Multi-Batch Sequencing with Decentralized Control using Highest-Density Conveyor Networks

Zur Erlangung des akademischen Grades einer

DOKTORIN DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

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

angenommene

DISSERTATION

von

M.Sc. Julia Simone Fleischmann

Tag der mündlichen Prüfung:	02.08.2023
Hauptreferent:	Prof. Dr.-Ing. Kai Furmans
Korreferent:	Prof. Dr.-Ing. Johannes Fottner

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Institut für Fördertechnik und Logistiksysteme des Karlsruher Instituts für Technologie. Es war eine prägende und lehrreiche Zeit, in der ich zahlreiche neue Dinge erlernen und mich persönlich wie beruflich weiterentwickeln durfte, sodass am Ende diese Dissertation entstehen konnte.

Ich bedanke mich bei meinem Doktorvater Prof. Dr.-Ing. Kai Furmans für die Ermöglichung meiner Promotion sowie die Übernahme des Hauptreferats. Seine Überzeugungen und Denkweisen haben mich stets inspiriert. Besonders geschätzt habe ich die Möglichkeit zum vollständig freien und selbstständigen Arbeiten auf Basis grundlegenden Vertrauens, welches er seinen Mitarbeiterinnen und Mitarbeitern von Anfang an entgegenbringt. Prof. Dr.-Ing. Johannes Fottner danke ich für die Übernahme des Korreferats sowie die angenehme Prüfungsatmosphäre. Bei Herrn Prof. Dr. rer. nat. Martin Dienwiebel bedanke ich mich für die Übernahme des Prüfungsvorsitzes meiner Promotionsprüfung.

Allen aktiven und ehemaligen Kolleginnen und Kollegen danke ich für die angenehme und entspannte Arbeitsatmosphäre am Institut und die gegenseitige Hilfsbereitschaft. Vor allem geht mein Dank hier an meine zwei Raumis Uta und Christoph, mit denen ich zahlreiche wertvolle und gleichzeitig humorvolle Stunden in unserem gemeinsamen Büro verbringen durfte. Für alles, was wir in dieser Zeit zusammen gelernt und erlebt haben, bin ich sehr dankbar.

Ich möchte mich bei meiner Familie und all meinen Freunden bedanken, die mich stets mit Verständnis in meinem Vorhaben bestärkt haben. Insbesondere gilt der Dank meinen Eltern Christiane und Arno dafür, dass sie mir diesen Bildungsweg ermöglicht haben. Ihr habt mich mein ganzes Leben als lehrreiche

Vorbilder begleitet und stets nach Kräften unterstützt. Dafür danke ich Euch von Herzen.

Ganz besonders bedanken möchte ich mich bei meinem Freund Simon, der einen großen Teil dieses Weges gemeinsam mit mir bestritten hat. Danke für Deine bedingungslose Unterstützung und unsere unzähligen konstruktiven Diskussionen, dafür, dass du immer ein offenes Ohr für mich hattest, aber vor allem dafür, niemals an mir zu zweifeln.

Karlsruhe, September 2023 Julia Fleischmann

Kurzfassung

Die Entwicklungen der letzten Jahre haben im industriellen Umfeld heutzutage eine nie dagewesene Komplexität, Dynamik und Unsicherheit erzeugt. Im Zuge anhaltender Globalisierung sind riesige, weltweit verteilte Netzwerke entstanden, die eine effiziente Gestaltung und Steuerung ihrer Informations- und Materialflüsse erfordern. Um langfristig am Markt zu bestehen, sind ausgefeilte automatisierte Systeme unerlässlich. Hier werden Steuerungsarchitekturen zunehmend dezentralisiert, indem die Intelligenz des Gesamtsystems auf mehrere, unabhängige funktionale Einheiten verteilt wird. Selbst für große und komplexe Systeme gewährleistet dies Flexibilität, Skalierbarkeit, Robustheit und Echtzeitfähigkeit auch unter unsicheren und volatilen Einsatzbedingungen. Darüber hinaus erfordern abnehmende Lagerbestände und reduzierte Pufferkapazitäten eine Umgestaltung gegenwärtiger Materialflusskonzepte, da benötigte Güter gemäß der Verbrauchsreihenfolge an ihrem Bedarfsort verfügbar sein müssen. Deshalb verspricht ein automatisiertes, dezentral gesteuertes System zur sequenzgetreuen Materialbereitstellung eine Optimierung der Materialflüsse für diverse Anwendungen im Logistik- wie Produktionssektor.

Die vorliegende Dissertation stellt einen dezentralen Algorithmus vor, der die gleichzeitige Sequenzierung mehrerer Batch-Aufträge in höchst-dichten Stetigfördernetzwerken ermöglicht. Dabei werden materielle Objekte, die von einem vorgelagerten Prozess in zufälliger Reihenfolge am System ankommen, so umgeordnet, dass sie einem nachgelagerten Prozess in der dort benötigten Reihenfolge am gewünschten Bedarfsort zur Verfügung stehen. Der entwickelte Algorithmus bedient mehrere Eingangs- und Ausgangspunkte zur parallelen Auftragsbearbeitung. Lokale Entscheidungen auf Basis lokaler Datenhaltung

führen durch die Interaktion autonomer Systemelemente zur Gesamtfunktionalität. Aufgrund des modularen, dezentralen Systemaufbaus sind eingesetzte Sequenziersysteme hochflexibel, robust und skalierbar, sodass sie für spezifische Anwendungsszenarien individuell zugeschnitten werden können.

Die Sequenzierung basiert auf einer dynamischen Pufferstrategie, bei der zwischengelagertes Material bei Bedarf verschoben wird, um Routen zeit- und entfernungseffizient durch das Netzwerk zu planen. Hierbei liegt das Konzept der logischen Zeit zugrunde, wodurch die Ausführung der geplanten Transporte dezentral koordiniert werden kann.

Der vorgestellte dezentrale Sequenzierungsalgorithmus schließt Deadlocks, Livelocks und Starvation sowohl innerhalb der algorithmischen Operationen als auch der Ressourcenallokation grundsätzlich aus, wodurch die Lebendigkeit des Systems zu jedem Zeitpunkt sichergestellt ist. Der erforderliche Kommunikationsaufwand skaliert mit der Systemgröße n. Auf Netzwerkebene liegt die algorithmische Komplexität damit in der Größenordnung $\mathcal{O}(n^3)$, was auf Ebene der dezentralen Systemelemente auf $\mathcal{O}(n^2)$ reduziert wird.

Die Qualität dezentral generierter Lösungen kann für kleine Probleminstanzen im Vergleich zu optimalen Ergebnissen bewertet werden. Daraus ergibt sich ein durchschnittliches Lösungsdefizit von weniger als 20 %. Gemessen an einer analytisch berechneten unteren Schranke zeigt der dezentrale Sequenzierungsalgorithmus außerdem eine konstante Lösungsqualität über Systemvariationen größerer Problemstellungen hinweg.

Mithilfe simulativer Leistungsanalysen werden Batchgröße, vorgegebene Vorgänger-Nachfolger Beziehungen, der Bedarf an Umordnung der Objekte innerhalb des Systems sowie die verfügbare Ausschleusekapazität des Netzwerks als wesentliche Einflussgrößen des erzielbaren Grenzdurchsatzes identifiziert. Außerdem erweisen sich kompakte, hochvernetzte Systeme und geradlinige Wege zwischen Ein- und Ausgängen als vorteilhaft.

Abstract

Due to the developments in recent years, today's industrial environments have become highly complex, dynamic, and uncertain. These are integrated within huge, globally distributed networks where the flow of information and materials need to be organized efficiently. Sophisticated, automated systems are crucial to ensure long-term productivity. Their control architectures are increasingly decentralized by distributing the overall intelligence to several independent functional entities. Even for large-scale and complex systems, this guarantees flexibility, scalability, robustness, and real-time capability under uncertain and volatile conditions. Additionally, decreasing inventory levels and limited buffer space complicate current material flow control, as required items need to be supplied at their point of use in the order of consumption. Therefore, an automated system based on decentralized control enabling sequenced item supply is able to enhance various material handling applications within warehouse operations or production systems.

This dissertation presents a decentralized algorithm for multi-batch sequencing using highest-density conveyor networks. Physical objects randomly arriving from an upstream process are reorganized within the system to be provided to a downstream process observing the correct sequence at the required point of consumption. The developed algorithm handles multiple input and output points which allows parallel order processing. Local decisions based on local databases yield the overall system functionality using the interaction of autonomous entities. Due to the modular, decentralized system setup, installed sequencing systems are highly flexible, robust, and scalable such that they can be individually customized for specific application scenarios.

Sequencing is realized using a dynamic buffering approach where buffered physical objects can be relocated to plan efficient paths regarding time and distance within the network. These are scheduled based on the concept of logical time which enables decentralized transport coordination.

The presented decentralized sequencing algorithm ensures system liveliness at any point in time by generally preventing deadlocks, livelocks and starvation within algorithmic operations as well as resource allocation. The required communication effort scales with system size n. At network level, we obtain an algorithmic complexity of order $\mathcal{O}(n^3)$ which reduces to $\mathcal{O}(n^2)$ at level of the decentralized system elements.

We assess the quality of solutions provided by the decentralized sequencing algorithm compared to optimal results for small problem settings. This yields an average decentralized solution deficit of less than 20 %. Larger problem settings are benchmarked against a lower bound approximation where the algorithmic quality level is maintained throughout the system variations with larger problem sizes.

Simulative performance evaluations indicate that batch size, predefined predecessor-successor dependencies, the rearrangement effort of objects within the system as well as the available outflow capacity of a network are key factors influencing the achieved limiting throughput. Additionally, we recommend compact system setups with highly connected network structures and straight paths from input to output nodes.

Contents

1 Introduction

Dynamics – uncertainty – complexity. These three factors fundamentally and comprehensively influenced industrial environments over the past decades (Spath et al. 2013). Globally distributed actors are intertwined in huge networks, where the flow of information and materials need to be organized such that performance goals are met (Wang 2016). To operate efficiently under these circumstances, sophisticated automated systems are vital, while additionally reducing manual work increases accuracy and speed of industrial applications. This sets the basis for the topic of this dissertation (cf. Section 1.1). We derive six research objectives to be achieved within the investigations (cf. Section 1.2). In Section 1.3, we outline the structure of this dissertation organized in nine chapters.

1.1 Problem Description

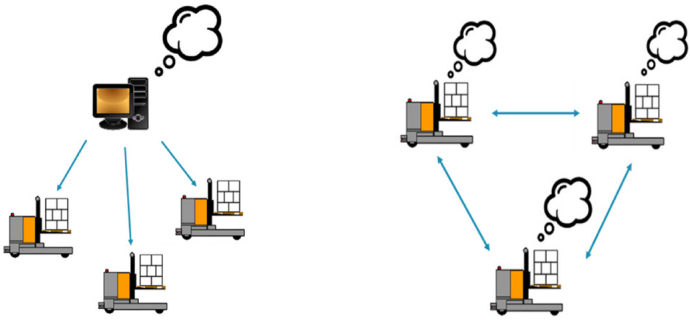
Due to increasing global interconnections of market participants, inside as well as outside their organizations, huge networks involving material, information and financial flows emerged (Lanza et al. 2019). Simultaneously, we observe substantial changes arising from customer requirements (Kartnig et al. 2012). They expect high-quality, uniquely individualized products which are immediately available at minimum cost (Kumar 2007). Therefore, companies offer a vast number of different product variants produced at low volume (Johansen et al. 2021). Additionally, striving for continuously decreasing stock levels and inventory investments favors a make-to-order production based on customer

demand reducing feasible lot sizes (Chikán et al. 2016, Spath et al. 2013). This results in high volatility and major sales fluctuations (Spath et al. 2013).

To cope with these changes, innovative production and material handling systems as well as processes are essential (Wang 2016, Spath et al. 2013, Günthner et al. 2008). The main success factors for long-term competitiveness in the market have shifted from costs or performance goals to flexibility, availability, and adaptability (Furmans et al. 2011). Thus, controlling the flow of information and materials requires agile processes and highly responsive automated systems, which cannot be realized using traditional rigid centralized control architectures (Kartnig et al. 2012). These are designed such that one single higher-level element monitors the entire system using a comprehensive database (Rana and Taneja 1988). This strives for global system optimization at the cost of a single point of failure, inflexibility concerning modifications, limited processing capacity, and long response times (Dilts et al. 1991, Rana and Taneja 1988, Duffie et al. 1988).

Therefore, control architectures of automated systems are increasingly decentralized by eliminating hierarchical layers (cf. Figure 1.1) (Martins et al. 2020, Rana and Taneja 1988). Splitting processing workload, information, and control allows the intelligence of the system to be distributed among several independent functional entities (de Ryck et al. 2020, Furmans et al. 2010, Duffie 1990). These are interconnected and operate autonomously by making local decisions based on local information (Dilts et al. 1991, Günthner and ten Hompel 2010, p. 6ff.). Even for large-scale and complex systems, this provides flexibility regarding system adaptions to changing requirements, scalability regarding varying system sizes, and robustness, i.e., fault tolerance regarding failures of single control entities (Furmans et al. 2011, 2010, Duffie 1990, Duffie et al. 1988). Overall, we achieve quick reactions and thus real-time capability under uncertain and volatile conditions (Trentesaux 2009).

In addition to redesigning traditional control architectures, automated systems in industrial applications need to provide increasingly sophisticated functionalities to guarantee efficiency. Decreasing inventories and buffer space allows for cost



(a) Centralized control architecture. **(b)** Decentralized control architecture.

Figure 1.1: Control architectures (de Ryck et al. 2020).

reductions but complicates current material flow control. Required items need to be supplied at their point of use in the order of consumption, as there is no space or capacity to reorganize them. Therefore, in the field of production, the concept of Just-in-Sequence (JiS) is widespread. It provides a high level of flexibility to operate competitively in dynamic and customer-oriented markets (Thun et al. 2007). Part deliveries are synchronized with predefined production sequences, in which these parts need to be assembled into the final product (Wannenwetsch 2014, p. 188ff.). The part required next is always available without wasting time for identification or searching (Boysen et al. 2015). This enables a high number of variants to offer individually customized products and shortens the time-to-customer due to efficient order processing. Additionally, inventory and therefore space requirements and capital lockup are reduced.

In other areas, such as warehouse operations or cross docking, inefficiencies occur due to additional searching or sorting if goods from an upstream process are not provided in the correct sequence as required for the downstream process. The overall flow of materials is only selectively optimized, missing an integrated consideration of interactions and dependencies among upstream and downstream processes (Azadeh et al. 2019). Retrieving from an automated storage and retrieval system typically focuses on throughput optimization. However, in-sequence supply from the warehouse to the picking zone, as required to

process customer orders, can increase the performance of the picking process (Lienert and Fottner 2018). When palletizing goods from an automated warehouse, the pick and pack process needs to satisfy a given sequence of customer orders. Finished packages are loaded into some kind of vehicle for delivery. Depending on the size of the vehicle and the number of parcels, the order in which the goods are delivered to their destinations is driving the amount of work involved when unloading the vehicle on its tour. A loading strategy based on the unloading sequence can reduce manual work during delivery and thus increase efficiency. In cross docking, sojourn times of trucks at the handling location can be reduced by supplying gates in the correct sequence based on the delivering order (Lienert and Fottner 2018).

All outlined applications have in common that supplying physical objects according to the sequence as required at their point of demand enhances the overall flow of materials. This includes, initially, *determining* a specific sequence, in which these objects should appear to benefit the underlying application. Subsequently, they need to be provided at the desired point of demand observing the predefined sequence, i.e. the sequence needs to be physically *realized*.

Sequence determination is addressed in several existing scientific problems in the area of warehouse operations, cross docking, or production (Lienert and Fottner 2018). These comprise aligning picking or retrieval operations with incoming customer orders in manual or automated storage systems (de Koster et al. 2007, Gu et al. 2007, Gagliardi et al. 2014), specifying the sequence in which trucks are processed at dock doors (Buakum and Wisittipanich 2019), scheduling jobs to machines in production systems to optimize certain performance criteria (Pinedo 2016, p. 7), or feeding an assembly line with a specific sequence of product variants to balance the workload of assembly stations (Boysen et al. 2009, Gravel et al. 2005). In this dissertation, we address sequence realization, i.e. we assume the predefined sequence of physical objects to be determined in advance. However, these objects do not observe this sequence when arriving at their point of use and need to be physically rearranged accordingly. Using an automated system based on decentralized control, we aim to provide sequenced supply of physical objects to support various practical

applications. This will contribute to efficient production and material handling processes within dynamic, uncertain, and complex industrial environments.

1.2 Research Objectives

We address the introduced problem by presenting a decentralized algorithm for multi-batch sequencing using highest-density[1] conveyor networks. This contribution comprises:

- defining requirements of network arrangements for sequencing,
- describing the decentralized sequencing algorithm,
- proving system liveliness,
- investigating the algorithmic complexity,
- assessing the quality of decentralized solutions, and
- deriving recommendations for practical applications.

This results in six research objectives, each of which we intend to address using one research question.

Current production and material handling systems are complex in terms of their structure, processes, and size. Furthermore, space in production and logistics facilities is becoming ever more limited (Gue 2006). High(est)-density systems are able to operate even when space is scarce. However, sequenced supply of physical objects requires certain network characteristics to be able to rearrange unordered physical objects. Thus, we pose the first research question:

> **1: Which requirements are necessary for conveyor networks to enable multi-batch sequencing at highest density?**

[1] *Highest-density* is an extreme case of *high-density* as defined in Gue (2006). Details are given in Chapter 2.

In order to cope with the increased challenges in complex industrial environments, decentralized control concepts emerged (cf. Section 1.1). These enable flexibility and responsiveness in dynamic and uncertain situations while preserving robustness and scalability. Therefore, they outperform traditional centralized rigid control concepts. Sequenced supply of materials to their point of use improves the performance within a variety of production and material handling applications and enables efficient order fulfillment processes. This results in the second research question:

2: How can we enable multi-batch sequencing of physical objects using decentralized controlled highest-density conveyor networks?

In decentralized systems, the overall functionality follows from the interactions of several autonomous entities. These independently claim available resources without a higher-level element supervising the whole system. Due to distributed information and decision making, decentralized systems are prone to create situations from which they – partially or entirely – can no longer progress (Trentesaux 2009). To guarantee stable system operation, we need to ensure system liveliness at any point in time. This yields the third research question:

3: How can we demonstrate the liveliness of the decentralized sequencing system?

Decentralized controlled systems involve communication efforts to coordinate the distributed entities (Monostori et al. 2015). It increases the more information is exchanged, which consumes available capacity for data processing and communication. Running the decentralized sequencing algorithm within a simulation environment, but especially on a physical system needs to satisfy real-time requirements. Using complexity analysis, we can estimate the algorithmic efficiency during runtime (Skiena 2008, p. 31ff.). Therefore, we investigate the fourth research question:

4: How can we evaluate the complexity of the decentralized sequencing algorithm?

Local decision-making of independent autonomous decisional entities complicates global optimization at system level (Dilts et al. 1991). Decentralized systems are likely to result in sub-optimal solutions, as decisions are based on incomplete information due to distributed local databases (de Ryck et al. 2020). This offers the known benefits of flexibility, robustness, and scalability. Nevertheless, sufficiently high system performance is necessary to achieve efficient processes (Trentesaux 2009). Thus, we derive the fifth research question:

5: How do decentralized solutions for sequencing problems compare to optimal solutions?

To efficiently support practical applications, theoretical results need to meet the requirements of industrial environments. The achievable sequencing throughput of a specific system configuration is decisive for whether an installed system will be profitable or not. System throughput is influenced by a variety of parameters concerning the individual underlying application. These interdependencies need to be studied and understood to purposefully improve practical applications. From this follows the sixth research question:

6: What throughput predictions and recommendations can we derive for sequencing in practical applications?

1.3 Structure of the Dissertation

To address the introduced research objectives, we structure this dissertation into nine chapters. Figure 1.2 assigns the resulting research questions to the corresponding chapter they are covered in.

After presenting the problem description and research objectives, we review related topics from scientific literature to derive the research gap in Chapter 2. Chapter 3 defines the system, which forms the basis for all concepts and models developed in the subsequent chapters. This includes specifying the necessary system requirements of conveyor networks for sequencing to respond to the

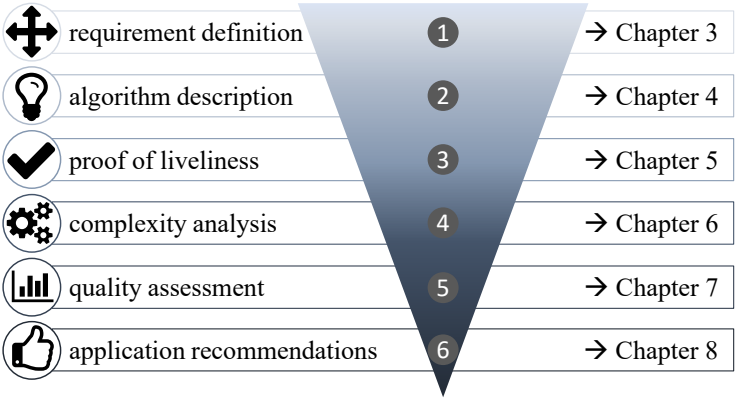


Figure 1.2: Structure of the dissertation.

first research question. In Chapter 4, we present the decentralized algorithm for multi-batch sequencing in highest-density conveyor networks providing an answer to the second research question. Referring to the third research question, we prove that liveliness is always guaranteed within the decentralized sequencing system in Chapter 5. Chapter 6 analyzes the algorithmic complexity in terms of the message processing effort to respond to the fourth research question. In Chapter 7, we assess the quality of decentralized solutions by investigating the deficit compared to optimal solutions. This enables answering the fifth research question. Throughput analyses to derive recommendations for practical application scenarios are given in Chapter 8, which responds to the sixth research question. In Chapter 9, we summarize the main findings of this dissertation and present an outlook on further research opportunities.

2 Literature Review

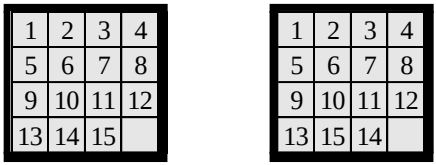
We structure the literature review according to the related research topics

- high-density conveyor systems (cf. Section 2.1),
- decentralized algorithms (cf. Section 2.2), and
- sequencing systems (cf. Section 2.3).

From the intersection of these three areas, we derive the research gap of this dissertation (cf. Section 2.4).

2.1 Centralized Algorithms for High-Density Conveyor Systems

Centralized algorithms for high-density conveyor systems originate from the 15-puzzle invented in the 19th century. On a square field containing 16 cells, 15 of them are occupied by tiles numbered 1, . . . , 15 leaving a blank space at the last position. Originally, the aim of the puzzle is to swap the positions of tiles number 14 and 15 using a series of legal moves, while the remaining tiles are found at their original positions (cf. Figure 2.1) (Archer 1999). Alternative variants intend to transform a randomly arranged puzzle into the initial numerically ordered configuration. Basically, the 15-puzzle can be generalized to any $(n \cdot m - 1)$-puzzle of size $n \times m$ for all integers m, n greater than 1 (Parberry 2015).



(a) Start configuration. **(b)** Goal configuration.

Figure 2.1: The 15-puzzle game.

A legal move of this puzzle consists of sliding an occupied cell adjacent to the empty cell horizontally or vertically into the empty cell, such that they swap their positions (Archer 1999). The puzzle should be transferred from the initial to the goal configuration using the minimum number of moves (Parberry 2015). Ratner and Warmuth (1990) prove this problem to be NP-hard. Using the theory of even and odd permutations, Johnson and Story (1879) demonstrate that even permutations can only be formed from even permutations and vice versa. Consequently, the 15-puzzle can only be solved for exactly half of the 16! possible initial configurations, i.e., the configurations of Figure 2.1 are impossible to obtain from each other (Archer 2007).

In the field of material handling systems, the 15-puzzle game is a fundamental base concerning storage system design, particularly in cases with limited space and high demand for storage locations (Kota et al. 2015). The density of a storage system depends on the space required to store a given number of items. Increasing storage density reduces storage costs. Therefore, *puzzle-based storage systems* aim to maximize storage density (Gue and Kim 2007). They consist of square storage cells arranged in a $m \times n$ grid, like in a 15-puzzle game. Each cell is either occupied with a single item or empty (Yalcin et al. 2019a). Empty cells are called *escorts* and enable moving the occupied cells (Kota et al. 2010). Thus, the maximum attainable storage density of a puzzle-based storage system corresponds to $\frac{m \cdot n - 1}{m \cdot n}$ (Gue and Kim 2007). A common measure used to evaluate the performance of storage systems is the retrieval time of an item (Yu et al. 2017). Generally, there is a tradeoff between storage density and the required retrieval time. Taylor and Gue (2008) investigate these

relations using simulation for different application scenarios in terms of initial escorts positioning, allocating storage locations, and demand distribution.

A variety of algorithms have been developed to determine item retrieval paths in puzzle-based storage systems using optimal and heuristic approaches. Table 2.1 classifies these regarding the presented sequencing problem. The main objective is to minimize the number of movements required to retrieve requested items. Different system densities – measured in the number of escorts – are investigated. We exclude from this analysis approaches which rely on the virtual aisle strategy (cf. Zaerpour et al. 2017a,b), as in the presented sequencing system the number of empty cells may not be sufficient to form a complete virtual aisle. If the system provides more than one escort, multiple independent movements are possible simultaneously, which may reduce retrieval times. Note that in this case, minimizing the retrieval time is no more equivalent to minimizing the number of movements. Additionally, we investigate the number of items being retrieved in parallel as well as that of available input/output (I/O) points within the studied systems. To reflect the material flows of a multi-batch sequencing application, both need to be greater than one.

Gue and Kim (2007) provide an optimal algorithm for retrieving items in puzzle-based storage systems with one single escort placed at the I/O point in the lower left corner. Furthermore, they investigate problem settings with multiple escorts lined up at the I/O point along the horizontal axis. A dynamic programming formulation to optimally solve small problem sizes as well as a heuristic to solve larger ones are presented. Based on these findings, Kota et al. (2010) develop optimal analytical results for retrieving items in puzzle-based storage systems with a single escort placed randomly within the grid and with two escorts – one placed at the I/O point, the other one remaining randomly positioned. For a general number of escorts, they propose an integer programming formulation. Puzzle-based storage systems with one I/O point and entirely random initial escort positioning are studied in Kota et al. (2015). They provide analytical results for one or two escorts, while heuristic solutions with associated worst-case bounds are presented for systems in which the number of escorts exceeds two. Ma et al. (2022) develop a heuristic search algorithm based on state

Table 2.1: Centralized algorithms for high-density conveyor systems.

Publication	Method	Escorts	Simultaneous movements	Item retrieval	I/O points
Gue and Kim (2007)	optimal	1[1]	no	1	1
	optimal, heuristic	k[1]	no	1	1
Kota et al. (2010)	optimal	1,2[1]	no	1	1
	optimal[2]	k	no	1	1
Kota et al. (2015)	optimal	1,2	no	1	1
	heuristic	k	no	1	1
Ma et al. (2022)	heuristic	k	no	1	1
Yu et al. (2017)	optimal	k	yes	1	1
Mirzaei et al. (2017)	optimal, heuristic	1[1]	no	m	1
Yalcin et al. (2019a)	optimal, heuristic	k	no	1	n
Rosenfeld (2022)	heuristic	k	no	m	n
Zou and Qi (2021)	heuristic	k	no	m	n
Yalcin et al. (2019b)	heuristic	k	yes	m	n
Bukchin and Raviv (2020)	optimal	k	yes	1	n
Bukchin and Raviv (2022)	optimal, heuristic	k	yes	1	n
This dissertation	optimal	k	yes	m	n

appraisal, neighborhood search, and beam search with improved accuracy and computational efficiency compared to existing approaches for the same problem setting.

[1] at least one of the escorts not placed at an arbitrary position
[2] only integer programming formulation without solutions

To improve system performance, developed approaches include simultaneous movements, where several items can be moved independently at the same time. Yu et al. (2017) study those systems with a single I/O point and multiple escorts at random positions. Based on an integer programming model, they develop an approach to find an optimal path for the requested item to the I/O point with minimum retrieval time considering necessary item movements. Moreover, simultaneously retrieving multiple items by moving them to a common location and then transferring them jointly to the I/O point allows reducing their overall retrieval time. Mirzaei et al. (2017) investigate systems in which one escort is initially located at the I/O point. They propose optimal solutions for the 2-item retrieval problem. A heuristic is suggested for larger numbers of retrieved items.

Several approaches address puzzle-based storage systems with multiple I/O points. Yalcin et al. (2019a) develop an exact and a heuristic A*-based search algorithm for retrieving a single item with multiple randomly positioned escorts moving sequentially. Extending this problem to multiple items to be retrieved is addressed in Rosenfeld (2022). Likewise, they formulate the problem as a graph search problem and propose a set of heuristic algorithms. Simulation is used to evaluate the different solution variants. Additionally, they investigate the runtime-memory tradeoff. Another heuristic approach for retrieving multiple items in puzzle-based storage systems with multiple randomly positioned escorts and multiple I/O points is given in Zou and Qi (2021). They model the problem as a Markovian decision process providing sequential movements based on the current system state. Yalcin et al. (2019b) develop an adapted multi-agent route planning heuristic for problem settings incorporating both storing and retrieving multiple items, where simultaneous movements are possible. The multiple item problem is reduced to a sequential single item problem. Using simulation, the authors evaluate their approach with regard to different storage and retrieval strategies. Optimal solutions to the single item retrieval problem in puzzle-based storage systems, where items can move simultaneously to multiple I/O points using an arbitrary number of randomly positioned escorts are proposed by Bukchin and Raviv (2020). As the developed dynamic programming algorithm guarantees optimality but is limited with respect to

solvable problem sizes, they build upon their previous studies to provide further solution approaches in Bukchin and Raviv (2022). In addition to an integer linear programming formulation, they develop a heuristic based on the former dynamic programming approach, which is used when the two exact methods fail to provide optimal solutions in reasonable time.

The overview of existing centralized algorithms for item retrieval in the field of puzzle-based storage systems reveals a high problem complexity resulting from the large number of relevant model parameters and their different investigated constellations. Heuristics frequently apply, as optimal approaches are limited regarding solvable problem sizes. All of them depend on global system information. Thus, for a decentralized system setup, the presented approaches may serve as a starting point, however, they need to be enhanced substantially to reduce processing load and regain run time efficiency.

Sequencing problems as considered in this dissertation correspond to systems with an arbitrary number of escorts, simultaneous item movements, parallel item processing, and multiple I/O points, where additionally the retrieval sequence of these items is observed. Moreover, the design of the system is dynamic in terms of handling a continuous material flow of arriving transport units which are processed within the system and depart from it afterwards. None of the existing approaches is able to provide optimal solutions to the specified problem setting, which will be necessary to assess the quality of decentralized solutions.

2.2 Decentralized Algorithms for General Conveyor Systems

Decentralized algorithms are already applied to realize specific material handling operations using modular conveyor systems. These are summarized in Table 2.2. Since deadlocks compromise system liveliness in a decentralized setup, we investigate whether and how the presented algorithms handle them.

There are four possible strategies differing in terms of the risk incurred (Tanenbaum and Bos 2015, p. 443):

1. Ignorance: Deadlocks occur and are not handled.
2. Detection: Deadlocks occur, are identified, and recovered.
3. Avoidance: Deadlocks do not occur, as they are actively averted.
4. Prevention: Deadlocks do not occur, as they are structurally excluded.

Furthermore, we classify the introduced approaches regarding their implemented routing strategy. Following the definition of Shiller (2015), we differentiate offline and online routing. With the former, item routes are planned entirely from start to destination before initiating transportation, whereas with the latter, they are planned incrementally with stepwise item movements. Based on Gue (2006), we rank the achievable density of a system as *high* if there are interfering items preventing from accessing desired items. We further define *highest*-density systems as those which can operate with just one empty module. The module size specifies whether a conveyor module is capable of handling an item on its own (1:1) or whether multiple modules are required (n:1).

Transport systems with decentralized control are provided by Mayer and Furmans (2010), Krühn et al. (2013), Firvida et al. (2018), and Sohrt and Overmeyer (2020). All of these approaches are based on an offline route planning algorithm.

The *FlexConveyor* of Mayer and Furmans (2010) consists of single unit-sized right-angle-transfer conveyor modules (cf. Figure 2.2). By combining multiple modules, various shapes for material flow systems can be created. The locally controlled modules interact via communication connections to their adjacent modules. For routing an incoming transport unit to its destination module, the selected path of modules is reserved using a reservation token. Pathfinding is based on shortest distances while considering the reservations of other transport units within the system. When executing scheduled transports, deadlock avoidance is guaranteed by special messages – so-called deadlock tokens – which

Table 2.2: Decentralized algorithms for general conveyor systems.

Publication	Functionality	Density	Routing	Deadlock handling	Module size
Mayer and Furmans (2010)	transport	low	offline	avoidance	1:1
Krühn et al. (2013)	transport	low	offline	avoidance	n:1
Firvida et al. (2018)	transport	low	offline	-/-	n:1
Sohrt and Overmeyer (2020)	transport	low	offline	avoidance	1/n:1
Gue et al. (2014)	store	high	online	prevention	1:1
Uludağ (2014)	pick	high	online	avoidance[3]	1:1
Shekari Ashgzari and Gue (2021)	pick	highest	online	avoidance	1:1
Seibold et al. (2022)	sort	low	offline	prevention	1:1
This dissertation	sequence	highest	offline	prevention	1:1

are sent between modules at critical route sections before initiating transportation. However, this mechanism is not intended for high(est)-density conveyor networks.

The transport systems of Krühn et al. (2013) and Firvida et al. (2018) are based on small, multidirectional conveying elements where multiple modules are required to carry a single transport unit.
For route planning, the *Cognitive Conveyors* (cf. Figure 2.3) presented in Krühn et al. (2013) use a distance vector routing metric. It considers existing routes in the system, dimensions and necessary rotations of the transport unit as well as the physical system boundaries. For deadlock avoidance during transport execution, the authors extend the deadlock handling approach of Mayer and

[3] not generally proven and only under certain conditions

Figure 2.2: Conveying network of FlexConveyor modules (Seibold et al. 2013).

Furmans (2010) to fit the smaller module size of their system. This is validated using simulation.

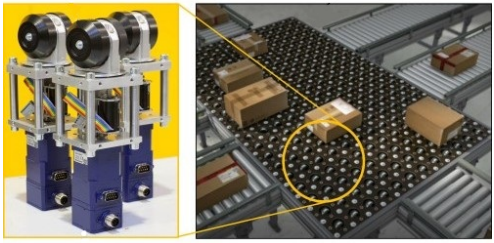


Figure 2.3: The Cognitive Conveyors (Krühn et al. 2013).

The decentralized routing algorithm of Firvida et al. (2018) applies to small conveyor elements in shape of equilateral triangles, squares, or regular hexagons. Shortest paths are identified following a breadth-first search based broadcast procedure towards the destination module considering the dimensions of the transport unit. The presented algorithm ignores other transport units in the system. Furthermore, route execution as well as deadlock and collision handling are not addressed.

In Sohrt and Overmeyer (2020), physical time windows are used for route planning of transport units within modular conveyor systems. These are negotiated by the corresponding modules with the objective of optimizing system

throughput. Each conveyor module is assigned a clock, which is synchronized throughout the network. This is necessary to guarantee reliable system behavior. The authors show that their decentralized algorithm avoids conflicts, deadlocks, livelocks, and starvation. Additionally, the algorithmic complexity in terms of the maximum number of messages is investigated.

Besides transporting, advanced material handling operations such as storing, picking, and sorting are realized using decentralized control algorithms. Of those, sorting is most closely related to sequencing, as this represents the special case of sequencing with batch size one.

Building upon the concept of puzzle-based storage systems (cf. Section 2.1), Gue et al. (2014) present *GridStore* – a high-density conveyor system with decentralized control for storage and retrieval of unit loads (cf. Figure 2.4). Identical square conveyor modules, each of them capable of conveying in the four cardinal directions, form a rectangular grid. Retrieval occurs at the southernmost system edge, replenishment at the northernmost one. The online route planning algorithm is based on an iterative procedure where adjacent conveyor modules negotiate item movements via messages. To match negotiation and movement phases at each module, the system is synchronized. North-south negotiation messages are used to convey requested items southwards to the retrieval edge. Interfering items on their paths are moved using east-west negotiations. This requires at least one empty module per row. The *GridStore* system operates in a constant work-in-process (CONWIP) mode, which is guaranteed by an external overall control scheme. In this way, the system is shown to be deadlock-free.

