 8. *In order fulfilment systems with high utilisation, the values of α- and β-service level achieved when using LOR are at least as high as the ones achieved when using FCFS.*

If it is possible to achieve higher values of α- and β-service level by using *LOR* instead of *FCFS*, as stated in hypothesis 8, this means the following in reverse with respect to capacity planning:

Hypothesis 9. *The total required capacity to guarantee predefined performance requirements concerning α- and β-service level in order fulfilment systems when using LOR is at most as high as the one when using FCFS.*

12.3 Comparison regarding Resulting System Performance

In the following, we conduct a numerical analysis based on samples A1, A2, C, and E to compare *LOR* and *FCFS* regarding the resulting system performance and to verify hypotheses 7 and 8. We focus on α- and β-service level for simplicity reasons. As evaluation criteria, we calculate the relative deviation of the values of α- and β-service level achieved when using *LOR* from the ones achieved when using *FCFS*, respectively.

We use a simulation model to model the performance analysis in multi-stage, stochastic order fulfilment systems with *FCFS*. The simulation model is based on the one used to verify the exact model for performance analysis (see Appendix B). The simulation models only differ regarding the modelling of order processing in the simulation iteration when the next order to be processed is selected based on its time of arrival and not based on its due date. β-service level is used to

determine the length of every simulation replication (precision of 1.0E-05) and the number of simulation replications (precision of 1.0E-04).

Samples A1 and A2 contain 120 and 350 single-stage order fulfilment systems, respectively. The ranges of the parameters are the same in both samples. Sample A2 focuses on order fulfilment systems with a utilisation of [0.5,1.0) since hypothesis 8 expects systems with high utilisation to be more relevant for the evaluation of *LOR*. Sample C contains 256 two-stage order fulfilment systems, and sample E consists of 353 three-stage order fulfilment systems. The samples differ regarding the ranges of the parameters, especially regarding the discretisation of the number of incoming orders per time period A: Its range is given by $\mathcal{A}^A = \{5, 6, \ldots, 40\}$ in samples A1 and A2, $\mathcal{A}^C = \{3, 4, \ldots, 14\}$ in sample C, and $\mathcal{A}^E = \{2, 3, \ldots, 6\}$ in sample E. A comprehensive description of the samples is given in Appendix C.2.

12.3.1 Analysis of α-Service Level

Table 12.1 presents the relative deviations of α-service level between *LOR* and *FCFS* in samples A1, A2, C, and E. Each sample is subdivided into multiple classes according to the system utilisation (see equation (6.7)). Table 12.1 gives the minimum, the maximum, the average value, the 5%-, and the 95%-quantile of the relative deviations for each class of utilisation as well as for the whole sample.

Table 12.1: Deviation of α-service level between the *Strategy of Levelled Order Release (LOR)* and *FCFS*.

		Relative deviation [%][1]					
	Utilisation	Min	Max	Average	$Q_{0.05}$	$Q_{0.95}$	No.
Sample A1	[0.1,0.2)	0.00	0.00	0.00	0.00	0.00	4
	[0.2,0.3)	0.00	0.00	0.00	0.00	0.00	14
	[0.3,0.4)	-0.01	0.18	0.01	-0.01	0.04	18
	[0.4,0.5)	0.00	0.19	0.03	0.00	0.16	17
	[0.5,0.6)	0.00	0.72	0.09	0.00	0.46	9
	[0.6,0.7)	0.00	8.74	1.49	0.00	5.26	14

		Relative deviation [%][1]					
	Utilisation	Min	Max	Average	$Q_{0.05}$	$Q_{0.95}$	No.
Sample A1	[0.7,0.8)	0.00	12.75	3.02	0.00	12.40	17
	[0.8,0.9)	0.00	40.91	6.95	0.00	28.88	15
	[0.9,1.0)	1.22	141.07	53.52	1.76	137.20	12
	[0.1,1.0)	-0.01	141.07	6.83	0.00	50.10	120
Sample A2	[0.5,0.55)	-0.01	1.18	0.13	-0.01	0.60	28
	[0.55,0.6)	-0.01	1.10	0.17	-0.01	0.96	39
	[0.6,0.65)	0.00	3.30	0.40	0.00	1.85	40
	[0.65,0.7)	-0.01	7.55	1.04	0.00	4.94	34
	[0.7,0.75)	-0.02	9.42	1.78	0.00	7.39	37
	[0.75,0.8)	-0.01	16.03	2.89	0.00	13.17	49
	[0.8,0.85)	0.00	26.95	10.24	0.00	22.52	35
	[0.85,0.9)	0.00	48.36	11.24	0.00	37.29	32
	[0.9,0.95)	0.00	70.66	20.82	0.00	56.00	31
	[0.95,1.0)	0.82	124.67	51.13	5.63	113.00	25
	[0.5,1.0)	-0.02	124.67	8.32	0.00	45.59	350
Sample C	[0.2,0.3)	0.00	0.00	0.00	0.00	0.00	5
	[0.3,0.4)	0.00	0.00	0.00	0.00	0.00	17
	[0.4,0.5)	0.00	0.03	0.00	0.00	0.01	27
	[0.5,0.6)	-0.01	0.63	0.06	0.00	0.36	39
	[0.6,0.7)	-0.01	1.81	0.13	0.00	0.42	45
	[0.7,0.8)	0.00	8.92	1.09	0.00	3.88	39
	[0.8,0.9)	0.00	21.45	3.91	0.01	11.37	51
	[0.9,1.0)	-0.07	74.04	21.37	0.12	62.51	33
	[0.2,1.0)	-0.07	74.04	3.73	0.00	25.47	256
Sample E	[0.3,0.4)	0.00	0.00	0.00	0.00	0.00	3
	[0.4,0.5)	-0.02	0.01	0.00	-0.01	0.00	14
	[0.5,0.6)	0.00	0.12	0.01	0.00	0.08	42
	[0.6,0.7)	-0.02	1.44	0.16	0.00	0.65	62
	[0.7,0.8)	0.00	4.96	1.21	0.02	3.25	85

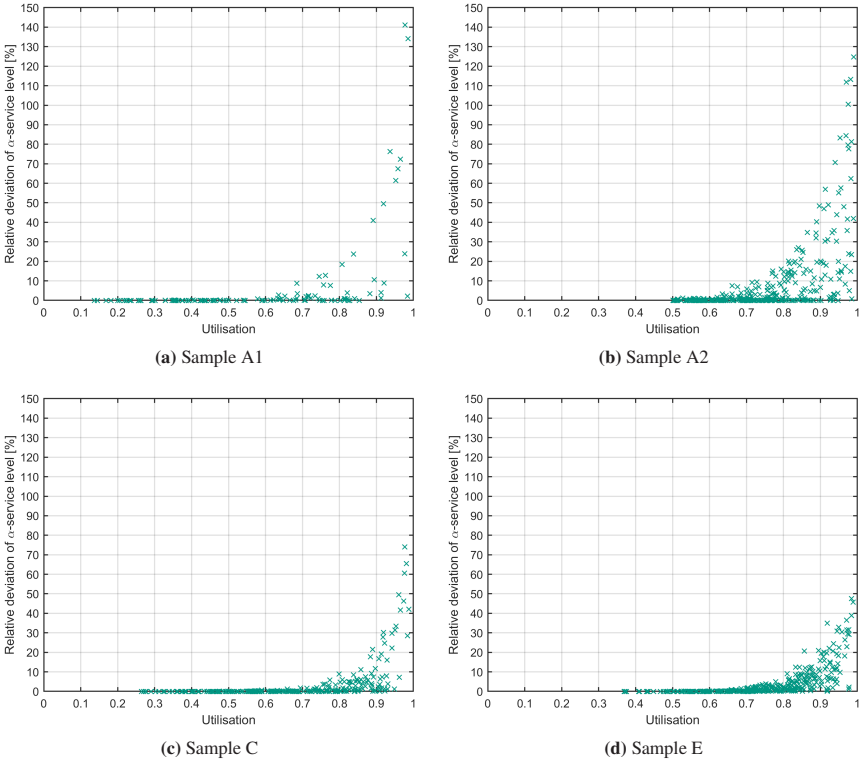
		Relative deviation [%][1]					
	Utilisation	Min	Max	Average	$Q_{0.05}$	$Q_{0.95}$	No.
Sample E	[0.8,0.9)	0.03	20.99	5.35	0.32	12.45	92
	[0.9,1.0)	0.03	47.53	16.49	1.02	37.18	55
	[0.3,1.0)	-0.02	47.53	4.29	0.00	20.80	353

[1] Difference between the value of α-service level when using *LOR* and the one when using *FCFS*, divided by the value of α-service level when using *FCFS*.

We observe three data points each in sample A1 and E, four data points in sample C, and nine data points in sample A2 to have a negative relative deviation. The maximum of the negative relative deviations is 0.01% in sample A1, 0.02% in samples A2 and E, and 0.07% in sample C. The corresponding maxima of the negative absolute deviations are smaller than 5.3E-04. Thus, we state the observed negative deviations to be negligible. They result from numerical errors during the calculation. Thus, for all data points of the considered samples, α-service level of *LOR* is at least as high as the one of *FCFS*. Furthermore, we observe a maximum relative deviation of 141% in sample A1, 125% in sample A2, 74% in sample C, and 48% in sample E. Thus, there are data points in every sample for which α-service level achieved when using *LOR* is significantly higher than the one achieved when using *FCFS*.

Figure 12.1 presents the relative deviations of α-service level between *LOR* and *FCFS* depending on the utilisation of the order fulfilment system in samples A1, A2, C, and E. It indicates that the utilisation of the order fulfilment system has a systematic impact on the relative deviations of α-service level since the relative deviations of α-service level increase as the utilisation increases.

In every sample, we observe a maximum relative deviation of at most 1.1% and an average relative deviation of at most 0.2% for systems with a utilisation smaller than 0.6. Consequently, for systems with a utilisation smaller than approximately 0.6, the deviations of α-service level between *LOR* and *FCFS* are negligible, and the strategy used to select the next order to be processed has a neglectable small impact on the resulting α-service level of the order fulfilment system.



(a) Sample A1 **(b)** Sample A2

(c) Sample C **(d)** Sample E

Figure 12.1: Deviation of α-service level between the *Strategy of Levelled Order Release (LOR)* and *FCFS*.

In contrast, in sample A1, the relative deviation is 141% at its maximum and 54% on average for systems with a utilisation of [0.9,1.0). In sample A2, we observe a maximum relative deviation of 71% and an average relative deviation of 21% for systems with a utilisation of [0.9,0.95), whereas the relative deviation is 125% at its maximum and 51% on average for systems with a utilisation of [0.95,1.0). In sample C, we observe a maximum relative deviation of 74% and an average relative deviation of 21% for systems with a utilisation of [0.9,1.0). In sample E, the relative deviation is 48% at its maximum and 16% on average for systems with a utilisation of [0.9,1.0). These results indicate that for systems with high

utilisation, the strategy used to select the next order to be processed has an impact on the resulting α-service level. For these systems, the α-service level achieved when using *LOR* can be significantly higher than the one when using *FCFS*.

By comparing the 90%-confidence intervals of the relative deviations of different classes of utilisation in each sample, we note that the width of the 90%-confidence interval increases as the utilisation of the order fulfilment systems increases. In every sample, the 90%-confidence intervals per class of utilisation are narrower than 1% for systems with a utilisation smaller than 0.6. In contrast, for systems with a utilisation of [0.9,1.0), the 90%-confidence interval is given by, for instance, [1.8%, 137%] in sample A1 and [1%, 37%] in sample E. These results indicate that the variance of the relative deviations of α-service level increases as the utilisation of the order fulfilment system increases.

By comparing the magnitude of the relative deviations of α-service level between the samples, we observe that the number of processes of the considered order fulfilment system has a systematic impact on the relative deviation of α-service level. The maximum relative deviation is 141% for the single-stage systems of sample A1, 74% for the two-stage systems of sample C, and 48% for the three-stage systems of sample E. These results indicate that the benefit achieved when using *LOR* instead of *FCFS* decreases as the number of processes of the considered order fulfilment system increases.

12.3.2 Analysis of β-Service Level

Table 12.2 presents the relative deviations of β-service level between *LOR* and *FCFS* in samples A1, A2, C, and E. Each sample is subdivided into multiple classes according to the system utilisation (see equation (6.7)). Table 12.2 gives the minimum, the maximum, the average value, the 5%-, and the 95%-quantile of the relative deviations for each class of utilisation as well as for the whole sample.

Table 12.2: Deviation of β-service level between the *Strategy of Levelled Order Release (LOR)* and *FCFS*.