Based on *GridStore*, Uludağ (2014) develops the decentralized controlled picking system *GridPick* (cf. Figure 2.5) which is likewise built from identical right-angle-transfer conveyor modules connected to a rectangular grid network. Requested items can be picked from the bottom edge of the system. In an extension of *GridPick*, picking from the upper system edge is additionally possible. The synchronized conveyor modules run an online route planning algorithm, where requested items move vertically while interfering items are

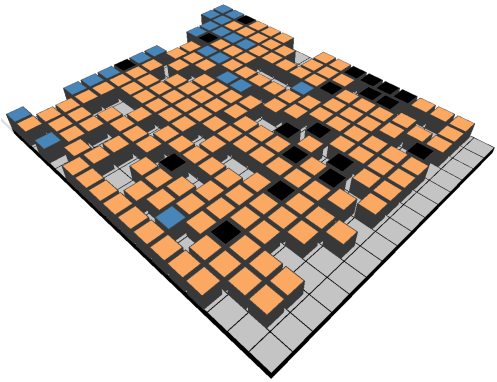


Figure 2.4: The GridStore System (Uludağ 2014).

relocated using east- and westward movements. The system requires at least one empty module per row. As the storage containers of *GridPick* do not leave the system after picking, additional vertical movements are necessary to balance the number of empty modules per row. The author analyzes system deadlock behavior using Petri nets. By limiting the maximum order size to the number of columns c and the total number of requested items to $(2 \cdot c - 1)$ deadlocks are avoided for small systems. Due to memory storage overflow, the absence of deadlocks cannot be proven in general.

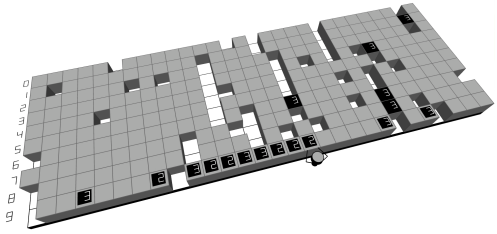


Figure 2.5: The GridPick System (Gue and Uludağ 2012).

Shekari Ashgzari and Gue (2021) introduce the picking system *GridPick+*, which allows picking items from all four edges of a rectangular modular conveyor grid (cf. Figure 2.6). Furthermore, the items are transferred to a specific picking position at the corresponding system edge, eliminating the need for the picker to move for collecting the requested items as with *GridPick*. The online route planning algorithm is based on the concepts of Hao (2020) (cf. Section 2.3), which are modified to enable order picking. It requires at least one empty module for item movements. To identify those, surrounding conveyor modules are explored sequentially. The system avoids deadlocks by following defined rules to exclude critical movements in each iteration.

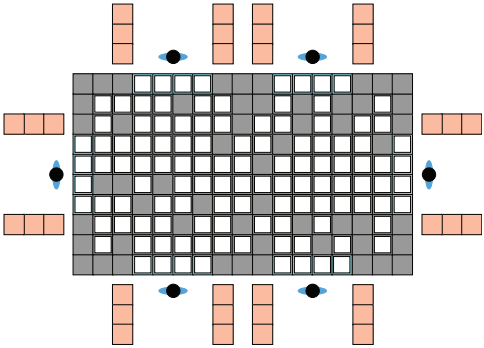


Figure 2.6: The GridPick+ system (Shekari Ashgzari and Gue 2021).

The *GridSorter* of Seibold et al. (2022) and Seibold (2016) represents a modular sorting system with decentralized control. Similarly to the aforementioned systems, it is built up of square conveyor modules transporting in the four cardinal directions (cf. Figure 2.7). The decentralized algorithm for routing transport units from their arrival location to the correct destination is based on the concept of logical time introduced by Lamport (1978). Pathfinding uses a decentralized iterative deepening A* algorithm (DIDA*) which identifies the shortest path in terms of logical time considering other reservations at the system modules. Transport execution follows the confirmed reservations at

each module in ascending logical time. This allows proving deadlock, livelock, and starvation prevention.

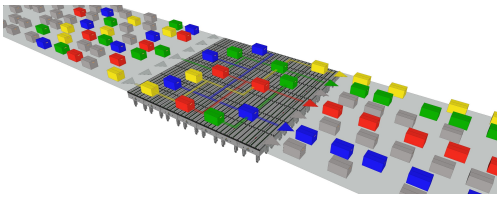


Figure 2.7: The GridSorter system (Seibold 2016).

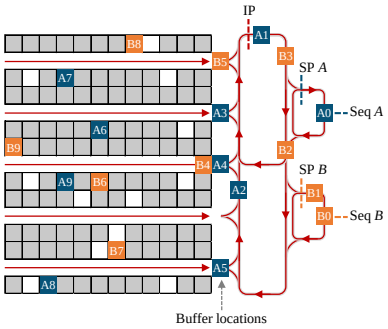
From the presented decentralized algorithms for general conveyor systems, we conclude that transporting, storing, picking, and sorting can be realized using decentralized control, while sequencing is not yet covered. Based on the definition of high-density systems (cf. Gue 2006), transporting and sorting systems are in general low-density systems, as arriving items are routed directly to their destinations. All of the low-density systems are based on offline route planning which allows operating in more complex, i.e., non-rectangular, network structures (Hao 2020). Route planning in high(est)-density systems requires relocating interfering items. So far, this is only possible using online approaches. These systems need to be synchronized involving a higher level coordination element to ensure that all modules execute the algorithmic phases consistently. Therefore, buffer times are incorporated within the negotiation cycles, which can reduce system performance (Uludağ 2014). None of the existing approaches provides a decentralized algorithm based on offline route planning applying to high(est)-density conveyor networks. Thus, preventing deadlocks or ensuring system liveliness in general under these conditions remains unresolved as well. Complexity aspects of the presented decentralized algorithms are poorly studied. Furthermore, a quality assessment of the obtained decentralized solutions is generally not provided.

2.3 Approaches for Sequencing Systems

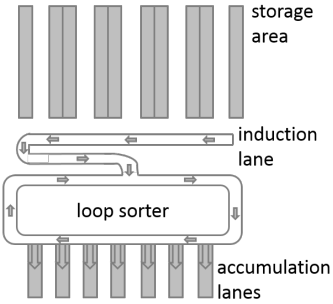
Conventional sequencing system configurations are highly specific and hence inflexible, consume large space, and are not controlled on a decentralized basis. Figure 2.8 (a) shows an exemplary setup for sequenced retrieval downstream from an automated storage and retrieval system. It is limited to two concurrent sequences. Additionally, sequences may block each other (Lienert and Fottner 2018). Alternatively, sortation conveyors can be used, as shown in Figure 2.8 (b). The items circulate inside until they are requested at their corresponding output point. Due to long travel distances of items, system throughput is reduced. In the field of production, the concept of selectivity banks arose in order to match multiple assembly systems (cf. Figure 2.8 (c)). If for two consecutive assembly systems the output sequence of the first one does not fit the input sequence of the subsequent one, selectivity banks allow to buffer and re-sequence parts, supplying them to the following system in the required order (Jayaraman et al. 1997). However, sequencing is limited due to the number of lines and the available buffer capacities per line.

To improve flexibility, space utilization, and efficiency of sequencing processes, several approaches are proposed in scientific research (cf. Table 2.3). We classify them according to their level of decentralization, as this determines their applicability within dynamic, uncertain, and complex industrial environments (cf. Section 1.1). Multi-batch processing results from the available I/O points in the system and is only possible within an $(m : n)$ setting with $m > 1$ input and $n > 1$ output points. We investigate density, routing algorithms, and integrated deadlock handling strategies to ensure system liveliness by analogy with Section 2.2.

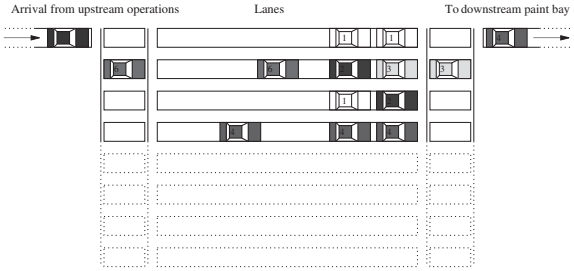
Referring to $(n^2 - 1)$-puzzle games (cf. Section 2.1), Alahmad and Ishii (2021) develop an A*-based search algorithm for rearranging an $(n \cdot m - 1)$ puzzle to a sequenced final configuration. This can be used for item sequencing within high(est)-density systems of logistic applications. To overcome unsolvable start configurations of the puzzle, they propose a pre-sorting strategy, where two



(a) Retrieval in sequence at an AS/RS (Lienert and Fottner 2018).



(b) Loop sortation conveyor (Boysen et al. 2019).



(c) Selectivity bank with eight lanes (Spieckermann et al. 2004).

Figure 2.8: Conventional sequencing system configurations.

specific items need to be switched within the system. The algorithm is entirely centralized such that deadlock handling is not addressed.

Single batch sequencing from one input to one output point is possible using the *GridSequence* system of Gue et al. (2012) (cf. Figure 2.9). Based on the length of a given sequence, each item is assigned a specified target position within the system depending on its index in this sequence. A transport unit cannot be unloaded until all of its predecessors reach their target positions. The decentralized control algorithm is based on the same concept as in Gue et al. (2014) and Uludağ (2014). All modules are synchronized and negotiate item movements in single steps. North-south negotiations are used to move items downwards to their target rows, while east-west negotiations serve for relocating

Table 2.3: Approaches for sequencing systems.

Publication	Decentralization	I/O points	Density	Routing	Deadlock handling
Alahmad and Ishii (2021)	none	1:1	highest	offline	-/-
Gue et al. (2012)	complete	1:1	high	online	-/-
Gue (2016)	partial	1:1	high	online	prevention
Sittivijan (2015)	partial	m:n	highest	online	centralized detection
Lieberoth-Leden and Fottner (2018)	partial	m:n	medium	offline	detection
Hao (2020)	complete	m:n	highest	online	prevention[4]
This dissertation	complete	m:n	highest	offline	prevention

interfering items. The length of the given sequence determines the necessary system size. Additionally, the system requires further space at its edges, which reduces the achievable density. Deadlock handling is not addressed.

Including a centralized element within the system architecture as in Gue (2016), Sittivijan (2015) and Lieberoth-Leden and Fottner (2018), allows reducing the complexity of a sequencing problem while accepting the drawbacks of inflexibility and high failure probability.

In the *GridHub* system of Gue (2016), trucks are loaded from a grid-based conveyor system following a specific sequence (cf. Figure 2.10). They are processed sequentially. Each truck holds the same number of containers, i.e., all order batches are assumed of equal size. The control architecture of *GridHub* comprises two hierarchical layers, where a centralized element on the top layer coordinates the unloading sequence by broadcasting requested containers together with their position within the sequence to all system modules on the

[4] livelocks not prevented

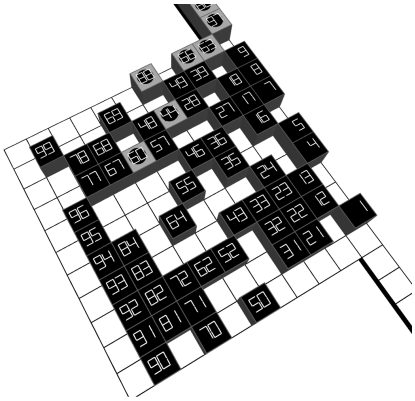


Figure 2.9: The GridSequence system (Gue et al. 2012).

layer below. At module level, the corresponding container transfers are organized. The developed decentralized online routing algorithm is based on the same synchronized iterative movement negotiations as Gue et al. (2012), but additionally requires eliminating fully occupied rows or columns to enable deadlock avoidance. Item priorities are used to resolve conflicts if empty modules are requested multiply.

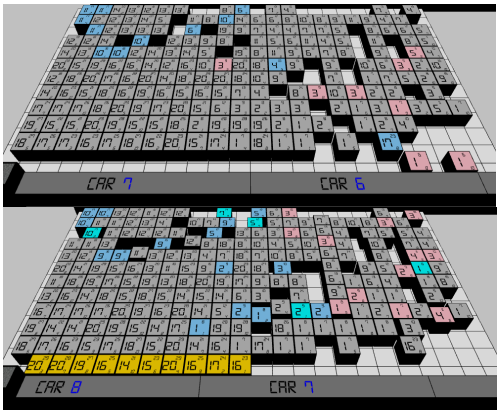


Figure 2.10: The GridHub system (Gue 2016).

Sittivijan (2015) develops an algorithm to transfer a modular warehouse consisting of unit-sized right-angle-transfer conveyor modules from an initial to a final configuration (cf. Figure 2.11). This may represent a sequencing problem if the final configuration forms a specific item sequence. The control algorithm combines centralized and decentralized elements and is based on unique priorities assigned to items. Supervising all item movements and detecting deadlock situations occurs at system level. At module level, an A*-based route planning is used to identify the paths from current to target item positions. Conflicts are resolved by a tagging procedure based on the assigned priorities. The results of Sittivijan (2015) are further refined in Dayıoğlu et al. (2020), but without increasing the level of decentralization.

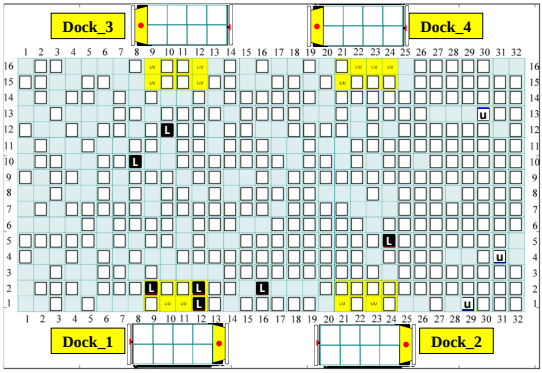


Figure 2.11: Modular warehouse system (Sittivijan 2015).

The automated material flow system of Lieberoth-Leden and Fottner (2018) (cf. Figure 2.12) is composed of autonomous modules providing multiple material handling tasks, including sequencing. Within its control architecture, a centralized element holds all data of the system to ensure consistent information for all other elements. Furthermore, it defines the necessary routes for processing incoming transport tasks. To reduce the disadvantages related to centralized control concepts, the centralized element changes dynamically (Lieberoth-Leden et al. 2018). For sequencing, transport units are buffered at

one or several intermediate destination modules, which are determined by the centralized element. Sufficient buffer capacity within the system for upcoming sequencing tasks needs to be ensured externally. Deadlock handling is limited to detection once the system already stagnates.

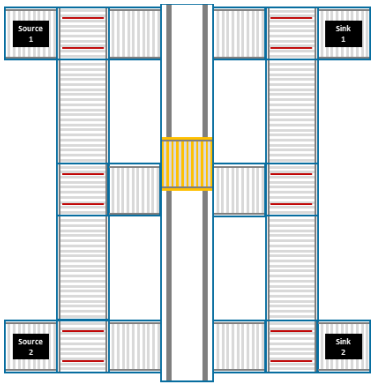


Figure 2.12: Layout of a simple automated material flow system (Lieberoth-Leden and Fottner 2018).

Hao (2020) presents *GridHub* – a modular decentralized controlled high(est)-density material handling system, where right-angle-transfer conveyor modules are arranged to a rectangular grid network (cf. Figure 2.13). Item loading and unloading is possible at all four system edges. The developed algorithm is capable of storing, retrieving, sorting, and sequencing while processing multiple items simultaneously. Similar to Gue et al. (2012) or Gue (2016), conveyor modules are synchronized and negotiate item movements iteratively in single steps. Thus, the approach is limited to rectangular network structures. Items proceed to their destinations using empty modules within the system. These are identified by sequential search phases with ascending number of additional steps for relocating interfering items. For sending and processing messages, rules are specified based on the current module state and the search phase of the message. At system level, priority directions are defined to resolve negotiation conflicts. The algorithm focuses on transferring items to their destinations via

shortest paths, which – in case of sequencing – can cause congestion at the output points if necessary predecessor items are missing. The system is proven to be deadlock-free, however, livelocks cannot be generally excluded.

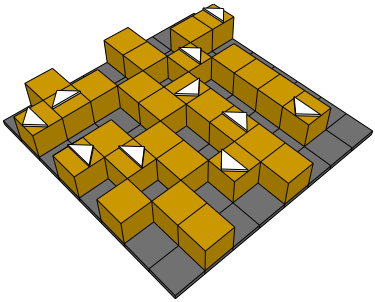


Figure 2.13: The GridHub system (Hao 2020).

From the presented approaches, we conclude that basically all of them offer a solution to realize a high(est)-density conveyor system for sequencing. The majority suffers from centralized system functionality where essential algorithmic parts, such as path finding, data holding, or deadlock handling, are delegated to a centralized element. Often, single batch problems are considered. Several of the proposed systems require additional space within the network, which reduces the achievable density. Online routing algorithms apply frequently, such that only synchronized systems with rectangular network structures can be realized (Hao 2020). None of the presented approaches provides decentralized sequencing from multiple input to multiple output points in general high(est)-density conveyor networks while ensuring system liveliness in terms of deadlock, livelock, and starvation prevention.

2.4 Chapter Conclusion

The results of the literature review state the research gap of this dissertation. First, there is no decentralized multi-batch sequencing algorithm which is able

to operate in highest-density conveyor systems with arbitrary network structures (cf. Sections 2.2 and 2.3). Second, ensuring system liveliness by preventing deadlocks, livelocks, and starvation in highest-density conveyor networks based on a decentralized offline route planning algorithm remains unresolved (cf. Section 2.2). Third, complexity aspects of decentralized algorithms are poorly studied (cf. Section 2.2). Fourth, none of the identified (partially) decentralized algorithms is benchmarked against a centralized approach to assesses the quality of its generated solutions (cf. Sections 2.2 and 2.3). Furthermore, a centralized algorithm applicable to multi-batch sequencing from multiple input to multiple output points for processing a continuous flow of materials as in the specified problem setting does not yet exist (cf. Section 2.1). Fifth, we found that the numerical studies of the presented approaches insufficiently address the requirements of practical applications within current industrial environments, as mainly rectangular networks are investigated. This dissertation addresses all of the identified shortcomings to comprehensively close this research gap.

3 System Definition

In decentralized systems, the intelligence of the system is distributed to several of its components eliminating hierarchical dependencies (de Ryck et al. 2020, Rana and Taneja 1988). This leads to a connected network of autonomous, equivalent decisional entities, which we can describe and formalize using graph theory (Trentesaux 2009, Dilts et al. 1991) (cf. Section 3.1). This formal description allows specifying the necessary system requirements of conveyor networks for sequencing (cf. Section 3.2). Transferring the elements of the graph-based system model to components in reality yields the showcase system used to demonstrate the developed decentralized sequencing algorithm (cf. Section 3.3). By summarizing the results of this chapter, Section 3.4 provides an answer to the first research question of this dissertation.

3.1 Graph-based System Model

A graph G is defined as a tuple $G(V, E)$ with V representing a nonempty set of nodes connected by the set of edges E with $E \subseteq [V]^2$. Each pair of nodes v_i and $v_j \in V$ is called *adjacent* if there is an edge $e \in E : e = \{v_i, v_j\}$ which directly connects both nodes. Two non-adjacent nodes v_k and v_l can be *connected* via a series of adjacent nodes $\{v_k, v_{k+1}, \ldots, v_{l-1}, v_l\}$ which is called a *path*. In *weighted* graphs, edges are assigned weights, such that $w : E \to \mathbb{R}$, representing certain costs to determine more or less favored paths. The graph G is connected if each pair of its vertices $v_i, v_j \in V (i \neq j)$ is connected. The *connectivity* indicates the intensity of its coherence. We denote a graph G as *k-connected* for $k \in \mathbb{N}$ if removing at least k nodes is necessary to disconnect

it. Transferring objects within a graph is modeled using flows between nodes, where *capacity constraints* serve for limiting the amount of objects which can be transferred simultaneously. (Diestel 2017, p. 2ff., 151; Erciyes 2013, p. 11ff.; Brandes and Erlebach 2005, p. 7ff.)

Using the analogy of graphs, we can understand a decentralized conveyor network in terms of a set of connected nodes. Between these nodes, physical objects as well as data objects are transferred. Each node has its own control unit for locally receiving, storing, processing, and forwarding data objects. For identifying and transferring physical objects, they are equipped with technical instruments. The underlying technology may vary and is not relevant for the developed algorithm. Nodes provide the overall system functionality. They are connected by edges to enable transferring objects via a path of adjacent nodes. Physical objects need to be sequenced within the conveyor network which is organized by sending data objects throughout the network representing the decentralized communication of the nodes. We divide the set of all nodes into input, output, and sequencing nodes. The flow of physical objects proceeds from input to output nodes. Therefore, edges for transferring physical objects connected to input and output nodes are unidirectional, while all other edges are bidirectional. The outflow of the network at the output nodes needs to observe the predefined unloading sequences. This is achieved by reordering the physical objects using the sequencing nodes in between.

3.2 System Requirements

We use the graph-based system description to define the system requirements for sequencing. These relate to both capacity and structural characteristics of the conveyor network, which we discuss in Sections 3.2.1 and 3.2.2, respectively.

3.2.1 Capacity

Within the decentralized conveyor network, physical and data objects are transferred between its nodes. To avoid congesting the system, we need to define capacity constraints. We derive the overall network capacity based on the capacity of its nodes (cf. Section 3.2.1.1). It limits the flow of objects throughout the network (cf. Section 3.2.1.2).

3.2.1.1 Nodes

Capacity constraints within decentralized conveyor networks differ depending on whether data objects or physical objects are exchanged (Qiu et al. 2002). Data objects are immaterial and can be transferred in negligible time. As decentralized communication processes are not the focus of this dissertation, we assume sufficient memory, communication, and data processing capacities at the network nodes such that node capacity regarding data objects becomes unconstrained. In contrast, transferring physical objects requires a non-negligible amount of time and space. This limits processing transport units simultaneously at a single node. Due to their spatial dimensions, two adjacent nodes need to be connected seamlessly to provide continuous movements. When transferring a physical object between two adjacent nodes, it temporarily occupies parts of both nodes until it is entirely located at the receiving node. Therefore, we describe the physical object capacity of the network by the physical object capacity of its nodes. It results from the relation of the space occupied by a physical object and that provided by a node for conveying – e.g. for 2-dimensional networks the base area is decisive, while for 3-dimensional networks the volume needs to be considered.

Let U be the set of all physical objects to be sequenced and φ_o^u the space required by physical object u. Each node offers space of φ_v. Then, the capacity κ_v of a single node v is given as

$$\frac{\varphi_v}{\max\limits_{u \in U} \varphi_o^u} \leq \kappa_v \leq \frac{\varphi_v}{\min\limits_{u \in U} \varphi_o^u}. \tag{3.1}$$

We suppose similar space requirements of all physical objects. Thus,

$$\kappa_v = \left\lfloor \frac{\varphi_v}{\max\limits_{u \in U} \varphi_o^u} \right\rfloor = \left\lfloor \frac{\varphi_v}{\min\limits_{u \in U} \varphi_o^u} \right\rfloor. \tag{3.2}$$

As all nodes have identical properties, we obtain the same capacity at each node, i.e.,

$$\kappa_v = \kappa_{\acute{v}} \qquad \forall\, v, \acute{v} \in V. \tag{3.3}$$

The *density* of a network depends on the number of occupied nodes compared to the overall number of nodes within the system. Moving a physical object within the network requires a set of unoccupied nodes V_e, such that

$$\sum_{v \in V_e} \kappa_v \geq 1 \tag{3.4}$$

holds. If

$$\kappa_v \geq 1 \tag{3.5}$$

is satisfied for each node of the network, transferring a physical object between each pair of adjacent nodes requires no more than one edge. Otherwise, several nodes need to cooperate to transfer the physical object collectively. We exclude such cases when presenting the decentralized sequencing algorithm. However, these can be covered without loss of generality by integrating grouping mechanisms into the algorithmic operations as described in Krühn et al. (2013),

Firvida et al. (2018), Sohrt and Overmeyer (2020), or Colling et al. (2016). If nodes provide capacity for several physical objects, i.e.

$$\kappa_v \geq \vartheta \qquad \forall\, v \in V, \tag{3.6}$$

with $\vartheta \geq 2$ ($\vartheta \in \mathbb{N}$), this allows splitting each node into ϑ single nodes each of which satisfies equation (3.5).

Thus, we assume the physical object capacity of each node within a conveyor network as

$$1 \leq \left\lfloor \frac{\varphi_v}{\max\limits_{u \in U} \varphi_o^u} \right\rfloor = \kappa_v = \left\lfloor \frac{\varphi_v}{\min\limits_{u \in U} \varphi_o^u} \right\rfloor < 2. \tag{3.7}$$

3.2.1.2 Network Flow

For sequencing physical objects, the outflow of the network needs to observe their predefined predecessor-successor dependencies. We denote a *batch* as a group of physical objects assigned to the same output node. Each batch contains an ordering of its objects according to which these need to arrive there. Assuming that input and output nodes do not provide capacity for buffering physical objects, but are only used for introducing and unloading them, the physical object capacity of a network follows from that of its sequencing nodes (cf. equation (3.7)).

The inflow of physical objects at the input nodes reduces remaining available network capacity, while the outflow releases occupied capacity. Due to the predefined predecessor-successor dependencies, sufficient capacity is required within the network to buffer physical objects which are introduced at a input node prior to one of their predecessors. Restricting the inflow of physical objects into the system allows relaxing capacity requirements, while an unrestricted inflow into the system requires increasing the provided network capacity to generally enable sequencing. Inflow and capacity of the system need to be matched to

prevent congestion. This means that the number of buffered physical objects maintains the system capable of operating at any given time without stagnating due to excessive capacity consumption. Therefore, we specify a relation between system inflow and capacity to ensure sequencing.

In the following, we assume a given conveyor network comprising the set of sequencing nodes C. Furthermore, occupied node capacity according to equation (3.7) is uniquely assigned to a physical object for each node at any point in time, i.e., each node holds at maximum one physical object at a time. Moving a physical object for relocation always requires at least one unoccupied sequencing node adjacent to the node occupied by this object (cf. equation (3.4)). As the graph of the conveyor network is entirely connected, where edges between each pair of sequencing nodes are bidirectional (cf. Section 3.1), any sequencing node can become unoccupied by moving the buffered physical objects. Therefore, we need to guarantee that there is always at least one arbitrary unoccupied sequencing node within the conveyor network.

We define a subset of *released* objects U_r within the set of all physical objects to be sequenced. This subset represents the physical objects which can be introduced from their assigned input node without compromising the operability of the system. U_r satisfies the following conditions:

(I1) If an object is included in U_r, then all of its predecessors within the unloading sequence at their output node are or were included as well.

(I2) The cardinality of U_r never exceeds $(|C| - 1)$.

(I3) Objects leaving the system at an output node are removed from U_r.

Any physical object originating at an input node which is not yet included in the set of released objects is prevented from being introduced into the network. Otherwise, we cannot guarantee that there is at least one unoccupied sequencing node. It needs to be ensured that creating a subset of released objects according to conditions (I1) to (I3) is always possible from the flow of physical objects arriving at the system. We refer to this as *inflow constraints*.

Using the defined inflow constraints, we are able to generally prevent congesting a conveyor network comprising $|C|$ sequencing nodes. Congestion implies a system arrangement in which a $|C|$-th physical object needs to be introduced into the network before another physical object currently occupying a sequencing node may be unloaded due to the predecessor-successor dependencies. Thus, all of the $|C|$ sequencing nodes are occupied, preventing any physical object movement. This occurs only if a successor is introduced into the system before one of its predecessors. According to condition (I1), this is only possible if the successor is included in the set of released objects together with all of its predecessors. Limiting the cardinality of this set to $(|C| - 1)$ (cf. condition (I2)) ensures that there is always sufficient capacity available in the system to introduce all necessary predecessors. When removing a sequenced physical object (cf. condition (I3)), a sequencing node becomes unoccupied again such that the next physical object is included in U_r to proceed in sequencing.

Note that different from other systems with decentralized control, such as *GridSorter* (cf. Seibold et al. 2022) or *GridStore* (cf. Gue et al. 2014), installing a work-in-process based load control is not feasible when sequencing predefined batches to their destinations, as without introducing a predecessor into the system its buffered successors may never be unloaded due to the predefined predecessor-successor dependencies.

3.2.2 Structure

The connections of input, output, and sequencing nodes define the structure of a conveyor network. They build up possible paths for transferring objects. The conveyor network represents an entirely connected graph. Paths of data objects are independent of specific node types, while for physical objects, any output node can be reached from any input node and any sequencing node can be reached from any other sequencing node via a path of sequencing nodes (cf. Section 3.1). The set of input and output nodes are non-empty to establish the flow of physical objects.

For sequencing, physical objects may be buffered within the network to observe the predefined unloading sequences at the output nodes. Each sequencing node provides capacity for buffering exactly one physical object (cf. Section 3.2.1). In case of highest-density system occupation, there is one unoccupied sequencing node within the network (cf. Section 2.2). To forward a requested physical object, the unoccupied node is iteratively positioned such that this object can be transferred one node further towards its destination node – much like moving the escort in an (n^2-1)-puzzle game to a selected location. This requires relocating other buffered physical objects on a path, which we refer to as *relocation path*. As each node cannot hold more than one physical object, this relocation path needs to omit the node currently occupied by the requested physical object to avoid collision (cf. Figure 3.1).

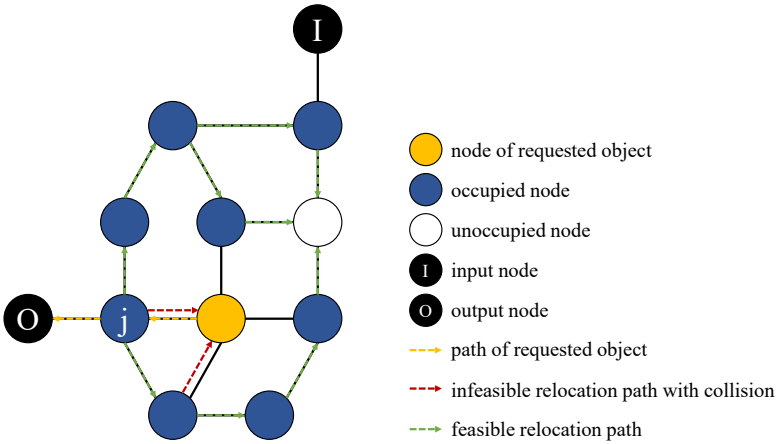


Figure 3.1: Collisions at relocation paths for relocating physical object j.

We can ensure that at highest-density system occupation, there is always a valid path for relocation if the network provides at least two *internally disjoint paths* between each pair of sequencing nodes. Internally disjoint paths only share their start and end nodes (Diestel 2017, p. 74). Thus, there is always an alternative path to relocate a buffered physical object, which is interfering on the path

of a requested physical object, without using the node the latter is currently occupying. If this holds for each pair of sequencing nodes, we can transfer a physical object on any path within the network using only one unoccupied sequencing node.

Following Menger's theorem, at least k internally disjoint paths exist between two distinct nodes $v_i, v_j \in V$ in a k-connected graph (Menger 1927). Therefore, the sub-graph of sequencing nodes within the given conveyor network needs to be at least 2-connected to provide sequencing at highest-density system occupation. Specifically, this means that there are no *cut-vertices* or *bridges* (cf. Figure 3.2).[1] In such networks, sequencing is not generally possible at highest-density occupation, as more than one unoccupied sequencing node is required to overcome the corresponding nodes.

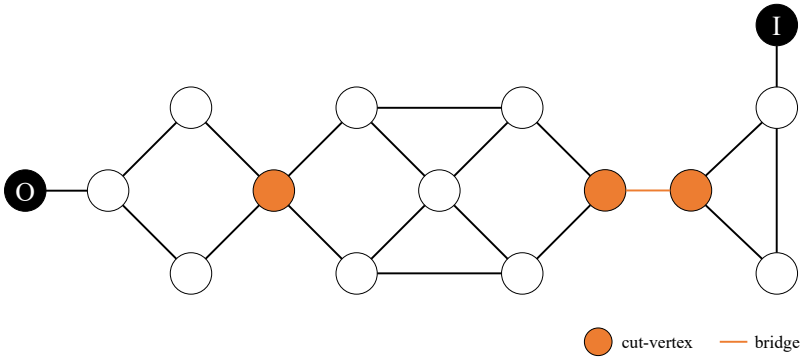


Figure 3.2: Graph with cut-vertices and a bridge.

[1] A *cut-vertex* represents a node, and a *bridge* represents an edge whose removal disconnects a graph (Diestel 2017, p. 11).

3.3 Showcase System

Sequencing problems arise in various industrial application environments (cf. Section 1.1). For demonstration, we transfer the graph-based network model to a practical showcase system. We introduce the different real-world system elements in Section 3.3.1. A running conveyor network, initially, requires system installation (cf. Section 3.3.2). Afterwards, it is ready to operate such that a flow of arriving physical objects can be processed for sequencing (cf. Section 3.3.3). Note that the terms *(sequencing) system* and *conveyor network (for sequencing)* can be understood synonymously throughout this dissertation.

3.3.1 System Elements

Within the showcase system, nodes, physical objects and data objects are represented using conveyor modules (cf. Section 3.3.1.1), transport units (cf. Section 3.3.1.2) and messages (cf. Section 3.3.1.3), respectively. Transferring objects is realized using corresponding mechanical and electronic connections (cf. Section 3.3.1.4).

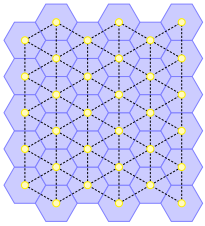
3.3.1.1 Conveyor Modules

We use standard rectangular conveyor modules for representing network nodes, such as the *FlexConveyor* of Mayer (2009) (cf. Figure 3.3). It is square shaped with an edge length of 500 mm. These modules can be adjusted to different heights and are equipped with lockable wheels guaranteeing stability during operation. They can easily be unlocked to flexibly modify an existing conveyor network. Loads are identified by photoelectric sensors or a distance measuring system detecting whether the module is currently occupied (Furmans et al. 2019).

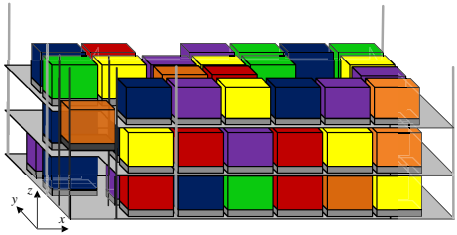
Although we refer to this standard rectangular module shape, the developed decentralized sequencing algorithm is not limited to right-angle-transfer conveyor

Figure 3.3: FlexConveyor module (Karlsruhe Institute of Technology: Institute for Material Handling and Logistics 2009).

modules. Generally, any module shape providing more than two movement directions is possible (cf. Figure 3.4 (a)), as long as the requirements regarding node capacity are met (cf. Section 3.2.1.1). This also allows for creating three-dimensional structures with additional vertical movement direction (cf. Figure 3.4 (b)) and includes conveyor networks consisting of different module shapes as well. However, we stick to the standard case of square conveyor modules within two-dimensional networks due to their broad practical applicability.



(a) Hexagonal module shape (Firvida et al. 2018).

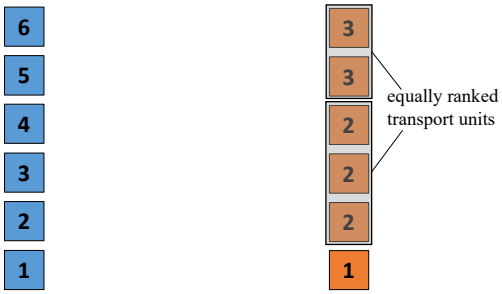


(b) Three-dimensional system configuration (Zaerpour et al. 2017a).

Figure 3.4: Advanced conveyor network structures.

3.3.1.2 Transport Units

We refer to physical objects as transport units, such as parcels, boxes, small load carriers etc., as these can be easily transferred using the specified conveyor modules (cf. Section 3.3.1.1). Bulk materials, liquids or gases may also be processed and sequenced as unit loads when combined with suitable packaging and/or load carriers (Arnold and Furmans 2019, p. 1). To meet the capacity requirements as defined in Section 3.2.1.1, the area covered by one transport unit fits onto the surface provided by a single conveyor module. A conveyor module can handle only one single transport unit simultaneously as given in Mayer (2009). Transport units are organized in *batches* of a certain *size* corresponding to the number of transport units included in the batch. Each transport unit of a batch is assigned a rank to specify the predefined unloading sequence at its output node. It may be completely or partially defined (cf. Figure 3.5). A partially defined sequence comprises several transport units sharing equal rank. These may arrive at their output node in arbitrary sequence while still preserving their distinct predecessor and successor transport units. Throughout this dissertation, we consistently characterize batches using colors, while the ranks of transport units within a batch are denoted with numbers.



(a) Completely defined. **(b)** Partially defined.

Figure 3.5: Batch unloading sequences.

Load specific information as destination or rank is encoded using RFID-tags or barcodes. These are attached to the transport units such that they can be identified by the conveyor modules.

3.3.1.3 Messages

Each data object is represented as an instance of the abstract message class to model the decentralized communication of the conveyor modules (cf. Figure 3.6). This allows arriving transport units to be processed for sequencing based on the local information and decisions at each node.

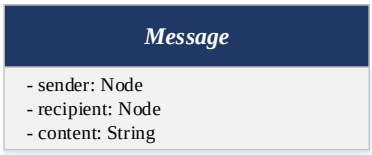


Figure 3.6: UML class diagram of message class.

3.3.1.4 Connections

Connecting several nodes of the network is necessary for exchanging objects between them. Each conveyor module provides two dimensional transportation in the four cardinal directions for transferring transport units. The primary movement direction is realized with rollers, while powered elastic bands are lifted up to the conveyor level for the secondary movement direction.