		Relative deviation [%][1]					
	Utilisation	Min	Max	Average	$Q_{0.05}$	$Q_{0.95}$	No.
Sample A1	[0.1,0.2)	0.00	0.00	0.00	0.00	0.00	4
	[0.2,0.3)	0.00	0.00	0.00	0.00	0.00	14
	[0.3,0.4)	-0.01	0.04	0.00	-0.01	0.01	18
	[0.4,0.5)	0.00	0.02	0.00	0.00	0.02	17
	[0.5,0.6)	0.00	0.10	0.01	0.00	0.06	9
	[0.6,0.7)	0.00	1.37	0.18	0.00	0.71	14
	[0.7,0.8)	-0.01	1.65	0.33	0.00	1.57	17
	[0.8,0.9)	0.00	4.94	0.81	0.00	4.16	15
	[0.9,1.0)	0.04	6.17	2.50	0.05	6.11	12
	[0.1,1.0)	-0.01	6.17	0.42	0.00	3.25	120
Sample A2	[0.5,0.55)	-0.01	0.15	0.01	0.00	0.07	28
	[0.55,0.6)	-0.01	0.19	0.02	-0.01	0.14	39
	[0.6,0.65)	0.00	0.48	0.05	0.00	0.28	40
	[0.65,0.7)	-0.01	1.17	0.12	0.00	0.65	34
	[0.7,0.75)	-0.01	1.16	0.19	0.00	0.94	37
	[0.75,0.8)	-0.01	1.99	0.33	0.00	1.80	49
	[0.8,0.85)	0.00	4.49	1.13	0.00	3.31	35
	[0.85,0.9)	0.00	5.71	1.06	0.00	4.02	32
	[0.9,0.95)	-1.78	6.26	1.29	-0.35	5.29	31
	[0.95,1.0)	-6.62	6.07	1.15	-5.20	5.40	25
	[0.5,1.0)	-6.62	6.26	0.49	0.00	3.16	350
Sample C	[0.2,0.3)	0.00	0.00	0.00	0.00	0.00	5
	[0.3,0.4)	0.00	0.00	0.00	0.00	0.00	17
	[0.4,0.5)	0.00	0.01	0.00	0.00	0.00	27
	[0.5,0.6)	0.00	0.13	0.01	0.00	0.07	39
	[0.6,0.7)	-0.01	0.35	0.02	0.00	0.09	45
	[0.7,0.8)	0.00	1.99	0.20	0.00	0.73	39

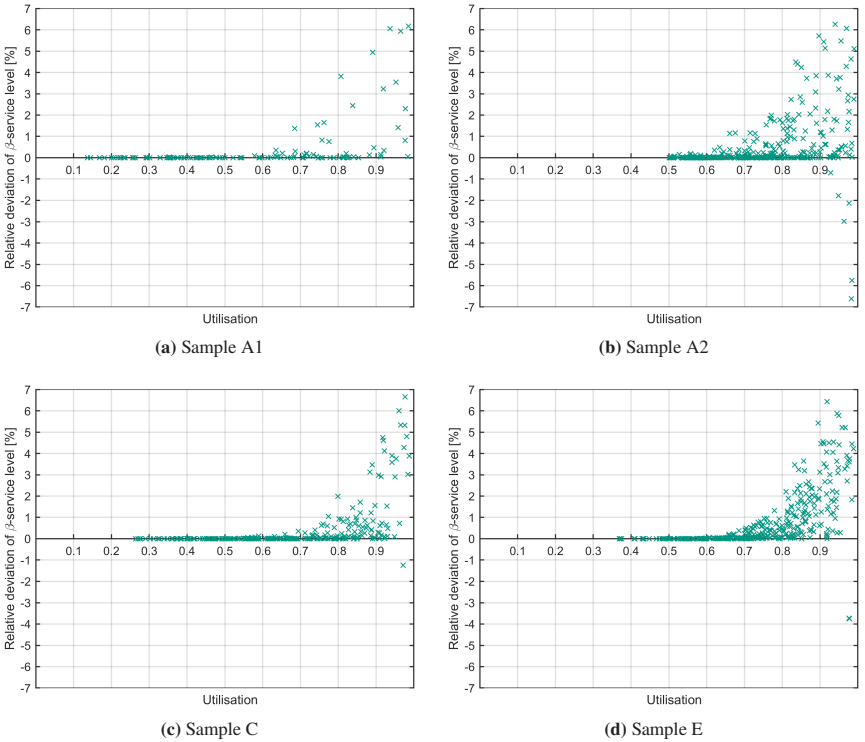
		Relative deviation [%][1]					
	Utilisation	Min	Max	Average	$Q_{0.05}$	$Q_{0.95}$	No.
Sample C	[0.8,0.9)	0.00	3.47	0.56	0.00	1.63	51
	[0.9,1.0)	-1.24	6.66	2.35	0.01	5.60	33
	[0.2,1.0)	-1.24	6.66	0.45	0.00	3.50	256
Sample E	[0.3,0.4)	0.00	0.00	0.00	0.00	0.00	3
	[0.4,0.5)	-0.01	0.00	0.00	0.00	0.00	14
	[0.5,0.6)	0.00	0.02	0.00	0.00	0.02	42
	[0.6,0.7)	0.00	0.38	0.04	0.00	0.17	62
	[0.7,0.8)	0.00	1.32	0.28	0.00	0.80	85
	[0.8,0.9)	0.01	5.42	1.23	0.06	3.16	92
	[0.9,1.0)	-3.76	6.43	2.54	0.07	5.38	55
	[0.3,1.0)	-3.76	6.43	0.79	0.00	3.68	353

[1] Difference between the value of β-service level when using *LOR* and the one when using *FCFS*, divided by the value of β-service level when using *FCFS*.

We observe a minimum relative deviation of -0.01% in all samples, except six data points in sample A2, whose relative deviation is between -7% and -0.7%, one data point in sample C with a relative deviation of -1.2%, and two data points in sample E with a relative deviation of -3.7%, respectively. Apart from these outliers that will be discussed in Section 12.3.3, we state the observed negative relative deviations to be negligible. They result from numerical errors during the calculation. Hence, we conclude that β-service level achieved when using *LOR* is at least as high as the one achieved when using *FCFS*. Furthermore, we observe a maximum relative deviation of approximately 6% in every sample. Thus, there are data points in every sample for which β-service level achieved when using *LOR* is significantly higher than the one achieved when using *FCFS*.

Figure 12.2 presents the relative deviations of β-service level between *LOR* and *FCFS* depending on the utilisation of the order fulfilment system in samples A1, A2, C, and E. It indicates that the utilisation of the order fulfilment system has a

systematic impact on the relative deviations of β-service level since the relative deviations of β-service level increase as the utilisation increases.



(a) Sample A1 **(b)** Sample A2

(c) Sample C **(d)** Sample E

Figure 12.2: Deviation of β-service level between the *Strategy of Levelled Order Release (LOR)* and *FCFS*.

In every sample, we observe a maximum relative deviation of at most 0.2% and an average relative deviation of at most 0.02% for systems with a utilisation smaller than 0.6. Consequently, for systems with a utilisation smaller than approximately 0.6, the deviations of β-service level between *LOR* and *FCFS* are negligible, and the strategy used to select the next order to be processed has a neglectable small impact on the resulting β-service level of the order fulfilment system.

In contrast, for systems with a utilisation of [0.9,1.0), the maximum relative deviation is between 6.1% and 6.7% in every sample. The corresponding average relative deviation is between 1.2% and 2.5%. These results indicate that for systems with high utilisation, the strategy used to select the next order to be processed has an impact on the resulting β-service level. In these systems, the β-service level achieved when using *LOR* can be significantly higher than the one achieved when using *FCFS*.

By comparing the 90%-confidence intervals of the relative deviations of different classes of utilisation in each sample, we note that the width of the 90%-confidence interval increases as the utilisation of the order fulfilment systems increases. In every sample, the 90%-confidence intervals per class of utilisation are narrower than 1% for systems with a utilisation smaller than 0.7. In contrast, for systems with a utilisation of [0.9,1.0), the 90%-confidence interval is given by, for instance, [0.05%, 6.1%] in sample A1 and [0.07%, 5.4%] in sample E. These results indicate that the variance of the relative deviations of β-service level increases as the utilisation of the order fulfilment system increases.

12.3.3 Discussion

For both α- and β-service level, we do not observe significant differences between *LOR* and *FCFS* for order fulfilment systems with a utilisation smaller than approximately 0.6 in all samples. These results confirm that for order fulfilment systems with low utilisation, the strategy used to select the next order to be processed has a neglectable small impact on the resulting system performance (see hypothesis 7).

In contrast, in every sample, we observe significantly higher values of both α- and β-service level when using *LOR* instead of *FCFS* in order fulfilment systems with high utilisation. For instance, for systems with a utilisation of [0.9,1.0), the average relative deviation over all samples is 27% for α-service level and 2% for β-service level. α-service level is more sensitive to the selected strategy than β-service level. These results confirm that the strategy used to select the next order to be processed has an essential impact on the resulting system performance

in order fulfilment systems with high utilisation and that system performance achieved when using *LOR* can be significantly higher than the one achieved when using *FCFS* (see hypothesis 8).

Samples A2, C, and E contain few data points whose β-service level achieved when using *LOR* is smaller than the one achieved when using *FCFS*. Table 12.3 gives a detailed analysis of selected performance measures – order income-related utilisation, β-service level, and expected total number of lost sales per time period – of the corresponding order fulfilment systems.

Table 12.3: Order income-related utilisation, β-service level, and expected total number of lost sales per time period of selected order fulfilment systems when using the *Strategy of Levelled Order Release (LOR)* and when using *FCFS* (A2.1-A2.6 of sample A2, C.1 of sample C, E.1-E.2 of sample E).

		SL_β			$E(S)$		
	$\tilde{U}$	*LOR*	*FCFS*	Deviation[1]	*LOR*	*FCFS*	Deviation[1]
A2.1	0.9638	0.8529	0.8791	-2.98%	0.0865	0.1396	-38.02%
A2.2	0.9771	0.7329	0.7488	-2.12%	0.1817	0.2515	-27.76%
A2.3	0.9278	0.8834	0.8896	-0.70%	0.0741	0.0925	-19.91%
A2.4	0.9823	0.8103	0.8677	-6.62%	0.1161	0.1784	-34.90%
A2.5	0.9489	0.8706	0.8863	-1.78%	0.0761	0.1068	-28.71%
A2.6	0.9838	0.7368	0.7818	-5.75%	0.1882	0.2636	-28.60%
C.1	0.9718	0.8699	0.8809	-1.24%	0.0221	0.0254	-12.80%
E.1	0.9773	0.8494	0.8826	-3.76%	0.0224	0.0326	-31.24%
E.2	0.9768	0.8535	0.8866	-3.73%	0.0213	0.0310	-31.30%

[1] Difference between the value of the key figure (SL_β or $E(S)$) when using *LOR* and the one when using *FCFS*, divided by the value of the key figure when using *FCFS*.

All of them are characterised by a high utilisation and a high proportion of backorders. In this kind of order fulfilment systems, *LOR*, which selects the next order to be processed based on the shortest due date to systematically avoid lost sales, has the negative side-effect that only a small proportion of the available processing performance per time period remains to process orders without failed due dates. Thus, the β-service level achieved when using *LOR* is relatively small

in these systems. In contrast, *FCFS*, which does not rely on due dates when selecting the next order to be processed, accepts a higher number of lost sales, and thus more orders can be completed on time. Consequently, the β-service level achieved when using *FCFS* is higher than the one achieved when using *LOR*.

12.4 Comparison regarding Required Capacity

In the following, we conduct a numerical analysis based on samples F and G to compare *LOR* and *FCFS* regarding the total required capacity and to verify hypothesis 9. As evaluation criteria, we use the absolute and relative deviation of the total required capacity in the order fulfilment system to guarantee a predefined performance requirement when using *LOR* from the one when using *FCFS*. The total required capacity corresponds to the objective function value of the capacity planning problem (see equation (9.5)).

We use the parameter setting SP/SMy-QM/NMn of the MADS algorithm for solving the capacity planning problem (see Chapter 11). In the case of capacity planning for order fulfilment systems using *LOR*, the exact model for performance analysis represents the blackbox. In contrast, in the case of capacity planning for order fulfilment systems using *FCFS*, the adapted simulation model, which is used in Section 12.3 for performance analysis of order fulfilment systems using *FCFS*, represents the blackbox.

Sample F consists of 200 single-stage order fulfilment systems, and sample G contains 200 two-stage order fulfilment systems. The samples differ regarding the ranges of the parameters, especially regarding the discretisation of the number of incoming orders per time period A: Its range is given by $\mathcal{A}^F = \{3, 4, \ldots, 15\}$ in sample F and $\mathcal{A}^G = \{2, 3, \ldots, 6\}$ in sample G. A comprehensive description of the samples is given in Appendix C.3. In both samples, the performance requirement of the capacity planning problem is specified by a predefined, individual value of α-service level.

12.4.1 Numerical Results

Table 12.4 presents the minimum, the maximum, and the average values of the absolute and relative deviations of the total required capacity between *LOR* and *FCFS* in samples F and G.

Table 12.4: Deviation of total required capacity between the *Strategy of Levelled Order Release (LOR)* and *FCFS*.

	Absolute deviation[1]			Relative deviation [%][2]		
	Min	Max	Average	Min	Max	Average
Sample F	-1.00	0.00	-0.14	-33.33	0.00	-2.85
Sample G	-1.00	0.00	-0.12	-20.00	0.00	-1.65

[1] Difference between the capacity required when using *LOR* and the one required when using *FCFS*.
[2] Difference between the capacity required when using *LOR* and the one required when using *FCFS*, divided by the capacity required when using *FCFS*.

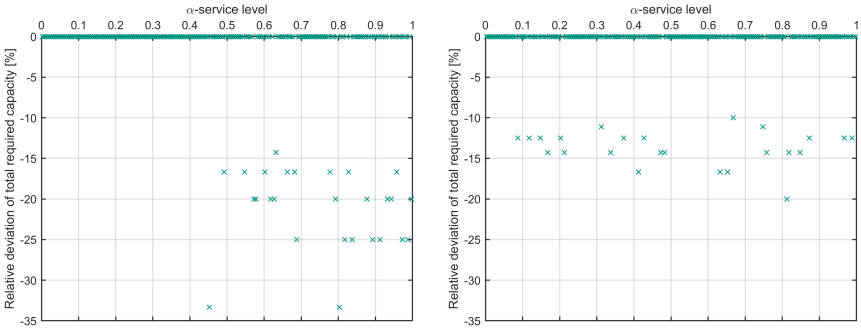
We observe a maximum absolute deviation of zero in both samples. Thus, for all data points of the considered samples, the total required capacity to guarantee a predefined α-service level when using *LOR* is at most as high as the one when using *FCFS*. Furthermore, for 13.5% of the data points in sample F and 12% of the data points in sample G, the total required capacity to guarantee a predefined α-service level when using *LOR* is smaller than the one when using *FCFS*. We observe a minimum absolute deviation of -1.00 in each sample. The corresponding minimum relative deviation is -33% in sample F and -20% in sample G, respectively.

Focusing on the data points for which we observe a capacity saving when using *LOR* instead of *FCFS*, the average absolute capacity saving is one time unit in each sample. The corresponding average relative capacity saving is 21% in sample F and 14% in sample G (see Table 12.5).

Figure 12.3 presents the relative deviation of the total required capacity between *LOR* and *FCFS* depending on the required α-service level in samples F and G. It indicates that the value of the required α-service level does not have any systematic

Table 12.5: Capacity savings when using the *Strategy of Levelled Order Release (LOR)* instead of *FCFS*.

	Proportion of data points [%]	Average absolute capacity saving	Average relative capacity saving [%]
Sample F	13.50	1.00	21.08
Sample G	12.00	1.00	13.71



(a) Sample F **(b)** Sample G

Figure 12.3: Deviation of total required capacity between the *Strategy of Levelled Order Release (LOR)* and *FCFS*.

impact on the capacity savings that can be achieved when using *LOR* instead of *FCFS*. The required α-service level of data points for which we observe a capacity saving when using *LOR* instead of *FCFS* is between 0.45 and 1.00 in sample F and between 0.08 and 1.00 in sample G.

12.4.2 Discussion

We observe no data point with a positive deviation of total required capacity between *LOR* and *FCFS* in both samples. Furthermore, we observe capacity savings of on average 18% when using *LOR* instead of *FCFS* for approximately 13% of the data points in each sample. These results confirm that the total required

capacity to guarantee a predefined performance requirement when using *LOR* is at most as high as the one when using *FCFS* (see hypothesis 9).

The absolute capacity saving equals one time unit for every data point, for which we observe a capacity saving when using *LOR* instead of *FCFS*. This is due to the fact that the considered order fulfilment systems in both samples are small (see Appendix C.3), and the capacity is an integer parameter (see equation (9.9)). Thus, the variable domain $\mathcal{C}$ of the capacity planning problem is limited: It is given by $\mathcal{C}^F = \{1, 2, \ldots, 8\}$ in sample F and by $\mathcal{C}^G = \{1, 2, \ldots, 6\} \times \{1, 2, \ldots, 12\}$ in sample G (see equation (9.4)).

12.5 Chapter Conclusion

In this chapter, we evaluated the *Strategy of Levelled Order Release* in multi-stage, stochastic order fulfilment systems with customer-required order deadlines by comparing its performance with the one of *FCFS*.

The *Strategy of Levelled Order Release* comprises two planning problems: Capacity planning and order dispatching. It is difficult to find suitable alternative strategies to evaluate the *Strategy of Levelled Order Release* since related research fields only focus on one of these planning problems: Strategies of flexible capacity adaption concentrate on capacity planning, whereas dispatching strategies and scheduling problems focus on order dispatching. Consequently, the comparison of the *Strategy of Levelled Order Release* with any alternative strategy is always limited to one of its planning problems. The *Strategy of Flexible Capacity Adaption* is an unsuitable alternative strategy since the models provided in this thesis are restricted to performance analysis of order fulfilment systems and do not consider further evaluation criteria, such as the flexibility costs of the *Strategy of Flexible Capacity Adaption*. Furthermore, processing-related dispatching policies, such as *Shortest remaining processing time*, and scheduling problems are unsuitable alternative strategies due to modelling reasons, and due-date related dispatching policies select the next order to be processed based on the same criterion as the

Strategy of Levelled Order Release. However, the dispatching strategy *FCFS* is a suitable alternative strategy to evaluate the *Strategy of Levelled Order Release*. Since *FCFS* is limited to the planning problem of order dispatching, it is combined with the capacity planning approach of the *Strategy of Levelled Order Release* in order to ensure the comparability of both strategies.

The comparison of the *Strategy of Levelled Order Release* and *FCFS* regarding the resulting system performance shows that the strategy used to select the next order to be processed has a neglectable small impact on the resulting values of α- and β-service level in order fulfilment systems with a utilisation smaller than approximately 0.6. In contrast, in order fulfilment systems with a utilisation higher than 0.6, one can achieve significantly higher values of α- and β-service level when using the *Strategy of Levelled Order Release* instead of *FCFS*. For instance, for systems with a utilisation of [0.9,1.0), we observe an average increase of α-service level of 27% and of β-service level of 2%.

The comparison of the *Strategy of Levelled Order Release* and *FCFS* regarding the total required capacity to guarantee a predefined α-service level in the order fulfilment system shows that the total required capacity when using the *Strategy of Levelled Order Release* is at most as high as the one when using *FCFS*. In approximately 13% of the considered capacity planning problems, we achieve a capacity saving of on average 18% when using the *Strategy of Levelled Order Release* instead of *FCFS*.

13 Conclusion

This chapter aims at summarising the most important results of this thesis and presenting an outlook on further research topics.

13.1 Summary

Order fulfilment systems are forced to efficiently manage a highly volatile customer demand consisting of low-volume orders while simultaneously meeting customer-required, short order deadlines. Hopp and Spearman (2004) provide three buffer types – inventory buffer, time buffer, and capacity buffer – to handle the volatile workload in production systems. In this thesis, we combine the potentials of time buffer and capacity buffer to develop and analytically investigate the *Strategy of Levelled Order Release* to balance workload in multi-stage, stochastic order fulfilment systems with customer-required order deadlines over time on a tactical level. The focus is on the two main application fields of order fulfilment: Warehouses and make-to-order systems. The contribution of this thesis is threefold: We develop (1) a workload balancing concept, (2) a discrete-time analytical model for performance analysis, and (3) an algorithm for capacity planning under performance constraints for multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines.

The results of the literature review state the research gap of this thesis. First, there is no workload balancing approach in the literature to balance the workload of order fulfilment systems over time on a tactical level by using a combination of time buffer and capacity buffer. Second, the discrete-time analytical model introduced

in this thesis is the first analytical model for performance analysis of workload balancing in order fulfilment systems that ensures a real-life representation of order fulfilment systems. It regards customer-required order deadlines and the interdependencies between the processing steps, and it models customer demand, customer-required deadlines, and processing performance as stochastic parameters. Furthermore, the discrete-time analytical model enables the exact calculation of complete probability distributions for several system- and customer-related performance measures. Third, capacity planning problems with due date-related performance constraints that rely on probability distributions of corresponding performance measures and service level in multi-stage systems with stochastic, customer-required order deadlines have not been studied in the literature so far.

Workload Balancing in Order Fulfilment Systems The order fulfilment system forms the basis for all concepts and models developed in this thesis. We formally describe an order fulfilment system by a finite set of order types, a finite set of processes, and a function mapping each order type to its processing sequence in the order fulfilment system. Each order type is characterised by the number of incoming orders per time period and the lead time of an order, and each process is specified by its processing performance and its capacity. We develop a levelling concept for order fulfilment systems, the so-called *Strategy of Levelled Order Release*, based on the key ideas of Heijunka levelling in production systems. It is impossible to directly apply the concept of Heijunka levelling in order fulfilment systems due to several differences in the system characteristics, such as order lot size, the role of customer demand, and characteristics of capacity. Moreover, the overarching question of the levelling concept is different: Heijunka levelling decides on a suitable buffer size of the finished-goods-inventory to meet the promised performance requirements, whereas, in order fulfilment systems, the decision is on a suitable amount of provided capacity to meet the promised performance requirements. The key characteristics of the *Strategy of Levelled Order Release* are the following:

- There is a fixed capacity per order type per scheduling interval reserved for order processing of orders of this order type in each scheduling interval.
- In each scheduling interval, the reserved capacity per order type is deployed to process orders of this order type according to ascending due dates.

In this way, the *Strategy of Levelled Order Release* systematically exploits the time buffer of each order between its time of arrival and its deadline to balance the variability of the customer demand. The remaining variability of the customer demand is either passed on to the customer, resulting in low service levels, or it is balanced using the capacity buffer. The extent to which capacity buffer is used to balance the remaining variability depends on the specific performance requirements of the order fulfilment system. Thus, the *Strategy of Levelled Order Release* provides an answer to the first research question of the thesis:

> **How can we balance workload over time in multi-stage, stochastic order fulfilment systems with customer-required order deadlines?**

Performance Analysis of Order Fulfilment Systems We choose a discrete-time Markov chain to model and analyse system behaviour and performance of a multi-stage, stochastic order fulfilment system with levelled order release and customer-required order deadlines since discrete-time analytical models provide several advantages regarding accuracy, level of detail, and the description of real processes compared to continuous-time analytical models. Above all, a discrete-time Markov chain enables the exact calculation of complete probability distributions of stochastic performance measures. To model system behaviour of multi-stage order fulfilment systems with levelled order release as a discrete-time Markov chain, we consider the order types of the order fulfilment system separately. We assume each order type to have a sequential processing sequence, and we limit the possible backlog duration of an order by the maximum accepted backlog duration. Furthermore, regarding the modelling of partially processed orders that are transmitted between the processing steps of a multi-stage order fulfilment

system, we differentiate between an exact and a simplified modelling approach: The exact modelling approach provides an exact recording of partially processed orders, whereas partially processed orders are neglected in the simplified modelling approach. Based on these modelling approaches, we introduce an exact and a simplified analytical model for performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines by

- modelling system behaviour of such order fulfilment systems according to the respective modelling approach as a discrete-time Markov chain, and
- calculating multiple stochastic and deterministic, system- and customer-related performance measures of order fulfilment systems exactly based on the limiting distribution of the Markov chain.

The simplified model can be seen as a special case of the exact model since it ensues from the exact model when modelling is restricted to single-stage order fulfilment systems. Based on the calculated performance measures, such as system throughput, α-, β- and γ-service level, utilisation, performance balance, number of unprocessed orders, number of lost sales, and time buffer and backlog duration of a completed order, the developed models enable a comprehensive performance analysis of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. These models for performance analysis provide an answer to the second research question of the thesis:

> **How can we determine the performance of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines?**

We show that the simplified model suffers from modelling inaccuracies for performance analysis of order fulfilment systems with high utilisation and a shifting bottleneck: We prove that for this system configuration, the expected value of system throughput calculated based on the simplified model is smaller than the actual one calculated based on the exact model. Furthermore, we numerically show that these modelling inaccuracies of the simplified model lead to inaccuracies in the

values of α- and β-service level. For instance, for systems with a utilisation of [0.9,1.0), the absolute value of the relative deviation of α-service level is at most 3.3% and on average 0.5% in the sample of two-stage systems, and it is at most 47.6% and on average 6.9% in the sample of three-stage systems. Hence, for order fulfilment systems with high utilisation and a shifting bottleneck, it is only reasonable to use the simplified model for worst-case analyses of system throughput and service level. In contrast, for order fulfilment systems with high utilisation and a static bottleneck, we prove that the expected value of system throughput calculated based on the simplified model corresponds to the actual one calculated based on the exact model. For order fulfilment systems with low utilisation, we numerically show that system performance calculated based on the simplified model is the same as the one calculated based on the exact model, apart from negligible numerical errors. However, we expect order fulfilment systems with high utilisation and a shifting bottleneck to be the main application field for workload balancing since systems with low utilisation are not competitive in the long run, and the most crucial measure for improvement in systems with a static bottleneck is not workload balancing but increasing the processing performance at the bottleneck. For performance analysis of these systems, one should prefer the exact model over the simplified model for reasons of modelling accuracy and accuracy of performance analysis, despite its drawbacks regarding computational effort and memory usage due to the larger number of reachable states (average reduction in the number of reachable states by more than 90% when using the simplified model instead of the exact one).

Capacity Planning in Order Fulfilment Systems The decision problem of capacity planning in multi-stage order fulfilment systems with performance requirements is a blackbox optimisation problem since the relationship between the provided capacity and the performance that is achieved with this capacity cannot be specified by a mathematical equation, but it is given by the analytical model for performance analysis. The analytical model that calculates the resulting system performance for any order fulfilment system with a given capacity represents the blackbox of the capacity planning problem. We select the direct search

method *Mesh Adaptive Direct Search* (MADS) and the model-based derivative-free method *Surrogate Optimisation Integer* (SO-I) as suitable solution algorithms for capacity planning since they meet the characteristics of the capacity planning problem, their convergence is mathematically proven, and their implementation is open source. The problem-specific configurations of the MADS algorithm and the SO-I algorithm enable a target-oriented determination of the minimum required, process-specific capacity to meet any performance requirement of the customers that is specified based on one or multiple performance measures of the order fulfilment system. Thus, both algorithms provide an answer to the third research question of the thesis:

> **How can we determine the capacity required to meet specific performance requirements in multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines?**

Numerical Analysis and Application Runtime optimisation and efficient memory usage become challenging issues when modelling system behaviour of real-life order fulfilment systems as a discrete-time Markov chain since the size of the state space of the Markov chain quickly increases. Therefore, we introduce the alternative procedure to determine the set of reachable states of a Markov chain based on a given initial state. By this procedure, we achieve an average reduction in the number of calculated states by 43.6% for single-stage systems and by 99.8% for two-stage systems. A further strategy to reduce memory usage of the Markov chain is to use a sparse storage scheme to store the transition matrix of the Markov chain. To reduce the computation time of the Markov chain, we recommend using the indirect method GMRES to solve linear systems and computing the Markov chain in parallel. By using GMRES instead of the Gaussian elimination, we observe an average reduction in computation time by approximately 46%. by calculating the Markov chain in parallel, we achieve an average reduction in computation time by 95% compared to sequential computation.

A numerical analysis on fine-tuning and evaluating the solution algorithms for capacity planning regarding solution quality and runtime efficiency identifies the parameter settings SP/SMy-QM/NMn and CANDglob/RBF to be the most suitable parameter settings of the MADS algorithm and the SO-I algorithm for solving the capacity planning problem. The MADS algorithm and the SO-I algorithm do not differ regarding solution quality since their most suitable parameter settings find the optimal solution for all considered data points, respectively. However, the MADS algorithm has a remarkable higher runtime efficiency than the SO-I algorithm (average reduction in the number of calculated blackbox instances per data point by 65%). Consequently, we recommend using the parameter setting SP/SMy-QM/NMn of the MADS algorithm for solving the capacity planning problem in order fulfilment systems.