Exchanging messages occurs via electronic and electrical connections between the modules. Each module offers a communication interface for each transport direction, i.e., to each of its adjacent modules. Communication to non-adjacent modules within the network requires a corresponding message transmission mechanism. It forwards messages via a series of adjacent modules to the respective recipient (cf. Figure 3.7).

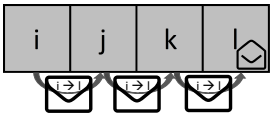


Figure 3.7: Message transmission mechanism – The message from module i to module l is forwarded via modules j and k.

3.3.2 System Installation

System installation involves all necessary steps until the system is available to operate. It needs to be set up (cf. Section 3.3.2.1) and configured (cf. Section 3.3.2.2) to acquire certain essential information concerning the established conveyor network. Sequencing may not start until the system is entirely installed.

3.3.2.1 Setup

System setup entails physically implementing the conveyor network before powering on. The underlying application determines the required number of conveyor modules. Each node of the graph-based network model corresponds to a single conveyor module. Therefore, we represent them as input, output, and sequencing *modules* in the following. Sequencing modules transfer transport units in all four cardinal directions, while input and output modules provide unidirectional conveying for introducing and unloading transfers, respectively (cf. Section 3.1).

All modules are arranged in the application environment while establishing electrical and electronic connections to their adjacent modules. Due to the Plug&Play technology, system setup requires less time than with conventional systems (Furmans et al. 2010). As soon as all system modules are correctly arranged and connected, the system is powered on for configuration.

3.3.2.2 Configuration

System configuration is initiated, as soon as the modules are supplied with electric power. Each module registers its adjacent modules via the connections established during system setup. All other installed modules of the conveyor network are identified using flooding mechanisms from information routing in distributed systems (cf. van Steen and Tanenbaum 2017, p. 226ff.).[2] Based on that, the shortest directional distance information related to the own position within the network is derived for all modules. It corresponds to the minimum number of modules that need to be passed in order to arrive at a destination module when choosing a specific direction as starting point of the path (cf. Figure 3.8). For this, path finding algorithms such as the Floyd–Warshall algorithm of Floyd (1962) and Warshall (1962) or the A* algorithm of Hart et al. (1968) apply. The calculated distance is saved locally for each combination of destination module and direction. Further global network information is not available at module level.

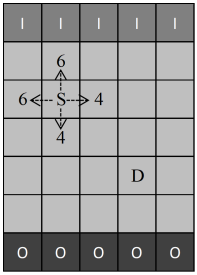


Figure 3.8: Directional distances from start module S to destination module D.

Based on the acquired system information, we verify whether the network is capable of sequencing at highest density. This requires redundant paths between

[2] We will not go into detail about the algorithms for system configuration, as they are not an integral part of the developed decentralized sequencing algorithm. These are rather necessary preceding steps to the main algorithmic operations described in Chapter 4.

all pairs of sequencing modules (cf. Section 3.2.2). Therefore, algorithms for determining the connectivity in graphs as proposed by Tarjan (1974) are applicable.

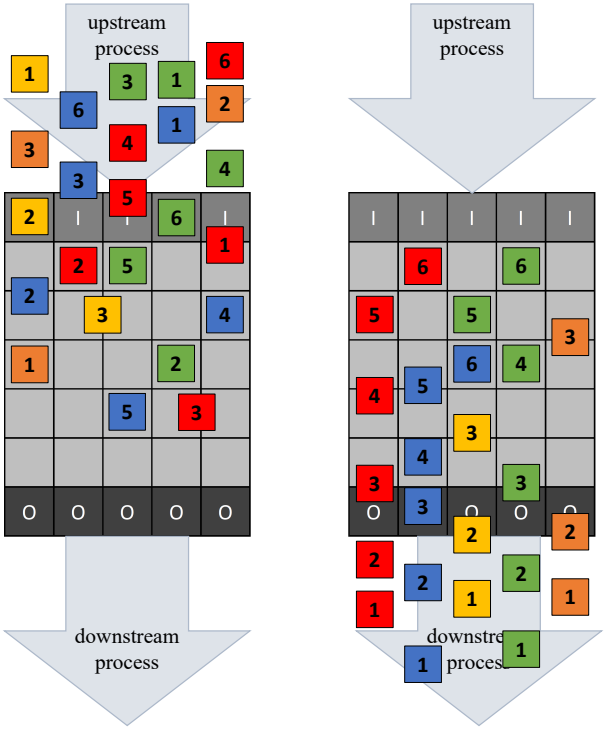
3.3.3 System Operation

For sequencing, transport units are processed by the installed conveyor network (cf. Section 3.3.3.1). To avoid congesting the system during operation, this needs to be coordinated with introducing and unloading of transport units (cf. Section 3.3.3.2).

3.3.3.1 Processing Transport Units

The sequencing system receives arriving transport units from an upstream process, sequences them, and forwards these to a downstream process. As the design of the upstream and downstream operations is irrelevant to the presented system model, these are not detailed further. The input modules link from the upstream process to the sequencing system to introduce arriving transport units. Likewise, the output modules provide these transport units to the downstream process after processing by the intermediate sequencing modules.

For sequencing, a set of arbitrarily arriving batches of transport units is reordered such that they can be unloaded at their assigned output modules adhering to the predefined batch sequences. As the system offers multiple input and output modules, this creates an $(m : n)$ setting ($m, n \in \mathbb{N}$) where each input module supplies to all output modules and each output module collects transport units from all input modules. Figure 3.9 illustrates an example. The $(m : n)$ setting allows batches of transport units with different destinations to be introduced, sequenced, and unloaded simultaneously, i.e., parallel order processing. In this way, the system is adaptable to many different applications and can be individually customized for specific use cases.



(a) Introducing arriving transport units for processing.

(b) Unloading transport units after processing.

Figure 3.9: Example sequencing system comprising five input, five output, and 5×5 sequencing modules.

3.3.3.2 Introducing and Unloading

The destination of the batches, i.e., the output module where they are to be supplied to the downstream process as well as the ranks of the transport units within each batch are assumed to be given upon arrival at the system. The arrival characteristics of transport units result from the upstream process. As transport units cannot be identified until they are physically positioned on their

input module, these need to comply with the defined inflow constraints to avoid congesting the system (cf. Section 3.2.1.2).

To handle the inflow constraints within the decentralized system setup, information concerning the inflow and outflow of the system, which is locally held at its input and output modules, needs to be combined. Each output module is aware of the batches it will receive including their size. Therefore, the set of released transport units is specified at the output modules. Input modules register arriving transport units at their assigned output module. The number of transport units missing within all incomplete batches at the output modules is decisive whether a transport unit can be introduced or not (cf. conditions (I1) and (I2)). This is responded to the respective input module accordingly. When unloading a transport unit at its output module, a new transport unit becomes released (cf. condition (I3)). If this is already registered, the corresponding input module, which is currently detaining the arrived transport unit, is notified to introduce it.

3.4 Chapter Conclusion

In this chapter, we present the general definition of a sequencing system which forms the basis for all approaches developed in the following. Overall, this responds to the first research question:

> **Which requirements are necessary for conveyor networks to enable multi-batch sequencing at highest density?**

The decentralized system design can be described as a graph-based model in which autonomous nodes transfer physical objects for sequencing as well as data objects for decentralized communication. We divide the set of all nodes into input, output, and sequencing nodes. The flow of physical objects proceeds from input to output nodes, where the predefined unloading sequences need to be observed. This is achieved by reordering the physical objects using the sequencing nodes in between.

Based on the graph-based system model, we derive capacity and structural system requirements for sequencing. Data object capacity can be assumed unconstrained, while for physical objects, node capacity is set to a minimum of one to represent their spatial dimensions. Sequencing requires buffering physical objects within the network to observe the predefined unloading sequences at the output nodes. Assuming that input and output nodes are only used for introducing and unloading, the physical object capacity of the network results from that of its sequencing nodes. Given a conveyor network comprising $|C|$ sequencing nodes, no more than $(|C| - 1)$ physical objects may occupy a sequencing node simultaneously to enable movements within the network. We define the subset of *released* physical objects U_r based on conditions (I1) to (I3). The *inflow constraints* require the physical objects to arrive at the system such that a feasible released set U_r can always be created. Physical objects arriving at their input node which are not yet released are prevented from being introduced into the network. The structure of a conveyor network results from the connections of input, output, and sequencing nodes establishing an entirely connected graph. Its sub-graph of sequencing nodes needs to satisfy at least 2-connectivity as sequencing at highest-density requires two internally disjoint paths between each pair of sequencing nodes.

For demonstration, we develop a practical showcase system where nodes, physical objects, and data objects are represented using conveyor modules, transport units, and messages, respectively. Objects are transferred by corresponding mechanical or electronic connections. Before operation, the system needs to be set up and configured. This gives the local directional distances for all modules and allows verifying whether the installed network is capable of sequencing. The showcase sequencing system receives arriving transport units from an upstream process at multiple input points, sequences them, and forwards these to a downstream process at multiple output points ($(m : n)$ setting).

4 Decentralized Sequencing

Chapter 4 of this dissertation is based on Fleischmann and Furmans (2023) (Chapter III).

Multi-batch sequencing from multiple input to multiple output modules requires the flow of arriving transport units to be processed correctly throughout the conveyor network. We present a decentralized sequencing algorithm starting with the overall design approach to outline its main ideas at system level (cf. Section 4.1). Based on that, we detail the concept of decentralized interactions at module level, where modules act as autonomous entities taking different roles (cf. Section 4.2). We use flow charts and state charts to demonstrate the algorithmic procedures based on the graphical notation of Figure 4.1. For demonstrating the developed algorithm, we implement an agent-based simulation model (cf. Section 4.3). By summarizing the results of this chapter, Section 4.4 provides an answer to the second research question of this dissertation.

4.1 Design Approach

To observe the predefined predecessor-successor dependencies for sequenced supply of physical objects, route planning needs to coordinate different routes. This requires buffering transport units in case at least one necessary predecessor within the unloading sequence is missing when it arrives at its input module. Therefore, we split the overall routes of buffered transport units into sub-routes (cf. Section 4.1.1). Buffer modules are used as intermediate destinations which need to be allocated to support efficient sequencing (cf. Section 4.1.2).

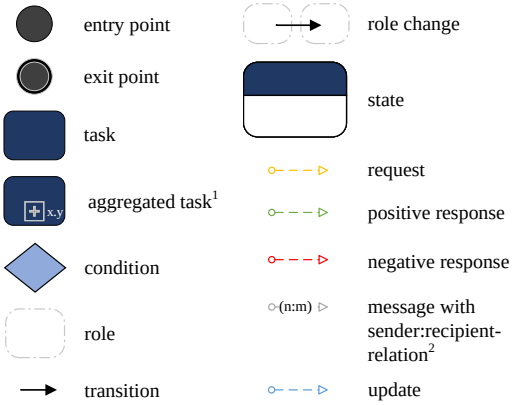


Figure 4.1: Graphical notation for flow charts and state charts.

Finding a path for each (sub-)route considers the resulting system occupation (cf. Section 4.1.3).

4.1.1 Transport Unit Flow

Sequencing differs from transporting, retrieving, picking, or sorting, as physical objects may only be unloaded if all of their predecessors have already been unloaded. To prevent a transport unit which is not yet requested when arriving at its input module from confounding the predefined unloading sequence, buffering is necessary (cf. Figure 4.2). We refer to a transport unit as *requested* if its corresponding output module is able to claim this transport unit, as all necessary preceding transport units have been scheduled there.

Processing a transport unit means specifying its route through the system. The overall route of each transport unit initially starts at the input module where it is introduced and ends at the output module where it is unloaded. In case of

1 This task is detailed in another flow chart, which can be found in Figure x.y of this dissertation.
2 The meaning of the colors yellow, green and red remains unchanged. If no sender:recipient-relation is specified, this corresponds to one sender and one recipient, i.e., (1:1).

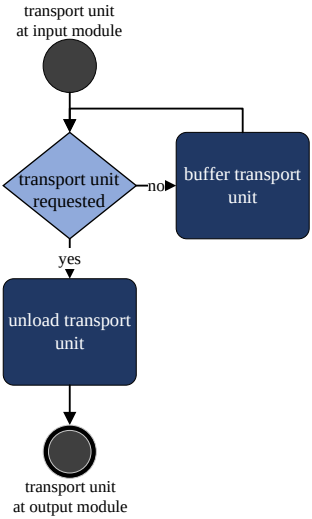


Figure 4.2: Processing transport units for sequencing.

buffering, this overall route is split into two sub-routes using a buffer module as intermediate destination. The first sub-route goes from the input module to the buffer module, while the second sub-route goes from the buffer module to the output module. If route splitting is limited to only these two sub-routes, this corresponds to a *static buffering* approach, as the transport unit occupies only a single buffer module until being unloaded at its output module. With *dynamic buffering*, a transport unit can switch between different buffer modules before being unloaded (Sohrt et al. 2014). This is crucial when operating in high(est)-density systems, where interfering items prevent directly accessing requested items.

We define two types of routes for sequencing transport units:

- *active routes* starting at an input module and/or ending at an output module and
- *passive routes* starting and ending at a sequencing module for buffering.

Active routes are necessary to introduce arriving transport units into the system and to unload requested transport units at their output modules. Thus, active routes generally guide the process of sequencing. Passive routes, instead, depend on active routes. If a not yet requested transport unit is buffered in the system such that it interferes with an active route, a passive route is initiated for relocation. This allows to represent the overall route of each buffered transport unit from the input to the output module as combination of one active route at the beginning and end and an arbitrary number of passive routes in between (cf. Figure 4.3). In case a transport unit is already requested when arriving at its input module, the two active routes are consolidated such that the transport unit is routed directly from the input to the output module.

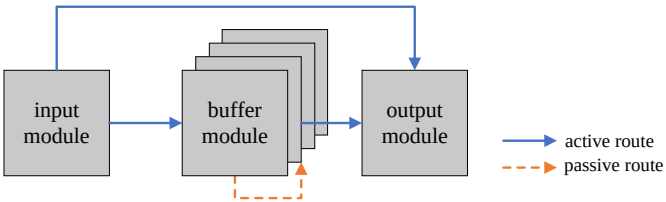


Figure 4.3: Splitting the overall route of a transport unit into active and passive routes.

By integrating the concept of active and passive routes, we can detail the necessary steps for processing a transport unit (cf. Figure 4.4). This generally represents the perspective of a single transport unit. However, the intelligence of the presented sequencing system is not provided by the transport units but by the autonomous conveyor modules (cf. Section 3.1). Thus, the objective of the decentralized sequencing algorithm is to achieve a transport unit behavior according to the flowchart of Figure 4.4 using decentralized interactions at module level (cf. Section 4.2).

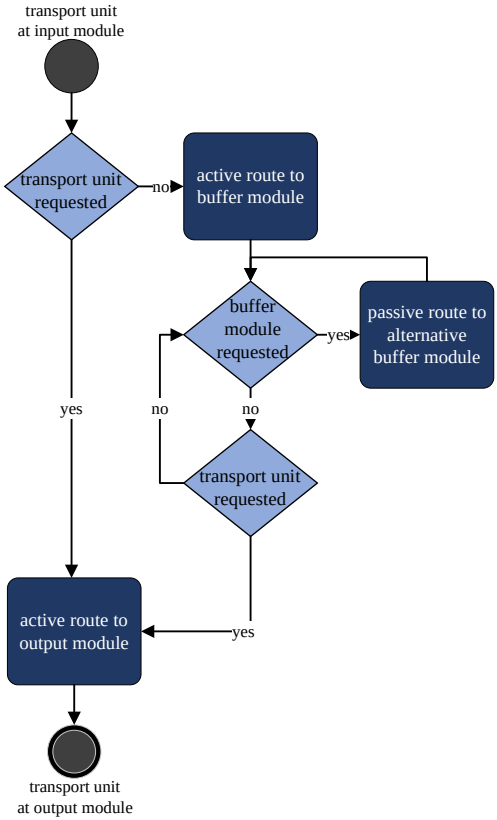


Figure 4.4: Flow chart for sequencing from the perspective of a transport unit.

4.1.2 Buffer Module Allocation

Active routes for not yet requested transport units as well as each passive route proceed from the currently allocated module of this transport unit to a sequencing module available for buffering. Generally, we only allow sequencing modules for buffering, as input and output modules are restricted to introducing and unloading transport units (cf. Section 3.2.1.2). To enable efficient sequencing, we develop the *unloading sequence-based buffer selection rule* for

active routes, while for passive routes, the *distance-based buffer selection rule* applies. Both are designed to meet different requirements of the specific route type when identifying suitable buffer modules (cf. Table 4.1).

Table 4.1: Buffer module allocation.

	Active route	Passive route
Requirement	Pre-sequenced buffering to enable fast unloading	Preserve pre-sequenced arrangement
Objective	Buffer predecessors closer to the output module than their successors	Move transport unit to an adjacent module
Selection rule	Unloading sequence-based buffer selection rule	Distance-based buffer selection rule
Selecting module	Output module	Buffer module of interfering transport unit

For active routes, buffering is organized to enable fast unloading as soon as the transport unit is requested. This is realized by allocating buffer modules according to the predefined unloading sequence. Transport units requested earlier are buffered closer to their corresponding output module than those requested later, which we refer to as *pre-sequencing*. As the number of missing transport units regarding an unloading sequence is locally available at each output module, these are responsible for requesting and selecting potential buffer modules.

Passive routes relocate transport units from previously allocated buffer modules. To preserve the pre-sequenced arrangement, a relocated transport unit is kept as close as possible to its previous position. Therefore, passive routes move a buffered transport unit to a module adjacent to its currently allocated buffer module the route originates from. In case the adjacent module is not available for buffering, as it is occupied by a buffered transport unit itself, this buffered transport unit is relocated to an adjacent module as well. The chain continues

until reaching the closest feasible non-buffering module. This distance-based selection is also preferable regarding energy consumption, measured in terms of the number of additional movements due to relocations.

4.1.3 Path Characteristics

While the path of an active route generally includes multiple modules, each passive route represents exactly one relocation step of a transport unit (cf. Section 4.1.2). Therefore, clearing the path of an active route due to an interfering buffered transport unit may require moving several transport units in a chain – especially when system density is high. We refer to this chain of relocation steps as *relocation route*, i.e. a relocation route may combine multiple passive routes in direction to a module available for buffering. Figure 4.5 shows a use case of two active routes from buffer to output and from input to buffer module, respectively, including their induced passive routes.

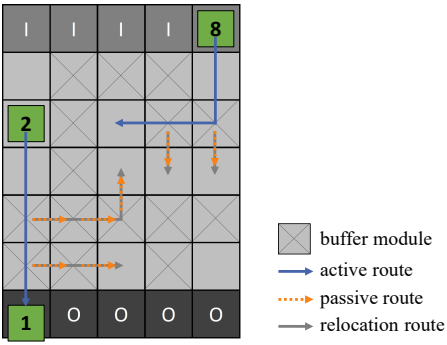


Figure 4.5: Use case of active and passive routes.

For efficient processing of transport units, paths of active as well as relocation routes aim to consider:

- the length of the route, as this defines the time for transferring the transport unit,
- the number of buffered transport units on the route, as these need to be relocated to be able to access there, and
- the number of directional changes on a route, as this implies a certain delay due to acceleration and deceleration as well as switching the conveyor modules between their rectangular transport directions.

4.2 Algorithm Concept

Processing transport units within the system as shown in the flowchart of Figure 4.4 is realized using decentralized communication of the autonomous system modules. To proceed in sequencing, active routes are crucial, which are initiated according to the predefined predecessor-successor dependencies (cf. Section 4.2.1). Using an offline route planning approach, all routes are scheduled entirely from their start module to their (intermediate) destination module (cf. Section 4.2.2). By aggregating the operations for route initiation and planning, each decentralized module can be represented as an independent state machine (cf. Section 4.2.3). After completing route planning, the corresponding transport units are physically transferred (cf. Section 4.2.4).

4.2.1 Route Initiation

Sequencing requires active routes for introducing arriving transport units from their input modules into the system as well as for unloading transport units out of the system at the output modules (cf. Figures 4.3 and 4.4). To observe the predefined predecessor-successor dependencies, information about newly arriving transport units at the input modules, buffered transport units at the sequencing modules, and requested transport units at the output modules needs

to be consolidated and coordinated. Due to the decentralized system setup, this information is held locally at the respective responsible modules.

Route initiation notifies the module currently assigned to a transport unit that planning an active route for this transport unit can start. Input modules identify newly arriving transport units and register them at their output module. Based on the current state of its unloading sequence, the latter is able to determine whether the route for the corresponding transport unit requires a buffer module or goes directly to the output module. In case of buffering, the output module requests potential buffer modules based on the unloading sequence-based buffer selection rule for active routes (cf. Section 4.1.2). To reduce the messaging load, the distance information from the initial system configuration (cf. Section 3.3.2.2) is used to iteratively modify the search distance until a suitable non-buffering module is found based on the current system occupation. The output module then returns the destination of the active route, i.e., either the identified module for buffering or the output module itself, to the requesting input module. Figures 4.6 and 4.7 illustrate the corresponding communication within the system for initiating an active route originating from an input module.

When scheduling a transport unit at its output module, currently buffered transport units might become requested. Output modules keep track of the respective buffer modules allocated to not yet requested transport units of their unloading sequence. Therefore, they can request such buffered transport units directly at their current buffer module to initiate an active route for unloading (cf. Figure 4.8).

Due to relocations of interfering buffered transport units, the local information at the output module about allocated buffer modules may be outdated up to this request. To deal with inconsistencies, the decentralized algorithm is designed to discard outdated requests at the receiving module if the allocation no longer applies. Additionally, each reallocated buffer module updates the corresponding output module, which we will discuss in more detail in Section 4.2.2.4. If an output module receives a relocation update for a requested transport unit, it

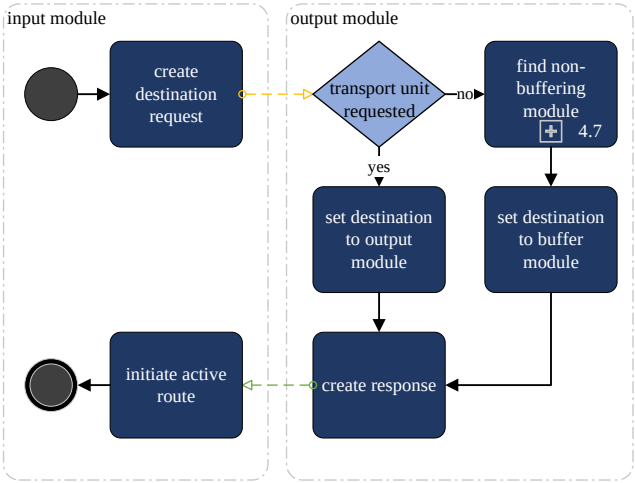


Figure 4.6: Initiating an active route starting from an input module.

resends the previous request to the newly allocated buffer module. This ensures that the overall route of the transport unit is correctly continued for sequenced unloading without deadlocks, livelocks, or starvation.

For each active route, the message received from the output module initiates route planning at the input and buffer module, respectively (cf. Figures 4.6 and 4.8).

4.2.2 Route Planning

Route planning originates from the currently allocated module of a transport unit to the (intermediate) destination module. Using a decentralized authorization concept, we are able to prevent deadlocks by ensuring consistent information in route planning (cf. Section 4.2.2.1). Path selection identifies an optimal series of modules to the destination of the active route according to the criteria defined in Section 4.1.3 (cf. Section 4.2.2.2). On this path, the corresponding modules negotiate reservations based on logical time windows (cf. Section 4.2.2.3). For

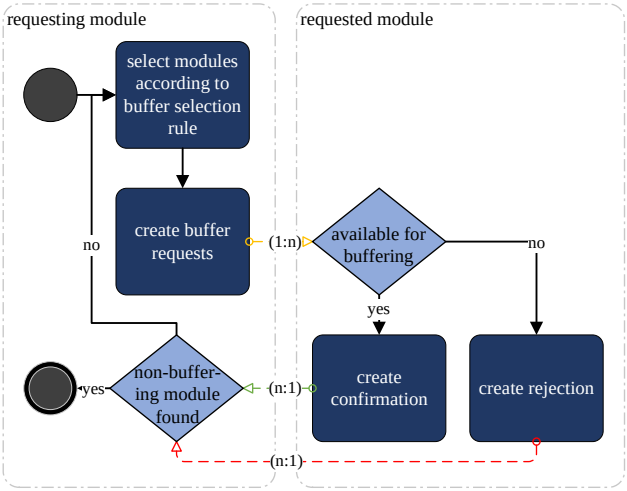


Figure 4.7: Iterative identification of non-buffering sequencing modules.

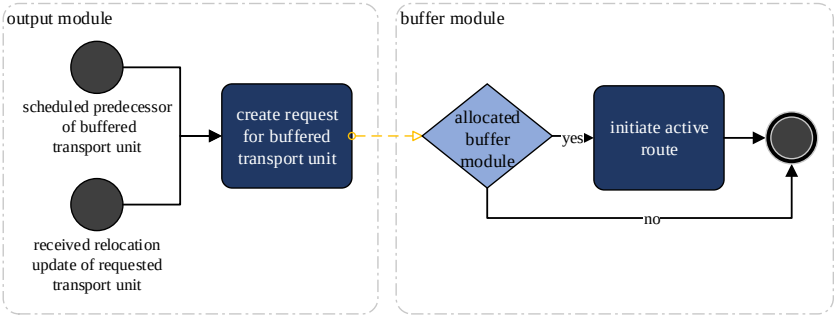


Figure 4.8: Initiating an active route starting from a buffer module.

interfering buffered transport units this requires specifying a relocation route using stepwise passive routes (cf. Section 4.2.2.4).

4.2.2.1 Authorization Concept

Within the decentralized system setup, all conveyor modules are independent and act autonomously. Planning several active routes simultaneously may create conflicts if requests are mutually exclusive. This could be solved by setting priorities among modules, transport units or directions as in Sohrt and Overmeyer (2020) or Hao (2020), but limits universality of the presented algorithm especially with respect to more complex network structures.

Therefore, we develop a *decentralized authorization concept* ensuring that all active routes are planned sequentially. Planning processes are decoupled such that deadlock-free system operation is guaranteed (cf. Section 5.1.1). Note that this only applies to route planning while not affecting their execution. As the time required for transferring data is negligible when compared to that for transferring physical objects, we do not expect limitations regarding system performance due to this sequential procedure.

For implementing the decentralized authorization concept, each system module acts as a state machine as illustrated in Figure 4.9. At any time, there is exactly one authorized module within the system. Only this module is entitled to plan an active route. At system configuration, it is initialized with the first input module of the system. Whenever the authorization is passed, the sequencing and input modules of the system are notified, as from these an active route may originate. Output modules never plan active routes and are therefore excluded from the authorization process.

Active routes are used to introduce and unload transport units into and out of the system (cf. Section 4.1.1). Each time a module intends to start planning an active route, it first sends a request to the currently authorized module to be authorized next. As soon as this completed planning its active route, it passes the authorization to the next module among the incoming requests. Active routes for unloading are prioritized over those for buffering, as the former directly contribute to system throughput. If an authorization request is not granted, the notification of changed authorization triggers a new request to the currently

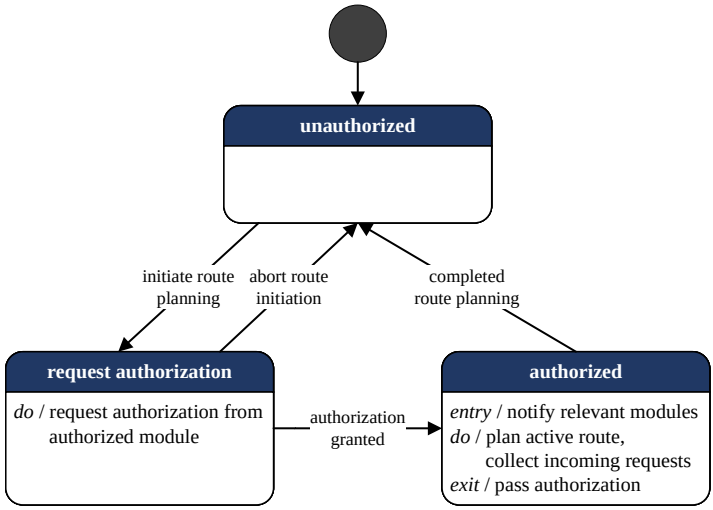


Figure 4.9: Authorization concept.

authorized module. To ensure that there is always an authorized module within the system, the authorized state can only be exited if and only if an authorization request has been received. In case a requested transport unit is relocated to an alternative buffer module, the former one is no longer responsible for initiating the active route of the corresponding transport unit. Thus, it aborts initiating the active route while the newly allocated buffer module is triggered by the output module to initiate route planning after receiving the update of the changed buffer module allocation (cf. Figure 4.8).

4.2.2.2 Path Selection

While paths of passive routes are rather clear defined (cf. Section 4.1.2), active routes entail a large range of possible paths. Based on the path characteristics defined in Section 4.1.3, we develop an adapted *decentralized A* algorithm* for path selection of active routes which is illustrated in Figure 4.10. Starting from the initiating module of the active route, the exploring module continues the

open list of known modules by its adjacent modules in each search iteration and identifies the best module to be explored next. In case of equivalent best modules, decisions are made randomly. The selected module is notified to continue the A* search, providing all currently available path information. Thus, the explored module becomes the next exploring module until the destination of the active route is found. Then, the best path is fixed by confirming the corresponding predecessor modules up to the initiating module.

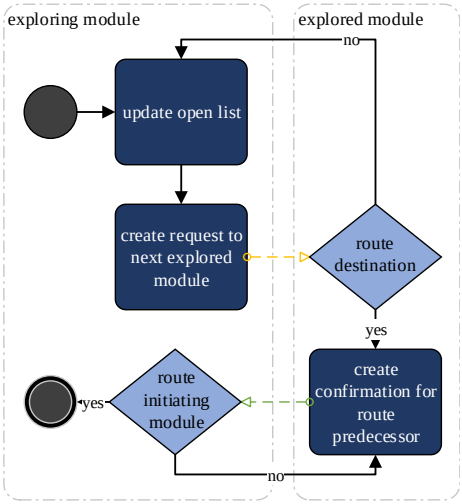


Figure 4.10: Path selection of active routes using the decentralized A* algorithm.

Selecting the next explored node is based on path lengths, interfering buffer modules, and necessary directional changes along the route (cf. Section 4.1.3). Due to the decentralized system setup, only the local information of exploring modules can be accessed directly in each iteration. It requires several iterations for specifying all necessary information concerning the path characteristics of a sub-path when determining an optimal solution.

Path lengths result from the directional distances specified during system configuration (cf. Section 3.3.2.2). Thus, this information can already be incorporated

to guide the A* search before notifying the next exploring module. Interfering buffered transport units cannot be detected before exploring the respective buffer module, as the allocation of buffered transport units is held locally. Directional changes can be derived based on the transmitted path predecessor combined with the direction towards the adjacent module. This implies a series of at least three modules for identifying directional changes along a potential path. Particularly, directional changes arising in later iterations cannot be anticipated.

We calculate the estimated path cost $\tilde{c}_n$ of an adjacent module n proceeding from module m as

$$\begin{aligned}\tilde{c}_n = \tilde{c}_m & \\ &+ \left(d^{\vec{n}}_{me} - d^{*}_{me}\right) \\ &+ \begin{cases} p_b & \text{if } m \text{ is buffer module} \\ 0 & \text{otherwise} \end{cases} \\ &+ \begin{cases} p_c & \text{if } n \text{ implies directional change} \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{4.1}$$

It incorporates the estimated path cost $\tilde{c}_m$ of module m from a prior iteration. Added to this is the length of the detour $(d^{\vec{n}}_{me} - d^{*}_{me})$ measured in module lengths when routing via its adjacent module n. It corresponds to the difference between the directional distance $d^{\vec{n}}_{me}$ to the destination module e via module n and the shortest possible distance d^{*}_{me} from module m to module e. The penalty terms p_b and p_c for buffer modules and directional changes can be parameterized to specifically guide path selection. However, adding p_c depends on the path predecessor and may retroactively change the cost estimate of different paths. For guaranteeing optimality, path cost estimation needs to be admissible according to the A* requirements, i.e. path costs must never be overestimated (Hart et al. 1968). To satisfy admissibility, p_c cannot be set arbitrarily.

We denote by

- $\tilde{c}^p$ the estimated cost for (sub-)path p,
- d^p the distance of (sub-)path p measured in module lengths,
- n_b^p the number of buffer modules on (sub-)path p ($n_b^p < d^p$),
- n_c^p the number of directional changes on (sub-)path p,
- p^* a sub-path of an optimal solution,
- $\acute{p}$ a sub-path of a non-optimal solution, and
- M the set of conveyor modules within the network.

As we aim to avoid buffer modules and directional changes within selected paths (cf. equation (4.1)), generally, the lower bound of

$$p_b, p_c \geq 0 \tag{4.2}$$

applies. This is also consistent with the restriction to non-negative edge weights required in the A* search.

The upper bound of p_c follows from Bellman's principle of optimality, where an overall optimal solution is derived from combining optimal partial solutions. Therefore, no estimated path costs may be generated for all possible sub-paths $\acute{p}$ and p^* with equal start and end nodes such that

$$\tilde{c}^{\acute{p}} \leq \tilde{c}^{p^*}. \tag{4.3}$$

This means that the estimated cost difference between each optimal and non-optimal sub-path must exceed p_c, as otherwise, the non-optimal sub-path may outperform the optimal one if the optimal sub-path includes a directional change prior to the non-optimal one. To generally satisfy admissibility within the decentralized A* search,

$$\begin{aligned} & \tilde{c}^{p^*} < \tilde{c}^{\acute{p}} \\ \Leftrightarrow \; & d^{p^*} + n_b^{p^*} \cdot p_b + n_c^{p^*} \cdot p_c < d^{\acute{p}} + n_b^{\acute{p}} \cdot p_b + n_c^{\acute{p}} \cdot p_c \end{aligned} \tag{4.4}$$

must hold for all possible sub-paths $\acute{p}$ and p^* with equal start and end nodes. This gives

$$p_c < \frac{\left(d^{\acute{p}} - d^{p^*}\right) + \left(n_b^{\acute{p}} - n_b^{p^*}\right) \cdot p_b}{n_c^{p^*} - n_c^{\acute{p}}}, \tag{4.5}$$

where $n_c^{\acute{p}} \neq n_c^{p^*}$. Note that overestimation is only critical in case of sub-paths with different numbers of directional changes, as otherwise, these are penalized equally such that their prioritization remains unchanged.

Due to equation (4.2) either

$$\left(d^{\acute{p}} - d^{p^*}\right) + \left(n_b^{\acute{p}} - n_b^{p^*}\right) \cdot p_b \overset{!}{>} 0 \qquad \wedge \qquad \left(n_c^{p^*} - n_c^{\acute{p}}\right) \overset{!}{>} 0 \tag{4.6}$$

or

$$\left(d^{\acute{p}} - d^{p^*}\right) + \left(n_b^{\acute{p}} - n_b^{p^*}\right) \cdot p_b \overset{!}{<} 0 \qquad \wedge \qquad \left(n_c^{p^*} - n_c^{\acute{p}}\right) \overset{!}{<} 0 \tag{4.7}$$

applies. Thus, we define the positive minimum of equation (4.5) for specifying p_c, i.e.

$$p_c \overset{!}{<} \min \left(\frac{\left(d^{\acute{p}} - d^{p^*}\right) + \left(n_b^{\acute{p}} - n_b^{p^*}\right) \cdot p_b}{n_c^{p^*} - n_c^{\acute{p}}} \right) \in \mathbb{R}^+. \tag{4.8}$$

Since the system is limited regarding its size

$$\left(d^{\acute{p}} - d^{p^*}\right), \left(n_b^{\acute{p}} - n_b^{p^*}\right), \left(n_c^{p^*} - n_c^{\acute{p}}\right) \in \mathbb{Z} \cap \{-|M|, \ldots, |M|\} \tag{4.9}$$

holds. Observing equations (4.6) and (4.7), the minimum of equation (4.8) results from

$$\max_{\substack{\acute{p},p^*:\\ \acute{p}\neq p^*}} \left| n_c^{p^*} - n_c^{\acute{p}} \right| \tag{4.10}$$

and

$$\min_{\substack{\acute{p},p^*:\\ \acute{p}\neq p^*}} \left| \left(d^{\acute{p}} - d^{p^*} \right) + \left(n_b^{\acute{p}} - n_b^{p^*} \right) \cdot p_b \right|. \tag{4.11}$$

Due to equation (4.9) and $M \neq \emptyset$, equation (4.10) can be restricted to

$$1 \leq \max_{\substack{\acute{p},p^*:\\ \acute{p}\neq p^*}} \left| n_c^{p^*} - n_c^{\acute{p}} \right| \leq |M|. \tag{4.12}$$

Equation (4.11) becomes arbitrarily small with growing system size, as it depends on the particular path characteristics of $\acute{p}$ and p^* as well as the value of p_b. Therefore, we set

$$\min_{\substack{\acute{p},p^*:\\ \acute{p}\neq p^*}} \left| \left(d^{\acute{p}} - d^{p^*} \right) + \left(n_b^{\acute{p}} - n_b^{p^*} \right) \cdot p_b \right| \leq \epsilon. \tag{4.13}$$

From this follows the upper bound of p_c as

$$p_c \overset{!}{<} \frac{\epsilon}{|M|} \tag{4.14}$$

covering all possible combinations of two sub-paths $\acute{p}$ and p^* to guarantee general optimality within the decentralized A* search for path selection.

4.2.2.3 Route Reservation

Route reservation aims to schedule the routes of transport units within the system. This comprises the selected path of the active route as well as its induced passive routes, where interfering buffered transport units are relocated

(cf. Section 4.2.2.4). We apply a time window-based approach using the concept of logical time. It builds upon the preliminary work of Lamport (1978) and Seibold et al. (2022).

Logical time is used to synchronize parallel processes in distributed systems. Each process comprises a defined causally related sequence of events (cf. Figure 4.11). Lamport (1978) defines the *"happened before"* relation '$\rightarrow$' representing an irreflexive partial ordering among the set of all events within the system: $e_i \rightarrow e_j$ implies event e_i happened before event e_j, i.e., e_i causally affects e_j. If neither $e_i \rightarrow e_j$ nor $e_j \rightarrow e_i$ holds, these events are stated concurrent. Each process q is assigned a logical clock C^q, where $C^q\langle e_i \rangle$ defines the logical time of event e_i within process q. From this follows the *clock condition*: If $e_i \rightarrow e_j$ then $C^q\langle e_i \rangle < C^{\acute{q}}\langle e_j \rangle$ for all events e_i, e_j of processes q and $\acute{q}$.

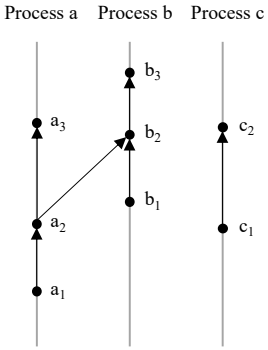


Figure 4.11: Parallel processes with multiple events and causal relations (based on Lamport 1978).

Seibold et al. (2022) transfer these findings to controlling decentralized material handling systems within a sorting application. A *process* relates to a transport route through the system, while each transfer between two adjacent modules corresponds to a single *event* of this process (cf. Figure 4.12). Each conveyor module is assigned a logical clock. Transferring a transport unit requires reserving a logical time window at all modules of its route. Therefore, each module of the system locally holds a reservation table in which all of its transfers are

scheduled. These are executed in ascending order of their reserved logical time windows. When completing a transfer, the involved modules update their logical clocks accordingly. Thus, clock times refer to events rather than physical time points.

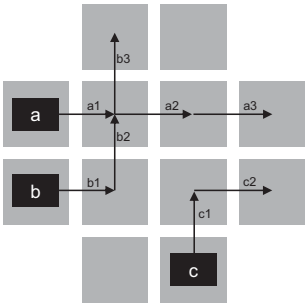


Figure 4.12: Describing transport routes using processes and events from Figure 4.11 (according to Seibold et al. 2022).

For route reservation within the developed decentralized sequencing algorithm, the involved modules of a transferred transport unit negotiate logical time windows matching their local reservation tables. Sequencing requires the following *reservation conditions* (R1) to (R6) to be satisfied. We use the notation according to Seibold et al. (2022) (cf. Table 4.2).

$$T_{in}\left(P_a, R_i\right) < T_{out}\left(P_a, R_i\right) \tag{R1}$$

The outgoing transport of each transport unit at each module is scheduled after its incoming transport there.

$$T_{out}\left(P_a, R_i\right) = T_{in}\left(P_a, R_j\right) \tag{R2}$$

Each pair of adjacent modules i and j along the transport route of a transport unit agrees on the timing regarding their common event. Combining reservation

Table 4.2: Notation for defining reservation conditions (based on Seibold et al. 2022).

Symbol	Physical equivalent
Processes P_a, P_b	Transport routes of transport units a and b
Process P_{s^a}	Transport route of a successor of transport unit a within their unloading sequence
Resource R_i, R_j	Conveyor modules i and j
Resource R_o	Output module o
Logical clock C_i	Logical clock of module i
Event $T_{in}\,(P_a, R_i)$	Incoming transfer of transport unit a at module i
Event $T_{out}\,(P_a, R_i)$	Outgoing transfer of transport unit a at module i

conditions (R1) and (R2) yields ascending logical time windows within the event sequence of each process.

$$T_{out}\,(P_a, R_i) < T_{in}\,(P_b, R_i) \tag{R3}$$

Each module can hold only one transport unit at a time. Thus, the respective logical time windows according to reservation condition (R1) for two routes scheduled at the same module may not overlap.

$$C_i < T_{in}\,(P_a, R_i) \tag{R4}$$

Upcoming events are scheduled in future logical time regarding the reserved module to satisfy the causal event dependencies (cf. Lamport 1978).

$$T_{in}\,(P_a, R_o) < T_{in}\,(P_{s^a}, R_o) \tag{R5}$$

Each succeeding transport unit may not enter at the assigned output module before any of its predecessors. This upholds the predefined unloading sequences.

$$T_{out}\,(P_a, R_i) = \infty \tag{R6}$$

Each buffered transport unit blocks its allocated buffer module for an unlimited time period until its predecessor is planned or it is relocated (cf. Figure 4.4). Thus, reservation condition (R6) defines a buffer module.

Establishing a set of logical time windows adhering to reservation conditions (R1) to (R6) is realized via decentralized negotiations along the selected path of active routes as shown in Figure 4.13. Adjacent modules iteratively suggest feasible logical time windows based on their local reservation tables until an agreement is found.

Each reservation request is created observing reservation condition (R1) at the requesting module. Furthermore, the suggested next open time window is always scheduled later than the requested one when rejecting reservation requests. Verifying the feasibility of the requested logical time by checking the local reservation table at the requested module reveals potential conflicts violating reservation conditions (R3) and (R4). Reservation condition (R5) only applies at output modules and represents an additional feasibility requirement. If the requested time disrupts the predefined unloading sequence, the output module suggests the next possible time window to guarantee the correct sequence required at the downstream process. Buffer modules need to satisfy reservation condition (R6). Thus, sequencing modules which are specified as the destination module of a route check a time window of indefinite length when receiving the corresponding reservation request. Reserving the destination module of the active route implies that an agreement among all involved modules is found, such that the reservations are confirmed up to the initiating module. Each pair of adjacent modules synchronizes the logical time of their common event to observe reservation condition (R2).

For active routes terminating at a buffer module, the overall route of the corresponding transport unit cannot yet be determined entirely (cf. Section 4.1.1). To correctly continue the subsequent active route from the buffer module to the output module, the buffer module registers the allocated transport unit at the responsible output module (cf. Figure 4.14). Thus, the latter is able to claim

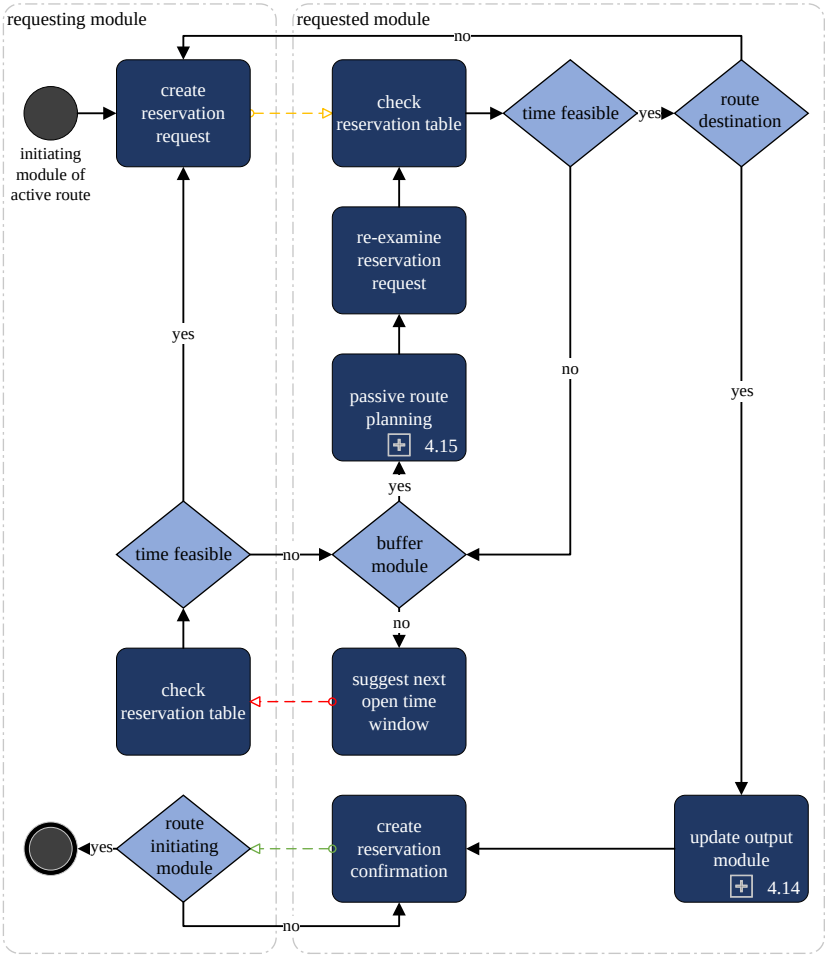


Figure 4.13: Active route reservation.

the buffered transport unit as soon as it is requested such that the active route for unloading at the output module is initiated (cf. Section 4.2.1).

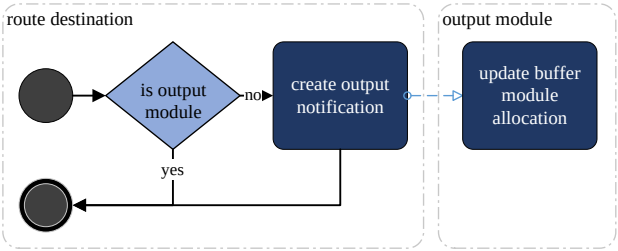


Figure 4.14: Updating buffer module allocation at output modules.

4.2.2.4 Relocation

In highest-density conveyor networks, a selected path of an active route may include several modules currently allocated for buffering transport units, which are not yet requested. Using a dynamic buffering approach (cf. Section 4.1.1), interfering transport units are relocated to alternative buffer modules to enable the path of the active route. Due to reservation condition (R6), buffer modules receiving a reservation request cannot accept any logical time window after the buffered transport unit enters there. Therefore, they initiate planning passive routes for relocating the interfering buffered transport unit before processing the received reservation request regarding the active route (cf. Figure 4.13).

A relocation route moves buffered transport units to access the module of the interfering transport unit at the active route. The number of buffered transport units to be moved results from the length of the relocation route. Based on the distance-based buffer selection rule (cf. Section 4.1.2), the currently allocated buffer module of the interfering transport unit first identifies the closest non-buffering module using the distance information from system configuration (cf. Figure 4.7).

Paths of relocation routes are planned stepwise, where passive routes are used to move buffered transport units towards the identified non-buffering module. Therefore, a relocation route is created by buffer modules successively requesting an adjacent module to pass their buffered transport unit. To guarantee system liveliness, relocation routes need to observe the following *path restrictions*:

(P1) The module the reservation request was received from is omitted.

(P2) No module is requested more than once for relocation.

Condition (P1) results from the spatial dimensions of the transferred transport units. When executing the active route, the interfering buffered transport unit needs to be relocated via an alternative path to avoid collision. This is always possible, as we claim 2-connectivity of the underlying conveyor network (cf. Section 3.2.2). Using condition (P2), we ensure cycle-free paths on passive routes for livelock prevention (cf. Section 5.2.1). We refer to modules not compliant with conditions (P1) and (P2) as *invalid modules*. With each further relocation request, these are updated to observe the path restrictions for passive routes.

Passive route reservation proceeds as illustrated in Figure 4.15. Starting from the allocated buffer module of the interfering transport unit, the shortest admissible path in direction to the identified non-buffering module is explored by successively sending relocation requests. An admissible path does not include any invalid modules. Therefore, each buffer module receiving a relocation request creates a subsequent relocation request directed to one of its admissible adjacent modules which represents a shortest path towards the identified non-buffering module. To avoid switching delays, adjacent modules are preferred if they are located on the same transport axis as the module from which the incoming relocation request was received. In case of equivalent adjacent modules, decisions are made randomly. If there is no admissible adjacent module available, the passive route to the predecessor of the relocation route is rejected. The predecessor in turn requests other admissible adjacent modules, if available. Otherwise, it also rejects to the sending module of its relocation request. Rejecting modules become invalid according to condition (P2).

The series of relocation requests continues until reaching a non-buffering module which can accept an indefinite buffer reservation according to reservation condition (R6). Then, the relocation route is defined enabling all adjacent modules of the included passive routes to negotiate logical time windows at which

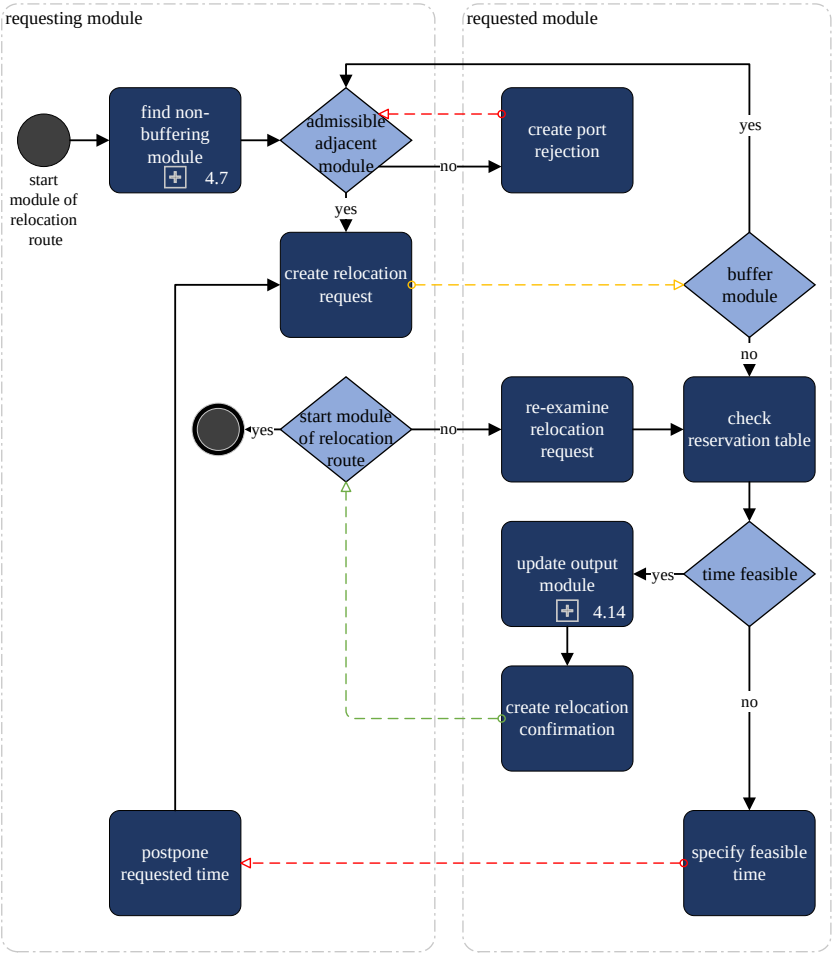


Figure 4.15: Passive route planning.

their allocated buffered transport unit will be passed to the subsequent module. When confirming a relocation request, the requesting module, i.e. the start module of this passive route, becomes a non-buffering module, as its indefinite buffer reservation is resolved. Thus, it is able to respond to the relocation request

it received from its predecessor on the relocation route. Each module accepting a relocation notifies the responsible output module such that it can correctly claim the buffered transport unit as soon as it is requested (cf. Section 4.2.1).

Especially in non-grid networks, the effective route length may be underestimated when searching for a non-buffering module based on shortest directional distances. Due to the decentralized system design, invalid modules extending a relocation route cannot be anticipated. Therefore, the relocation route terminates at the first requested non-buffering module. The identified non-buffering module solely directs the path of a relocation route while guaranteeing capacity to take a buffered transport unit in any case. This allows avoiding unnecessary relocations.

4.2.3 Routing State Machine

Aggregating the operations for route initiation and planning yields a routing state machine at decentralized module level as illustrated in Figure 4.16. It runs concurrently to the state machine of the decentralized authorization concept (cf. Figure 4.9). Buffer modules hold an indefinite buffer reservation according to reservation condition (R6), otherwise they are in state *transport*. Requested transport units require an active route from their allocated buffer module to the output module. Therefore, buffer modules of requested transport units become *impulsive buffers*, while buffer modules of transport units which are not yet requested remain in state *non-impulsive buffer*. Impulsive buffer modules aim to initiate active routes. This creates interdependencies with the decentralized authorization concept (cf. Section 4.2.2.1), as planning an active route may not be started before being authorized.

Input and output modules are generally not used for buffering transport units such that their routing state machines can be simplified (cf. Figure 4.17). Introducing transport units into the system requires an active route starting from the input module the transport unit arrives at. Therefore, input modules are included within the decentralized authorization concept and intend to initiate

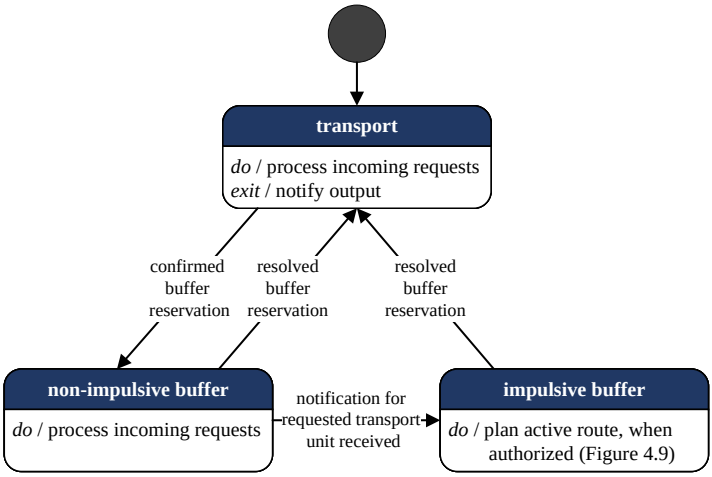
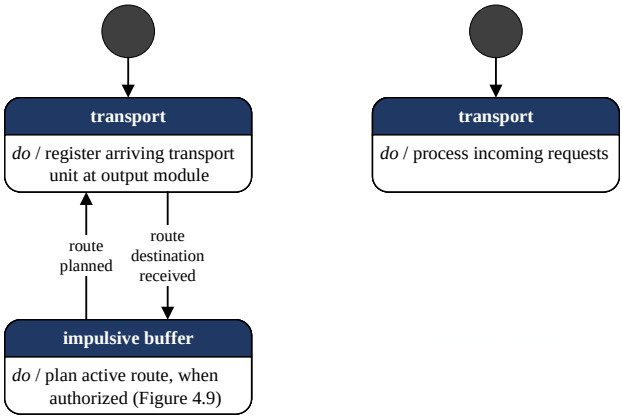


Figure 4.16: Routing state machine of sequencing modules.

an active route as impulsive buffers upon receiving the destination for their currently allocated transport unit from the output module. As the next active route is not required until another transport unit arrives, input modules switch to state transport after completing route planning. Output modules are only in state transport, as they do not initiate active routes. Transport units which enter there are immediately passed to the downstream process.

4.2.4 Transport Execution

In offline route planning, transports are not executed until route planning is complete (cf. Section 2.2). Each active and passive route represents an independent sub-route. Therefore, the physical transfers of the corresponding transport unit are initiated after confirming the reservation at the start module of an active or passive route. These are processed following their reserved logical time windows in ascending order (cf. Seibold et al. 2022). Within the decentralized system setup, this is realized by exchanging transport requests and confirmations (cf. Figure 4.18). A transport unit is transferred if and only



(a) Input modules. **(b)** Output modules.

Figure 4.17: Routing state machine of non-sequencing modules.

if both adjacent modules aligned their transport direction and agree on the next logical time step within their respective reservation tables. After completing a transfer, the involved modules forward their logical clocks. The sending module updates its logical clock to T_{out}, the receiving one to T_{in} of their reserved logical time windows for the corresponding transport unit. This guarantees an unambiguous transfer sequence at each module within the system providing deadlock-free system operation during transport execution (cf. Section 5.1.2).

4.3 Agent-Based Simulation Model

To validate the developed decentralized sequencing algorithm, we implement an agent-based simulation model using the Java-based simulation software *Anylogic*. As the proposed decentralized sequencing system consists of several autonomous decisional entities, which act independently, the overall system behavior cannot be described analytically (Fottner et al. 2021). For such systems, agent-based simulation models are increasingly used to analyze the complex

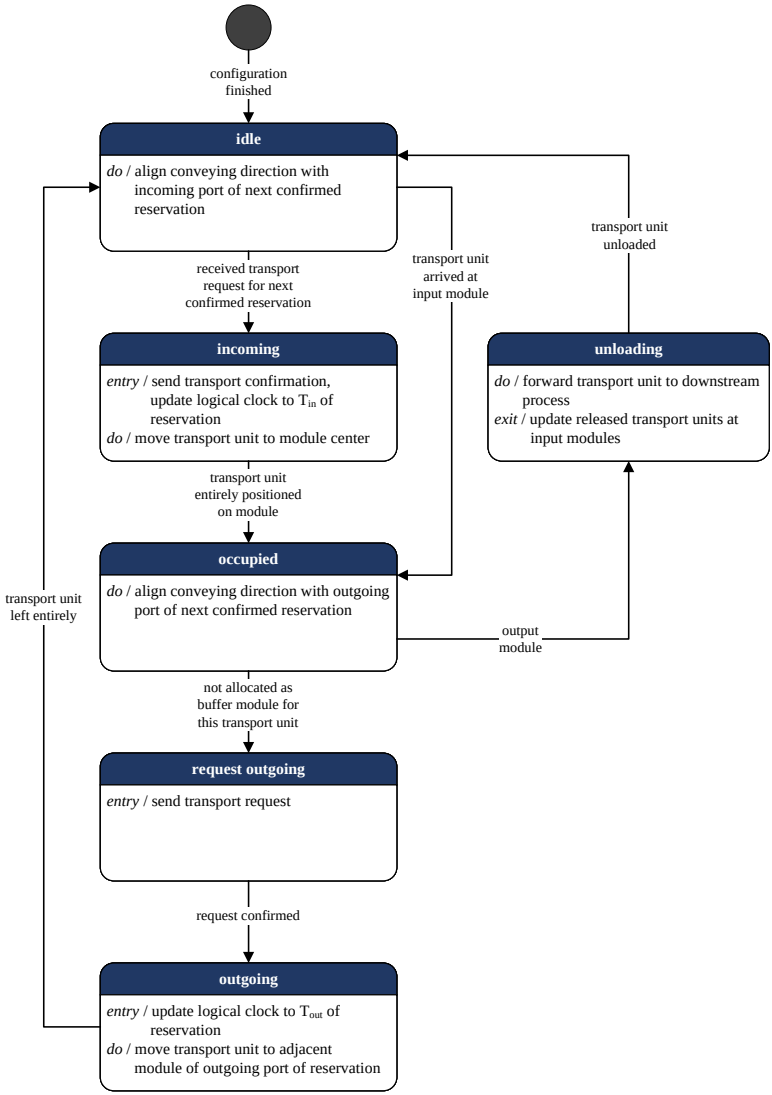


Figure 4.18: State machine for transport execution.

interdependencies (Macal and North 2014). Object-oriented programming languages are suitable, as attributes and methods can be encapsulated into the entities themselves (Law 2015, p. 695). Creating an agent-based model comprises defining the simulation environment, the agent types, as well as their interactions (Macal and North 2014), which we describe in Sections 4.3.1, 4.3.2, and 4.3.3, respectively.

4.3.1 Simulation Environment

System installation (cf. Section 3.3.2) and system-related data recording (cf. Section 8.1.2) are included in the central simulation environment to provide a more user-friendly simulation model. These are already realized within existing decentralized applications such that we can reduce model complexity without loss of generality. We outline the simulative implementation of system installation and system-related data recording in Sections 4.3.1.1 and 4.3.1.2, respectively. As the flow of transport units arriving from the upstream process needs to be compatible with the available sequencing capacity of an installed conveyor network (cf. Section 3.2.1.2), we generate the network inflow within the implemented simulation model to meet the defined inflow constraints (cf. Section 4.3.1.3).

4.3.1.1 System Installation

For system setup, the network of the considered sequencing system is read from an external file defining the number and arrangement of sequencing, input, and output modules. Each required conveyor module is created as an independent conveyor module agent and positioned within the simulation area accordingly. The lowest indexed input module is identified to be authorized within the decentralized authorization concept at each module (cf. Section 4.2.2.1). Additionally, the parameters of the decentralized sequencing algorithm p_b and p_c are set locally (cf. Section 4.2.2.2).

From the network structure, adjacent modules including their connecting directions are derived for each module. This allows to calculate the shortest directional distances – measured in module lengths – from each module to all other modules in the system via each of its adjacent modules. Based on that, we can verify the feasibility requirements for sequencing networks as defined in Section 3.2.2. If the conveyor network is capable of sequencing, system configuration is finished such that processing of arriving transport units starts.

4.3.1.2 Data Recording

Providing system-related key figures allows to assess the performance of the decentralized sequencing system directly at runtime. All local data of the module agents is continuously consolidated to an overall system result and visualized using different diagrams (cf. Figure 4.19). After terminating a simulation run, aggregated key figures are calculated from all recorded data. Additionally, heat maps and diagrams are generated. All evaluations are exported to an external file to be used for further analysis.

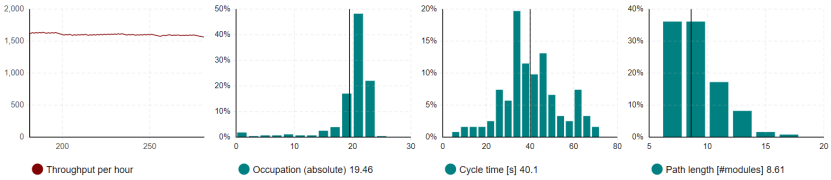


Figure 4.19: Snippet of simulation model – Performance Evaluation.

4.3.1.3 Network Inflow

To ensure running operations without congestion, we need to generate the flow of transport units arriving at the system such that the defined inflow constraints can be satisfied (cf. Section 3.2.1.2). Generally, we describe the set of transport

units for sequencing within the implemented simulation model by the numerical parameters

- *batch size* $k \in \mathbb{N}$
- *sequencing rate* $s \in [0, 1]$, and
- *arrival rate* $\lambda \in \mathbb{R}$

as well as the non-numerical parameters

- *arrival sequence* and
- *input-output assignment*.

These can be set stochastically or deterministically. The batch size corresponds to the number of transport units included in each batch. Completely and partially defined unloading sequences, as shown in Figure 3.5, are generated using the sequencing rate. The rank r^b of a transport unit b within its batch sequence for unloading follows as

$$r^b = \begin{cases} 1 & \text{if } \nexists\, p^b \\ r^{p^b} + 1 & \text{if } \exists\, p^b \wedge \varrho < s \\ r^{p^b} & \text{if } \exists\, p^b \wedge \varrho \geq s, \end{cases} \tag{4.15}$$

where p^b represents the direct predecessor of transport unit b within the batch sequence and ϱ a $[0, 1)$-uniformly distributed random number. This means that for completely defined unloading sequences, $s = 1$ applies. The share of equally ranked transport units within a batch increases with decreasing value of s. We obtain a batch sorting use case for $s = 0$, as no predecessor-successor dependencies arise within a single batch.

The arrival sequence of a batch corresponds to a permutation of the predefined unloading sequence. The more these two sequences diverge, the more capacity needs to be provided by the sequencing modules for buffering not yet requested transport units. Within the arrival sequence of a batch, each transport unit is

assigned to one of the input modules of the network for being introduced into the system. Furthermore, each batch entails a specified destination, i.e., one of the output modules where it will be unloaded. This is captured using the input-output assignment. To model a continuous flow of materials, transport units arrive at their assigned input modules according to a defined arrival rate based on exponentially distributed interarrival times. To compare different conveyor networks regardless of their number of input modules, λ relates to the overall sequencing system. Assuming that arriving transport units are on average equally distributed among all input modules within the network, the arrival rate λ_i of input module $i \in I$ follows as

$$\lambda_i = \frac{\lambda}{|I|}. \tag{4.16}$$

Observing the inflow constraints means that creating a subset of released transport units according to conditions (I1) to (I3) is always possible from the flow of transport units arriving at the system. We can ensure conditions (I1) and (I2) by restricting the maximum processable batch size based on the given conveyor network as well as the mixing of batches within the arrival characteristics at the input modules. Condition (I3) is intrinsically satisfied.

Consider a single batch sequencing system where only one batch of size n with completely defined unloading sequence is processed. If the transport units of this batch arrive in entirely reverse order, the network is initially filled with buffered transport units of rank $2 \dots n$. When introducing the last transport unit, i.e., the first one to be unloaded at the assigned output module, $n-1$ sequencing modules are already occupied. In this case, n sequencing modules are necessary to accept all transport units of this batch. Moving transport units within the network requires at least one unoccupied sequencing module. Thus, we limit a conveyor network comprising the set C of sequencing modules to processing batches with a maximum of $(|C|-1)$ transport units to handle arbitrary batch arrival sequences. For single batch sequencing systems, this requirement suffices to observe conditions (I1) and (I2).

To extend the single batch sequencing system such that multiple batches – still with a maximum size of $(|C|-1)$ – can be processed simultaneously within an $(m:n)$ setting, we specify a definite batch processing sequence. This needs to be observed within the arrival characteristics of transport units at all input modules. Let π_i^b and $\pi_i^{\tilde{b}}$ denote the positions of two transport units b and $\tilde{b}$ within the queue of arriving transport units at their input module i. Both transport units are associated with different batches, but assigned to the same input module. If the batch containing transport unit b is processed before that of transport unit $\tilde{b}$, then

$$\pi_i^b < \pi_i^{\tilde{b}} \tag{4.17}$$

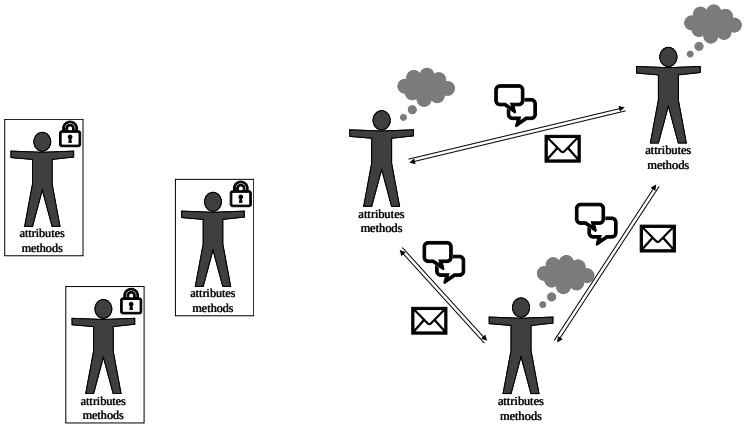
needs to be satisfied for all input modules where transport units assigned to both of these batches arrive.

Due to the maximum processable batch size of $(|C|-1)$, the available capacity of the conveyor network suffices to entirely sequence each batch. Restricting the batch mixing within the arrival characteristics at the input modules allows this capacity to be applied to multi-batch sequencing without requiring additional sequencing modules. Transport units are introduced if and only if they can be included in the set of released transport units, which is limited to $(|C|-1)$ (cf. condition (I2)). Thus, available capacity within the network is allocated to buffered transport units while ensuring that a transport unit requested next within the unloading sequence of an output module can still be introduced when arriving at its input module. Each batch can be processed entirely such that we achieve a continuous flow of materials in sequencing.

Note that the effective arrival *times* of transport units result from the interarrival time distribution at the corresponding input modules. Transport units arriving at different input modules are independent in terms of their arrival times, i.e., a transport unit assigned to a subsequently processed batch may arrive earlier at its input module than a transport unit assigned to an previously processed batch. To comply with the defined inflow constraints, only their *positions* within the queue of arriving transport units are relevant.

4.3.2 Agent Types

In the literature, there is no universal agreement on precisely defining the term *agent* in the context of agent-based simulation models (Macal and North 2014). Based on the definitions given in Macal (2016), we define two agent categories to model the decentralized sequencing system: active and passive agents. Passive agents can be understood as self-centered entities incorporating several attributes and methods, whereas active agents additionally take decisions and are able to interact with other active agents (cf. Figure 4.20).



(a) Passive agents. **(b)** Active agents.

Figure 4.20: Agent types.

Transport units (cf. Section 3.3.1.2) represent passive agents within the simulation model. They are illustrated as colored, numbered squares, where the color corresponds to their batch and the number to their rank within the batch. The decentralized controlled conveyor modules (cf. Section 3.3.1.1) represent active agents. They are illustrated as gray squares sized to cover the area of a transport unit, where input, output, and sequencing modules can be distinguished by their hue. Additionally, an output module inherits the color of the batch it is

currently unloading at runtime. Figure 4.21 shows a snippet of the simulation model of a 5×5 square arrangement of sequencing modules including 5 input and 5 output modules at the top and bottom, respectively.

Figure 4.21: Snippet of simulation model – Sequencing System.

The arrival of transport units follows a given interarrival time distribution (cf. Section 4.3.1.3). Once a transport unit enters the system at the assigned input module, it becomes visible there. The transport unit disappears when reaching its assigned output module. Further illustration of the upstream and downstream processes is not included, as these fall outside the defined system boundaries (cf. Section 3.1). Identifying transport units in terms of their batch and rank is not modeled explicitly, but assumed to be given and without errors.

4.3.3 Agent Interactions

Active agents are able to interact with each other using events, i.e. sending messages and transferring transport units. These events trigger reactions at the receiving agent, such as

- initiating an active route,
- forwarding a message,

- sending a response message,
- taking an incoming transport unit, or
- updating its logical clock.

To reduce the processor load during simulation execution, the message transmission mechanism (cf. Figure 3.7) is modeled as direct messaging events with a temporal delay proportional to the distance between message sender and recipient. Thus, sending a message from module i to module j takes a time of

$$t_{msg}^{ij} = d_{ij}^{*} \cdot t_{msg}^{m\acute{m}}, \tag{4.18}$$

where $t_{msg}^{m\acute{m}}$ denotes the message sending delay between two adjacent modules and d_{ij}^{*} the length of a shortest path of modules from module i to module j. We assume faultless communication within the simulation model.

In reality, the drives of conveyor modules, which are controlled by attached sensors and actuators, enable physically moving transport units (cf. Section 3.3.1.1). We omit specifically modeling the corresponding hardware control within the simulation to reduce model complexity. Transport units are transferred between adjacent modules, which takes a time of t_{conv}. We neglect acceleration and deceleration by assuming them to be included in t_{conv}. Changing between orthogonal conveying directions at a module requires switching from roller to belt drive or vice versa (cf. Section 3.3.1.4). This causes a time delay of t_{switch}. We assume faultless transport execution within the simulation model, i.e. transport units are always transferred along their planned route according to the negotiated reservations at each module.

Unless otherwise stated, we set $t_{msg}^{m\acute{m}} = 0.005\,s$, $t_{conv} = 1\,s$, and $t_{switch} = 0.5\,s$.

4.4 Chapter Conclusion

In this chapter, we develop the overall design, the decentralized concept as well as the simulative implementation of the decentralized sequencing algorithm. Overall, this responds to the second research question:

> **How can we enable multi-batch sequencing of physical objects using decentralized controlled highest-density conveyor networks?**

Sequencing is realized by splitting the overall route of a transport unit from its input to its output module into sub-routes. We differentiate active and passive routes. Active routes generally guide the process of sequencing. They enable introducing arriving transport units into the system as well as unloading requested transport units at their output modules. Passive routes depend on active routes and are used to relocate buffered transport units interfering with active routes. Allocating buffer modules for active routes is based on the *unloading sequence-based buffer selection rule*, while for passive routes the *distance-based buffer selection rule* applies. These are designed to create a pre-sequenced arrangement of buffered transport units to enable fast unloading as soon as these are requested at their output modules. Pathfinding for active and passive routes considers the length of a route as well as buffered transport units and directional changes.

Decentralized communication of the system modules achieves processing transport units within the system for sequencing. Active routes are initiated based on the scheduled reservations at the output modules such that the predefined predecessor-successor dependencies are observed. Due to an offline route planning approach, all routes are planned entirely from the start module to the (intermediate) destination module before being executed. Planning an active route comprises

- being granted the authorization for route planning,
- selecting the path of modules forming the active route, and

- reserving feasible logical time windows for the active route as well as all induced passive routes for relocation.