Finally, we evaluate the *Strategy of Levelled Order Release* in multi-stage, stochastic order fulfilment systems with customer-required order deadlines by comparing its performance with the one of *FCFS*. Since the *Strategy of Levelled Order Release* comprises two planning problems – capacity planning and order dispatching –, it is difficult to find suitable alternative strategies in related research fields to evaluate the *Strategy of Levelled Order Release*. Strategies of flexible capacity adaption concentrate on capacity planning, whereas dispatching strategies and scheduling problems focus on order dispatching. The *Strategy of Flexible Capacity Adaption* is an unsuitable alternative strategy since the models provided in this thesis are restricted to performance analysis of order fulfilment systems and do not consider further evaluation criteria, such as the flexibility costs of the *Strategy of Flexible Capacity Adaption*. Furthermore, processing-related dispatching policies, such as *Shortest remaining processing time*, and scheduling problems are unsuitable alternative strategies due to modelling reasons, and due-date related dispatching policies select the next order to be processed based on the same criterion as the *Strategy of Levelled Order Release*. However, the dispatching strategy *FCFS* is a suitable alternative strategy to evaluate the *Strategy of Levelled Order Release*. Since *FCFS* is limited to the planning problem of order dispatching, it is combined with the capacity planning approach of the *Strategy of Levelled Order Release* in order to ensure the comparability of both strategies.

The comparison of the *Strategy of Levelled Order Release* and *FCFS* regarding the resulting system performance shows that the strategy used to select the next order to be processed has a neglectable small impact on the resulting values of α- and β-service level in order fulfilment systems with a utilisation smaller than approximately 0.6. In contrast, in order fulfilment systems with a utilisation higher than 0.6, one can achieve significantly higher values of α- and β-service level when using the *Strategy of Levelled Order Release* instead of *FCFS*. For instance, for systems with a utilisation of [0.9,1.0), we observe an average increase of α-service level of 27% and of β-service level of 2%. The comparison of the *Strategy of Levelled Order Release* and *FCFS* regarding the total required capacity to guarantee a predefined α-service level in the order fulfilment system shows that the total required capacity when using the *Strategy of Levelled Order Release* is at most as high as the one when using *FCFS*. In approximately 13% of the considered capacity planning problems, we achieve a capacity saving of on average 18% when using the *Strategy of Levelled Order Release* instead of *FCFS*.

13.2 Outlook

After answering the research questions and summarising the results, we encounter the boundaries of this thesis. Some aspects go beyond the scope of this thesis but are interesting and relevant to analyse.

First, further research can be deducted on generalising the models developed in this thesis. In real-life order fulfilment systems, customer demand is often seasonally fluctuating. By modelling the number of incoming orders as a time-dependent, stochastic parameter, the time-dependency of customer demand can be incorporated into the models of this thesis. Furthermore, the processing sequence of orders in order fulfilment systems usually contains multiple splits and merges, whereas the models of this thesis are limited to a sequential processing sequence. We can model an order type whose processing sequence contains splits by dividing this order type into multiple artificial order types, each of which has a sequential

processing sequence. However, modelling merges in order fulfilment systems is beyond the scope of this thesis.

Second, parallel computing of the Markov chain is limited to the available processors of the CPU in this thesis. However, we expect the additional use of graphics processing units to provide further potentials in reducing computation time and memory usage of the Markov chain.

Finally, the concept of variability buffers of Hopp and Spearman (2004) provides further strategies for workload balancing in multi-stage, stochastic order fulfilment systems apart from the *Strategy of Levelled Order Release*. For instance, the *Strategy of Flexible Capacity Adaption* uses the capacity buffer to match provided capacity as precisely as possible to demanded capacity. Even inventory buffer provides some potentials for workload balancing in order fulfilment systems by doing preparatory processing steps in advance. Moreover, various combined strategies are reasonable. A comprehensive evaluation and comparison of these different approaches for workload balancing in multi-stage, stochastic order fulfilment systems require a holistic evaluation framework. Based on the models developed in this thesis, the comparison of the *Strategy of Levelled Order Release* with alternative strategies is limited to several performance measures so far. However, a holistic evaluation framework has to incorporate further evaluation criteria, such as cost aspects.

A Methodology of Literature Review

The state of the art of academic literature in the research segments of this thesis, which is presented in Chapter 2, results from a keyword-based systematic literature review in the Scopus database. We searched for each pair of keywords given in Table A.1 in the article title, abstract, and keywords. We limited the resulting set of publications to journal articles and conference papers in English language. We additionally removed all papers of unrelated research fields. In the remaining relevant publications, we analysed the references and the citations to find further relevant publications that did not emerge previously.

Table A.1: Keywords of literature review.

Keyword 1	Keyword 2
capacity	deadline
capacity	service level, service requirement
flow shop	order release
Heijunka	-
make to order, MTO	analytic
make to order, MTO	capacity
make to order, MTO	deadline
make to order, MTO	due date
make to order, MTO	flow shop
make to order, MTO	levelling, leveling
make to order, MTO	linear system, linear flow
make to order, MTO	multi-stage, multi stage
make to order, MTO	order release

Keyword 1	Keyword 2
make to order, MTO	performance analysis
make to order, MTO	queueing, queuing
make to order, MTO	review
make to order, MTO	sequential
make to order, MTO	stochastic
make to order, MTO	workload balance, workload balancing
order fulfilment, order fulfillment	capacity
order fulfilment, order fulfillment	deadline
order fulfilment, order fulfillment	due date
order fulfilment, order fulfillment	levelling, leveling
order fulfilment, order fulfillment	make to order, MTO
order fulfilment, order fulfillment	multi-stage, multi stage
order fulfilment, order fulfillment	order release
order fulfilment, order fulfillment	performance analysis
order fulfilment, order fulfillment	queueing, queuing
order fulfilment, order fulfillment	review
order fulfilment, order fulfillment	stochastic
order fulfilment, order fulfillment	uncertainty
order fulfilment, order fulfillment	warehouse, order picking
order fulfilment, order fulfillment	workload balance, workload balancing
performance analysis	deadline
performance analysis	due date
performance analysis	multi-stage, multi stage
performance analysis	stochastic
queueing, queuing	deadline
warehouse, order picking	analytic
warehouse, order picking	capacity
warehouse, order picking	deadline
warehouse, order picking	dispatch
warehouse, order picking	due date
warehouse, order picking	flow shop
warehouse, order picking	levelling, leveling
warehouse, order picking	linear system, linear flow

Keyword 1	Keyword 2
warehouse, order picking	multi-stage, multi stage
warehouse, order picking	order release
warehouse, order picking	performance analysis
warehouse, order picking	queueing, queuing
warehouse, order picking	review
warehouse, order picking	sequential
warehouse, order picking	stochastic
warehouse, order picking	uncertainty
warehouse, order picking	workload balance, workload balancing

B Simulation Model

The simulation model depicts and analyses system behaviour and performance of multi-stage, stochastic order fulfilment systems with levelled order release and customer-required order deadlines. It is based on the exact modelling approach (see Figure 5.1), and it considers a multi-stage, stochastic order fulfilment system with a finite set of processes $\mathcal{P}$, a unique order type, and levelled order release considering a finite set of due dates $\mathcal{K}$. The order type and each process are specified by the parameters introduced in Sections 3.2.2 and 3.2.3, respectively. Analogous to the analytical models for performance analysis, we observe the order fulfilment system at discrete-time points in time $t \in \mathbb{N}_0$. The state of the simulation model specifies the number of unprocessed orders according to their due dates (see equation (6.8)).

B.1 Simulation Iteration

A simulation iteration models order processing in the multi-stage order fulfilment system in any time period t. It consists of the same $(p_{max} + 2)$ sub-steps as the state transition of the Markov chain of the exact model (see Section 6.2.2):

- Order processing in time period t:
  - Order processing at process $p = 1$,
  - $\ldots$,
  - Order processing at process p_{max},

- Due date update at the end of time period t, and
- Order income at the beginning of time period $(t+1)$.

We use the same notation as in the exact model for performance analysis and denote the interim states of a simulation iteration by the variables $\mathbf{y}^{(m)}$, $m \in \{0, \ldots, (p_{max}+2)\}$, whereby m is used as a counter, and $\mathbf{y}^0$ specifies the state of the simulation before the start of order processing in time period t. The realisations of the order income per time period $\mathbf{G} = \mathbf{g}$ and the processing performance per time period $\mathbf{H} = \mathbf{h}$ in time period t are generated based on independent random generators. Depending on these realisations, the interim states $\mathbf{y}^{(m)}$, $m \in \{0, \ldots, (p_{max}+2)\}$, of the simulation iteration in time period t are calculated using the equations of the state transition of the Markov chain of the exact model (see equations (6.9)-(6.12)). Based on the values of the interim states, we calculate the values of the performance measures of the order fulfilment system (see Table 6.1) in time period t as follows

- Number of unprocessed orders at process p at the beginning of the time period

$$q_p = \sum_{k \in \mathcal{K}} y_{p,k}^{(0)} \qquad \forall p \in \mathcal{P} \tag{B.1}$$

- Number of unprocessed orders in the system at the beginning of the time period

$$q = \sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} y_{p,k}^{(0)} \tag{B.2}$$

- Number of unprocessed orders at process p immediately before the start of order processing at process p

$$\tilde{q}_p = \sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)} \qquad \forall p \in \mathcal{P} \tag{B.3}$$

- Number of lost sales per time period at process p

$$s_p = y_{p,-R}^{(p)} \qquad \forall p \in \mathcal{P} \tag{B.4}$$

- Total number of lost sales per time period

$$s = \sum_{p \in \mathcal{P}} y_{p,-R}^{(p)} \tag{B.5}$$

- Performance balance of process p

$$w_p = h_p - \sum_{k \in \mathcal{K}} y_{p,k}^{(p-1)} \qquad \forall p \in \mathcal{P} \tag{B.6}$$

- Order backlog-related utilisation of process p

$$u_p = \min \left\{ 1; \frac{y_{p,k}^{(p-1)}}{h_p} \right\} \qquad \forall p \in \mathcal{P} \tag{B.7}$$

- Processed order per time period

$$\begin{gathered} \mathbf{m} = \begin{pmatrix} m_{1,-R} & \dots & m_{1,e_{max}} \\ \vdots & \ddots & \vdots \\ m_{p_{max},-R} & \dots & m_{p_{max},e_{max}} \end{pmatrix} \\ m_{p,k} = \min \left\{ y_{p,k}^{(p-1)}; \max \left\{ 0; h_p - \sum_{l=-R}^{k-1} y_{p,l}^{(p-1)} \right\} \right\} \\ \forall p \in \mathcal{P}, k \in \mathcal{K} \end{gathered} \tag{B.8}$$

- Number of processed orders per time period at process p

$$f_p = \sum_{k \in \mathcal{K}} m_{p,k} \qquad \forall p \in \mathcal{P} \tag{B.9}$$

- Number of completed orders per time period

$$f = \sum_{k \in \mathcal{K}} m_{p_{max},k} \tag{B.10}$$

- α-service level

$$sl_\alpha = \begin{cases} 1 & \sum_{k=-R}^{-1} m_{p_{max},k} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{B.11}$$

- β-service level

$$sl_{\beta} = \frac{\sum_{k=0}^{e_{max}} m_{p_{max},k}}{f+s} \tag{B.12}$$

- γ-service level

$$sl_{\gamma} = 1 - \left(\frac{\sum_{k=-R}^{-1} |k| \cdot m_{p_{max},k} + (R+1) \cdot s}{R \cdot f + (R+1) \cdot s} \right). \tag{B.13}$$

We register the time difference to the order deadline d, the backlog duration $d^{backlog}$, and the time buffer d^{buffer} of all orders completed in time period t by recording the due date of every completed order at its time of completion.

B.2 Warm-up Period

The simulation model suffers from the problem of initial transient. Since the initial state of the simulation is randomly selected, there is no guarantee that initial simulation iterations correctly represent system behaviour in steady-state. Instead, the average value of any key figure calculated based on a finite number of replications will be a biased estimator of the real value of this key figure in steady-state. To avoid this bias in estimators, a certain number of simulation iterations at the beginning of every replication, the so-called *warm-up period* of the simulation, is deleted, and the key figures are estimated based on the remaining observations that are said to represent steady-state behaviour of the simulation (Law 2015, p.511).

There are various methods in the literature to determine the end of the warm-up period τ. We use the *Marginal Standard Error Rule* (MSER-5) introduced by White Jr (1997) to identify the end of warm-up period τ based on β-service level and 100 replications, each of which consists of 5,000 simulation iterations.

B.3 Stopping Criteria

The length of every replication and the number of replications are determined based on β-service level. $SL_{\beta,i,j}$ denotes β-service level of simulation iteration j in replication i. $SL_{\beta,i}$ denotes β-service level of replication i, and it is calculated as the average value of the values $SL_{\beta,i,j}$ of β-service level of the simulation iterations in steady-state of replication i

$$SL_{\beta,i} = \frac{1}{m_i - \tau} \sum_{k=\tau}^{m_i} SL_{\beta,i,k}, \tag{B.14}$$

whereby m_i denotes the number of simulation iterations of replication i.

To determine the length of replication i^*, we calculate the absolute deviation of the average value of β-service level measured after the current simulation iteration j^* from the average value of β-service level measured after simulation iteration j'

$$\Delta_{i^*,j^*,j'} = \left| \frac{1}{j^* - \tau} \sum_{k=\tau}^{j^*} SL_{\beta,i^*,k} - \frac{1}{j' - \tau} \sum_{k=\tau}^{j'} SL_{\beta,i^*,k} \right| \quad \forall j^*, j' \geq \tau, (j^* - j') \in \{1, 2, \ldots, 10\}. \tag{B.15}$$

Replication i^* stops after simulation iteration j^* if the following condition holds

$$\Delta_{i^*,j^*,j'} < \epsilon_R \qquad \forall (j^* - j') \in \{1, 2, \ldots, 10\}, \tag{B.16}$$

whereby ϵ_R denotes the predefined precision of a replication.