The decentralized authorization concept ensures that all active routes are planned sequentially. This avoids conflicts and enables deadlock-free system operation. For path selection, we develop an adapted decentralized A* algorithm including the defined path characteristics. Optimality is generally guaranteed if it is parameterized as

$$p_c < \frac{\epsilon}{|M|}. \tag{4.19}$$

Route reservation follows a time window-based approach using the concept of logical time. Adjacent modules of active routes negotiate feasible logical time windows according to the defined reservation conditions (R1) to (R6). As in high(est)-density conveyor networks buffered transport units may interfere with active routes, relocations and even chains of relocations are required. These are planned as sub-processes during route reservation using stepwise passive routes. Satisfying 2-connectivity always enables specifying a feasible relocation route starting from any sequencing module within the conveyor network. After confirming all reservations up to the start module of an active or passive route, the corresponding transport units are physically transferred by processing the scheduled reservations in ascending order at each module.

For demonstrating the developed decentralized sequencing algorithm, we implement an agent-based simulation model where each conveyor module is represented by an individual agent instance. These interact with each other using events, i.e. sending messages and transferring transport units. This allows predicting the behavior of the physical decentralized conveyor network for sequencing in reality.

5 System Liveliness

Chapter 5 of this dissertation is based on Fleischmann and Furmans (2023) (Chapter IV).

Decentralized systems operate based on local decisions of several autonomous entities. Each of them is only aware of a small portion of the information about all processes and resources within the system (cf. Section 3.3.1). This makes it prone to entering states from which it cannot recover such that – despite faultless operation – the system can no longer progress. These states are referred to as deadlock, livelock or starvation situations (Tai 1994). In the presented sequencing system, the algorithmic operations as well as the allocation of resources for physically moving transport units are processed at decentralized module level. In the following, we show that system liveliness is generally ensured by demonstrating deadlock, livelock, and starvation prevention in Sections 5.1, 5.2 and 5.3, respectively. By summarizing the results of this chapter, Section 5.4 provides an answer to the third research question of this dissertation.

5.1 Deadlocks

A deadlock results from a set of processes where each process is waiting for an event of another process in this set (Tanenbaum and Bos 2015, p. 439). We subdivide the investigations concerning deadlock prevention into algorithmic operations (cf. Section 5.1.1) and resource allocation (cf. Section 5.1.2).

5.1.1 Algorithmic Operations

The algorithmic operations are deadlock-free if we can guarantee that for each arriving transport unit a complete route from its input to its output module is initiated in finite time, observing the predefined predecessor-successor dependencies. This allows a continuous flow of all transport units through the system. At system level, a feasible solution needs to exist for the set of all necessary active and passive routes. This requires all relevant modules to use consistent information. Outdated or incomplete local information may create requests which are mutually exclusive. We initially demonstrate how these inconsistencies are prevented within the algorithmic operations. Based on that, we can ensure that the system can always proceed in processing arriving transport units if the defined system requirements (cf. Section 3.2) are satisfied.

Local module information is updated when receiving confirmations. The decentralized authorization concept (cf. Section 4.2.2.1) guarantees that active routes are planned sequentially based on updated system information. Each request is confirmed before receiving a new request concerning another active route. This ensures that each module is in the correct state to respond to an incoming request at any time. Planning multiple active routes simultaneously is possible if and only if there is more than one module in state authorized. This requires either an authorized module to pass on the authorization before completing route planning of its current active route, or multiple modules to be notified of being authorized next. The former case contradicts the specification of the state transition from authorized to unauthorized (cf. Figure 4.9). Assuming faultless system behavior, simultaneously authorizing multiple modules is impossible, as exactly one sending module is selected from the incoming requests. Thus, we can guarantee that all active routes are planned sequentially which prevents interdependencies of different active routes and guarantees consistent system information.

Due to sequential route planning, we can consider each active route individually. Active routes are necessary to keep the system operating for sequencing. We show that for each arriving transport unit, an active route to its output module

is initiated in finite time. This ensures that the algorithmic operations required for processing each transport unit are always triggered and prevents the system or parts of it from stagnating in a deadlock situation.

Initiating an active route to the output module of a transport unit is possible as soon as it is requested (cf. Section 4.1.1). This requires all of its necessary predecessors to be introduced into the system and scheduled at the output module. We guarantee this using the defined inflow constraints (cf. Section 3.2.1.2). Condition (I1) ensures that each missing transport unit within the unloading sequence of an output module is available in finite time such that the route of the direct successor can be planned while observing reservation condition (R5). The corresponding notification of the output module triggers route initiation at the allocated buffer module (cf. Section 4.2.1). Using condition (I3), the set of released transport units is updated for processing newly arriving transport units. Condition (I2) limits the set of released transport units such that moving transport units is always possible. As passive route planning interrupts route reservation on the active route, multiple relocation routes are never planned at the same time. Therefore, one non-buffering sequencing module is generally sufficient within the network. The algorithmic operations for sequencing can continuously proceed preventing the system or parts of it from becoming deadlocked.

5.1.2 Resource Allocation

Resource allocation for physically transferring the corresponding transport units implies coordinating the utilization of the common system resources. According to Coffman et al. (1971), a resource deadlock requires the following four conditions to coexist:

(D1) *Mutual exclusion:* Processes claim exclusive control of their required resources.

(D2) *Hold and wait:* Processes hold allocated resources while waiting for further resources.

(D3) *No preemption:* Allocated resources are used to completion and released by the process holding them.

(D4) *Circular waiting:* There is a circular chain of processes, each of which is requesting a resource held by the next process in the chain.

To prevent deadlocks, at least one of these four conditions needs to be excluded (Coffman et al. 1971). In modular conveyor systems, conditions (D1) to (D3) are generally true, as (Mayer and Furmans 2010):

- each conveyor module can handle only one transport unit at a time satisfying mutual exclusion (cf. condition (D1)),
- each transport unit occupies a conveyor module until the next conveyor module on its overall route is specified and becomes available satisfying hold and wait (cf. condition (D2)), and
- occupying a conveyor module cannot be interrupted by another transport unit satisfying no preemption (cf. condition (D3)).

Thus, we need to demonstrate that circular waiting (cf. condition (D4)) is generally excluded when allocating scheduled resources. Circular waiting requires a set of transport units – referred to as processes $P_1 \dots P_n$ – holding a chain of conveyors – referred to as resource $R_1 \dots R_n$ – forming a closed loop while each process requests the resource currently held by the next process in the chain (cf. Figure 5.1).

Referring to Table 4.2, the overall route of each transport unit through the system corresponds to a cohesive process starting at an input module and ending at an output module. To observe the predefined predecessor-successor dependencies, planning the overall process of a transport unit is decomposed by defining active and passive routes (cf. Section 4.1.1). For each buffered transport unit, an indefinite buffer reservation is created at the allocated resource for

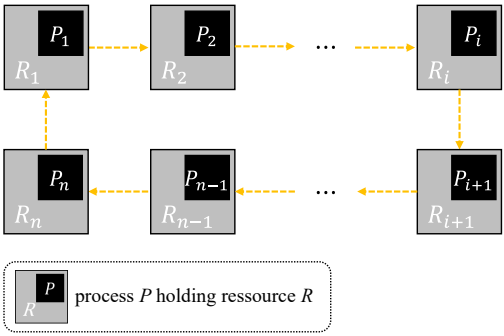


Figure 5.1: Circular waiting condition.

buffering according to reservation condition (R6). It is resolved entirely when initiating the active route from the buffer module to the output module (cf. Section 5.1.1). Thus, we obtain a finite overall process for each transport unit ending up at its output module. It represents a series of ascending logical time windows according to reservation conditions (R1) and (R2) even if the route of the transport unit is partially specified as composition of active and passive routes. This allows demonstrating deadlock prevention for resource allocation similar to Seibold et al. (2022).

Due to reservation conditions (R1) and (R3), satisfying circular waiting (cf. condition (D4)) as illustrated in Figure 5.1 results in the following reservation dependencies for processes $P_1 \dots P_n$:

$$\begin{aligned} T_{in}\left(P_1, R_1\right) &< T_{out}\left(P_1, R_1\right) < T_{in}\left(P_n, R_1\right) \\ T_{in}\left(P_2, R_2\right) &< T_{out}\left(P_2, R_2\right) < T_{in}\left(P_1, R_2\right) \\ &\vdots \\ T_{in}\left(P_n, R_n\right) &< T_{out}\left(P_n, R_n\right) < T_{in}\left(P_{n-1}, R_n\right). \end{aligned} \tag{5.1}$$

Based on reservation condition (R2), we obtain for the circular chain of adjacent resources $R_1 \ldots R_n$:

$$\begin{aligned} T_{out}\left(P_1, R_1\right) &= T_{in}\left(P_1, R_2\right) \\ &\vdots \\ T_{out}\left(P_{n-1}, R_{n-1}\right) &= T_{in}\left(P_{n-1}, R_n\right) \\ T_{out}\left(P_n, R_n\right) &= T_{in}\left(P_n, R_1\right). \end{aligned} \tag{5.2}$$

From this, it follows

$$\begin{aligned} T_{out}\left(P_1, R_1\right) < T_{out}\left(P_n, R_n\right) &< T_{out}\left(P_{n-1}, R_{n-1}\right) \\ < \quad \ldots \quad &< T_{out}\left(P_2, R_2\right) < T_{out}\left(P_1, R_1\right) \end{aligned} \tag{5.3}$$

which is a contradiction. This demonstrates that observing reservation conditions (R1) to (R6) within the algorithmic operations prevents deadlocks during resource allocation by excluding circular waiting (cf. condition (D4)). The scheduled reservations create a consistent ordering for processing transport units at each module. As completing the transfer of a transport unit includes forwarding the logical clock at the involved resources according to the logical time of their common event (cf. Section 4.2.4), this ordering is preserved due to reservation condition (R4).

5.2 Livelocks

In a livelock, a process indefinitely repeats the same sequence of actions without progressing (Holzmann 1991, p. 38). We subdivide the investigations concerning livelock prevention into algorithmic operations (cf. Section 5.2.1) and resource allocation (cf. Section 5.2.2).

5.2.1 Algorithmic Operations

Within the algorithmic operations, livelocks exist if modules send and receive messages in an endless loop. In Section 5.1.1 we demonstrate that a feasible solution for processing each arriving transport unit using active routes exists at system level. This ensures that the corresponding active routes are initiated. For livelock prevention, we show that this solution is found at decentralized module level such that the algorithmic operations terminate at the initiating module. We can guarantee this if path selection, route reservation, as well as all induced relocation operations terminate.

Path selection iteratively explores the network of conveyor modules from the start module to the destination module of an active route using the decentralized A* algorithm. This requires a path to exist which is given by the defined network requirements (cf. Section 3.2.2). Messaging loops imply that the same modules are notified more than once to continue the A* search, i.e., the selected path needs to include a cycle. As the decentralized A* algorithm is based on non-negative edge weights (cf. Section 4.2.2.2), a path including a cycle is always less valuable than the same path without the cycle. Thus, modules which have already been explored are never selected for re-exploration such that messaging loops are generally excluded. This guarantees that path selection always terminates with confirming the active route at its initiating module.

Route reservation includes the routes of multiple transport units when interfering buffered transport units are relocated using passive routes. This requires all involved modules to achieve a feasible reservation schedule according to the defined reservation conditions (R1) to (R6). At any time during reservation negotiations, there are no more than two actively negotiating modules (cf. Figures 4.13 and 4.15). All other involved modules either already received a confirmation or are awaiting a response to their request to continue negotiations. Therefore, we show that each pair of adjacent modules always reaches an agreement on the logical time of transferring the corresponding transport unit while observing reservation conditions (R1) to (R6).

Reservation requests to non-buffering modules on the active route may create livelock situations if the rejection loop from two negotiating modules (cf. Figure 4.13) is never interrupted. The next open time window the requested module suggests is later than the requested one. This avoids repeating identical reservation requests. Non-buffering modules only hold finite reservations within their local reservation tables. Therefore, the reservation request is feasible at the latest when the new reservation is scheduled last at both negotiating modules. The initiating module of the active route does not hold any reservation scheduled later than that of its allocated transport unit such that postponing reservations is always possible without contradicting reservation conditions (R1) and (R2) along the active route. Due to sequential route planning using the decentralized authorization concept, this also applies in the given $(m : n)$ setting.

Reservation requests to buffer modules induce passive route planning for relocating the interfering buffered transport unit before continuing reservation on the active route. This resolves the indefinite buffer reservation according to reservation condition (R6) such that it becomes a non-buffering module. Due to the defined inflow constraints (cf. Section 3.2.1.2), there is always at least one module in the system which is not a buffer module and thus accepts the indefinite relocation request (cf. Figure 4.15).

Livelocks occur if relocation requests are passed on indefinitely in a circular chain of buffer modules without arriving at a non-buffering module. For such a cycle of requests, no solution can be found, as negotiating a feasible solution requires $T_{out} < \infty$ to be satisfied for at least one buffer module of the cycle, which contradicts reservation condition (R6). Due to the defined path restrictions (P1) and (P2), cycles within a relocation route are generally excluded. Before sending a subsequent relocation request, an admissible adjacent module is initially verified (cf. Figure 4.15). Continuously updating the invalid modules of the relocation route ensures that (P1) and (P2) are always met. The graph of the conveyor network satisfies 2-connectivity (cf. Section 3.2.2) such that there is always a feasible relocation path to move an interfering buffered transport unit without collision to proceed on the active route. When negotiating passive routes, the non-buffering module sets the time at which it confirms the

indefinite buffer reservation. At the requesting module of the passive route, reservation condition (R6) holds. Therefore, it can postpone the scheduled outgoing transport of the relocated buffered transport unit such that the relocation request is accepted. This guarantees that the algorithmic operations for each relocation route always terminate at the initiating module of the interfering buffered transport unit.

5.2.2 Resource Allocation

When allocating the scheduled resources, a livelock refers to a transport unit circulating indefinitely within a cycle of conveyor modules without arriving at its output module. Processing a transport unit throughout the system consists of active and passive routes. Combined, they form the overall route of the transport unit from its input to its output module. Resource allocation follows the reserved logical time windows of each transport unit in ascending order. Thus, the sequence of occupied modules is clearly defined for each transport unit. As input and output modules are distinct, each transport unit is guaranteed to be unloaded at its output module in finite time, even if – due to relocations – the same modules may be reused along its overall route. This prevents livelocks when allocating resources for transferring transport units within the system.

5.3 Starvation

Starvation describes a situation where a process – not being deadlocked – is waiting indefinitely, as it is never allocated its requested resource (Ramesh 2010, p. 78). Thus, it never terminates. We subdivide the investigations concerning starvation prevention into algorithmic operations (cf. Section 5.3.1) and resource allocation (cf. Section 5.3.2).

5.3.1 Algorithmic Operations

Within the algorithmic operations, requesting modules await the response of the requested module. Starvation means that a requesting module waits indefinitely for a response which is never received. In Section 5.2.1, we demonstrate that route planning always terminates at the initiating module. This initially requires being granted the authorization (cf. Section 4.2.2.1). We show that any module initiating an active route is authorized to start route planning in finite time.

For this, we first need to ensure that there is always at least one module within the system in state authorized. As the authorization is only passed in case of incoming requests (cf. Section 4.2.2.1), we can guarantee that the authorization is never lost. Combined with the findings of Section 5.1.1, we can restrict the number of authorized modules to exactly one at any point in time.

If the authorized module changes, it notifies all relevant modules such that initiating modules of active routes address their authorization requests correctly. The number of simultaneously initiated active routes results from the buffer modules allocated a transport unit requested at its output module and the input modules occupied by a transport unit to be introduced into the system. These may compete to be granted the authorization for route planning.

Sequencing requires observing the predefined predecessor-successor dependencies when processing arriving transport units. An initiating module of an active route cannot be disadvantaged indefinitely when passing the authorization, as after a limited number of requests, there will be no other module awaiting to be authorized. Due to the defined inflow constraints, all input modules restrain a transport unit from being introduced which cannot be released yet (cf. Section 3.2.1.2). These will not generate an authorization request. Unloading buffered transport units requires all necessary predecessors to be available within the system. This limits the set of transport units requested at the output modules, as it depends on the inflow into the system. Each buffer module of a requested transport unit is therefore authorized for route planning in finite time.

Likewise, buffered transport units cannot become requested if their predecessors are not introduced. As the set of released transport units is limited and matched to the system size (cf. condition (I2)), the authorization is granted to each requesting input module in finite time. Thus, starvation is excluded within the algorithmic operations.

5.3.2 Resource Allocation

Starvation within resource allocation means that a transport unit is continuously denied from accessing a module on its overall route through the system. In Section 5.1, we show that each transport unit is processed in finite time such that its set of reservations represents a series of ascending logical time windows of definite length ending up at its output module. The order of transferring transport units at each module is given by their local reservation tables. If different transport units use the same modules on their routes through the system, a requested module may not immediately be available to receive the corresponding transport unit, as other transport units are scheduled there before. However, this waiting time is limited due to the reserved logical time windows. These unambiguously specify the order of transport units to be transferred for each module such that any process is able to use a requested resource in finite time. This excludes starvation within resource allocation.

5.4 Chapter Conclusion

In this chapter, we show that the presented sequencing system ensures system liveliness at any point in time by generally preventing deadlocks livelocks, and starvation. Overall, this responds to the third research question:

> **How can we demonstrate the liveliness of the decentralized sequencing system?**

We investigate the algorithmic operations as well as the allocation of resources for physically moving transport units, as these are processed at decentralized module level. Due to the specified system requirements, reservation conditions, and module interactions, we can guarantee that each arriving transport unit is processed for sequencing in finite time.

The defined inflow constraints specified by conditions (I1) to (I3) ensure route initiation for each transport unit up to its output module. Route planning is based on the decentralized authorization concept, which excludes mutually exclusive requests and guarantees consistent system information. For each initiated route, the planning authorization is granted in finite time.

Within route planning, a feasible solution is always found at decentralized module level such that the algorithmic operations terminate at the initiating module of an active route. This is given, as

- path selection using the decentralized A* algorithm excludes re-exploration,
- route reservation based on the defined reservation conditions (R1) to (R6) always results in an agreement between each pair of adjacent negotiating modules, and
- all induced relocation routes end up at a non-buffering module via an admissible path of stepwise passive routes due to path restrictions (P1) and (P2).

Thus, the algorithmic operations yield a finite overall process for each transport unit ending up at its output module. The set of reservations represents a series of ascending logical time windows of definite length even if the route of the transport unit is partially specified as composition of active and passive routes. At each module, the scheduled reservations create a consistent ordering for processing transport units. This ensures that any transport unit is able to enter the next module on its route in finite time to be unloaded at its assigned output module.

6 Complexity Analysis

Chapter 6 of this dissertation is based on Fleischmann and Furmans (2023) (Chapter V).

Due to localized information and decision making, decentralized systems involve increased communication and coordination efforts (Monostori et al. 2015). The presented sequencing algorithm is based on decentralized interactions of the conveyor modules in terms of exchanging messages. Algorithmic complexity usually relies on the required elementary computing operations an algorithm performs based on a given input size (Arora and Barak 2009, p. 13). As the presented algorithm is event-based such that sending and receiving messages drive the algorithmic operations, we rather evaluate the complexity of the decentralized sequencing algorithm using the number of messages required for sequencing. Section 6.1 introduces the general assumptions underlying this complexity analysis. Based on that, we derive the messaging effort of the decentralized sequencing algorithm (cf. Section 6.2). This yields its overall algorithmic order of complexity (cf. Section 6.3). By summarizing the results of this chapter, Section 6.4 provides an answer to the fourth research question of this dissertation.

6.1 General Assumptions

We denote by $\mathcal{C}$ the number of messages, i.e., the messaging effort, required to sequence a given set of transport units B using a conveyor network comprising the set of conveyor modules M, where M includes all sequencing, input, and

output modules of the network. The algorithmic operations for processing the set of transport units are triggered equally for each transport unit. Therefore, $\mathcal{C}$ linearly depends on $|B|$, i.e.,

$$\mathcal{C} = |B| \cdot \mathcal{C}_b, \tag{6.1}$$

where $\mathcal{C}_b$ represents the average number of messages incurred per transport unit. This allows analyzing the messaging effort at the level of a single transport unit.

Routing a transport unit from its input to its output module is organized via active and passive routes, where passive routes arise if this transport unit interferes with *other* active routes. We consider processing each transport unit as the composition of its active routes together with its *induced* passive routes. This will not change the overall result, as each transport unit just incorporates the relocations of its interfering buffered transport units rather than its own ones. These are considered at the transport units where this transport unit interferes.

We denote by $\mathcal{C}_a$ the average number of messages per active route, including its passive routes to relocate interfering buffered transport units. For each transport unit, no more than two active routes are required. This gives

$$\mathcal{C}_b \leq 2 \cdot \mathcal{C}_a \tag{6.2}$$

and thus,

$$\mathcal{C} \leq 2 \cdot |B| \cdot \mathcal{C}_a. \tag{6.3}$$

To determine the order of complexity the presented decentralized sequencing algorithm can be classified to, we need to specify an upper bound of the required messaging effort. Thus, we can ensure that any sequencing problem is covered. Let $\hat{\mathcal{C}}$ be the maximum number of messages required for a given sequencing problem and $\hat{\mathcal{C}}_a$ the maximum number of messages required per active route, then based on equation (6.3), it follows

$$\hat{\mathcal{C}} \leq 2 \cdot |B| \cdot \hat{\mathcal{C}}_a. \tag{6.4}$$

Each active route includes messages for initiation, planning, and execution (cf. Section 4.2). The total number of messages per active route results from the sum of messages required for these partial components. We denote by $\mathcal{C}_a^I$, $\mathcal{C}_a^P$, and $\mathcal{C}_a^E$ the average messaging effort required per active route for initiation, planning, and execution, respectively, while $\hat{\mathcal{C}}_a^I$, $\hat{\mathcal{C}}_a^P$, and $\hat{\mathcal{C}}_a^E$ refer to their upper bounds. Generally,

$$\mathcal{C}_a = \mathcal{C}_a^I + \mathcal{C}_a^P + \mathcal{C}_a^E \tag{6.5}$$

and thus,

$$\hat{\mathcal{C}}_a = \hat{\mathcal{C}}_a^I + \hat{\mathcal{C}}_a^P + \hat{\mathcal{C}}_a^E \tag{6.6}$$

applies.

6.2 Messaging Effort

Based on the given conveyor network comprising $|M|$ conveyor modules, we are able to quantify the messaging effort for initiating, planning, and executing an active route. This allows estimating the upper bounds $\hat{\mathcal{C}}_a^I$, $\hat{\mathcal{C}}_a^P$, and $\hat{\mathcal{C}}_a^E$, which we present in Sections 6.2.1, 6.2.2 and 6.2.3, respectively. We use the factor ξ to represent linear dependencies resulting from responses to requests or forwarding messages, but do not affect the order of algorithmic complexity. For all ξ^X introduced in the following $1 \leq \xi^X << |M|$ holds, where index X represents an algorithmic component.

6.2.1 Route Initiation

Initiating an active route is triggered when the output module notifies the module currently allocated to a transport unit (cf. Figures 4.6 and 4.8). In case of relocations, the allocated modules of buffered transport unit change. This requires updating the respective output module for each relocation (cf. Figures 4.15 and 4.14). Each active route incorporates less than $|M|$ interfering

buffered transport units, each of which is moved on a relocated route comprising less than $|M|$ passive routes. An upper bound for the messaging effort $\hat{\mathcal{C}}_a^U$ of each active route to update the corresponding output modules due to relocations is therefore given by

$$\hat{\mathcal{C}}_a^U \leq \xi^U \cdot |M|^2. \tag{6.7}$$

Additionally, identifying non-buffering modules requires requesting potential buffer modules (cf. Figure 4.7). This messaging effort arises for the active route if the corresponding transport unit is not yet requested at its output module (cf. Figure 4.6), as well as for each of its induced relocation routes (cf. Figure 4.15). As the length of the active route is limited by $|M|$, no more than $|M|$ non-buffering modules can be used in total. These are identified using no more than $|M|$ requests to potential buffer modules. This yields an upper bound for the messaging effort $\hat{\mathcal{C}}_a^B$ of each active route to identify non-buffering modules as

$$\hat{\mathcal{C}}_a^B \leq \xi^B \cdot |M|^2. \tag{6.8}$$

From equations (6.7) and (6.8), we obtain the overall messaging effort for route initiation as

$$\begin{aligned} & \hat{\mathcal{C}}_a^I \leq \xi^U \cdot |M|^2 + \xi^B \cdot |M|^2 \\ \Leftrightarrow\ & \hat{\mathcal{C}}_a^I \leq \xi^I \cdot |M|^2. \end{aligned} \tag{6.9}$$

6.2.2 Route Planning

Planning an active route starts with acquiring the authorization for route planning (cf. Figure 4.9). As the number of simultaneously initiated active routes is limited by the defined inflow constraints and the set of transport units requested at the output modules, the number of requests to the currently authorized module until being authorized is definitely less than $|M|$. When passing the authorization on to the module initiating the corresponding active route, all other modules, an active route may originate from, are notified. Thus, we derive an

upper bound for the messaging effort $\hat{\mathcal{C}}_a^A$ of each active route to be granted the authorization for route planning as

$$\hat{\mathcal{C}}_a^A \leq \xi^A \cdot |M|^2. \tag{6.10}$$

Messaging effort for path selection of an active route (cf. Figure 4.10) is driven by the size of the conveyor network, as this determines the number of modules to be explored. For each active route, the decentralized A* algorithm takes no more than $|M|$ search iterations to reach the destination module. This gives an upper bound for the messaging effort $\hat{\mathcal{C}}_a^S$ of each active route to select the path of modules as

$$\hat{\mathcal{C}}_a^S \leq \xi^S \cdot |M|. \tag{6.11}$$

Route reservation comprises scheduling the transfers of an active route along the selected path of modules as well as of all induced passive routes for relocating interfering buffered transport units. We subdivide the investigations into the messaging effort for reserving the modules of the active route and that for reserving all necessary relocation routes. Thus, we first consider only the negotiations between non-buffering modules of the active route. Transforming interfering buffer modules to non-buffering modules before being able to process an incoming reservation request at a buffer module is investigated within passive route planning.

Negotiations for route reservation on the active route are based on the existing reservations at the selected modules. In case of conflicting reservations, incoming reservation requests are rejected while suggesting an alternative logical time window (cf. Figure 4.13). The number of rejected reservation requests is limited by the number of reservations within the local reservation table at each module. At any time, no more than $|M|$ transport units are scheduled within the system simultaneously due to the necessary capacity requirements (cf. Section 3.2.1). The maximum number of reservations arises at a module assuming that it is included in each relocation route. For each transport unit scheduled within the system, no more than $|M|$ relocation reservations can be generated at the same module. Therefore, a module can never hold more than

$|M|^2$ reservations in its reservation table simultaneously. Each active route incorporates less than $|M|$ negotiating adjacent modules. With this, we obtain

$$\hat{\mathcal{C}}_a^{R_a} \leq \xi^{R_a} \cdot |M|^3 \tag{6.12}$$

as an upper bound for the messaging effort $\hat{\mathcal{C}}_a^{R_a}$ of each active route to negotiate feasible reservations along its selected path of modules.

On relocation routes induced by an active route, each two adjacent modules of a passive route agree on the time at which the currently non-buffering module can accept the transport unit of the buffer module ahead of it (cf. Figure 4.15). Due to reservation conditions (R3) and (R6), the buffer reservation needs to be scheduled at the non-buffering module after the last of its existing finite reservations. The latter specifies the feasible time such that the messaging effort to negotiate a single passive route can be considered constant. Each active route incorporates less than $|M|$ interfering buffered transport units, each of which is moved on a relocated route comprising less than $|M|$ passive routes. Thus, an upper bound for the messaging effort $\hat{\mathcal{C}}_a^{R_r}$ of each active route to plan all required relocation routes is given by

$$\hat{\mathcal{C}}_a^{R_r} \leq \xi^{R_r} \cdot |M|^2. \tag{6.13}$$

From equations (6.10), (6.11), (6.12) and (6.13), the overall messaging effort of route planning follows as

$$\begin{aligned} & \hat{\mathcal{C}}_a^{P} \leq \xi^{A} \cdot |M|^2 + \xi^{S} \cdot |M| + \xi^{R_a} \cdot |M|^3 + \xi^{R_r} \cdot |M|^2 \\ \Leftrightarrow\ & \hat{\mathcal{C}}_a^{P} \leq \xi^{P} \cdot |M|^3. \end{aligned} \tag{6.14}$$

6.2.3 Transport Execution

For transport execution, each two adjacent modules agree on the physical time for transferring a scheduled transport unit. Transfers are only requested if the currently occupied module is ready to forward the corresponding transport

unit and confirmed if the adjacent next receiving module is ready to take it (cf. Figure 4.18). Thus, the messaging effort for transferring a transport unit between two adjacent modules can be considered constant. For each active route, no more than $|M|^2$ reservations are scheduled, as its length as well as the length of its induced relocation routes is limited by $|M|$. This yields an upper bound for the messaging effort $\hat{\mathcal{C}}_a^E$ of each active route for transport execution as

$$\hat{\mathcal{C}}_a^E \leq \xi^E \cdot |M|^2. \tag{6.15}$$

6.3 Order of Complexity

Based on equations (6.9), (6.14), and (6.15), an upper bound of the total messaging effort per active route follows as

$$\begin{aligned} & \hat{\mathcal{C}}_a \leq \xi^I \cdot |M|^2 + \xi^P \cdot |M|^3 + \xi^E \cdot |M| \\ \Leftrightarrow\ & \hat{\mathcal{C}}_a \leq \xi \cdot |M|^3. \end{aligned} \tag{6.16}$$

Combined with equation (6.4), we obtain an upper bound of the messaging effort for a given sequencing problem:

$$\hat{\mathcal{C}} \leq 2 \cdot \xi \cdot |B| \cdot |M|^3. \tag{6.17}$$

As the message sender and recipient are not necessarily directly connected, the message is transmitted via a shortest path of adjacent modules (cf. Section 3.3.1.4). The length of this path is limited by the maximum shortest distance of two modules within the conveyor network and can therefore be included in the factor ξ.

This implies an overall complexity of

$$\mathcal{O}^n = \mathcal{O}(n^3) \tag{6.18}$$

at network level scaling with system size, i.e. the number of conveyor modules n within the network.

Equation (6.17) provides an upper bound of the messaging effort arising within the system, i.e., at $|M|$ conveyor modules. As we always assume $|M|$ conveyor modules involved in each partial component of the algorithm when determining $\hat{\mathcal{C}}_a$, we can derive an upper bound of the messaging effort at module level as well. Due to the decentralized system setup, there are $|M|$ control units for message processing within the system – one installed at each module (cf. Section 3.3.1.1). Therefore, the messaging effort per module $\mathcal{C}_m$ decreases to

$$\mathcal{C}_m \leq \xi \cdot |B| \cdot |M|^2. \tag{6.19}$$

This corresponds to a complexity of

$$\mathcal{O}^m = \mathcal{O}(n^2) \tag{6.20}$$

at module level scaling with system size, i.e. the number of conveyor modules n within the network.

6.4 Chapter Conclusion

In this chapter, we analyze the complexity of the decentralized sequencing algorithm in terms of the number of exchanged messages for sequencing a set of transport units using an installed conveyor network. Overall, this responds to the fourth research question:

> **How can we evaluate the complexity of the decentralized sequencing algorithm?**

The order of complexity follows from the upper bound of the required messaging effort. It increases linearly with the number of transport units. Routing a transport unit from its input to its output module comprises initiating, planning, and executing active routes including the induced passive routes. The total

messaging effort per active route results from the sum of messages required for these partial components. Overall, this gives a complexity of order $\mathcal{O}(n^3)$ at network level which reduces to $\mathcal{O}(n^2)$ at module level, both scaling with the number of modules n.

7 Quality Assessment

In decentralized systems, the dynamic interactions of several autonomous decisional entities aggravate analyzing efficiency and providing insightful metrics to derive a general performance evaluation (Trentesaux 2009). Moreover, there is no generally applicable centralized benchmark algorithm for the investigated sequencing problem of this dissertation (cf. Section 2.1). In this chapter, we develop centralized concepts providing solutions to the problem of sequencing in highest-density conveyor networks. This allows assessing the quality of solutions obtained by the decentralized sequencing algorithm. In Section 7.1, we present approaches for determining optimal solutions to given sequencing problems based on mixed-integer programming. Due to NP-hardness, only small problem sizes are solvable given the currently available computing power. To compensate for bigger and more realistic systems, we additionally develop a lower bound approach to analytically approximate the minimum sequencing time (cf. Section 7.2). The deficit of solutions generated by the decentralized sequencing algorithm is investigated in Section 7.3. By summarizing the results of this chapter, Section 7.4 provides an answer to the fifth research question of this dissertation.

7.1 Optimal Approaches

To benchmark solutions of the decentralized sequencing algorithm against optimal solutions, we develop two optimization models. The single stage optimization model is able to provide globally optimal solutions but suffers from extensive computation time. The iterative segmented optimization model intends

to solve larger problem settings than the single stage one, while guaranteeing partial optimality. Section 7.1.1 introduces the preliminary assumptions of both models. Based on that, we present the single stage and the iterative segmented optimization model in Sections 7.1.2 and 7.1.3, respectively. Section 7.1.4 details the performance indicators used to quantify the deficit of decentralized solutions.

7.1.1 Preliminary Assumptions

Table 7.1 outlines the notation for formally describing the single stage as well as the iterative segmented optimization model. Sets, elements and parameters are specified by the given problem setting, while the values of decision variables are determined by solving the corresponding optimization model. They define the optimal result of a sequencing problem.

Table 7.1: Notation of the single stage optimization model.

Symbol	Description
Sets	
I	Set of input modules ($I \subseteq M$)
O	Set of output modules ($O \subseteq M$)
C	Set of sequencing modules ($C \subseteq M$)
M	Set of conveyor modules ($I \cup O \cup C = M$; $I \cap O \cap C = \emptyset$)
N_m	Set of adjacent modules directly connected to module m ($N_m \subseteq M$)
B	Set of transport units
B_i	Set of transport units arriving at input module i ($B_i \subseteq B$)
B_o	Set of transport units unloaded at output module o ($B_o \subseteq B$)

Continued on next page

Table 7.1 – *Continued from previous page*

Symbol	Description
P_o^b	Set of direct predecessors of transport unit b within the unloading sequence at output module o ($P_o^b \subseteq B_o$)[1]
T	Set of time points defining the time frame of the sequencing problem ($T \subseteq \mathbb{R}_0^+ : n \cdot \Delta t,\ n \in \mathbb{N}_0$)
Elements	
i^b	Input module transport unit b is assigned to
o^b	Output module transport unit b is assigned to
f_i^b	Direct successor of transport unit b within the queue of arriving transport units at input module i ($f_i^b, b \in B_i$)
Parameters	
a^b	Arrival time of transport unit b at the system
$d^*_{m\acute{m}}$	Length of a shortest path of modules from module m to module $\acute{m}$
t_{conv}	Transfer time of a transport unit between two adjacent modules
Δt	Incremental length of successive time points
L	Large positive number
Decision variables	
x_m^{bt}	1 if transport unit b occupies module m at time t
	0 otherwise
$y_{m\acute{m}}^{bt}$	1 if transport unit b moves from module m to an adjacent module $\acute{m}$ at time t
	0 otherwise

[1] In case of partially defined unloading sequences (cf. Section 3.3.1.2), the set P_o^b may comprise more than one transport unit, otherwise it refers to the single directly preceding transport unit within the unloading sequence.

In the decentralized sequencing system, the system arrangement may change continuously over time, as modules are independent and operate autonomously. However, formulating the optimization models using the time-based decision variables x_m^{bt} and $y_{m\acute{m}}^{bt}$ requires a discrete time frame T. In the presented optimization models, we discretize the continuous time frame of the real system into time increments of length $\Delta t = \frac{t_{conv}}{2}$.

At any point in time, the positions of all transport units define the current system arrangement, which changes as transport units are moved within the system. Discretizing the continuous time frame into time segments of length $\Delta t = \frac{t_{conv}}{2}$ matches the physical movements of transport units. Transferring a transport unit currently occupying module m to an adjacent module $\acute{m}$ requires a period of t_{conv}. While transitioning, the transport unit occupies both modules. This is modeled using the specified decision variables defining whether a transport unit is currently moved – $y_{m\acute{m}}^{bt}$ – or stationary positioned – x_m^{bt} (cf. Figure 7.1). Assuming that transport units move at constant speed between modules, from the time when the transport unit is positioned on module m until it occupies both of these modules equally, a time period of $\frac{t_{conv}}{2}$ elapses. Similarly, it takes $\frac{t_{conv}}{2}$ until the transport unit completely arrives at the adjacent module $\acute{m}$. This yields the defined time frame $T \subseteq \mathbb{R}_0^+ : n \cdot \Delta t,\ n \in \mathbb{N}_0$ (cf. Table 7.1).

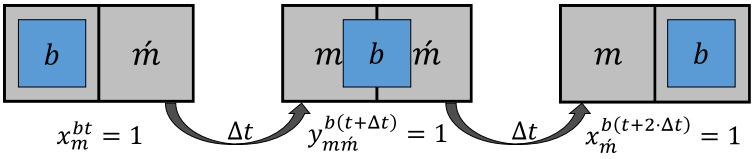


Figure 7.1: Moving transport unit b between adjacent modules m and $\acute{m}$.

The time frame T needs to be predefined when solving the optimization model for a given problem setting. If it is set too small, there will be no solution where all transport units are sequenced to their assigned output modules. We determine the time frame T based on the solution of the decentralized sequencing

algorithm. Let H_{dec} be the physical time required for sequencing the given set of transport units within the decentralized sequencing system, then

$$|T| \leq \left\lceil \frac{H_{dec}}{\Delta t} \right\rceil = \left\lceil \frac{2 \cdot H_{dec}}{t_{conv}} \right\rceil \tag{7.1}$$

holds, as H_{dec} represents an upper bound of the optimal result.

7.1.2 Single Stage Optimization Model

The multi-criteria weighted objective function of the single stage optimization model (7.2) minimizes the sequencing time of the given set of transport units using the least possible number of movements:

$$\min\ W_{ss} = -\,|T| \cdot \sum_{t \in T} \sum_{b \in B} x_{o^b}^{bt} + \sum_{t \in T} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt}. \tag{7.2}$$

Minimizing the sequencing time is equal to maximizing the time the transport units spend at their assigned output module. The specified formulation focuses on the minimum sequencing time of all transport units keeping movement minimization as secondary objective. In Appendix A.1, we attach the underlying calculations proving this.

We minimize W_{ss} subject to constraints (7.3) to (7.13) representing the requirements of the given sequencing problem. For reasons of clarity and comprehensibility, we structured them according to their field of impact.

Introducing and Unloading

Constraints (7.3) initially assign all transport units to their respective input modules. They are introduced according to their arrival sequences (cf. constraints (7.4)) and arrival times (cf. constraints (7.5)). Constraints (7.6) ensure

that unloading observes the predefined sequences at the assigned output modules.

$$x_{i^b}^{b0} \geq 1 \qquad \forall\, b \in B \tag{7.3}$$

$$x_{i^b}^{bt} \leq x_{i^b}^{f_{i^b}^b t} \qquad \forall\, t \in T, b \in B \tag{7.4}$$

$$t \cdot y_{i^b m}^{bt} \geq a^b + \Delta t - \left(1 - y_{i^b m}^{bt}\right) \cdot L \qquad \forall\, t \in T, b \in B, m \in N_{i^b} \tag{7.5}$$

$$y_{m o^b}^{bt} \leq x_{o^b}^{pt} \qquad \forall\, t \in T, b \in B, p \in P_{o^b}^b, m \in N_{o^b} \tag{7.6}$$

Capacity Requirements

Transport units and modules need to be assigned such that capacity requirements are met. At any time, each transport unit occupies a defined position within the system (cf. constraints (7.7)) while observing the capacity of the sequencing modules (cf. constraints (7.8)).

$$\sum_{m \in M} x_m^{bt} + \sum_{m \in M} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt} = 1 \qquad \forall\, t \in T, b \in B \tag{7.7}$$

$$\sum_{b \in B} x_m^{bt} + \sum_{b \in B} \sum_{\acute{m} \in N_m} \left(y_{m\acute{m}}^{bt} + y_{\acute{m}m}^{bt}\right) \leq 1 \qquad \forall\, t \in T, m \in C \tag{7.8}$$

Transport Unit Movements

Movements of transport units are restricted to directly adjacent modules. The system arrangement at time $t + \Delta t$ results from that at time t (cf. constraints (7.9) and (7.10)). To avoid collisions, entering a module is not possible

until the movement of a preceding transport unit is finished entirely (cf. constraints (7.11)).

$$x_m^{bt} \leq x_m^{b(t+\Delta t)} + \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{b(t+\Delta t)} \tag{7.9}$$
$$\forall\, t \in \{0, \ldots, (|T| - 2) \cdot \Delta t\}, b \in B, m \in M$$

$$y_{m\acute{m}}^{bt} \leq x_{\acute{m}}^{b(t+\Delta t)} \tag{7.10}$$
$$\forall\, t \in \{0, \ldots, (|T| - 2) \cdot \Delta t\}, b \in B, m \in M, \acute{m} \in N_m$$

$$1 - \sum_{b \in B} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt} \geq \sum_{b \in B} \sum_{\acute{m} \in N_m} y_{\acute{m}m}^{b(t+\Delta t)} \tag{7.11}$$
$$\forall\, t \in \{0, \ldots, (|T| - 2) \cdot \Delta t\}, m \in C$$

Value Range of Decision Variables

Constraints (7.12) and (7.13) restrict the binary decision variables of the optimization model.

$$x_m^{bt} \in \{0; 1\} \;\; \forall\, t \in T, b \in B, m \in C \cup i^b \cup o^b \tag{7.12}$$

$$y_{m\acute{m}}^{bt} \in \{0; 1\} \;\; \forall\, t \in T, b \in B, m \in \left(C \cup i^b\right), \acute{m} \in \left(N_m \cap \left(C \cup o^b\right)\right) \tag{7.13}$$

7.1.3 Iterative Segmented Optimization Model

In contrast to the single stage optimization model, the iterative segmented optimization model creates smaller sub-problems (cf. Section 7.1.3.1). This requires adjusting the optimization objective (cf. Section 7.1.3.2). In Section 7.1.3.3, we formally describe the iterative segmented optimization model based on the formal description of the single stage optimization model.

7.1.3.1 Creating Sub-problems

The problem size of an optimization model which can be solved in reasonable time is highly dependent on its number of non-trivial decision variables. Regarding the defined decision variables of the formulated sequencing problem – x_m^{bt} and $y_{m\acute{m}}^{bt}$ – it results from three dimensions:

- the number of transport units $b \in B$,
- the number of modules $m \in M$ and their connections within the conveyor network,
- the number of time points $t \in T$, i.e. the amount of time it takes until the last transport unit of set B is positioned at its assigned output module o^b.

Figure 7.2 shows how the set of time points influences the number of decision variables for exemplary problem settings.

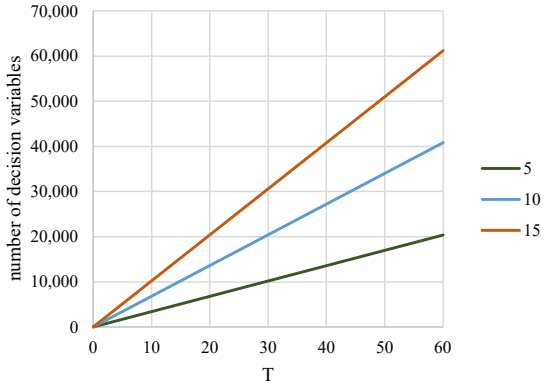
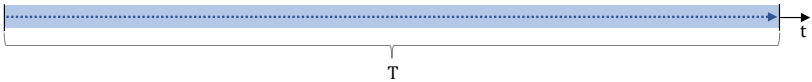
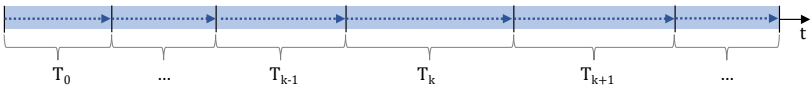


Figure 7.2: Number of decision variables depending on time frame T for single batch problems of size 5, 10, and 15 and a square 4×4 arrangement of sequencing modules (cf. Figure 7.10).

The iterative segmented optimization model reduces the problem size regarding the time frame T. For each time point $t > 0$, the respective system arrangement follows from the system arrangement at time $t - \Delta t$, as a transport unit either keeps its position at the current module or is moved to an adjacent module (cf. constraints (7.9) and (7.10)). Splitting the total time frame T into subsets T_k with $T_k \subset T$ results in several smaller sub-problems which are formulated and solved iteratively. The system arrangement at the end of an iteration k is fixed and used as initial arrangement for the optimization problem of iteration $k + 1$. This iterative procedure terminates when the system reaches the final arrangement, where all transport units are positioned at their assigned output modules. Figure 7.3 opposes the solution generation of the single stage and the iterative segmented optimization model.



(a) Single stage optimization.



(b) Iterative segmented optimization.

Figure 7.3: Comparison of the developed optimization models.

Based on equation (7.1), we derive the time frame T_k of each single iteration from the decentralized sequencing time H_{dec} using the *time split factor* l as

$$T_k = \left\lceil \frac{|T|}{l} \right\rceil = \left\lceil \frac{1}{l} \cdot \left\lceil \frac{2 \cdot H_{dec}}{t_{conv}} \right\rceil \right\rceil, \tag{7.14}$$

where $l \in \mathbb{N}$.

As the given time frame of each sub-problem is limited, the system proceeds from the initial arrangement to a certain terminal arrangement following the

defined objective of the iterative segmented optimization (cf. Section 7.1.3.2). The movements of the transport units necessary for this transformation may consume less time than provided by the time frame T_k. In this case, idle time within the solution of this iteration is cut off accordingly for continuing the optimization in the subsequent iteration. Therefore, the overall solution of a sequencing problem, which results from combining all iteratively calculated partial solutions, may comprise solutions of different time frame lengths (cf. Figure 7.3 (b)).

7.1.3.2 Objective of a Single Iteration

For the single stage optimization model, we formulate the objective of minimum sequencing time using the final arrangement of the sequencing problem where all transport units are positioned at their assigned output modules (cf. equation (7.2)). Reaching this final arrangement requires a sufficient time frame T, which is not necessarily given within the iterative segmented optimization due to the decomposition into sub-problems with $T_k \subset T$. Thus, we define the objective of the iterative segmented optimization model such that the system proceeds in sequencing as far as possible in each iteration. Each sub-problem strives for the best pre-sequenced arrangement in the given time frame T_k. Pre-sequencing supports fast unloading in the subsequent iterations to enable an efficient overall solution of the given problem setting.

A pre-sequenced arrangement intends to prevent successors within the unloading sequence from interfering with their predecessors when accessing the assigned output module. We evaluate the position of each transport unit by comparing its distance to the assigned output module to that of its direct predecessors and penalize arrangements where a predecessor occupies a sequencing module more distant to the output module than its successor. This enables lower ranked transport units to be unloaded primarily to proceed in sequencing.

We denote by z^{bt} the pre-sequenced arrangement of transport unit b at time t in relation to its direct predecessors $p \in P^b_{o^b}$ as

$$z^{bt} = \max \left\{ \left(\sum_{m \in M} d^*_{mo^b} \cdot x^{bt}_m + \sum_{m \in M} \sum_{\acute{m} \in N_m} \frac{d^*_{mo^b} + d^*_{\acute{m}o^b}}{2} \cdot y^{bt}_{m\acute{m}} \right), \right. \tag{7.15}$$

$$\left. z^{pt} - x^{pt}_{ip} \cdot L \right\} \qquad \forall\, t \in T_k, b \in B, p \in P^b_{o^b}$$

The pre-sequenced arrangement of the overall system at a given time results from combining the arrangements of all transport units at this time. Minimizing yields the best pre-sequenced system arrangement according to this definition.

Generally, an overall solution of the iterative segmented optimization model consists of local optima resulting from the solutions of the single sub-problems. The objective of pre-sequencing cannot guarantee global optimality, as future movements, which may be necessary in later iterations, are not anticipated when solving a sub-problem. Nevertheless, the presented iterative segmented optimization enables reducing the size of a given sequencing problem such that larger problem settings can be solved compared to the single stage optimization (cf. Section 7.3.2).

7.1.3.3 Formal Description

The iterative segmented optimization model uses the same notation as the single stage optimization model, but requires extensions to model the optimization using sub-problems (cf. Table 7.2).

In each iteration, the multi-criteria weighted objective function of the iterative segmented optimization model (7.16) minimizes the pre-sequenced terminal

Table 7.2: Additional notation of the iterative segmented optimization model.

Symbol	Description
Sets	
K	Set of iterations, i.e. number of sub-problems ($K \subseteq \mathbb{N}_0$)
T_k	Set of time points in iteration k ($T_k \subseteq T,\ \lvert T_k \rvert \geq 2$)
X_0^k	Set of non-zero x-variables defining the system arrangement iteration k starts from ($k > 0$)
Y_0^k	Set of non-zero y-variables defining the system arrangement iteration k starts from ($k > 0$)
Elements	
t_0^k	Initial time point of iteration k
Decision variables	
z^{bt}	Pre-sequenced arrangement of transport unit b at time t

system arrangement within the given time frame using the least possible number of movements:

$$\min\ W_{it} = |T_k|^2 \cdot \sum_{b \in B} \left(z^{b\left(t_0^k + (|T_k|-1)\cdot \Delta t\right)} + \Delta t \right) + \sum_{t \in T_k} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} \left(t - t_0^k\right) \cdot y_{m\acute{m}}^{bt}. \tag{7.16}$$

Deliberately, we only include the terminal arrangement at time point $t_0^k + (|T_k| - 1) \cdot \Delta t$ to accept intermediately poorer system arrangements if they yield an improved solution at the end of the given time frame. As the movements of transport units from their initial arrangement to the optimal pre-sequenced terminal arrangement of an iteration may consume less time than the given time frame T_k, movement minimization is time-dependent by including the weighting factor $\left(t - t_0^k\right)$. The specified formulation of the objective function (7.16) focuses on pre-sequencing keeping movement minimization as secondary objective. The underlying calculations proving this can be found in Appendix A.2.

Minimizing W_{it} within the iterative segmented optimization model requires the additional or adapted constraints (7.17) to (7.22) compared to the single stage optimization model.

Pre-sequenced Arrangement

Constraints (7.17) and (7.18) define the pre-sequenced arrangement according to equation (7.15).

$$z^{bt} \geq \sum_{m \in M} d^*_{mo^b} \cdot x^{bt}_m + \sum_{m \in M} \sum_{\acute{m} \in N_m} \frac{d^*_{mo^b} + d^*_{\acute{m}o^b}}{2} \cdot y^{bt}_{m\acute{m}} \qquad (7.17)$$

$$\forall\, t \in T_k, b \in B$$

$$z^{bt} \geq z^{pt} - x^{pt}_{ip} \cdot L \qquad \forall\, t \in T_k, b \in B, p \in P^b_o \quad (7.18)$$

Initial Arrangement

The initial arrangement of an iteration results from the positions of all transport units within the terminal arrangement of the previous iteration (cf. constraints (7.19) and (7.20)).

$$x^{bt^k_0}_m \geq 1 \qquad \forall\, b \in B, m \in M : x^{bt^k_0}_m \in X^k_0 \qquad (7.19)$$

$$y^{bt^k_0}_{m\acute{m}} \geq 1 \qquad \forall\, b \in B, m \in M, \acute{m} \in N_m : y^{bt^k_0}_{m\acute{m}} \in Y^k_0 \quad (7.20)$$

Capacity Requirement

Achieving a partial solution from which a feasible solution can be calculated in the subsequent iteration necessarily requires at least one unoccupied sequencing module (cf. constraints (7.21)).

$$\sum_{b \in B} \sum_{m \in C} x^{bt}_m + \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} y^{bt}_{m\acute{m}} \leq |C| - 1 \qquad \forall\, t \in T_k \quad (7.21)$$

Time Frame Adaptions

The time frame needs to be adapted to the given sub-problem as

- $\forall\, t \in T_k$ for constraints (7.4) to (7.8), (7.12), (7.13) and
- $\forall\, t \in \{t_0^k, \ldots, t_0^k + (|T_k| - 2) \cdot \Delta t\}$ for constraints (7.9) to (7.11).

All other introduced specifications remain unchanged.

Value Range of Decision Variables

The additional decision variables to specify the pre-sequenced arrangement are positive continuous (cf. constraints (7.22)).

$$z^{bt} \in \mathbb{R}_0^+ \qquad \forall\, t \in T_k, b \in B \quad (7.22)$$

7.1.4 Performance Indicators

We apply the sequencing time of a given problem setting as primary performance indicator while the number of movements is included supplementarily. We denote these by H and F, respectively. The sequencing time represents the time required until unloading all transport units of a given sequencing problem at their assigned output modules. It depends on the time of unloading the highest-ranked transport unit of the last batch.

We derive the decentralized sequencing time H_{dec} from the simulation runtime of the given sequencing problem. As the decentralized sequencing algorithm incorporates stochastic decisions (cf. Sections 4.2.2.2 and 4.2.2.4), we conduct several replications within a simulation experiment to obtain reliable results. The number of replications is based on a 95 %-confidence interval related to the mean sequencing time of the single simulation runs for a given parameter setting as well as the specifications given in Appendix B.3.

For the optimal result, we denote by H^* the optimal sequencing time of the overall problem, while h^{b^*} refers to the optimal sequencing time of transport unit b, i.e., the unloading time of transport unit b within the optimal solution. With this, we obtain

$$H^* = \max_{b \in B} h^{b^*}. \tag{7.23}$$

Based on a solution of the single stage optimization model or a composite solution of all iterations of the iterative segmented optimization model, we can derive h^{b^*} for all transport units from the values of their respective decision variables. It corresponds to the minimum time at which a transport unit is positioned at its assigned output module, i.e.,

$$h^{b^*} = \min_{t \in T} t : x_{o^b}^{bt} = 1. \tag{7.24}$$

Thus,

$$H^* = \max_{b \in B} \left\{ \min_{t \in T} t : x_{o^b}^{bt} = 1 \right\}. \tag{7.25}$$

Due to discretizing the time frame T, the solutions of the optimization models correspond to a synchronized system with time steps of incremental length $\Delta t = \frac{t_{conv}}{2}$ (cf. Section 7.1.1). Within optimal solutions, changes in the system arrangement can only be captured at integer multiples of Δt. The conveyor modules of the decentralized sequencing system, however, operate independently such that transport units can be transferred continuously. Therefore, the actual optimal value H^*_{real} may deviate from H^* within the interval of $\left(H^* - \frac{t_{conv}}{2}; H^*\right]$. H^* represents the least possible multiple of $\frac{t_{conv}}{2}$ which is greater than or equal to the actual value H^*_{real} of the decentralized system. This means

$$\frac{2 \cdot H^*}{t_{conv}} = \left\lceil \frac{2 \cdot H^*_{real}}{t_{conv}} \right\rceil. \tag{7.26}$$

The number of movements results from the necessary transfers of all transport units when being processed within the system. For a decentralized solution, we derive F_{dec} from the corresponding path lengths of the simulation experiment. The number of movements F^* resulting from the optimization models is given as

$$F^* = \sum_{t \in T} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt}. \tag{7.27}$$

Within the numerical evaluations of Chapter 7.3.2, we focus on presenting the sequencing time results while providing the number of movements in the attached research data (Fleischmann 2023).

7.2 Lower Bound Approach

The presented optimization models serve as a benchmark to evaluate the deficit of decentralized solutions for small problem settings. As problem size increases, they suffer from excessive computation times (cf. Figure 7.4). For bigger and more realistic sequencing problems, we develop an analytical lower bound of the minimum sequencing time. Comparing it with achievable optimal solutions allows assessing its suitability as a reference for real-world problem settings. Section 7.2.1 outlines the dependencies within the given set of transport units the lower bound calculation relies on. Section 7.2.2 provides the formal description of this approach.

7.2.1 Recursive Dependencies

The sequencing time of a given problem setting results from the sequencing times of all transport units (cf. Section 7.1.4). Specifically, the time of unloading the highest-ranked transport unit of the last batch is crucial (cf.

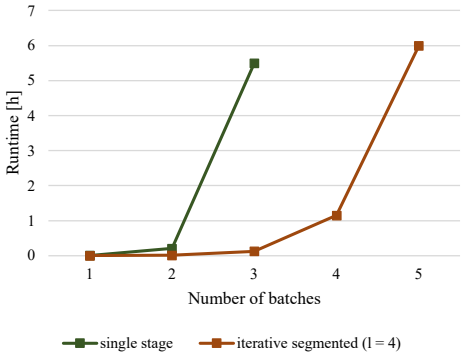


Figure 7.4: Overall runtime of optimization models for problem settings with batch size 5 and average-case arrival sequence using a square 4×4 arrangement of sequencing modules.

equation 7.23). Sequencing times of transport units depend on the predecessor-successor relations. If any predecessor is missing within the flow of arriving transport units, all of its succeeding transport units cannot be unloaded. This creates recursive dependencies, which we use to derive a simplified lower bound for the sequencing time of each transport unit. From this follows the sequencing time of the overall problem.

Each lowest ranked transport unit within the first batch assigned to an output module can always be unloaded directly upon its arrival, as it does not depend on any predecessor. In this case, sequencing times can be determined solely based on their arrival times and path lengths. Due to the predefined unloading sequences, this restricts the sequencing times of all succeeding transport units assigned to the same output module. Thus, we can deduce the sequencing times of all transport units by combining their arrival times, path lengths as well as the sequencing time of their corresponding direct predecessors within the unloading sequence at the assigned output module to obtain the overall sequencing time.

7.2.2 Formal Description

We derive a lower bound H_{LB} of the minimum sequencing time H^* for a given sequencing problem using the specified recursive dependencies (cf. Section 7.2.1). It results from the lower bound of the sequencing times h^b_{LB} of all transport units. These are at the minimum limited by the earliest possible unloading times u^b_{min}. The time required for unloading a transport unit at its assigned output module follows from (cf. Figure 7.5)

- its arrival time at the system,
- its waiting time until it is introduced into the system, and
- its processing time within the system until unloading.

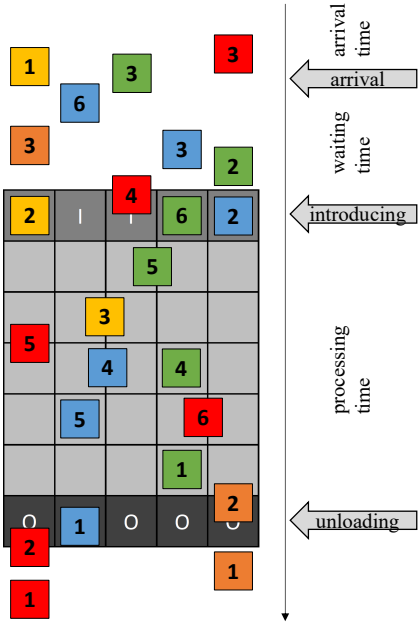


Figure 7.5: Time components until unloading a transport unit.

We denote by

- a^b the arrival time of transport unit b at the system,
- e^b_{min} the earliest possible time for introducing transport unit b into the system at its assigned input module i^b,
- t^b_{min} the minimum time required to transfer transport unit b from its assigned input module i^b to its assigned output module o^b,
- l_{i^b} the loading delay at input module i^b, and
- l_{o^b} the unloading delay at output module o^b.

The arrival times of all transport units are defined by the given sequencing problem.

The earliest possible time for introducing a transport unit results from the arrival characteristics of the set of all transport units assigned to the same input module including its loading delay. Let $Q^b_{i^b}$ be the set of predecessors of transport unit b within the queue of arriving transport units at input module i^b, then

$$e^b_{min} = \begin{cases} a^b & \text{if } Q^b_{i^b} = \emptyset \\ \max\left\{a^b,\ \max\limits_{q \in Q^b_{i^b}} \left\{e^q_{min}\right\} + l_{i^b}\right\} & \text{if } Q^b_{i^b} \neq \emptyset \end{cases} \tag{7.28}$$

with

$$l_{i^b} = \begin{cases} t_{conv} & \text{if } |N_{i^b} \cap C| > 1 \\ 2 \cdot t_{conv} & \text{if } |N_{i^b} \cap C| = 1 \\ \infty & \text{if } |N_{i^b} \cap C| = 0.^2 \end{cases} \tag{7.29}$$

[2] This case represents an invalid sequencing network (cf. Section 3.2.2), but is included here for reasons of completeness.

For all transport units arriving first at their assigned input module, $Q^b_{i^b} = \emptyset$ holds such that these may be introduced immediately. The introducing rate at each input module is limited by the physical transfer time t_{conv} between two adjacent modules (cf. Figure 7.6). Additionally, the connectivity of the input modules within the network is relevant, as introducing an arriving transport unit requires an empty sequencing module adjacent to the corresponding input module (cf. Figure 7.7). Since we do not restrict the number of conveyor modules adjacent to input or output modules within the conveyor networks under consideration, both constellations need to be included when defining the loading delay.

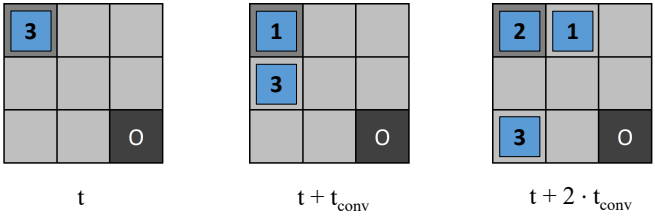


Figure 7.6: Loading delay of t_{conv} with more than one adjacent sequencing module – Introducing is possible at time t, $t + t_{conv}$, and $t + 2 \cdot t_{conv}$.

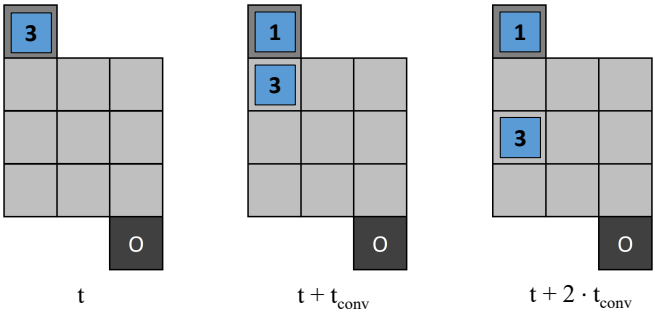


Figure 7.7: Loading delay of $2 \cdot t_{conv}$ with one adjacent sequencing module – Introducing is possible at time t and $t + 2 \cdot t_{conv}$.

The processing time of a transport unit results from its path through the conveyor network. At minimum, it corresponds to the shortest distance between its assigned input and output module. Therefore, we set

$$t^b_{min} = d^*_{i^b o^b} \cdot t_{conv}, \tag{7.30}$$

where $d^*_{i^b o^b}$ represents the minimum number of modules forming a path from input module i^b to output module o^b. This gives the earliest possible unloading time of transport unit b at the assigned output module:

$$\begin{aligned} u^b_{min} &= e^b_{min} + t^b_{min} \\ &= \max\left\{ a^b,\ \max_{q \in Q^b_{i^b}} \left\{ e^q_{min} \right\} + l_{i^b} \right\} + d^*_{i^b o^b} \cdot t_{conv}. \end{aligned} \tag{7.31}$$

Including the recursive dependencies (cf. Section 7.2.1), the lower bound of the sequencing time h^b_{LB} follows as

$$h^b_{LB} = \begin{cases} u^b_{min} & \text{if } P^b_{o^b} = \emptyset \\ \max\left\{ u^b_{min},\ \max_{p \in P^b_{o^b}} \left\{ h^p_{LB} \right\} + l_{o^b} \right\} & \text{if } P^b_{o^b} \neq \emptyset, \end{cases} \tag{7.32}$$

where

$$l_{o^b} = \begin{cases} t_{conv} & \text{if } |N_{o^b} \cap C| > 1 \\ 2 \cdot t_{conv} & \text{if } |N_{o^b} \cap C| = 1 \\ \infty & \text{if } |N_{o^b} \cap C| = 0.^3 \end{cases} \tag{7.33}$$

As with the loading delay, the physical transfer time of transport units induces an unloading delay depending on the connectivity of the corresponding output module within the network (cf. Figures 7.8 and 7.9).

[3] This case represents an invalid sequencing network (cf. Section 3.2.2), but is included here for reasons of completeness.

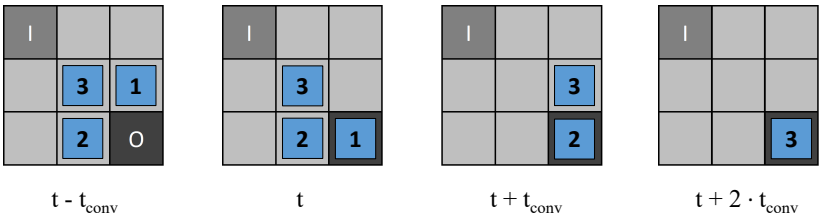


Figure 7.8: Unloading delay of t_{conv} with more than one adjacent sequencing module – Unloading is possible at time t, $t + t_{conv}$, and $t + 2 \cdot t_{conv}$.

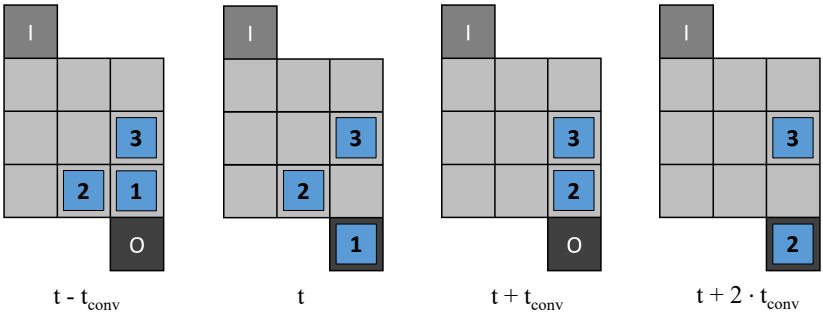


Figure 7.9: Unloading delay of $2 \cdot t_{conv}$ with one adjacent sequencing module – Unloading is possible at time t and $t + 2 \cdot t_{conv}$.

For partially defined unloading sequences, we can define an ordering $b^{eq^{(1)}}$, $b^{eq^{(2)}}$, $\ldots$, $b^{eq^{(n)}}$ within a set of equally ranked transport units based on their minimum unloading times at the assigned output module. This implies

$$u_{min}^{b^{eq^{(1)}}} \leq u_{min}^{b^{eq^{(2)}}} \leq \ldots \leq u_{min}^{b^{eq^{(n)}}} . \tag{7.34}$$

Combined with the set of direct predecessors $P^b_{o^b}$ within their unloading sequence at the output module, we obtain

$$\begin{aligned} h_{LB}^{b^{eq(1)}} &= \max\left\{ u_{min}^{b^{eq(1)}},\ \max_{p \in P^b_{o^b}} \left\{ h_{LB}^{p} \right\} + l_{o^b} \right\} \\ h_{LB}^{b^{eq(2)}} &= \max\left\{ u_{min}^{b^{eq(2)}},\ h_{LB}^{b^{eq(1)}} + l_{o^b} \right\} \\ &\vdots \\ h_{LB}^{b^{eq(n)}} &= \max\left\{ u_{min}^{b^{eq(n)}},\ h_{LB}^{b^{eq(n-1)}} + l_{o^b} \right\}. \end{aligned} \tag{7.35}$$

By analogy with equation (7.23), the lower bound of the overall sequencing problem follows as

$$H_{LB} = \max_{b \in B} h^b_{LB}, \tag{7.36}$$

where

$$H_{LB} \leq H^*. \tag{7.37}$$

This enables providing benchmark values for problem settings of arbitrary size to assess the quality of the obtained decentralized solutions (cf. Chapter 8).

7.3 Decentralized Solution Deficit

We assess the quality of the decentralized sequencing algorithm by comparing its solutions to the results of the presented centralized approaches. The evaluations are based on the design of experiments defined in Section 7.3.1. We present the numerical results in Section 7.3.2. These yield the implications of Section 7.3.3.

7.3.1 Design of Experiments

The design of experiments comprises the problem settings specified in Section 7.3.1.1. The obtained decentralized and optimal solutions result from the parameterization given in Section 7.3.1.2.

7.3.1.1 Problem Setting

We use a square 4×4 arrangement of sequencing modules to compromise between solvable problem sizes using the optimization models and available network capacity (cf. Section 3.2.1.1). This allows investigating sequencing problems with a batch size of up to 15 transport units per batch without restricting the arrival sequence within a single batch. For introducing and unloading, there are four input and four output modules at the top and bottom of the network, respectively (cf. Figure 7.10).

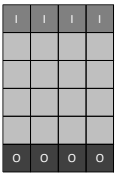


Figure 7.10: Reference network to compare centralized and decentralized solutions.

As part of these evaluations, we aim to define a set of sequencing problems usable for future sequencing approaches and systems to establish a common basis for comparing different solution methods. To ensure that all specified sequencing problems can be identically replicated, all parameters related to the network inflow (cf. Section 4.3.1.3) are set deterministically.

We investigate batches of 5, 10 and 15 transport units with completely defined unloading sequence, i.e., $s = 1$. Regarding their arrival sequence at the system, we consider best-case (b-c), worst-case (w-c) and average-case (a-c) situations.

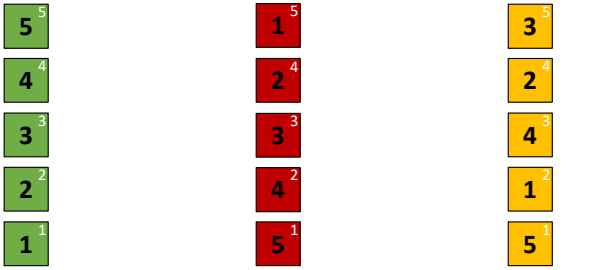
These result from comparing the positions of the transport units within the unloading sequence of a batch in contrast to the arrival sequence. We denote by σ^b the position of transport unit b within the arrival sequence of its batch. The best-case arrival sequence of a batch is given if for all of its transport units b together with their direct successors $s^b_{o^b}$ within the unloading sequence at their output module

$$\sigma^b < \sigma^{s^b_{o^b}} \tag{7.38}$$

applies (cf. Figure 7.11 (a)). Analogously, for the worst-case arrival sequence

$$\sigma^b > \sigma^{s^b_{o^b}} \tag{7.39}$$

holds (cf. Figure 7.11 (b)).



(a) Best-case. **(b)** Worst-case. **(c)** Average-case.

Figure 7.11: Deterministic batch arrival sequences for batch size 5.[4]

Concerning the average case, a subset of the transport units of a batch needs to satisfy equation (7.38), while for the remaining ones equation (7.39) is true. We select an alternating allocation depending on whether the position of a transport

[4] The black numbers refer to the rank of a transport unit within the unloading sequence at the output module, while the white numbers indicate its position within the arrival sequence of its batch at the input module.

unit within the arrival sequence is even or odd. Even positions correspond to a best-case arrival sequence, while odd positions are associated with a worst-case arrival sequence (cf. Figure 7.11 (c)). Note that the defined arrival sequences apply at batch level excluding dependencies that may arise between several batches.

The input-output assignment is organized from left to right, i.e., the first batch is assigned to the leftmost output module, the second to the one right next to it, and so on, starting again from the left after $|O|$ batches are assigned. Likewise, the first arriving transport unit of the first batch is assigned to the leftmost input module, the second to the one right next to it, and so on, starting again from the left as soon as $|I|$ transport units have been assigned. Batch sizes are irrelevant in this case, i.e. the first transport unit of a subsequent batch continues the assignment depending on the last transport unit of the preceding batch. To eliminate interarrival delays of transport units when evaluating different solutions to sequencing problems, we assume that all transport units of a sequencing problem are initially available in front of the assigned input module according to their arrival sequence. The design of experiments for comparing optimal and decentralized solutions using the conveyor network of Figure 7.10 is summarized in Table 7.3.

Table 7.3: Design of experiments to assess the quality of decentralized solutions.

Parameter	Values
Batch size	5; 10; 15
Arrival sequence	best-case; average-case; worst-case
Sequencing rate	1
Input-output assignment	deterministic
Arrival rate	∞

As the presented optimization models do not include delays for switching a conveyor module between its orthogonal transport axes, we exclude switching

times within the decentralized solutions accordingly. Likewise, we neglect delays when sending messages within the decentralized conveyor network to measure the effective conveying time for sequencing. Thus, for these evaluations, $t_{msg} = t_{switch} = 0.0\,\text{s}$ applies.

7.3.1.2 Model Parameterization

Due to extensive computation times as problem size increases, we apply a time limit of 3600 s (per iteration) for calculating optimal solutions. The number of batches within the specified problem settings (cf. Section 7.3.1.1) is iteratively increased as long as an optimal solution can be calculated within this time limit. To determine the time frame for the iterative segmented optimization model, we set the time split factor $l \in \{2, \ldots, 10\}$ and determine the minimum value, where an optimal solution can be calculated within the given time limit for each iteration. The smaller l, the less the compound partially optimal solution of the iterative segmented optimization deviates from the global optimum of the overall problem.

To generate decentralized solutions, we parameterize the decentralized sequencing algorithm with $p_b = 1$ and $p_c = 0$. Based on the results of Fleischmann and Furmans (2023), it achieves the best performance for $0 < p_b \leq 2$. p_c is set to neglect directional changes as switching delays between orthogonal transport axes of conveyor modules are not captured within the optimization models (cf. Section 7.3.1.1).

7.3.2 Numerical Results

In Tables 7.4 and 7.5, we summarize the problem settings for which we obtained an optimal solution within the given time limit for the single stage optimization model and all iterations of the iterative segmented optimization model, respectively. For solving, we applied Gurobi Optimizer version 9.5.0 with its Java interface. All results of this section were conducted using a desktop computer

with 3.60 GHz Intel Core i7-4790 CPU, 16.0 GB of RAM and Windows 10 Enterprise LTSC (version 1809).

Generally, the single stage optimization model could be solved for problem settings with a maximum of 15 transport units arriving at the given conveyor network (cf. Figure 7.10). Applying the iterative segmented optimization model yields solutions for problem settings of up to 35 transport units, i.e., seven batches of size five, enabling a more comprehensive quality assessment of decentralized solutions. As the investigated batch size increases, fewer problem settings are solved. Larger batches require a sufficiently long time frame T_k per iteration for pre-sequencing. The number of transport units assigned to the same output module increases such that observing the predefined predecessor-successor dependencies becomes more complex. But enlarging the time frame T_k induces excessive computation time.