The replications of the simulation are independent and identically distributed since the initial state of every replication is selected randomly (Law 2015, p.489). To determine the number of replications, we calculate the 95%-confidence interval of β-service level based on the values $SL_{\beta,i}$ of β-service level of the already calculated replications. The 95%-confidence interval of β-service level after replication i^* is calculated as follows

$$\left[\overline{SL}_{\beta,i^*} - t_{(i^*-1),0.975} \cdot \sqrt{\frac{S^2(SL_{\beta,i^*})}{i^*}}; \; \overline{SL}_{\beta,i^*} + t_{(i^*-1),0.975} \cdot \sqrt{\frac{S^2(SL_{\beta,i^*})}{i^*}} \right], \tag{B.17}$$

whereby $\overline{SL}_{\beta,i^*}$ denotes the average value of β-service level after replication i^*

$$\overline{SL}_{\beta,i^*} = \frac{1}{i^*} \sum_{k=1}^{i^*} SL_{\beta,k},$$

$S^2(SL_{\beta,i^*})$ denotes the variance of β-service level after replication i^*

$$S^2(SL_{\beta,i^*}) = \frac{1}{i^* - 1} \sum_{k=1}^{i^*} (SL_{\beta,k} - \overline{SL}_{\beta,i^*})^2,$$

and $t_{(i^*-1),0.975}$ denotes the 97.5%-quantile of the t-distribution with $(i^* - 1)$ degrees of freedom. The simulation model terminates after replication i^* if the 95%-confidence interval of β-service level is narrower than the predefined precision ϵ_S of the simulation

$$2 \cdot t_{(i^*-1),0.975} \cdot \sqrt{\frac{S^2(SL_{\beta,i^*})}{i^*}} < \epsilon_S. \tag{B.18}$$

B.4 Performance Measures of the Order Fulfilment System

The values of each performance measure of the order fulfilment system (see Table 6.1) are derived from the corresponding absolute frequency distribution that results from the values of this performance measure in all steady-state simulation iterations of the simulation model. The probability distribution of each stochastic performance measure is estimated by the corresponding relative frequency distribution, whereas the value of each deterministic performance measure is estimated by the average value of the corresponding absolute frequency distribution.

C Design of Experiments

We use a space-filling design to generate samples of order fulfilment systems. The main characteristic of space-filling designs is that the design becomes increasingly dense in the design space by increasing the sample size (Santner et al. 2018, p.149). Vazquez and Bect (2011) provide some justification for choosing space-filling designs by showing that a space-filling design is the design with the highest rate of the mean square prediction error decreases as design size increases. The *Latin hypercube design*, which we use in this thesis, is a popular space-filling design whose sample is evenly spread over the range of each input variable separately. Santner et al. (2018, p.152f.) provide a detailed description of the general procedure for obtaining a Latin hypercube sample. To determine a reasonable sample size, Jones et al. (1998) propose the rule of thumb to use a sample size of $10d$ data points for an input space of dimension d. Due to curse of dimensionality, a $10d$ sample becomes very sparse as d increases. Loeppky et al. (2009) investigate this issue and conclude that a sample size of $10d$ data points is reasonable for experiments with an input space of dimension $d \leq 5$.

C.1 Latin Hypercube Designs for Order Fulfilment Systems

An order fulfilment system with a unique order type is specified by the following parameters (see Sections 3.2 and 5.3):

- Number of incoming orders per time period A,

- Lead time E of an order,
- Maximum backlog duration R,
- Set of processes $\mathcal{P}$ that results from the number of processing steps p_{max} of the considered order type,
- Processing performance per time unit L_p of process $p \in \mathcal{P}$, and
- Capacity c_p of process $p \in \mathcal{P}$.

The maximum backlog duration R and the number of processes p_{max} are not included into the Latin hypercube design. They are both constant for every data point of a sample. The maximum backlog duration is given $R = 1$ in every sample. Regarding the number of processes p_{max}, we differentiate between samples of single-stage, two-stage, and three-stage order fulfilment systems. The remaining parameters are systematically varied in every sample.

Some of them – A, E, and L_p, $p \in \mathcal{P}$ – are stochastic parameters. To reduce the degree of freedom of the stochastic parameters and thus to reduce the number of input variables of the Latin hypercube design, we assume each stochastic parameter to belong to a particular class of theoretical probability distribution: We assume that the number of incoming orders per time period A is approximately Poisson-distributed with parameter λ and a finite range since arrival processes are commonly modelled as Poisson-distributed random variables in the literature (Law 2015, p.312). Appendix E specifies the methodology to generate an approximately Poisson-distributed random variable with finite range. Furthermore, we focus on order fulfilment systems with short lead times by assuming the lead time E of an order to be Bernoulli-distributed with parameter η in order to incorporate the customers' requirements for short order deadlines (see Chapter 1). Finally, we assume low variation for the processing performance per time unit L_p of any process $p \in \mathcal{P}$ and model L_p as Bernoulli-distributed random variable with parameter θ_p and transformed range. Due to these assumptions, each stochastic parameter of the order fulfilment system is specified by one input variable in the Latin hypercube design.

C.2 Samples for Performance Analysis

The Latin hypercube design for performance analysis of order fulfilment systems with p_{max} processes consists of $(2 + 2 \cdot p_{max})$ input variables, one for each of the following parameters:

- Number of incoming orders per time period A,
- Lead time E of an order,
- Processing performance per time unit L_p of each process p, and
- Capacity c_p of each process p.

The ranges of the parameters are sample-specific. For the stochastic parameters, we differentiate between the range of the random variable and the range of the corresponding input variable of the Latin hypercube design. To ensure that the resulting order fulfilment systems correspond to reasonable system configurations, the chosen ranges are based on the following considerations: First, the maximum possible processing performance per time unit $l_{p,max}$ of any process $p \in \mathcal{P}$ corresponds to the minimum possible number of incoming orders per time period a_{min}. Second, the minimum (maximum) possible value of the provided capacity $c_{p,min}$ ($c_{p,max}$) at process $p \in \mathcal{P}$ is determined by the ratio of the minimum (maximum) possible number of incoming orders per time period a_{min} (a_{max}) to the maximum (minimum) possible processing performance per time unit $l_{p,max}$ ($l_{p,min}$) of process p. The discretisation of the lead time E and the processing performances per time unit L_p of the processes $p \in \mathcal{P}$ is constant in all samples since these parameters are Bernoulli-distributed. In contrast, the range of the number of incoming orders per time period A and thus its discretisation are sample-specific. Unstable order fulfilment systems (see equation (6.7)) are removed from the sample.

In this thesis, we generate six samples for performance analysis of order fulfilment systems, three of which consider single-stage systems (sample A1, A2, B) and two

of which consider two-stage systems (sample C, D). Sample E contains three-stage systems. The specification of each sample is given in Table C.1.

Table C.1: Samples for performance analysis of order fulfilment systems.

Sample A1	Configuration	Single-stage order fulfilment systems $\mathcal{P} = \{1\}$		
	Parameter ranges	A	$\mathcal{A} = \{5, 6, \ldots, 40\}$	$\lambda \in \{6, 7, \ldots, 39\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{4, 5\}$	$\theta_p \in [0, 1]$
		$c_p, p \in \mathcal{P}$		$c_p \in \{2, 3, \ldots, 10\}$
	Sample size	200 data points, 120 of which specify stable systems		
	Application	Evaluation of the *Strategy of Levelled Order Release* in Chapter 12		
Sample A2	Configuration	Single-stage order fulfilment systems $\mathcal{P} = \{1\}$		
	Parameter ranges	A	$\mathcal{A} = \{5, 6, \ldots, 40\}$	$\lambda \in \{6, 7, \ldots, 39\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{4, 5\}$	$\theta_p \in [0, 1]$
		$c_p, p \in \mathcal{P}$		$c_p \in \{2, 3, \ldots, 10\}$
	Sample size	1000 data points, 350 of which specify stable systems with a utilisation of [0.5,1.0]		
	Application	Evaluation of the *Strategy of Levelled Order Release* in Chapter 12		
Sample B	Configuration	Single-stage order fulfilment systems $\mathcal{P} = \{1\}$		
	Parameter ranges	A	$\mathcal{A} = \{3, 4, \ldots, 15\}$	$\lambda \in \{4, 5, \ldots, 14\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{2, 3\}$	$\theta_p \in [0, 1]$
		$c_p, p \in \mathcal{P}$		$c_p \in \{1, 2, \ldots, 7\}$
	Sample size	200 data points, 133 of which specify stable systems		
	Application	Runtime and memory optimisation of the Markov chain in Chapter 10		

Sample C	Configuration	Two-stage order fulfilment systems $\mathcal{P} = \{1, 2\}$		
	Parameter ranges	A	$\mathcal{A} = \{3, 4, \ldots, 14\}$	$\lambda \in \{4, 5, \ldots, 13\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{2, 3\}$	$\theta_p \in [0, 1]$
		$c_p, p \in \mathcal{P}$		$c_p \in \{1, 2, \ldots, 7\}$
	Sample size	500 data points, 256 of which specify stable systems		
	Application	Evaluation of the models for performance analysis in Chapter 8 Evaluation of the *Strategy of Levelled Order Release* in Chapter 12 Verification of the exact model for performance analysis in Appendix D		
Sample D	Configuration	Two-stage order fulfilment systems $\mathcal{P} = \{1, 2\}$		
	Parameter ranges	A	$\mathcal{A} = \{2, 3, \ldots, 6\}$	$\lambda \in \{3, 4, 5\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{1, 2\}$	$\theta_p \in [0, 1]$
		$c_p, p \in \mathcal{P}$		$c_p \in \{2, 3, \ldots, 5\}$
	Sample size	500 data points, 256 of which specify stable systems		
	Application	Runtime and memory optimisation of the Markov chain in Chapter 10		
Sample E	Configuration	Three-stage order fulfilment systems $\mathcal{P} = \{1, 2, 3\}$		
	Parameter ranges	A	$\mathcal{A} = \{2, 3, \ldots, 7\}$	$\lambda \in \{3, \ldots, 6\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{1, 2\}$	$\theta_p \in [0, 1]$
		$c_p, p \in \mathcal{P}$		$c_p \in \{2, 3, \ldots, 6\}$
	Sample size	1000 data points, 353 of which specify stable systems		
	Application	Evaluation of the models for performance analysis in Chapter 8 Evaluation of the *Strategy of Levelled Order Release* in Chapter 12 Verification of the exact model for performance analysis in Appendix D		

C.3 Samples for Capacity Planning

Samples for capacity planning differ from the ones for performance analysis regarding the input variables of the Latin hypercube design. For capacity planning, we do not require input variables for capacity c_p, $p \in \mathcal{P}$, but one input variable to specify the performance requirement. Thus, the Latin hypercube design for capacity planning in order fulfilment systems with p_{max} processes consists of $(3 + p_{max})$ input variables, one for each of the following parameters:

- Number of incoming orders per time period A,
- Lead time E of an order,
- Processing performance per time unit L_p of each process p, and
- Performance requirement.

We determine the ranges of the parameters of the samples for capacity planning based on the same considerations as the ones of the samples for performance analysis (see Section C.2). To specify the performance requirement, any performance measure of the order fulfilment system (see Table 6.1) can be used. In this thesis, we focus on α-service level SL_α. Thus, the range of the corresponding input variable of the Latin hypercube design is given by the interval [0,1].

We generate two samples (sample F, G) for capacity planning in order fulfilment systems, whereby sample F considers single-stage systems and sample G contains two-stage systems. The specification of each sample is given in Table C.2.

Table C.2: Samples for capacity planning in order fulfilment systems.

Sample F	Configuration	Single-stage order fulfilment systems $\mathcal{P} = \{1\}$		
	Parameter ranges	A	$\mathcal{A} = \{3, 4, \ldots, 15\}$	$\lambda \in \{4, 5, \ldots, 14\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{2, 3\}$	$\theta_p \in [0, 1]$
		SL_α		$SL_\alpha \in [0, 1]$
	Sample size	200 data points		
	Application	Evaluation of the *Strategy of Levelled Order Release* in Chapter 12		
Sample G	Configuration	Two-stage order fulfilment systems $\mathcal{P} = \{1, 2\}$		
	Parameter ranges	A	$\mathcal{A} = \{2, 3, \ldots, 6\}$	$\lambda \in \{3, 4, 5\}$
		E	$\mathcal{E} = \{0, 1\}$	$\eta \in [0, 1]$
		$L_p, p \in \mathcal{P}$	$\mathcal{L}_p = \{1, 2\}$	$\theta_p \in [0, 1]$
		SL_α		$SL_\alpha \in [0, 1]$
	Sample size	200 data points		
	Application	Evaluation of the capacity planning algorithms in Chapter 11 Evaluation of the *Strategy of Levelled Order Release* in Chapter 12		

D Model Verification

To verify the implementation of the exact model for performance analysis, we conduct a model comparison between the exact model and a simulation model. The simulation model depicts system behaviour of a multi-stage, stochastic order fulfilment system with levelled order release and customer-required deadlines, analogous to the Markov chain. Based on the simulation results, we calculate the same performance measures of the order fulfilment system as in the exact model (see Table 6.1). β-service level is used to determine the length of each simulation replication (precision of 1.0E-05) and the number of simulation replications (precision of 1.0E-04). Appendix B introduces the simulation model in detail.

As evaluation criteria, we calculate the absolute and relative deviations of selected performance measures – α-service level, β-service level, expected number of unprocessed orders, and expected number of completed orders – between the exact and the simulation model based on samples C and E. Sample C contains 256 two-stage order fulfilment systems, and sample E consists of 353 three-stage order fulfilment systems. The samples differ regarding the ranges of the parameters, especially regarding the discretisation of the number of incoming orders per time period A: Its range is given by $\mathcal{A}^C = \{3, 4, \ldots, 14\}$ in sample C and $\mathcal{A}^E = \{2, 3, \ldots, 7\}$ in sample E. A comprehensive description of the samples is given in Appendix C.2.

Table D.1 presents the absolute and relative deviations of the selected performance measures between the exact model and the simulation model in samples C and E. For each performance measure, Table D.1 gives the minimum, the maximum, and the average value, as well as the 2.5%- and the 97.5%-quantile of the absolute and the relative deviations, respectively.

Table D.1: Deviation of selected performance measures between the exact model and the simulation model.