Table 7.4: Solved problem settings using the single stage optimization model.

Batches		Arrival sequence		
size	number	best-case	average-case	worst-case
5	1	X	X	X
	2	X	X	X
10	1	X	X	X
15	1	X		

Figure 7.12 shows the deficit of the decentralized solutions compared to the globally optimal solutions for the problem settings solved with the single stage optimization model. Using x|y|z, we identify a problem setting with batch size x, where y batches arrive in sequence z. Illustrated error bars describe the length of the 95 %-confidence interval for the decentralized sequencing time resulting from the underlying simulation experiment (cf. Section 7.1.4). For the best-case arrival sequence and one batch, i.e. x|1|b-c, the decentralized solution deviates

Table 7.5: Solved problem settings using the iterative segmented optimization model – Indicated values denote the minimum possible time split factor the calculation is based on.

Batches		Arrival sequence		
size	number	best-case	average-case	worst-case
5	3	3	3	4
	4	4	4	7
	5	4	7	7
	6	6	8	
	7	8	9	
10	2	4	6	
	3	6		
15	1		4	

from the optimal solution by a maximum of 3.7 % for all investigated batch sizes. The deficit increases with the number of transport units to be sequenced. Additionally, average-case and worst-case arrival sequences complicate batch processing such that we observe a deficit of up to 40.4 % for problem setting 10|1|w-c. Overall, the comparison of decentralized solutions to all available globally optimal solutions yields an average deficit in sequencing time of 19.8 %, where the decentralized number of movements exceeds the optimal one by 4.6 % on average.

In seven of the ten solvable problem settings, we obtain identical results for the global optimal sequencing time compared to the approximation using the lower bound approach. For the remaining problem settings, the underestimation averages 5.9 %, resulting in an overall deviation between the optimal solutions and the lower bound estimate of 1.8 % on average.

Figure 7.13 shows the deficit of the decentralized solutions compared to the partially optimal solutions for the problem settings solved with the iterative segmented optimization model. The mean relative deviation from partially optimal

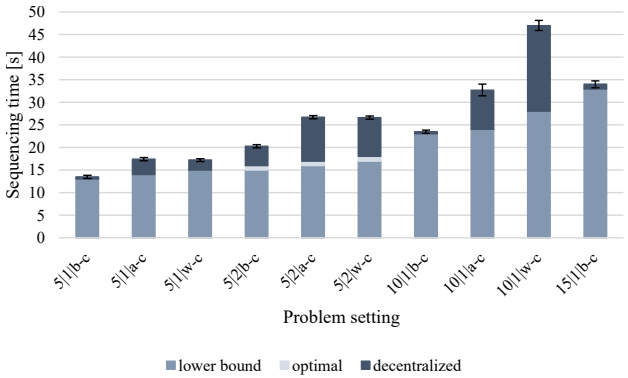


Figure 7.12: Deficit of decentralized solutions for the problem settings solved by the single stage optimization model.[5]

to decentralized sequencing time is significantly negatively linearly correlated with the number of partial solutions an overall solution is composed of within the iterative segmented optimization (p-value of 0.0018). This indicates that the sub-optimality of iteratively calculated solutions expands with the number of iterations increasing the effective deficit compared to globally optimal solutions. The deficit of the decentralized solutions compared to all available partially optimal solutions averages 19.9 %. For the problem setting 10|2|b-c the decentralized solution outperforms the partially optimal solution by 1.7 %, which again underlines the non-global optimality. The maximum deficit observed amounts to 35.2 %. The partially optimal number of movements exceeds that of the decentralized solutions by 14.2 % on average, as the system arrangement is always only locally optimized for each given iteration. Upcoming movements in later iterations are not included.

Comparing the partially optimal sequencing time to the approximation using the lower bound approach yields a deviation of 35.8 % on average. The mean

[5] The bars of the optimal and decentralized results represent the additional sequencing time incurred within the respective solutions.

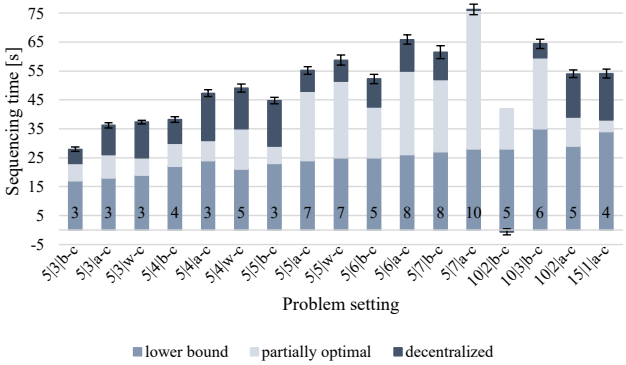


Figure 7.13: Deficit of decentralized solutions for the problem settings solved by the iterative segmented optimization model – The given numbers indicate the number of partial solutions the overall solution is composed of.[6]

deficit is significantly positively linearly correlated with the number of partial solutions an overall solution is composed of within the iterative segmented optimization (p-value of 1.08E-06). This supports the stated interdependencies between level of sub-optimality and number of iterations outlined above. The deficit of the decentralized solutions compared to the results of the lower bound approach averages 49.6 %. However, this might overestimate the actual deficit compared to an optimal solution, as the lower bound represents a theoretical, underestimating limit.

We provide all numerical results of these evaluations for further use in the attached research data (Fleischmann 2023).

[6] The bars of the partially optimal and decentralized results represent the additional sequencing time incurred within the respective solutions.

7.3.3 Implications

The developed centralized optimization models differ from the presented decentralized algorithm in terms of the available information for decision making. Decentralized solutions are composed of local decisions based on the locally available information at module level. The system arrangement can never be analyzed in its entirety, as each module can only access its own state, i.e. whether it is currently occupied or not. Furthermore, the set of transport units, including their arrival characteristics, are not known in advance. A transport unit is not registered within the system until it is physically positioned at its input module. Using the optimization models, the arrangement of transport units is optimized to minimize the required time for sequencing based on global system information. Arrival locations, i.e., the input modules where transport units are introduced into the system, arrival sequences, i.e., the positions within the permutation of the required unloading sequences at the assigned output modules, and arrival times, i.e., the physical times when transport units reach the system, are incorporated when calculating optimal solutions. Reordering transport units can consider the predefined predecessor-successor dependencies even before introducing transport units.

Despite this incomplete information, we obtain a decentralized solution deficit of less than 20 % on average for all solvable problem settings. The decentralized system design guarantees flexibility, robustness, scalability, as well as runtime efficiency, i.e., it is not limited in terms of the problem sizes to be solved. Due to missing comparative benchmarks of existing decentralized systems (cf. Section 2.2) and especially the additional benefits of the system design, we classify the developed algorithm based on the obtained results as relevant for practical use. Furthermore, we suppose that the lower bound approach is capable of providing reasonable estimates of the optimal sequencing time. Therefore, it will be used for further quality assessment of the achieved decentralized solutions within the performance evaluations of Chapter 8.

7.4 Chapter Conclusion

In this chapter, we assess the quality of the decentralized sequencing algorithm by evaluating its deficit compared to optimal solutions. Overall, this responds to the fifth research question:

> **How do decentralized solutions for sequencing problems compare to optimal solutions?**

We present two optimization models to the problem of sequencing in highest-density conveyor networks based on mixed-integer programming. The single stage optimization model provides globally optimal solutions, while the iterative segmented optimization model provides partially optimal solutions by splitting the overall sequencing problem into smaller sub-problems. This allows increasing solvable problem sizes.

Due to NP-hardness, both optimization models suffer from extensive computation times which prevents evaluating the decentralized solution deficit for real-world problem settings. Therefore, we provide a lower bound approach for approximating the minimum sequencing time for a given sequencing problem. It underestimates the true optimal result but enables assessing the quality of decentralized solutions without limitation in terms of solvable problem sizes.

Based on the obtained numerical results, we conclude that the decentralized sequencing algorithm is able to provide valuable solutions for use in practical applications. The decentralized solution deficit is less than 20 % on average for all problem settings that could be solved using the presented optimization models. Additionally, the decentralized system design guarantees flexibility, robustness, scalability, as well as runtime efficiency. By comparing the results of the lower bound approach to the obtained optimal results we found that the former is capable of providing reasonable estimates of the optimal sequencing time.

8 Performance Evaluation

Due to the modular design, installed sequencing systems can be flexibly customized. Additionally, the flow of arriving transport units highly depends on the underlying practical requirements. We provide recommendations for real-world sequencing applications within a comprehensive parameter analysis using simulation studies. Section 8.1 presents the preliminary assumptions these evaluations are based on. To capture the impacts of the various problem parameters, we subdivide the investigations according to the batch characteristics (cf. Section 8.2) and the installed conveyor network (cf. Section 8.3). All numerical results of these evaluations are available for further use in the attached research data (Fleischmann 2023). By summarizing the results of this chapter, Section 8.4 provides an answer to the sixth research question of this dissertation.

The simulated systems within these evaluations intend to represent material flow systems processing transport units arriving continuously over time as in practical applications. To obtain results that relate to stable system operation, we start recording the performance indicators after a warm-up phase, which we determine according to the MSER-5 method (for details see Appendix B.1). The stopping criterion is calculated using an admissible deviation range based on the mean system throughput of the overall simulation run time in steady-state (for details see Appendix B.2). To represent fluctuating order demands, the network inflow (cf. Section 4.3.1.3) is parameterized stochastically. We perform several replications for each parameter setting to provide reliable results based on a 95 %-confidence interval related to the average throughput result (for details see Appendix B.3).

8.1 Preliminaries

We present the general principles the obtained simulation results are based on starting with the selected parameterization of the decentralized sequencing algorithm (cf. Section 8.1.1). Section 8.1.2 outlines the investigated performance indicators used to define stable sequencing systems within their operating processes (cf. Section 8.1.3). These requirements yield the numerical results shown in Sections 8.2 and 8.3, respectively.

8.1.1 Algorithm Parameterization

The decentralized sequencing algorithm is parameterized by the relocation penalty p_b and the directional change penalty p_c guiding the decentralized A* search for path selection (cf. Section 4.2.2.2). Within the simulation studies of this chapter, we consistently set $p_b = 1$ and $p_c = 0.1$ for all simulation runs.

The relocation penalty determines whether path selection for active routes favors buffer modules inducing a relocation route or detours circumventing the blocking buffered transport unit. From the results of Fleischmann and Furmans (2023), we conclude that preferring relocations instead of detours improves system throughput. While a detour increases solely the transport time of the transport unit on the active route, active and passive routes can be executed simultaneously. That means that scheduled relocations can already be executed before the transport unit of the active route moves towards the blocking buffered transport unit. This requires p_b to be set greater than 0 and smaller than the difference between the shortest path and the shortest detour path between all pairs of modules. Distances within the conveyor network are measured in module lengths. Due to the underlying module characteristics, transport units can only be transferred via one of its four edges. Thus, a detour between two modules takes at least two module lengths compared to a shortest path, as each step leads either towards the path destination or away from it. A detour moving one module length away from the destination needs to be compensated accordingly

to reach the destination. The chosen parameterization satisfies all of these requirements.

In Section 4.2.2.2, we derive the upper bound of the directional change penalty p_c according to equation (4.14) to generally guarantee optimality. This covers all possible combinations of two sub-paths $\acute{p}$ and p^*, of which some may be highly infrequent. The precise value for the upper bound of p_c depends on the corresponding conveyor network and cannot be determined without applying additional optimization methods. The results of Fleischmann and Furmans (2023) indicate that $p_c < p_b$ represents a sufficient requirement within practical use cases, as throughput is increased for these parameter constellations. Using the chosen parameterization, this is satisfied.

8.1.2 Performance Indicators

The sequencing system can be evaluated regarding different performance indicators. These either relate to the system or single modules and transport units:

- System-related performance indicators:
  - Throughput [TU/h][1]: average number of transport units being unloaded according to the predefined sequence at the assigned output module
  - Share of lower bound [%]: ratio of the sequencing time of the lower bound solution to that of the decentralized solution
  - Density [%]: average share of sequencing modules occupied by a transport unit
- Transport unit-related performance indicators:

[1] *TU* within units implies *transport unit(s)* for all specified performance indicators.

  - Cycle time [s/TU]: average time for processing a transport unit, i.e. from introducing at the input module to unloading at the output module
  - Sojourn time [s/TU]: average time for handling a transport unit, i.e. from arriving at the system to unloading at the output module
  - Waiting time [s/TU]: average time a transport unit waits for being processed, i.e. from arriving at the system to introducing at the input module
  - Waiting time queue [s/TU]: average time a transport unit spends in the queue in front of the input module, i.e. from arriving at the system to reaching the input module
  - Waiting time input [s/TU]: average time a transport unit positioned at the input module is prevented from being introduced due to the defined inflow constraints
  - Path length [#modules]: average number of modules a transport unit passes from the input to the output module when being processed

- Module-related performance indicators:
  - Message processing [#msg/module/s]: average number of messages initiated to be processed by a recipient
  - Message transmission [#msg/module/s]: average number of messages being forwarded to an adjacent module due to the decentralized system setup

Unless otherwise stated, we present average values within steady-state of a simulation experiment for all described indicators. Illustrated error bars describe the length of the 95 %-confidence interval for the respective indicator resulting from the underlying simulation experiment.

In practical applications, system throughput is mainly relevant, as this is decisive for operational suitability in industrial processes. Therefore, the numerical results of these simulation studies (cf. Sections 8.2.2, 8.3.1.2, and 8.3.2.2) focus on illustrating system throughput results measured in transport units per hour. All other performance indicators are included in the attached research data (Fleischmann 2023). We apply the presented lower bound approach (cf. Section 7.2) to benchmark the obtained decentralized results within these evaluations. It refers to the theoretically minimum possible sequencing time for a given sequencing problem the sequencing time of the decentralized algorithm is compared to. As throughput and sequencing time are inversely related, this indicates a maximum possible system performance in terms of throughput.[2]

8.1.3 Stability Requirements

To provide reasonable simulation results in terms of the presented performance indicators, the sequencing system needs to be stable. A stable system is able to continuously process the incoming flow of materials such that there is no overload. Using principles from queuing theory, we define a stability criterion for the presented sequencing system. Based on the requirements of a simple queuing system (cf. Section 8.1.3.1), we derive the time components involved when handling a transport unit within the sequencing system (cf. Section 8.1.3.2). This yields the adapted stability criterion (cf. Section 8.1.3.3).

[2] For the evaluations of Section 7.3, we use the sequencing time as key performance indicator due to the limited problem size. However, when simulating continuous material flow systems of practical applications, steady-state system throughput is more meaningful.

8.1.3.1 Principles of Queuing Theory

Considering a simple queuing system composed of one arrival stream, one processing station, and one departure stream (cf. Figure 8.1). The arrival rate λ results from the expected interarrival time $E(t_a)$ of the arriving items, i.e.,

$$\lambda = \frac{1}{E\left(t_a\right)}. \tag{8.1}$$

Equally, the processing rate μ results from the expected processing time of the system $E(t_p)$. So,

$$\mu = \frac{1}{E\left(t_p\right)}. \tag{8.2}$$

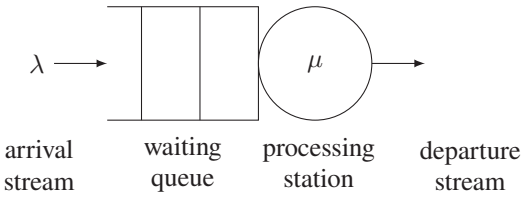


Figure 8.1: Simple queuing system.

System stability is expressed using the system utilization ρ. The system is stable if

$$\rho = \frac{\lambda}{\mu} \overset{!}{<} 1. \tag{8.3}$$

Otherwise, the arrival stream of the system will exceed its departure stream creating a continuously increasing waiting queue in front of it. (Arnold and Furmans 2019, p. 114ff.)

8.1.3.2 Time Components in Sequencing

We formalize the time components involved when handling a transport unit for sequencing such that reorganizing transport units within the system as well as the defined inflow constraints (cf. Section 3.2.1.2) can be represented. This serves to specify the adapted stability criterion (cf. Section 8.1.3.3).

Generally, the sojourn time t_s^b of a transport unit b for sequencing is defined by its arrival time a^b within the queue of arriving transport units at the assigned input module and its unloading time u^b at the assigned output module. This implies

$$t_s^b = u^b - a^b. \tag{8.4}$$

This sojourn time comprises the waiting time t_w^b until transport unit b is introduced into the system at time e^b together with its processing time for sequencing t_p^b. Therefore,

$$t_s^b = t_w^b + t_p^b \tag{8.5}$$

with

$$t_w^b = e^b - a^b \qquad \text{and} \qquad t_p^b = u^b - e^b \tag{8.6}$$

applies.

Due to the defined inflow constraints which prevent congesting the sequencing system in an $(m : n)$ setting, unlike with common queuing systems, the waiting time t_w^b consists of

- the waiting time within the queue of transport units *in front of* the assigned input module $t_{w^q}^b$ and

- the waiting time *on* the assigned input module until the necessary requirements for introducing the transport unit according to the inflow constraints are met $t^b_{w^e}$.

Thus,

$$t^b_w = t^b_{w^q} + t^b_{w^e}. \tag{8.7}$$

Overall, this gives

$$\begin{aligned} u^b &= a^b + t^b_s \\ &= a^b + t^b_w + t^b_p \\ &= a^b + t^b_{w^q} + t^b_{w^e} + t^b_p. \end{aligned} \tag{8.8}$$

8.1.3.3 Adapted Stability Criterion

In contrast to simple queuing systems (cf. Figure 8.1), the presented sequencing system provides multiple input and output modules for simultaneously introducing and unloading transport units. Furthermore, multiple transport units are present within the system for being processed. This yields a queuing model with multiple arrival and departure streams and parallel processing.

To evaluate system stability, we define the arrival and processing rate at system level such that equation (8.3) can be applied. These can be derived from the interarrival and processing times of the single transport units, respectively. We denote by $E(t_{a_i})$ the expected interarrival time of the transport units arriving at input module i while $E(t^b_p)$ denotes the expected processing time for sequencing a single transport unit b within the system.

Basically, the arrival rate λ_i at a single input module i is determined by the expected interarrival time $E(t_{a_i})$. However, a transport unit arriving on its assigned input module might need to wait until a sufficient number of other transport units have been unloaded due to the defined inflow constraints (cf.

condition (I3)). Processing a transport unit is on average delayed by $E\left(t_{w^e}^{b_i}\right)$ for each transport unit b_i assigned to input module i. This requires adjusting the arrival rate considering the expected delay when introducing to represent the arrival process appropriately. We define the expected delayed interarrival time $E(t_{d_i})$ at input module i as

$$E\left(t_{d_i}\right) = E\left(t_{a_i}\right) + E\left(t_{w^e}^{b_i}\right). \tag{8.9}$$

From this follows the delayed arrival rate $\check{\lambda}_i$ at input module i

$$\check{\lambda}_i = \frac{1}{E\left(t_{d_i}\right)}. \tag{8.10}$$

We obtain the delayed arrival rate $\check{\lambda}$ of the system based on the expected delayed interarrival time $E(t_d)$ at system level as

$$\check{\lambda} = \sum_{i \in I} \check{\lambda}_i = \frac{1}{E\left(t_d\right)}, \tag{8.11}$$

where

$$E\left(t_d\right) = \frac{1}{\sum_{i \in I} \frac{1}{E\left(t_{d_i}\right)}}. \tag{8.12}$$

The processing rate of the system μ depends on the expected processing time for sequencing a single transport unit $E(t_p^b)$ together with the expected number of transport units $E(n^b)$ which are simultaneously within the system for being sequenced. This can be derived from the average density δ related to the total number of sequencing modules $|C|$ observed for this simulation run, i.e.,

$$E\left(n^b\right) = \delta \cdot |C|. \tag{8.13}$$

This results in the expected processing time of the system

$$E\left(t_p\right) = \frac{E\left(t_p^b\right)}{E\left(n^b\right)} = \frac{E\left(t_p^b\right)}{\delta \cdot |C|} \tag{8.14}$$

and thus,

$$\mu = \frac{1}{E\left(t_p\right)} = \frac{\delta \cdot |C|}{E\left(t_p^b\right)}. \tag{8.15}$$

With equations (8.11) and (8.15), we are able to describe the arrival and processing rate at system level based on the parallel arrival and processing rates at module level. Using equation (8.3), the stability criterion for the multi-batch sequencing system results as

$$\rho = \frac{\check{\lambda}}{\mu} = \frac{E\left(t_p\right)}{E\left(t_d\right)} = \sum_{i \in I} \frac{1}{E\left(t_{a_i}\right) + E\left(t_{w^e}^{b_i}\right)} \cdot \frac{E\left(t_p^b\right)}{\delta \cdot |C|} \overset{!}{<} 1. \tag{8.16}$$

If equation (8.16) holds, we denote the system as stable when processing the stream of arriving transport units. The limiting throughput γ of the system results from the highest possible arrival rate λ^{γ} satisfying equation (8.16).

All simulation results within these evaluations are based on limiting throughput simulation experiments. We incrementally increase the expected interarrival time at system level in steps of 0.05 s as long as equation (8.16) holds. This yields the simulation experiment of the limiting throughput for the given parameter setting.

8.2 Impact of Batch Characteristics

Analyzing the impact of batch characteristics is based on the design of experiments defined in Section 8.2.1. We present the numerical results in Section 8.2.2. These yield the implications of Section 8.2.3.

8.2.1 Design of Experiments

We apply a reference network comprising a square 7×7 arrangement of sequencing modules including five input and five output modules for introducing and unloading at the top and bottom, respectively (cf. Figure 8.2).

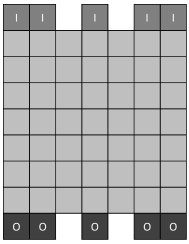


Figure 8.2: Reference network to evaluate batch characteristics.

Batch sizes are normally distributed, i.e., $k \sim \mathcal{N}(\mu, \sigma^2)$, with mean values $\mu \in [1, 45]$ and a fixed standard deviation of $\sigma = 2$. Note that the maximum processable batch size of the conveyor network in Figure 8.2 is 48 according to the specifications given in Section 4.3.1.3. As arrival sequence and sequencing rate apply at batch level, these need to be analyzed depending on the corresponding batch size. To examine the impact of different batch sizes in isolation, we investigate batches with completely defined unloading sequence ($s = 1$). These arrive at the system according to randomly generated permutations of their unloading sequence.

For analyzing different batch arrival sequences and sequencing rates, we focus on mean batch sizes of $\mu \in [5, 20]$ transport units. Based on discussions with practitioners, we deduce that these are most relevant for real-world applications. By investigating best-case and worst-case arrival sequences, we can estimate upper and lower bounds of the expected throughput achieved in practical operation. For the best case, the arrival and unloading sequences of batches are equal, while for the worst case, batches arrive in reverse order as they are unloaded at their output modules. Random arrival sequences are intended to

represent general circumstances of practical applications. The sequencing rate is varied in steps of 0.25.

Batches are randomly assigned to output modules. Likewise, arriving transport units of these batches are randomly assigned to input modules. The arrival rate is set to provide the limiting system throughput (cf. Section 8.1.3). Table 8.1 summarizes the design of experiments for analyzing the impact of batch characteristics for the given conveyor network of Figure 8.2.

Table 8.1: Design of experiments to evaluate the impact of batch characteristics.

Parameter		Values
Batch size	μ	1[3]; 5; 10; 15; 20; 25[3]; 30[3]; 35[3]; 40[3]; 45[3]
	σ	2
Arrival sequence		random; best-case; worst-case
Sequencing rate		0; 0.25; 0.5; 0.75; 1
Input-output assignment		random
Arrival rate		λ^{γ}

8.2.2 Numerical Results

Figure 8.3 shows the limiting throughput results for the specified normally distributed batch sizes. We observe a significant negative linear correlation between limiting throughput and batch size related to their mean values (p-value of 2.20E-09). Compared to a mean batch size of 1, increasing the mean batch size decreases the limiting system throughput by up to 58.0 % for a mean batch size of 45 transport units. Thus, the limiting throughput is reduced by 1.3 % on average for each increase in mean batch size by one transport unit per batch. Due to the strong linear correlation, we assume that the trend line in

[3] only with sequencing rate 1 and random arrival sequence

Figure 8.3 can be used to approximate the limiting throughput results for all further mean batch sizes ranging from 1 to 45 transport units per batch.

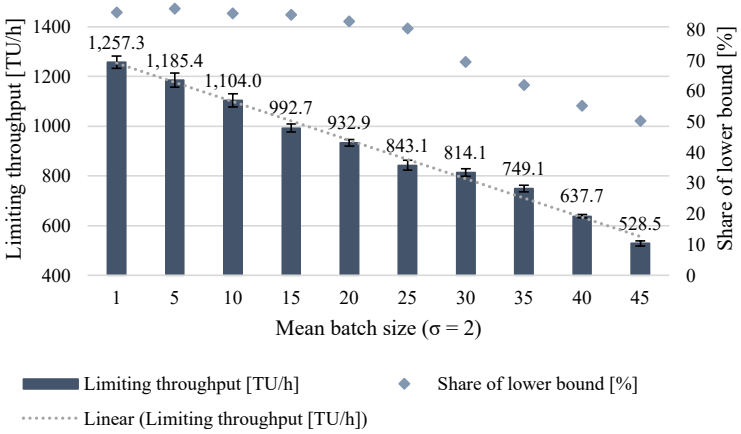


Figure 8.3: Limiting throughput depending on batch size ($s = 1$, random arrival sequence).

The solution deficit of the decentralized solutions averages 25.7 % compared to the analytically calculated lower bound. For mean batch sizes of up to 25 transport units, the decentralized solutions and the results of the lower bound approach differ by less than 20 %. With increasing mean batch sizes, the deficit of the decentralized solutions increases to a maximum of 49.6 % at mean batch size 45. The analytically calculated lower bound is prone to underestimate in case of larger batch sizes. These increase the average time a transport unit is buffered within the system until all necessary predecessors are available, which is not captured when estimating its processing time using t^b_{min}.

Best-case and worst-case arrival sequences change the limiting throughput by an average of +3.9 % and -2.9 %, respectively, compared to random arrival sequences for the selected mean batch sizes between 5 and 20 (cf. Figure 8.4). The effects are amplified with increasing mean batch size as more transport units arrive already observing the unloading sequence in the best case or need

to be reordered in the worst case. Thus, for a mean batch size of 20 transport units, the limiting throughput is improved by up to 6.7 % and reduced by up to 5.2 % for the best- and worst-case arrival sequence, respectively.

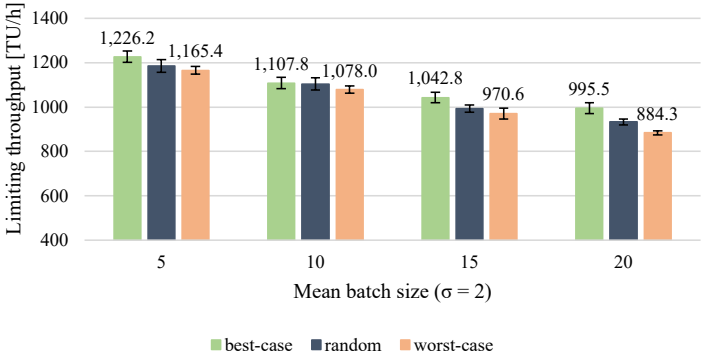


Figure 8.4: Limiting throughput depending on arrival sequence for selected batch sizes ($s = 1$).

The average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 14.8 % for these evaluations with a minimum of 10.9 % and a maximum of 24.9 %.

Partially defined unloading sequences imply less predefined predecessor-successor dependencies and enable increasing system throughput (cf. Figure 8.5). Compared to a completely defined unloading sequence ($s = 1$), the limiting throughput is improved by an average of 30.0 % at sequencing rate 0 for the selected mean batch sizes between 5 and 20. The number of transport units of non-equal rank within a batch increases with the mean batch size at a given sequencing rate. Therefore, the differences between partially defined unloading sequences with $s > 0$ decrease with increasing batch size. This yields a maximum increase of 42.2 % for mean batch size 20, while the minimum increase amounts to 17.4 % at mean batch size 5.

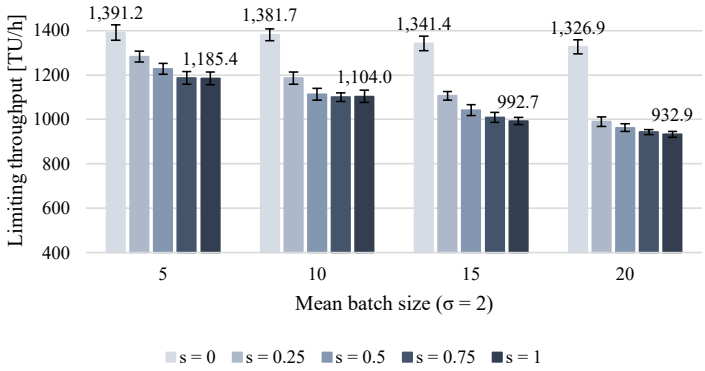


Figure 8.5: Limiting throughput depending on sequencing rate for selected batch sizes (random arrival sequence).

For these evaluations, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 13.8 % with a minimum of 10.9 % and a maximum of 17.3 %.

8.2.3 Implications

Sequencing becomes increasingly sophisticated the more arriving transport units need to be reordered within the system. Increased batch sizes and higher sequencing rates imply more predefined predecessor-successor dependencies within each batch to be observed at the output modules. This reduces the number of transport units already requested upon their arrival such that they can be routed directly from the input to the output module. Buffering non-requested transport units within the system increases their processing time, which decreases system throughput. These effects are intensified or diminished by the arrival sequences of batches. If batches arrive at the system in reverse order compared to the predefined unloading sequence, i.e. a worst-case arrival sequence, buffering becomes more likely. The opposite is true the more the arrival sequence of batches follows their predefined unloading sequence as

with a best-case arrival sequence. Due to the defined inflow constraints (cf. Section 3.2.1.2), the number of batches which can be processed simultaneously within the system decreases with increasing batch size. Therefore, higher mean batch sizes reduce the number of transport units requested next within the unloading sequences. This limits the outflow of transport units from the system.

Overall, system throughput in sequencing is enhanced

- by reducing the size of the batches to be processed,
- by matching their arrival sequences with the predefined unloading sequences, and
- by reducing the predefined predecessor-successor dependencies such that there are more transport units of equal rank within each batch.

8.3 Network Analysis

A conveyor network is characterized by the number and arrangement of its sequencing, input, and output modules. Input and output modules provide the inflow and outflow of the system, while sequencing modules specify available paths within the network as well as its buffering capacity. The buffer capacity of the network determines the inflow constraints at its input modules (cf. Section 3.2.1.2). For reasons of comparability, we assume similar buffer capacity for all conveyor networks studied in the following. The decentralized system design offers scalability such that much larger networks than those simulated in this dissertation are feasible. As these do not reveal essentially further findings, we focus on network sizes where the structural impact can be comprehensibly demonstrated within our evaluations. In Section 8.3.1, we analyze how inflow and outflow capacity of a network affect system throughput. Different arrangements of sequencing modules are examined in Section 8.3.2. All evaluations

regarding the network analysis are based on a network inflow parameterization as given in Table 8.2.

Table 8.2: Network inflow parameterization for investigated conveyor networks.

Parameter		Value
Batch size	μ	5
	σ	2
Arrival sequence		random
Sequencing rate		1
Input-output assignment		random
Arrival rate		λ^{γ}

8.3.1 Inflow and Outflow Capacity

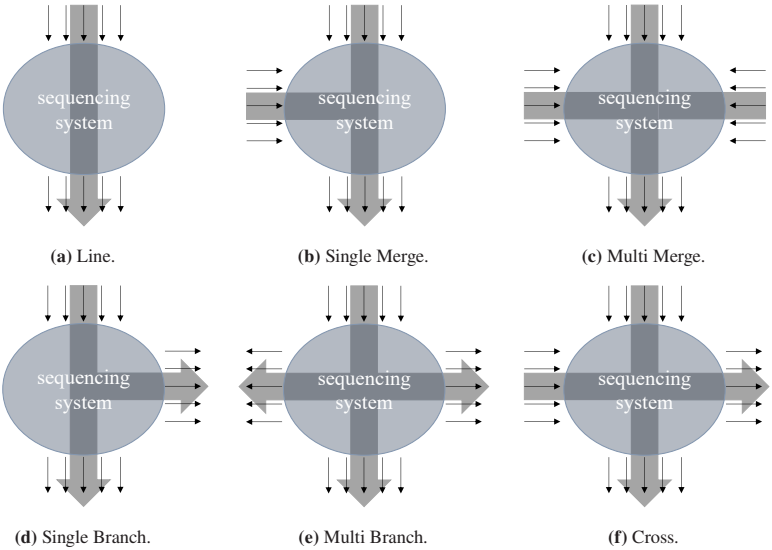
Analyzing the impact of network inflow and outflow capacity is based on the design of experiments defined in Section 8.3.1.1. We present the numerical results in Section 8.3.1.2. These yield the implications of Section 8.3.1.3.

8.3.1.1 Design of Experiments

The necessary inflow and outflow capacity of a sequencing network depend on the upstream and downstream process of the underlying application. Each arrival and departure stream requires an input and output module to be installed within the sequencing network. Based on the number and arrangement of input and output modules, we define the following system level material flow types (cf. Figure 8.6):

- *Line* with $|I| = |O|$,
- *Single Merge* with $|I| > |O|$,

- *Multi Merge* with $|I| >> |O|$,
- *Single Branch* with $|I| < |O|$,
- *Multi Branch* with $|I| << |O|$, and
- *Cross* with $|I| = |O|$.



(a) Line. **(b)** Single Merge. **(c)** Multi Merge.

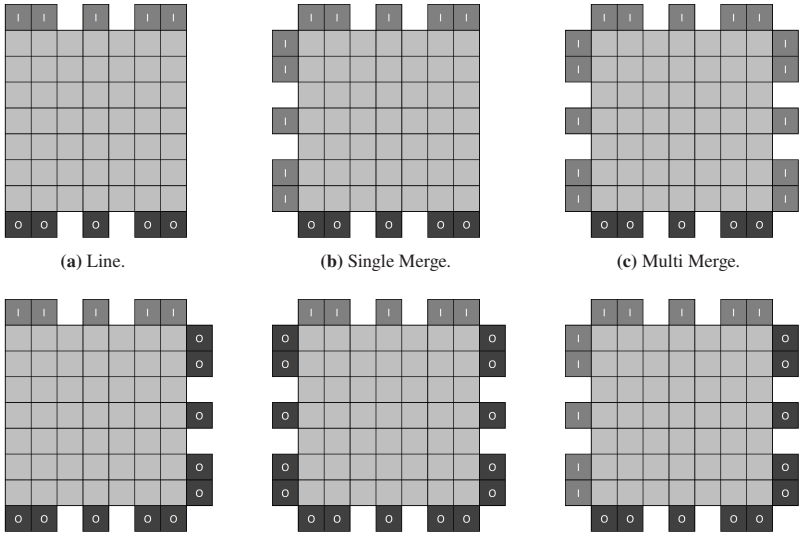
(d) Single Branch. **(e)** Multi Branch. **(f)** Cross.

Figure 8.6: System level material flow types.

To reduce the total number of possibilities, we exploit symmetries and fix the top and bottom of the sequencing system for introducing and unloading, respectively. This defines the system level material flow of linear networks, where we assume an equal number of input and output modules. For merging networks, one or both remaining system sides are additionally used for introducing transport units, increasing its inflow capacity. The opposite holds for branching networks. Crossing networks result from combining branching and merging networks, where the number of input and output modules is equal but exceeds that of linear networks. More complex constellations in which, for instance, one or more

system sides simultaneously provide introducing and unloading can be handled using the presented sequencing algorithm, but are not further investigated in this context due to their lack of practical relevance.

We use a 7×7 square arrangement of sequencing modules as reference network and set the number of input and output modules to five modules per system edge used for introducing and unloading, respectively. Combined with the system level material flow types according to Figure 8.6, this results in the six sequencing networks as shown in Figure 8.7.



(a) Line. **(b)** Single Merge. **(c)** Multi Merge.

(d) Single Branch. **(e)** Multi Branch. **(f)** Cross.

Figure 8.7: Reference networks to evaluate inflow and outflow capacity.

8.3.1.2 Numerical Results

We apply the input/output ratio $|I|/|O|$, which relates the inflow and outflow capacity of a sequencing network, to analyze their impact on system throughput for these evaluations. Figure 8.8 shows the limiting throughput results for the

six sequencing networks of Figure 8.7. We observe a significant negative linear correlation between mean limiting throughput values and input/output ratio (p-value of 0.0083).