		Absolute deviation[1] (Relative deviation [%][2])				
	Criterion	Min	Max	Average	$Q_{0.025}$	$Q_{0.975}$
Sample C	SL_α	-6.04E-04 (-0.0684)	4.13E-04 (0.0438)	-9.95E-06 (-0.0011)	-1.07E-04 (-0.0103)	1.01E-04 (0.0105)
	SL_β	-4.68E-04 (-0.0487)	3.96E-04 (0.0403)	-9.25E-06 (-0.0009)	-6.89E-05 (-0.0069)	3.16E-05 (0.0032)
	$E(Q)$	-1.23E-02 (-0.1371)	3.10E-02 (0.6455)	4.86E-03 (0.0851)	-6.24E-03 (-0.0787)	2.02E-02 (0.4099)
	$E(F)$	-6.88E-03 (-0.1086)	3.10E-02 (0.6455)	4.83E-03 (0.0838)	-4.02E-03 (-0.0556)	2.03E-02 (0.4012)
Sample E	SL_α	-2.36E-04 (-0.0248)	3.60E-04 (0.0363)	-6.94E-06 (-0.0006)	-1.37E-04 (-0.0140)	1.24E-04 (0.0140)
	SL_β	-1.67E-04 (-0.0170)	1.68E-04 (0.0172)	-5.45E-06 (-0.0005)	-6.99E-05 (-0.0070)	5.89E-05 (0.0069)
	$E(Q)$	-4.61E-03 (-0.1232)	2.05E-02 (0.4289)	1.86E-03 (0.0492)	-2.50E-03 (-0.0578)	1.03E-02 (0.2879)
	$E(F)$	-5.31E-03 (-0.1610)	1.14E-02 (0.3081)	1.46E-03 (0.0398)	-2.16E-03 (-0.0564)	8.45E-03 (0.2512)

[1] Difference between the value of the considered performance measure calculated based on the exact model and the one calculated based on the simulation model.
[2] Difference between the value of the considered performance measure calculated based on the exact and the one calculated based on the simulation model, divided by the value calculated based on the simulation model.

We observe absolute deviations of β-service level between -4.68E-04 and 3.96E-04 in sample C and between -1.67E-04 and 1.68E-04 in sample E. The corresponding 95%-confidence interval is given by [-6.98E-05, 3.16E-05] in sample C and [-6.99E-05, 5.89E-05] in sample E. Regarding α-service level, we observe absolute and relative deviations of the same magnitude as for β-service level. However, the absolute and relative deviations of α-service level are more fluctuating than those of β-service level as indicated by the wider 95%-confidence intervals of [-1.07E-04, 1.01.E-04] in sample C and [-1.37E-04, 1.24E-04] in sample E.

We observe absolute deviations of the expected number of unprocessed orders between -1.23E-02 and 3.10E-02 in sample C and between -4.61E-03 and 2.05E-02 in sample E. The corresponding 95%-confidence interval is given by [-6.24E-03, 2.02E-02] in sample C and [-2.5E-03, 1.03E-02] in sample E. The absolute and relative deviations of the expected number of completed orders are of the same magnitude as the ones of the expected number of unprocessed orders.

The deviations of the expected number of unprocessed orders and the expected number of completed orders are higher than the ones of α- and β-service level. However, we measure the expected number of unprocessed orders and the expected number of completed orders in number of orders, whereas α- and β-service level are calculated as a ratio with a range of [0,1]. Thus, for each performance measure, the deviations between the exact model and the simulation model are negligible despite the different magnitudes of the deviations. In conclusion, the results of the model comparison confirm that the exact model is implemented correctly.

E Generation of Approximately Poisson-distributed Random Variables with Finite Range

Random variable K is said to be *Poisson-distributed* $Poi(\lambda)$ with parameter $\lambda > 0$ and discrete range $\{0, 1, \ldots\}$ if its probability distribution is given by

$$P(K = k) = \begin{cases} \frac{1}{k!} \cdot e^{-\lambda} \cdot \lambda^k & k \in \{0, 1, \ldots\} \\ 0 & \text{otherwise.} \end{cases} \tag{E.1}$$

The expected value and the variance are calculated as follows (Law 2015, p.312)

$$E(K) = \lambda \qquad Var(K) = \lambda. \tag{E.2}$$

In the following, we present a methodology to generate the probability distribution of an approximately Poisson-distributed random variable J with finite range $\mathcal{J} = \{j_{min}, \ldots, j_{max}\}$ whose expected value is given by $E(J)$.

Initially, we calculate the parameter λ_0 of the corresponding Poisson-distributed random variable K as follows

$$\lambda_0 = E(K) = E(J) - j_{min}. \tag{E.3}$$

Subsequently, an iterative procedure starts. Each iteration $i \in \mathbb{N}$ consists of the following steps:

1. Determine the approximate upper bound Π_i of the range of $K \sim Poi(\lambda_{i-1})$ based on the following condition:

$$P(K \leq \Pi_i) \geq 1 - \epsilon_P, \tag{E.4}$$

whereby ϵ_P is set to 1.0E-09.

2. Calculate the scale parameter Φ_i:

$$\Phi_i = \frac{j_{max} - j_{min}}{\Pi_i}. \tag{E.5}$$

3. Update the Poisson parameter λ_i:

$$\lambda_i = \frac{E(J) - j_{min}}{\Phi_i}. \tag{E.6}$$

4. Calculate the normalisation constant Ψ_i based on $K \sim Poi(\lambda_i)$:

$$\Psi_i = \sum_{k=0}^{\Pi_i} P(K = k). \tag{E.7}$$

5. Check the stopping criterion:

$$\Psi_i > 1 - \epsilon_P, \tag{E.8}$$

whereby ϵ_P is set to 1.0E-09. If the stopping criterion is met, the iterative procedure terminates with $\lambda = \lambda_i$, $\Pi = \Pi_i$, $\Phi = \Phi_i$, and $\Psi = \Psi_i$. Otherwise, iteration $(i + 1)$ starts.

Finally, the probability distribution of J is derived from the probability distribution $Poi(\lambda)$ of K as follows

$$\begin{aligned} P(J = j) &= \frac{1}{\Psi} \cdot \sum_{k \in \mathcal{I}(j)} P(K = k) \qquad \forall j \in \mathcal{J} \\ \mathcal{I}(j) &= \left\{ k \in \{0, \ldots, \Pi\} \mid \left(j - \frac{1}{2}\right) \leq (\Phi \cdot k + j_{min}) < \left(j + \frac{1}{2}\right) \right\}. \end{aligned} \tag{E.9}$$

Glossary of Notation

Description of an Order Fulfilment System

$\mathcal{I}$	Set of order types	32
i	Order type	32
i_{max}	Number of order types	32
$\mathcal{P}$	Set of processes	32
p	Process	32
p^*	Static bottleneck	92
p_{max}	Number of processes	32
$\mathbf{V}_i$	Processing sequence of order type i specified by an adjacency matrix	33
v	Function mapping each order type i to its processing sequence $\mathbf{V}_i$	33
A	Number of incoming orders (random variable)	34
a	Number of incoming orders (realisation)	34
$\mathcal{A}$	Range of the number of incoming orders	34
a_{min}	Minimum possible number of incoming orders	34
a_{max}	Maximum possible number of incoming orders	34
A_i	Number of incoming orders of order type i (random variable)	34
a_i	Number of incoming orders of order type i (realisation)	34
$\mathcal{A}_i$	Range of the number of incoming orders of order type i	34

$a_{i,min}$	Minimum possible number of incoming orders of order type i . . 34
$a_{i,max}$	Maximum possible number of incoming orders of order type i . . 34
E	Lead time of an order (random variable) 34
e	Lead time of an order (realisation) 34
$\mathcal{E}$	Range of the lead time of an order 34
e_{min}	Minimum possible lead time of an order 34
e_{max}	Maximum possible lead time of an order 34
E_i	Lead time of an order of order type i (random variable) 34
e_i	Lead time of an order of order type i (realisation) 34
$\mathcal{E}_i$	Range of the lead time of an order of order type i 34
$e_{i,min}$	Minimum possible lead time of an order of order type i 34
$e_{i,max}$	Maximum possible lead time of an order of order type i 34
L_p	Processing performance of process p (random variable) 35
l_p	Processing performance of process p (realisation) 35
$\mathcal{L}_p$	Range of the processing performance of process p 35
$l_{p,min}$	Minimum possible processing performance of process p 35
$l_{p,max}$	Maximum possible processing performance of process p 35
$L_{i,p}$	Processing performance of order type i at process p (random variable) 35
$l_{i,p}$	Processing performance of order type i at process p (realisation) 35
$\mathcal{L}_{i,p}$	Range of the processing performance of order type i at process p 35
$l_{i,p,min}$	Minimum possible processing performance of order type i at process p 35
$l_{i,p,max}$	Maximum possible processing performance of order type i at process p 35
c_p	Capacity provided at process p 35

$c_{i,p}$	Capacity provided for order type i at process p 35
$\bar{c}_{i,p}$	Capacity reserved for order type i at process p in the planning step of system parametrisation 44
$\mathcal{K}$	Set of due dates 53
k	Due date 53
R	Maximum accepted backlog duration 53
$\mathbf{G}$	Order income per time period (random vector) 59
$\mathbf{g}$	Order income per time period (realisation) 59
$\mathcal{G}$	Range of the order income per time period 59
G_k	Number of incoming orders per time period with a lead time of k time periods (random variable) 59
g_k	Number of incoming orders per time period with a lead time of k time periods (realisation) 59
$\mathbf{H}$	Processing performance per time period of the order fulfilment system (random vector) 58
$\mathbf{h}$	Processing performance per time period of the order fulfilment system (realisation) 58
$\mathcal{H}$	Range of the processing performance per time period of the order fulfilment system 58
H_p	Processing performance per time period of process p (random variable) 58
h_p	Processing performance per time period of process p (realisation) 58
$\mathcal{H}_p$	Range of the processing performance per time period of process p 58
$\tilde{H}$	Aggregated processing performance per time period of the order fulfilment system (random variable) 82
$\tilde{h}$	Aggregated processing performance per time period of the order fulfilment system (realisation) 82

$\tilde{\mathcal{H}}$ Range of the aggregated processing performance per time period of the order fulfilment system 82

Performance Measures of an Order Fulfilment System

Q_p Number of unprocessed orders at process p at the beginning of a time period (random variable) 68

q_p Number of unprocessed orders at process p at the beginning of a time period (realisation) 68

$\mathcal{Q}_p$ Range of the number of unprocessed orders at process p at the beginning of a time period 68

Q Number of unprocessed orders in the order fulfilment system at the beginning of a time period (random variable) 68

q Number of unprocessed orders in the order fulfilment system at the beginning of a time period (realisation) 68

$\mathcal{Q}$ Range of the number of unprocessed orders in the order fulfilment system at the beginning of a time period 68

$\tilde{Q}_p$ Number of unprocessed orders at process p immediately before the start of order processing at process p (random variable) 69

$\tilde{q}_p$ Number of unprocessed orders at process p immediately before the start of order processing at process p (realisation) 69

$\tilde{\mathcal{Q}}_p$ Range of the number of unprocessed orders at process p immediately before the start of order processing at process p 69

S_p Number of lost sales per time period at process p (random variable) 69

s_p Number of lost sales per time period at process p (realisation) 69

$\mathcal{S}_p$	Range of the number of lost sales per time period at process p 69
S	Total number of lost sales per time period (random variable) . . . 70
s	Total number of lost sales per time period (realisation) 70
$\mathcal{S}$	Range of the total number of lost sales per time period 70
W_p	Performance balance of process p (random variable) 71
w_p	Performance balance of process p (realisation) 71
$\mathcal{W}_p$	Range of the performance balance of process p 71
U_p	Order backlog-related utilisation of process p 71
u_p	Order backlog-related utilisation of process p (realisation) 207
$\tilde{U}$	Order income-related utilisation 60
$\mathbf{M}$	Processed orders per time period (random vector) 72
$\mathbf{m}$	Processed orders per time period (realisation) 72
$\mathcal{M}$	Range of the processed orders per time period 72
$m_{p,k}$	Number of processed orders per time period at process p with a due date of k time periods at their time of processing (realisation) 72
F_p	Number of processed orders per time period at process p (random variable) 73
f_p	Number of processed orders per time period at process p (realisation) 73
$\mathcal{F}_p$	Range of the number of processed orders per time period at process p 73
F	Number of completed orders per time period (random variable) . 74
f	Number of completed orders per time period (realisation) 74
$\mathcal{F}$	Range of the number of completed orders per time period 74
D	Time difference to order deadline of a completed order (random variable) 74

d	Time difference to order deadline of a completed order (realisation) 74
$\mathcal{D}$	Range of the time difference to order deadline of a completed order 74
$D^{backlog}$	Backlog duration of a completed order (random variable) 75
$d^{backlog}$	Backlog duration of a completed order (realisation) 75
$\mathcal{D}^{backlog}$	Range of the backlog duration of a completed order 75
D^{buffer}	Time buffer of a completed order (random variable) 75
d^{buffer}	Time buffer of a completed order (realisation) 75
$\mathcal{D}^{buffer}$	Range of the time buffer of a completed order 75
SL_α	α-service level 76
sl_α	α-service level (realisation) 207
SL_β	β-service level 76
sl_β	β-service level (realisation) 208
SL_γ	γ-service level 77
sl_γ	γ-service level (realisation) 208

Discrete-time Markov Chain

$\mathcal{T}$	Time parameter set 49
t	Time parameter 48
t_{inc}	Length of discretisation interval 48
$\mathcal{X}$	State space 49
X^t	State observed at time t (random variable) 49
i^t	State observed at time t (realisation) 49
$\mathbf{P}$	Transition matrix 49
$p_{i,j}$	Single-step transition probability from state i at time t to state j at time $(t+1)$ 50

$p_{i,j}^{(n)}$	n-step transition probability from state i at time t to state j at time $(t+n)$ ... 50
i	State (realisation) ... 50
j	State (realisation) ... 50
π^0	Initial state probability distribution ... 51
π^t	State probability distribution at time t ... 51
π_i^t	State probability of state i at time t ... 51
π	Stationary distribution ... 51
π_i	Stationary state probability of state i ... 51
$\tilde{\pi}$	Limiting distribution ... 52
$\tilde{\mathbf{\Pi}}$	Matrix of limiting distributions ... 52
$\mathbf{E}$	Sparse matrix ... 143
E_v	Array of real values of sparse matrix $\mathbf{E}$ ... 143
E_c	Array of column indices of sparse matrix $\mathbf{E}$ ... 143
E_r	Array of pointers to the beginning of the rows of sparse matrix $\mathbf{E}$ ... 143

Analytical Model for Performance Analysis

$\mathbf{X}$	System state; number of unprocessed orders in the order fulfilment system at the beginning of a time period (random vector) ... 61
$\mathbf{x}$	System state; number of unprocessed orders in the order fulfilment system at the beginning of a time period (realisation) ... 61
$x_{p,k}$	Number of unprocessed orders at process p with a due date of k time periods at the beginning of a time period (exact model) ... 61

x_k	Number of unprocessed orders in the order fulfilment system with a due date of k time periods at the beginning of a time period (simplified model) 84
$\mathbf{z}$	System state; number of unprocessed orders in the order fulfilment system at the beginning of a time period (realisation) 61
z_k	Number of unprocessed orders in the order fulfilment system with a due date of k time periods at the beginning of a time period (simplified model) 84
$\mathbf{y}^{(m)}$	m^{th} interim system state of the state transition; number of unprocessed orders in the order fulfilment system at the beginning of the m^{th} sub-step of the state transition 61
$y_{p,k}^{(m)}$	Number of unprocessed orders at process p with a due date of k time periods at the beginning of the m^{th} sub-step of the state transition 61
$\mathcal{I}(\mathbf{x}, \mathbf{y})$	Set of tuples $(\mathbf{g}, \mathbf{h})$ resulting in a state transition from state $\mathbf{x}$ to state $\mathbf{z}$ 63
$h_p^{res}(k)$	Residual processing performance available for order processing of orders with a due date of k time periods at process p in time period t 95
k_p^*	Critical due date of process p in time period t 95
N	Proportionality factor of the size of the state space 114

Derivative-free and Blackbox Optimisation

ξ	Decision variable 123
$\Theta(\xi)$	Objective function 123
$\Lambda(\xi)$	Constraint 123
$\mathcal{J}$	Set of constraints 123
Ω	Variable domain 123

$\Gamma(\xi)$	Constraint violation function	131
$\Theta_p(\xi)$	Penalty-augmented objective function	137
ρ	Penalty factor	137
Θ_{max}	Current worst feasible objective function value	137
$\Sigma(\xi)$	Surrogate function	133
$\chi(\xi)$	Polynomial basis function of polynomial response surface	133
$\psi(\xi)$	Basis function of Kriging model	133
$\epsilon(\xi)$	Random error of Kriging model	133
$\phi(\xi)$	Kernel function	133
$\delta(\xi, \xi_i)$	Distance function	133
ω	Weight	133
n_F	Number of independent basis functions	133
n_S	Sample size	133

Capacity Planning Problem

$\mathbf{c}$	Capacity provided at the processes of the order fulfilment system	120
$\mathcal{C}$	Range of provided capacity	122
$c_{p,min}$	Lower limit of provided capacity at process p	121
$c_{p,max}$	Upper limit of provided capacity at process p	121
$SL(\mathbf{c})$	Performance achieved with the provided capacity $\mathbf{c}$	120
SL^*	Performance requirement	120
$\tilde{U}(\mathbf{c})$	Order income-related utilisation of the order fulfilment system with the provided capacity $\mathbf{c}$	121

Algorithmic Parameters of the MADS Algorithm

SP	Initial point is calculated based on equation (4.1) 158
LHD	Latin hypercube design with a sample size of $0.25 \cdot \lvert\mathcal{C}\rvert$ 158
SMn	No surrogate model is used 158
SMy-QM	Use of a quadratic model 158
SMy-PRS	Use of a polynomial response surface of degree two 158
SMy-RBF	Use of a radial basis function with Gaussian kernel and l_1-norm 158
SMy-KSM	Use of a kernel smoothing model with Gaussian kernel and l_1-norm 158
SMy-KM	Use of a Kriging model with l_1-norm 158
NMn	No use of Nelder-Mead algorithm 158
NMy	Use of Nelder-Mead algorithm 158

Algorithmic Parameters of the SO-I Algorithm

CANDglob	Perturb the best feasible point found so far and uniformly select integer points in the variable domain 162
SurfMin	Select the local minimum of the surrogate model as new candidate point 162
RBF	Cubic radial basis function with l_2-norm 162
PRS	Quadratic polynomial response surface 162

Simulation Model for Performance Analysis

i	Replication 209
i^*	Current replication 209
j	Simulation iteration 209
j^*	Current simulation iteration 209

m_i	Number of simulation iterations of replication i 209
τ	End of the warm-up period 208
$SL_{\beta,i,j}$	β-service level of simulation iteration j in replication i 209
$SL_{\beta,i}$	β-service level of replication i 209
$\Delta_{i^*,j^*,j'}$	Absolute deviation of the average value of β-service level measured after the current simulation iteration j^* from the average value of β-service level measured after simulation iteration j' of the current replication i^* 209
ϵ_R	Precision of a replication 209
$\overline{SL}_{\beta,i^*}$	Average value of β-service level after replication i^* 210
$S^2(SL_{\beta,i^*})$	Variance of β-service level after replication i^* 210
$t_{(i-1),0.975}$	97.5%-quantile of t-distribution with $(i-1)$ degrees of freedom 210
ϵ_S	Precision of the simulation 210

Design of Experiments

λ	Parameter of Poisson distribution 212
η	Parameter of Bernoulli distribution 212
θ	Parameter of Bernoulli distribution 212
K	Poisson-distributed random variable 223
J	Approximately Poisson-distributed random variable 223
$\mathcal{J}$	Range of random variable J 223
j_{min}	Minimum possible value of random variable J 223
j_{max}	Maximum possible value of random variable J 223
Π	Approximate upper bound of Poisson-distributed random variable 224
ϵ_P	Residual probability mass 224

List of Figures

List of Tables

List of Publications

Mohring, U., M. Baumann and K. Furmans (2020). Discrete-Time Analysis of Levelled Order Release and Staffing in Order Picking Systems. *Logistics Research 13*(9), p. 1.

References

Abramson, M. A., C. Audet, G. Couture, J. E. Dennis Jr, V. Rochon Montplaisir and C. Tribes (2018). The NOMAD project. Software available at https://www.gerad.ca/nomad (last visited on 31.05.2021).

Abramson, M. A., C. Audet, J. E. Dennis Jr and S. Le Digabel (2009). OrthoMADS: A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization 20*(2), p. 948–966.

Altendorfer, K. (2014). *Capacity and Inventory Planning for Make-to-Order Production Systems: The Impact of a Customer Required Lead Time Distribution*. Lecture Notes in Economics and Mathematical Systems; 671SpringerLink. Cham: Springer.

Altendorfer, K., A. Hübl and H. Jodlbauer (2014). Periodical capacity setting methods for make-to-order multi-machine production systems. *International Journal of Production Research 52*(16), p. 4768–4784.

Altendorfer, K. and H. Jodlbauer (2011). An analytical model for service level and tardiness in a single machine MTO production system. *International Journal of Production Research 49*(7), p. 1827–1850.

Altendorfer, K. and S. Minner (2011). Simultaneous optimization of capacity and planned lead time in a two-stage production system with different customer due dates. *European Journal of Operational Research 213*(1), p. 134–146.

Altendorfer, K. and S. Minner (2012). Optimal composition of number and size of machines in a multi-stage make-to-order system with due dates. *International Journal of Production Research 50*(3), p. 603–621.

Audet, C. (2014). A Survey on Direct Search Methods for Blackbox Optimization and Their Applications. In: P. M. Pardalos and T. M. Rassias (eds.),

Mathematics Without Boundaries: Surveys in Interdisciplinary Research, p. 31–56. New York, NY: Springer New York.

Audet, C., J. E. Dennis and S. Le Digabel (2010). Globalization strategies for Mesh Adaptive Direct Search. *Computational Optimization and Applications 46*(2), p. 193–215.

Audet, C. and J. E. Dennis Jr (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization 17*(1), p. 188–217.

Audet, C. and J. E. Dennis Jr (2009). A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization 20*(1), p. 445–472.

Audet, C. and W. Hare (2017). *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Cham: Springer.

Audet, C., M. Kokkolaras, S. Le Digabel and B. Talgorn (2018). Order-based error for managing ensembles of surrogates in mesh adaptive direct search. *Journal of Global Optimization 70*(3), p. 645–675.

Audet, C., S. Le Digabel and C. Tribes (2009). NOMAD user guide. Technical Report G-2009-37. Technical report.

Audet, C. and C. Tribes (2018). Mesh-based Nelder–Mead algorithm for inequality constrained optimization. *Computational Optimization and Applications 71*(2), p. 331–352.

Bartholdi III, J. J. and D. D. Eisenstein (1996). A production line that balances itself. *Operations Research 44*(1), p. 21–34.

Baumann, M. (2019). *Discrete time analysis of multi-queue systems with multiple departure streams in material handling and production under different service rules*. Dissertation, Karlsruhe Institute of Technology, Institute for Material Handling and Logistics, Karlsruhe, Germany.

Becker, C. and A. Scholl (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research 168*(3), p. 694–715.

Bergamaschi, D., R. Cigolini, M. Perona and A. Portioli (1997). Order review and release strategies in a job shop environment: A review and a classification. *International Journal of Production Research 35*(2), p. 399–420.

Bertrand, J. W. M. and V. Sridharan (2001). A study of simple rules for subcontracting in make-to-order manufacturing. *European Journal of Operational Research 128*(3), p. 509–531.

Bhosekar, A. and M. Ierapetritou (2018). Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering 108*, p. 250–267.

Blackstone, J. H., D. T. Philips and G. L. Hogg (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research 20*(1), p. 27–45.

Bolch, G. (eds.) (2006). *Queueing networks and Markov chains: Modeling and performance evaluation with computer science applications* (2., rev. and enl. ed.). Hoboken, NJ: Wiley-Interscience.

Boysen, N., R. De Koster and F. Weidinger (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research 277*(2), p. 396–411.

Boysen, N., M. Fliedner and A. Scholl (2009). Level Scheduling for batched JIT supply. *Flexible Services and Manufacturing Journal 21*(1-2), p. 31–50.

Buyukkaramikli, N. C., J. W. M. Bertrand and H. P. Van Ooijen (2013). Periodic capacity management under a lead-time performance constraint. *OR Spectrum 35*(1), p. 221–249.

Buzacott, J. A. and J. G. Shanthikumar (1993). *Stochastic models of manufacturing systems*. Prentice Hall International series in industrial and systems engineering. Upper Saddle River, NJ: Prentice Hall.

Çeven, E. and K. R. Gue (2017). Optimal Wave Release Times for Order Fulfillment Systems with Deadlines. *Transportation Science 51*(1), p. 52–66.

Chen, C.-S., S. Mestry, P. Damodaran and C. Wang (2009). The capacity planning problem in make-to-order enterprises. *Mathematical and Computer Modelling 50*(9-10), p. 1461–1473.

Chew, E. P. and L. C. Tang (1999). Travel time analysis for general item location assignment in a rectangular warehouse. *European Journal of Operational Research 112*(3), p. 582–597.

Cinlar, E. (1975). *Introduction to stochastic processes*. Englewood Cliffs, NJ: Prentice-Hall.

Conn, A. R. and S. Le Digabel (2013). Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software 28*(1), p. 139–158.

Cooper, M. C., D. M. Lambert and J. D. Pagh (1997). Supply chain management: More Than a New Name for Logistics. *The International Journal of Logistics Management 8*(1), p. 1–14.

Croxton, K. L. (2003). The order fulfillment process. *The International Journal of Logistics Management 14*(1), p. 19–32.

Croxton, K. L., S. J. Garcia-Dastugue, D. M. Lambert and D. S. Rogers (2001). The supply chain management processes. *The International Journal of Logistics Management 12*(2), p. 13–36.

Dallery, Y. and S. B. Gershwin (1992). Manufacturing flow line systems: A review of models and analytical results. *Queueing Systems 12*(1-2), p. 3–94.

Davarzani, H. and A. Norrman (2015). Toward a relevant agenda for warehousing research: literature review and practitioners' input. *Logistics Research 8*(1), p. 1–18.

De Koster, R., T. Le-Duc and K. J. Roodbergen (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research 182*(2), p. 481–501.

Dehdari, P. (2014). *Measuring the Impact of Lean Techniques on Performance Indicators in Logistics Operations*. Dissertation, Karlsruhe Institute of Technology, Institute for Material Handling and Logistics, Karlsruhe, Germany.

Dhamala, T. N. and W. Kubiak (2005). A brief survey of just-in-time sequencing for mixed-model systems. *International Journal of Operations Research 2*(2), p. 38–47.

Doytchinov, B., J. Lehoczky and S. Shreve (2001). Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *Annals of Applied Probability 11*(2), p. 332–378.

Fermi, E. (1952). Numerical solution of a minimum problem. Technical Report No. LA-1492, Los Alamos National Laboratory, Los Alamos.

Fernandes, N. O., M. Thürer, T. M. Pinho, P. Torres and S. Carmo-Silva (2020). Workload control and optimised order release: an assessment by simulation. *International Journal of Production Research 58*(10), p. 3180–3193.

Furmans, K. (2005). Models of Heijunka-levelled Kanban-Systems. In: *Proceedings of the Fifth International Conference on "Analysis of Manufacturing Systems - Production Management", Zakynthos, Greece, May 20-25, 2005*, p. 243–248.

Furmans, K., C. Huber and J. Wisser (2009). Queueing Models for manual order picking systems with blocking. *Logistics Journal 1*(1), p. 1–16.

Furmans, K. and M. Veit (2013). Models of Leveling for Lean Manufacturing Systems. In: J. M. Smith and B. Tan (eds.), *Handbook of stochastic models and analysis of manufacturing system operations*, p. 115–138. New York, NY: Springer.

Gong, Y. and R. B. M. De Koster (2011). A review on stochastic models and analysis of warehouse operations. *Logistics Research 3*(4), p. 191–205.

Govil, M. K. and M. C. Fu (1999). Queueing theory in manufacturing: A survey. *Journal of Manufacturing Systems 18*(3), p. 214–240.

Gu, J., M. Goetschalckx and L. F. McGinnis (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research 177*(1), p. 1–21.

Haskose, A., B. G. Kingsman and D. Worthington (2002). Modelling flow and jobbing shops as a queueing network for workload control. *International Journal of Production Economics 78*(3), p. 271–285.

Haskose, A., B. G. Kingsman and D. Worthington (2004). Performance analysis of make-to-order manufacturing systems under different workload control regimes. *International Journal of Production Economics 90*(2), p. 169–186.

Hong, S. (2019). A performance evaluation of bucket brigade order picking systems: Analytical and simulation approaches. *Computers & Industrial Engineering 135*, p. 120–131.

Hooke, R. and T. A. Jeeves (1961). Direct Search Solution of Numerical and Statistical Problems. *Journal of the ACM (JACM) 8*(2), p. 212–229.

Hopp, W. J. and M. L. Spearman (2004). To pull or not to pull: what is the question? *Manufacturing & Service Operations Management 6*(2), p. 133–148.

Huang, H. C., C. Lee and Z. Xu (2006). The workload balancing problem at aircargo terminals. *OR Spectrum 28*(4), p. 705–727.

Irastorza, J. C. and R. H. Deane (1974). A loading and balancing methodology for job shop control. *AIIE Transactions 6*(4), p. 302–307.

Jane, C. C. (2000). Storage location assignment in a distribution center. *International Journal of Physical Distribution & Logistics Management 30*(1), p. 55–71.

Jane, C.-C. and Y.-W. Laih (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. *European Journal of Operational Research 166*(2), p. 489–496.

Jones, D. R., M. Schonlau and W. J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization 13*(4), p. 455–492.

Khodja, L. Z., R. Couturier, A. Giersch and J. M. Bahi (2014). Parallel sparse linear solver with GMRES method using minimization techniques of communications for GPU clusters. *Journal of Supercomputing 69*(1), p. 200–224.

Koziel, S., D. Echeverría Chiaurri and L. Leifsson (2011). Surrogate-Based Methods. In: S. Koziel and X.-S. Yang (eds.), *Computational optimization, methods and algorithms*, p. 33–59. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kundu, K., M. J. Land, A. Portioli-Staudacher and J. A. Bokhorst (2020). Order review and release in make-to-order flow shops: Analysis and design of new methods. *Flexible Services and Manufacturing Journal 33*(3), p. 750–782.

Kundu, K. and A. P. Staudacher (2017). Order review and release methods in internal logistics of a MTO company. In: *Proceedings - 2017 IEEE International Conference on Service Operations and Logistics, and Informatics, Bari*, p. 133–138.

Lai, X., H. Shui, D. Ding and J. Ni (2021). Data-driven dynamic bottleneck detection in complex manufacturing systems. *Journal of Manufacturing Systems 60*(1), p. 662–675.

Larson, J., M. Menickelly and S. M. Wild (2019). Derivative-free optimization methods. *Acta Numerica 28*, p. 287–404.

Law, A. M. (2015). *Simulation modeling and analysis* (5., and international ed.). New York, NY: McGraw-Hill Education.

Lawrence, S. R. and A. H. Buss (1994). Shifting production bottlenecks: causes, cures, and conundrums. *Production and Operations Management 3*(1), p. 21–37.

Le Digabel, S. L. and S. M. Wild (2015). A taxonomy of constraints in simulation-based optimization. *arXiv preprint arXiv:1505.07881*.

Le-Duc, T. and R. M. De Koster (2007). Travel time estimation and order batching in a 2-block warehouse. *European Journal of Operational Research 176*(1), p. 374–388.

Liker, J. K. (2004). *The Toyota way: 14 management principles from the world's greatest manufacturer*. New York, NY: McGraw-Hill.

Lin, F.-R. and M. J. Shaw (1998). Reengineering the order fulfillment process in supply chain networks. *International Journal of Flexible Manufacturing Systems 10*(3), p. 197–229.

Lippolt, C. R. and K. Furmans (2008). Sizing of Heijunka-controlled Production Systems with Unreliable Production Processes. In: T. Koch (eds.), *Lean Business Systems and Beyond*, p. 11–20. Boston, MA: Springer US.

Liu, L. and X.-M. Yuan (2001). Throughput, flow times, and service level in an unreliable assembly system. *European Journal of Operational Research 135*(3), p. 602–615.

Loeppky, J. L., J. Sacks and W. J. Welch (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics 51*(4), p. 366–376.

Ma, J., Y. T. Leung and M. Kamath (2019). Service system design under information uncertainty: Insights from an M/G/1 model. *Service Science 11*(1), p. 40–56.

MacCarthy, B. L., L. Zhang and L. Muyldermans (2019). Best performance frontiers for buy-online-pickup-in-store order fulfilment. *International Journal of Production Economics 211*, p. 251–264.

Marchet, G., M. Melacini and S. Perotti (2015). Investigating order picking system adoption: a case-study-based approach. *International Journal of Logistics Research and Applications 18*(1), p. 82–98.

Matl, P., R. F. Hartl and T. Vidal (2018). Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science 52*(2), p. 239–260.

Matzka, J., M. Di Mascolo and K. Furmans (2012). Buffer sizing of a Heijunka Kanban system. *Journal of Intelligent Manufacturing 23*(1), p. 49–60.

Mezzogori, D., G. Romagnoli and F. Zammori (2021). Defining accurate delivery dates in make to order job-shops managed by workload control. *Flexible Services and Manufacturing Journal 33*(4), p. 956–991.

Mincsovics, G. and N. P. Dellaert (2009). Workload-dependent capacity control in production-to-order systems. *IIE Transactions 41*(10), p. 853–865.

Mohring, U., M. Baumann and K. Furmans (2020). Discrete-Time Analysis of Levelled Order Release and Staffing in Order Picking Systems. *Logistics Research 13*(9), p. 1.

Müller, J. (2014). MATSuMoTo: The MATLAB surrogate model toolbox for computationally expensive black-box global optimization problems. *arXiv preprint arXiv:1404.4261*.

Müller, J., C. A. Shoemaker and R. Piché (2014). SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. *Journal of Global Optimization 59*(4), p. 865–889.

Müller, J. (2014). The MATLAB Surrogate Model Toolbox MATSuMoTo. Software available at https://github.com/Piiloblondie/MATSuMoTo (last visited on 31.05.2021).

Nelder, J. A. and R. Mead (1965). A simplex method for function minimization. *The Computer Journal 7*(4), p. 308–313.

Okongwu, U., M. Lauras, L. Dupont and V. Humez (2012). A decision support system for optimising the order fulfilment process. *Production Planning & Control 23*(8), p. 581–598.

Öner-Közen, M. and S. Minner (2017). Impact of priority sequencing decisions on on-time probability and expected tardiness of orders in make-to-order production systems with external due-dates. *European Journal of Operational Research 263*(2), p. 524–539.

Pan, J. C.-H. and M.-H. Wu (2012). Throughput analysis for order picking system with multiple pickers and aisle congestion considerations. *Computers & Operations Research 39*(7), p. 1661–1672.

Pan, J. C.-H., M.-H. Wu and W.-L. Chang (2014). A travel time estimation model for a high-level picker-to-part system with class-based storage policies. *European Journal of Operational Research 237*(3), p. 1054–1066.

Papadopoulos, C. T., J. Li and M. E. O'Kelly (2019). A classification and review of timed Markov models of manufacturing systems. *Computers & Industrial Engineering 128*, p. 219–244.

Papadopoulos, H. and C. Heavey (1996). Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research 92*(1), p. 1–27.

Peeters, K. and H. Van Ooijen (2020). Hybrid make-to-stock and make-to-order systems: a taxonomic review. *International Journal of Production Research 58*(15), p. 4659–4688.

Pinedo, M. (2009). *Planning and Scheduling in Manufacturing and Services* (2 ed.). New York, NY: Springer New York.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems* (5 ed.). Cham: Springer.

Portioli-Staudacher, A. and M. Tantardini (2012). A lean-based ORR system for non-repetitive manufacturing. *International Journal of Production Research 50*(12), p. 3257–3273.

Privault, N. (2013). *Understanding Markov Chains: Examples and Applications*. Springer Undergraduate Mathematics Series. Singapore: Springer Singapore.

Rieck, J., J. Zimmermann and T. Gather (2012). Mixed-integer linear programming for resource leveling problems. *European Journal of Operational Research 221*(1), p. 27–37.

Rios, L. M. and N. V. Sahinidis (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization 56*(3), p. 1247–1293.

Rouwenhorst, B., B. Reuter, V. Stockrahm, G.-J. Van Houtum, R. Mantel and W. H. Zijm (2000). Warehouse design and control: Framework and literature review. *European Journal of Operational Research 122*(3), p. 515–533.

Saad, Y. (2003). *Iterative methods for sparse linear systems* (2 ed.). Other titles in applied mathematics; 82. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM).

Saad, Y. and M. H. Schultz (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing 7*(3), p. 856–869.

Santner, T. J., B. J. Williams and W. I. Notz (2018). *The Design and Analysis of Computer Experiments* (2 ed.). Springer Series in Statistics. New York, NY: Springer New York.

Schleyer, M. (2007). *Discrete time analysis of batch processes in material flow systems*. Dissertation, Karlsruhe Institute of Technology, Institute for Material Handling and Logistics, Karlsruhe, Germany.

Schleyer, M. and K. Furmans (2007). An analytical method for the calculation of the waiting time distribution of a discrete time G/G/1-queueing system with batch arrivals. *OR Spectrum 29*(4), p. 745–763.

Schleyer, M. and K. Gue (2012). Throughput time distribution analysis for a one-block warehouse. *Transportation Research Part E: Logistics and Transportation Review 48*(3), p. 652–666.

Slotnick, S. A. (2011). Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research 212*(1), p. 1–11.

Smalley, A. and J. Womack (2004). *Creating level pull: A lean production-system improvement guide for production-control, operations, and engineering professionals; A lean toolkit method and workbook* (1 ed.). Lean toolkit workbooks. Brookline, MA: The Lean Enterprise Institute.

Sóbester, A., A. I. Forrester, D. J. Toal, E. Tresidder and S. Tucker (2012). Engineering design applications of surrogate-assisted optimization techniques. *Optimization and Engineering 15*(1), p. 243–265.

Souza, G. C. and M. E. Ketzenberg (2002). Two-stage make-to-order remanufacturing with service-level constraints. *International Journal of Production Research 40*(2), p. 477–493.

Stewart, W. J. (1994). *Introduction to the numerical solution of Markov chains*. Princeton, NJ: Princeton Univ. Press.

Tang, L. C. and E.-P. Chew (1997). Order picking systems: batching and storage assignment strategies. *Computers & Industrial Engineering 33*(3-4), p. 817–820.

Thürer, M., M. Stevenson, M. J. Land and L. D. Fredendall (2019). On the combined effect of due date setting, order release, and output control: An assessment by simulation. *International Journal of Production Research 57*(6), p. 1741–1755.

Thürer, M., M. Stevenson and C. Silva (2011). Three decades of workload control research: A systematic review of the literature. *International Journal of Production Research 49*(23), p. 6905–6935.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization 7*(1), p. 1–25.

Van Der Gaast, J. P., R. B. de Koster, I. J. Adan and J. A. Resing (2020). Capacity analysis of sequential zone picking systems. *Operations Research 68*(1), p. 161–179.

Van Gils, T., K. Ramaekers, A. Caris and M. Cools (2017). The use of time series forecasting in zone order picking systems to predict order pickers' workload. *International Journal of Production Research 55*(21), p. 6380–6393.

Van Gils, T., K. Ramaekers, A. Caris and R. B. De Koster (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research 267*(1), p. 1–15.

Van Nieuwenhuyse, I. and R. B. De Koster (2009). Evaluating order throughput time in 2-block warehouses with time window batching. *International Journal of Production Economics 121*(2), p. 654–664.

Vanheusden, S., T. van Gils, K. Braekers, K. Ramaekers and A. Caris (2021). Analysing the effectiveness of workload balancing measures in order picking operations. *International Journal of Production Research 0*(0), p. 1–25.

Vanheusden, S., T. Van Gils, A. Caris, K. Ramaekers and K. Braekers (2020). Operational workload balancing in manual order picking. *Computers & Industrial Engineering 141*, p. 106269.

Vazquez, E. and J. Bect (2011). Sequential search based on kriging: convergence analysis of some algorithms. *arXiv preprint arXiv:1111.3866*.

Viana, F. A., R. T. Haftka and L. T. Watson (2013). Efficient global optimization algorithm assisted by multiple surrogate techniques. *Journal of Global Optimization 56*(2), p. 669–689.

Vu, K. K., C. d'Ambrosio, Y. Hamadi and L. Liberti (2017). Surrogate-based methods for black-box optimization. *International Transactions in Operational Research 24*(3), p. 393–424.

White Jr, K. P. (1997). An effective truncation heuristic for bias reduction in simulation output. *Simulation 69*(6), p. 323–334.

Xu, X., T. Liu, K. Li and W. Dong (2014). Evaluating order throughput time with variable time window batching. *International Journal of Production Research 52*(8), p. 2232–2242.

Yaman, H., O. E. Karasan and B. Y. Kara (2012). Release time scheduling and hub location for next-day delivery. *Operations Research 60*(4), p. 906–917.

Yan, T., E. Rabinovich, K. Dooley and P. T. Evers (2010). Managing backlog variation in order fulfillment: The case of Internet retailers. *International Journal of Production Economics 128*(1), p. 261–268.

Yu, M. and R. B. De Koster (2009). The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research 198*(2), p. 480–490.