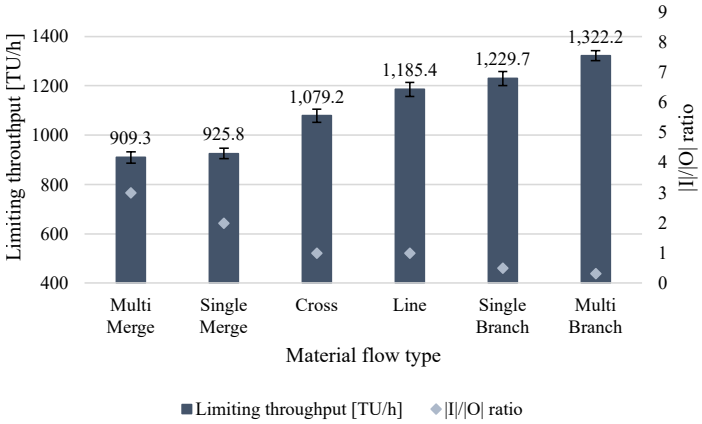


Figure 8.8: Limiting throughput depending on input/output ratio.

In merging networks, the limiting throughput is reduced by up to 23.3 % compared to the network of flow type Line when tripling the inflow capacity up to the Multi Merge network. Branching networks increase the limiting throughput by up to 11.5 % compared to the network of the flow type Line when tripling the outflow capacity up to the Multi Branch network. Thus, the impact of changing the inflow capacity exceeds that of analogously changing the outflow capacity, as the predefined unloading sequences additionally need to be observed when unloading transport units at the output modules. This can also be seen by comparing the networks of the flow types Cross and Line. Despite equal inflow and outflow capacity in each network, doubling both capacities within the Cross network reduces the limiting throughput by 9.0 %.

For these evaluations, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 20.0 %. The

minimum of 7.7 % is achieved at the smallest input/output ratio within the Multi Branch network, while the maximum of 33.1 % occurs at the highest input/output ratio within the Multi Merge network. The lower bound is prone to underestimate in case of increased inflow capacity, as it does not incorporate inflow constraints to avoid congesting the system. Introducing times are calculated solely based on the arrival characteristics of the transport units. When reducing the inflow capacity of the network, sequencing times of transport units become more dependent on the arrival times of missing predecessors. This yields a solution deficit of the decentralized algorithm of at most 15.5 % for the Line, Single Branch, and Multi Branch network compared to the lower bound solutions.

8.3.1.3 Implications

Additional outflow capacity within the sequencing network enhances system throughput, while it is reduced by additional inflow capacity. As the number of input modules increases, more transport units can be introduced into the system simultaneously such that system density increases. In case of insufficient outflow capacity for unloading, this creates a bottleneck at the output modules. Higher system density requires more relocations or detours within active routes increasing the processing times of transport units. As the number of output modules increases, more transport units are requested next within their unloading sequences simultaneously. This facilitates unloading, as the flow of transport units out of the system becomes less restrictive. To enable efficient sequencing, sufficient outflow capacity is essential within the sequencing network to prevent the output modules from becoming the bottleneck of the system when unloading sequenced transport units.

8.3.2 Arrangement of Sequencing Capacity

The developed decentralized sequencing algorithm not necessarily requires rectangular and compact network structures such as in Hao (2020), Gue et al. (2014), or Uludağ (2014). It is capable of operating in networks with notches, holes, or input and output modules positioned inside the system, as long as the requirements specified in Section 3.2 are satisfied. We denote a conveyor network as *compact* if the shape defined by the outermost system border of its arrangement of sequencing modules is completely covered with sequencing modules. Otherwise, it is *non-compact*. A system border is formed by a set of adjacent sequencing modules each of which is connected to less than four other sequencing modules.

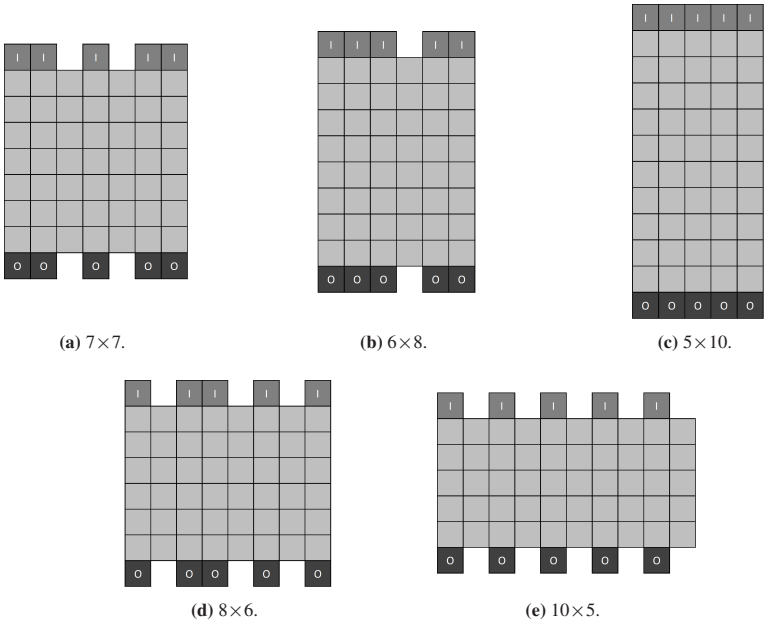
Analyzing different network arrangements is based on the design of experiments defined in Section 8.3.2.1. We present the numerical results in Section 8.3.2.2. These yield the implications of Section 8.3.2.3.

8.3.2.1 Design of Experiments

Industrial environments usually evolved over time and are limited in terms of available space. Structural elements or other permanently installed objects restrict the range of feasible options for setting up new systems. Due to the modular and decentralized system definition, the presented sequencing system can be adapted to comply with application-specific requirements.

To cover the needs of practical applications, we classify the five scenarios *Greenfield Planning*, *Brownfield Planning*, *Vertical In- or Outflow*, *Investment Minimization*, and *Defective Elements*. Each of them incorporates certain constraints regarding the arrangement of its sequencing modules. For reasons of comparability, all investigated conveyor networks include five input and five output modules each with an system level material flow based on type Line (cf. Figure 8.6 (a)).

Greenfield Planning using Network Type Rectangle The least complex network structure is rectangular- or even square-shaped. It is easy to set up if the available space has few or no restrictions, such as in new buildings. However, the length and width of the rectangle modify distances between input and output modules. We define the five conveyor networks shown in Figure 8.9 for investigating how the length-to-width ratio in rectangular, compact networks affects system throughput.

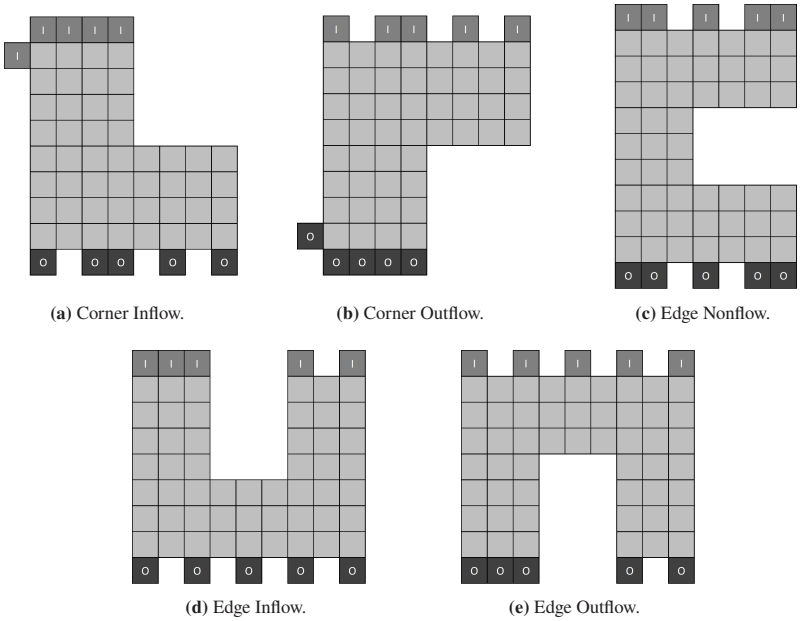


(a) 7×7. **(b)** 6×8. **(c)** 5×10. **(d)** 8×6. **(e)** 10×5.

Figure 8.9: Investigated conveyor networks of type Rectangle.

Brownfield Planning using Network Type Notch When adding new installations within existing applications, the available space is usually limited by other stationary resources or structural elements. These may represent immovable obstacles and restrict feasible setup options, as the network needs to

integrate them. Using the network type Notch, we investigate how rectangular notches at outer system edges of initially rectangular networks influence system throughput. For this, we define the five compact conveyor networks shown in Figure 8.10.



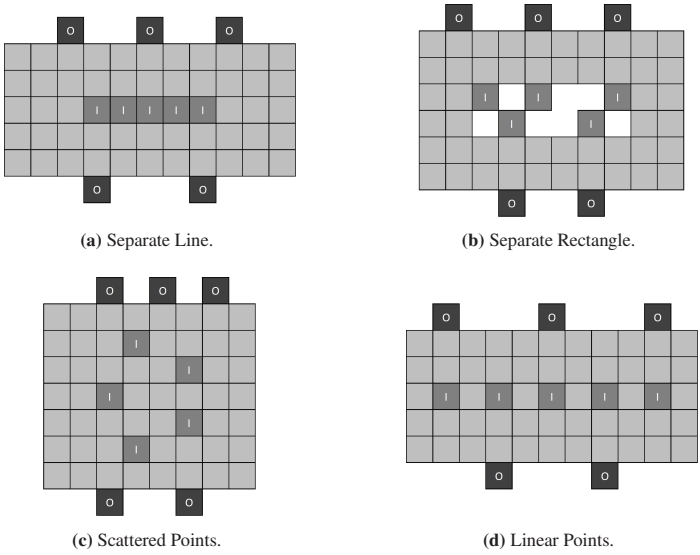
(a) Corner Inflow. **(b)** Corner Outflow. **(c)** Edge Nonflow.

(d) Edge Inflow. **(e)** Edge Outflow.

Figure 8.10: Investigated conveyor networks of type Notch.

Vertical In- or Outflow using Network Type Hub-&-Spoke In practical applications, the flow of materials is often not only guided in two dimensions on a flat surface but also upwards or downwards in vertical directions to better exploit available space. Therefore, introducing or unloading transport units is not necessarily restricted to the outer system borders, but is also possible using input and output modules located inside the network. This creates non-compact networks which enable shortening transport distances between input

and output modules compared to a compact setup. Using the network type Hub-&-Spoke, we investigate the throughput of networks where transport units are sequenced from the center to the edges and vice versa. We define the four conveyor networks shown in Figure 8.11 for investigating an inner system inflow. The corresponding networks with inner system outflow result analogously by interchanging the positions of input and output modules.



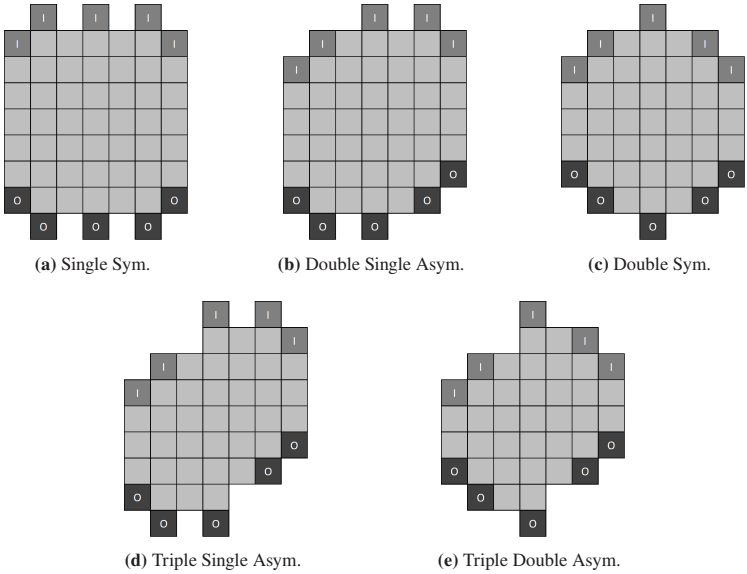
(a) Separate Line. **(b)** Separate Rectangle.

(c) Scattered Points. **(d)** Linear Points.

Figure 8.11: Investigated conveyor networks of type Hub-&-Spoke.

Investment Minimization using Network Type Fringe Monetary factors are crucial when designing a new system. The investment costs of the presented decentralized sequencing system can mainly be influenced by the number of sequencing modules installed within the conveyor network. Removing the outermost corner modules of rectangular sequencing module arrangements creates circular or elliptical compact networks with reduced investment costs. Based

on the 7×7 square arrangement of sequencing modules, we define the five conveyor networks shown in Figure 8.12 to investigate how these modifications affect system throughput.



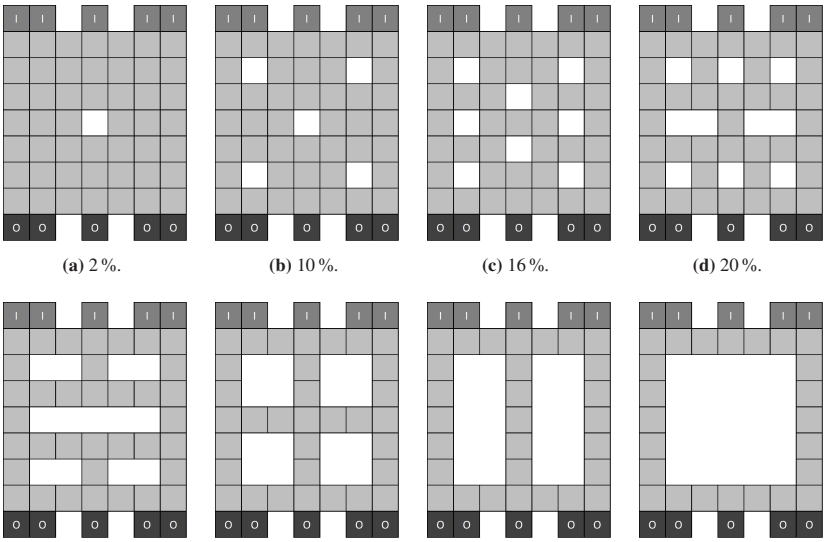
(a) Single Sym. **(b)** Double Single Asym. **(c)** Double Sym.

(d) Triple Single Asym. **(e)** Triple Double Asym.

Figure 8.12: Investigated conveyor networks of type Fringe.

Defective Elements using Network Type Hole Modules may fail during operation such that they are no longer available for sequencing. Due to the decentralized setup, the system is still operable as long as a feasible conveyor network, meeting the specified requirements for sequencing capability (cf. Section 3.2.2), is created from the remaining modules. The non-compact network type Hole serves to investigate the impact of reduced system capacity due to defective sequencing modules. We assume that the failure is already present when configuring the system, but it runs without the defective modules. Starting from the 7×7 square arrangement of sequencing modules, we define the eight conveyor networks shown in Figure 8.13. Holes intend to represent defective

sequencing modules. Particularly, we focus on defects within the system center, where we suppose major impacts on system performance.



(a) 2 %. **(b)** 10 %. **(c)** 16 %. **(d)** 20 %.

(e) 27 %. **(f)** 33 %. **(g)** 41 %. **(h)** 51 %.

Figure 8.13: Investigated conveyor networks of type Hole.

8.3.2.2 Numerical Results

Figure 8.14 shows the limiting throughput results for network type Rectangle. The square 7×7 arrangement outperforms all other rectangular arrangements with a limiting throughput of 1185.4 TU/h. It decreases by up to 15.7 % for the 10×5 network when modifying the length-to-width ratio. More square-like arrangements, where input and output modules are placed along the two shorter system edges, yield a smaller reduction in throughput. This can be illustrated by comparing the throughput heat maps of the 7×7, 5×10, and 10×5 networks (cf. Figure 8.15). Generally, the vertical path from the input to the output module in the network center is used most frequently. The wider

the rectangular network arrangement within the investigated conveyor networks of Figure 8.9, the more workload is distributed among the parallel vertical paths. Sequenced unloading requires sufficient space between input and output modules for reordering transport units within the system. However, excessive distances cause unnecessarily long transport paths. The 7×7 arrangement resolves this trade-off most effectively. Its limiting throughput result is also taken as a reference for the further investigations concerning the networks of the other application scenarios.

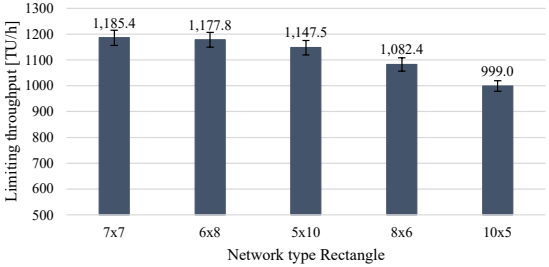
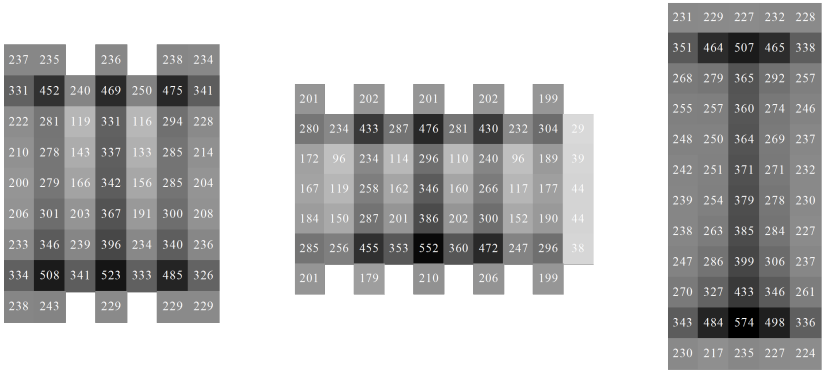


Figure 8.14: Limiting throughput results for application scenario Greenfield Planning.



(a) 7×7. **(b)** 10×5. **(c)** 5×10.

Figure 8.15: Network heat maps of type Rectangle – numbers and shading indicate throughput per hour and module.[4]

For the evaluations concerning network type Rectangle, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 14.3 % with a minimum of 12.9 % and a maximum of 16.6 %.

Figure 8.16 shows the limiting throughput results for network type Notch. Compared to the 7×7 square arrangement, it decreases by 27.5 % on average. Networks containing a corner notch outperform those containing an edge notch. In both corner notch networks, a bottleneck occurs where the input or output modules are densely spaced as illustrated in Figure 8.17. As at the output modules the predefined unloading sequences need to be observed additionally, the Corner Inflow network achieves the smallest throughput reduction of 13.3 %. Edge notches cut into the horizontal path in the center of the system, which is used most frequently in rectangular networks (cf. Figure 8.15). Compared to the 7×7 square arrangement, the limiting throughput is reduced by up to 41.4 % for the Edge Outflow network. The Edge Nonflow network outperforms the Edge Inflow and Edge Outflow network. Notches at the edge of the input and output modules, as with both latter networks, divide the material flow around the notch into two separate flows. Reordering of transport units is primarily accomplished at the bottom or top of the system, which is not affected by the notch. The lateral edge notch within the Edge Nonflow network merely narrows the network around the notch, while the entire length of the network can be used for reordering transport units.

For the evaluations concerning network type Notch, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 18.4 % with a minimum of 7.0 % and a maximum of 25.0 %.

Figure 8.18 shows the limiting throughput results for network type Hub-&-Spoke. Although all investigated networks of this type provide a smaller average distance between input and output modules than the 7×7 square arrangement, none of them achieves an equivalent limiting throughput. The average limiting

[4] Shading range is linear from light gray to black based on the minimum and maximum throughput values over all networks of type Rectangle.

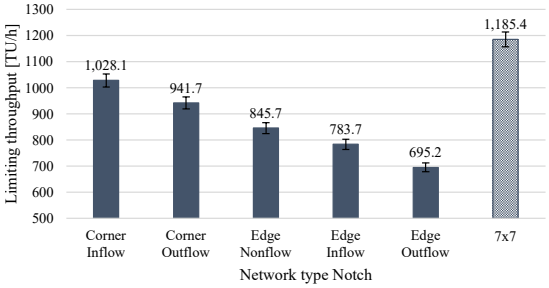
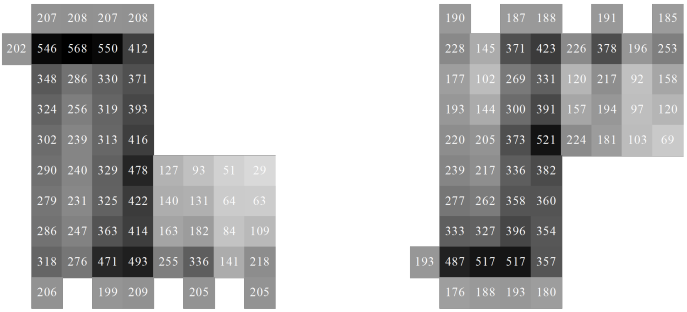


Figure 8.16: Limiting throughput results for application scenario Brownfield Planning.



(a) Corner Inflow. **(b)** Corner Outflow.

Figure 8.17: Network heat maps of type Notch – numbers and shading indicate throughput per hour and module.[5]

throughput reduction amounts to 16.0 %. Nevertheless, we observe a significant negative linear correlation between mean limiting throughput values and average input-output distance within the network for the investigated Hub-&-Spoke networks (p-value of 1.96E-06). Within the Separate Rectangle networks, several paths from input to output modules include the detour around the rectangular area in the network center increasing the average input-output distance. These networks yield the highest limiting throughput reduction of 29.8 % and 30.4 %,

[5] Shading range is linear from light gray to black based on the minimum and maximum throughput values over all networks of type Notch.

respectively. Scattering input and output modules within the network, as in the Scattered Points and Linear Points networks, increases their connectivity to sequencing modules. However, these also represent obstacles which need to be bypassed when routing transport units within the network. Due to the densely spaced inner input and output modules and the closest average input-output distance, the Separate Line networks yield the smallest throughput reduction of 9.0 % and 5.1 %, respectively, compared to the 7×7 square arrangement.

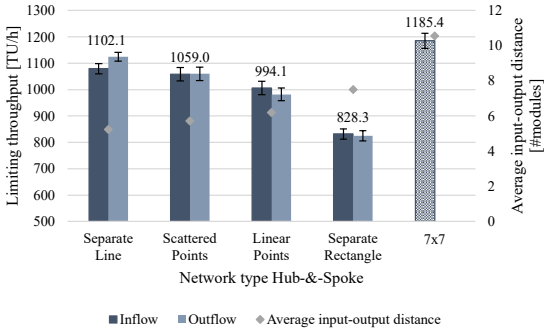


Figure 8.18: Limiting throughput results for application scenario Vertical In- or Outflow.

For the evaluations concerning network type Hub-&-Spoke, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 11.2 % with a minimum of 7.0 % and a maximum of 15.4 %.

Before presenting the evaluation results for network types Fringe and Hole, we would initially like to remark that reducing available sequencing capacity within a network always implies increasingly restrictive inflow constraints to prevent congesting the system (cf. Section 3.2.1.2). Thus, when reducing monetary investments or accepting defective modules within the network, enhanced inflow control is required during running system operation. The network inflow parameterization (cf. Table 8.2) allows for these smaller capacities of network types Fringe and Hole, as $k << (|C| - 1)$ holds (cf. Section 4.3.1.3).

Figure 8.19 shows the limiting throughput results for network type Fringe. They outperform the limiting throughput of the 7×7 square arrangement by 10.0 % on average due to two benefits: First, replacing sequencing modules at the system corners by input or output modules decreases the average input-output distance within the network. Second, the fringed system edges allow input and output modules to be positioned such that they can introduce and unload transport units from two directions. This reduces bottlenecks at input and output modules and enables routes to be planned more efficiently based on the current system occupation. Comparing the heat maps of the Single Sym and Double Sym network (cf. Figure 8.20 (a) and (b)) shows that horizontal paths are used more frequently as the system edges become more frayed. System workload can better be balanced among available sequencing modules. The limiting throughput increases by up to 14.3 % for the Double Sym network, while the Triple Single Asym network provides the least improvement of 7.1 % compared to the 7×7 square arrangement. Within asymmetrically fringed networks, horizontal paths contain additional directional changes. A higher amount of workload occurs at vertical paths (cf. Figure 8.20 (c)), such that symmetric fringing should be preferred to asymmetric fringing.

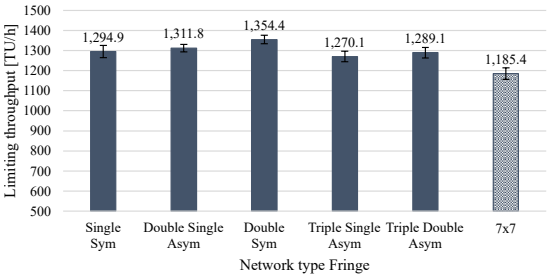
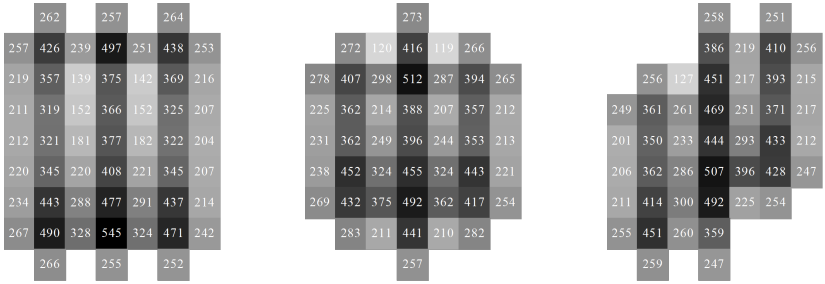


Figure 8.19: Limiting throughput results for application scenario Investment Minimization.

For the evaluations concerning network type Fringe, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 14.4 % with a minimum of 11.3 % and a maximum of 16.3 %.



(a) Single Sym. **(b)** Double Sym. **(c)** Triple Single Asym.

Figure 8.20: Network heat maps of type Fringe – numbers and shading indicate throughput per hour and module.[6]

Figure 8.21 shows the limiting throughput results for network type Hole. We observe a significant negative linear correlation between mean limiting throughput values and network perforation (p-value of 6.03E-06). While the 2 % network achieves 98.1 % of the limiting throughput of the compact 7×7 square arrangement, it is reduced by up to 59.9 % for the 51 % network. Defective modules represent obstacles when routing transport units within the system such that the flow of materials is increasingly displaced towards the intact horizontal paths (cf. Figure 8.22 (a) and (b)). Extending the network perforation limits possible paths between input and output modules. Reordering transport units is concentrated in front of the output modules creating a bottleneck when unloading them according to their predefined sequences (cf. Figure 8.22 (c)).

However, even if single modules fail, system functionality is maintained by the remaining operational ones. The presented sequencing system is still capable of processing arriving transport units due to its decentralized setup. Centralized controlled systems cannot guarantee such robustness, as they are inoperative in case a centralized component fails.

[6] Shading range is linear from light gray to black based on the minimum and maximum throughput values over all networks of type Fringe.

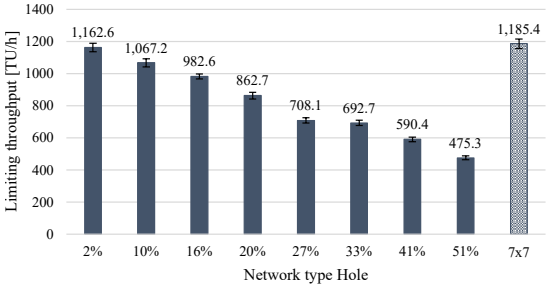
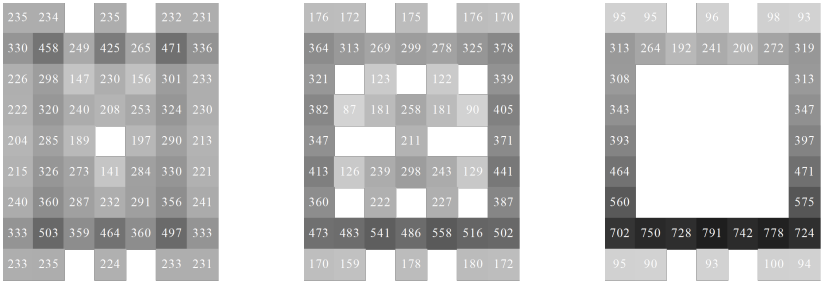


Figure 8.21: Limiting throughput results for application scenario Defective Elements.



(a) 2 %. **(b)** 20 %. **(c)** 51 %.

Figure 8.22: Network heat maps of type Hole – numbers and shading indicate throughput per hour and module.[7]

For the evaluations concerning network type Hole, the average deficit of the decentralized solutions compared to the results of the lower bound approach amounts to 20.4 %. It increases from 13.3 % for the 2 % network to up to 29.4 % for the 51 % network. As network perforation increases, the number of possible input-output paths decreases. Detours or relocations occur more frequently which is not captured when estimating the processing time of a transport unit using t^b_{min}.

[7] Shading range is linear from light gray to black based on the minimum and maximum throughput values over all networks of type Hole.

8.3.2.3 Implications

The decentralized system design enables to flexibly adapt installed sequencing systems to the individual spatial requirements of practical applications. Modifying the network structure affects the achieved system throughput during operation. If most frequently used paths of compact networks are obstructed, rearranging the transport units becomes more complex which reduces system throughput. A suitable network design enables workload balancing between the sequencing modules. This allows to improve system performance, as the flow of materials is distributed more evenly. System throughput decreases if workload is concentrated at certain parts of the system – especially in front of the output modules.

Based on the results of the investigated application scenarios, we recommend setting up conveyor networks for sequencing such that

- the installed conveyor modules form a compact network (cf. network type Rectangle, Fringe, Hub-&-Spoke, Hole),
- the length corresponds to the width of the system (cf. network type Rectangle, Fringe),
- the connectivity of input and output modules to sequencing modules within the system is increased (cf. network type Fringe, Hub-&-Spoke),
- rearranging transport units is not restricted to certain areas within the system (cf. network type Notch[8], Hole),
- paths from input to output modules are straight and without obstacles (cf. network type Rectangle, Notch, Hole),
- bottlenecks are avoided (cf. network type Notch[9], Hole), and

[8] specifically investigated edge notch networks
[9] specifically investigated corner notch networks

- input-output distances are reduced (cf. network type Hub-&-Spoke).

Moreover, the presented decentralized sequencing algorithm shows consistent performance regarding the lower bound reference throughout all investigated application scenarios. This indicates robust and scalable solutions.

8.4 Chapter Conclusion

This chapter evaluates the performance of the presented decentralized sequencing algorithm within a comprehensive parameter analysis using simulation studies. Overall, this responds to the sixth research question:

> **What throughput predictions and recommendations can we derive for sequencing in practical applications?**

We focus on the limiting system throughput as key performance indicator for these evaluations to derive recommendations for design and operation of sequencing systems running in real-world applications.

Due to the modular design, installed sequencing systems can be flexibly customized. We describe a conveyor network by the number and arrangement of its sequencing, input, and output modules. Input and output modules provide the inflow and outflow of the system, while sequencing modules specify the paths within the network as well as its buffering capacity. Additional outflow capacity within the sequencing network enhances system throughput, while it is reduced by additional inflow capacity.

Modifying the network structure affects the achieved system throughput during operation. Based on the results obtained, we deduce compact network structures, straight paths from input to output modules, reduced input-output distances, increased connectivity within the network – especially for input and output modules – and continuous sequencing areas as drivers for high system throughput.

Concerning the batch characteristics within the flow of arriving transport units, system throughput in sequencing can be improved by decreasing batch sizes, reducing predecessor-successor dependencies as well as pre-sequenced arrival characteristics.

Benchmarking the obtained decentralized solutions within these evaluations against the presented lower bound approach (cf. Section 7.2) shows that the presented algorithm is valuable for practical use.

9 Conclusion

To conclude this dissertation, we summarize its main achievements (cf. Section 9.1). This provides a new scientific contribution in the field of autonomous material handling systems from which future research opportunities can be derived (cf. Section 9.2).

9.1 Achievements

In this dissertation, we present a decentralized algorithm for multi-batch sequencing using highest-density conveyor networks. In contrast to existing systems of current research, we enable

- an entirely decentralized system setup,
- parallel processing of multiple batches within an $(m : n)$ setting,
- highest-density space utilization, as well as
- arbitrary network structures.

This offers a valuable contribution to enhance various production and material handling applications by sequenced supply of physical objects within dynamic, uncertain, and complex industrial environments.

Decentralized sequencing applies within a network of identical, autonomous input and output nodes for introducing and unloading physical objects together with sequencing nodes for processing them. Transferring physical objects as

well as exchanging data for communication occurs between each pair of adjacent nodes.

Sequencing physical objects at highest density imposes capacity and structural requirements on the underlying network. To consider the spatial dimensions of the physical objects being processed, node capacity is set to a minimum of one. Based on that, the defined *inflow constraints* comprising conditions (I1) to (I3) prevent congesting the system during operation. If the given network of sequencing nodes satisfies at least 2-connectivity, we can guarantee a feasible relocation path between each pair of distinct sequencing nodes for each highest-density arrangement.

To observe the predefined unloading sequences at the output nodes, sequencing nodes are used to buffer physical objects, which are not yet requested when being introduced at their input nodes. Highest-density system operation requires a dynamic buffering approach, where allocated buffer nodes can be switched. The overall routes of physical objects from input to output nodes are specified using active routes for introducing arriving and/or unloading requested physical objects as well as passive routes for relocating buffered physical objects interfering with active routes. To achieve efficient sequencing, we develop the *unloading sequence-based buffer selection rule* for buffer allocation on active routes, while for passive routes the *distance-based buffer selection rule* applies.

Active routes are crucial to proceed in sequencing and are initiated based on the scheduled reservations at the output nodes. Using an offline route planning approach, all routes are scheduled entirely from the start node to the (intermediate) destination node before being executed. Route planning comprises:

- the decentralized authorization concept, which avoids conflicts and enables deadlock-free system operation, as all active routes are planned sequentially,
- path selection of active routes using the adapted decentralized A* algorithm, which considers distances, buffer nodes, and directional changes,

- route reservation of active routes as well as all induced passive routes based on the concept of logical time according to the defined reservation conditions (R1) to (R6),
- relocation as sub-process during active route reservation, where buffered physical objects are successively moved to an adjacent node such that the occupied node of the active route can be accessed.

Transport execution follows the series of reserved logical time windows in ascending order at each network node. After completing a transfer, the involved nodes forward their logical clocks accordingly.

For demonstration within a practical showcase system, we apply rectangular conveyor modules providing transportation in the four cardinal directions. Each of them has its own control unit and offers an interface for connecting an adjacent module at each of its edges. The system processes transport units, which are organized in batches. Each batch entails an ordering of its transport units specifying its unloading sequence. The decentralized communication is realized via messages sent between adjacent modules, incorporating a transmission mechanism to enable exchanging information between non-adjacent modules. Using an agent-based simulation model, we implement the presented decentralized sequencing algorithm. This allows assessing the behavior of the physical, decentralized conveyor network for real-world sequencing applications.

The developed decentralized sequencing algorithm ensures system liveliness at any point in time by generally preventing deadlocks, livelocks, and starvation within algorithmic operations as well as resource allocation.

From the complexity analysis, we obtain the algorithmic complexity – measured in the required communication effort – of order $\mathcal{O}(n^3)$ at network level which reduces to $\mathcal{O}(n^2)$ at node level, both scaling with the number of nodes n.

To assess the quality of the presented sequencing algorithm, we use a single stage optimization model providing globally optimal solutions as well as an iterative segmented optimization model providing partially optimal solutions

to the problem of sequencing in highest-density conveyor networks. Due to NP-hardness, solvable problem sizes are limited. To compensate for bigger and more realistic systems, we present a lower bound approach, which analytically approximates the minimum sequencing time for a given sequencing problem. Despite incomplete local information for decision making, the solution deficit of the decentralized sequencing algorithm is less than 20 % on average for all problem settings that could be solved using the presented optimization models. Based on all benchmark results of the lower bound approach, the algorithmic quality level is maintained throughout the system variations with larger problem sizes.

The performance evaluations demonstrate that system throughput can be generally improved by decreasing batch sizes, reducing predecessor-successor dependencies as well as pre-sequenced arrival characteristics of physical objects. When designing sequencing systems running in real-world applications, sufficient outflow capacity within the sequencing network is essential. Additionally, compact network structures with straight paths from input to output nodes enable increasing system throughput. To prevent bottlenecks when introducing and unloading physical objects, we recommend increasing the connectivity of input and output nodes to sequencing nodes within the network.

9.2 Outlook

In this dissertation, we provide the basis for establishing a physical decentralized sequencing system used in practical applications. Reliable system operation under real-time conditions requires a suitable communication protocol to ensure that all messages reach their corresponding recipient. Continuous automated functional diagnostics at module level help updating available network parts as well as system capacity information. In case of failures, recovery strategies can be used to restore the system to a state from which it can continue processing.

The presented sequencing system consists of identical autonomous entities. However, this does not constitute a necessary requirement for applying the developed decentralized sequencing algorithm. As long as the relevant network structures are given, we can imagine that the presented ideas and concepts are transferable to applications with heterogeneous entities. Thus, complex production systems or even entire supply chains can be modeled, where the flow of materials satisfies sequenced supply at the points of consumption. Moreover, Industry 4.0 environments increasingly incorporate intelligence encapsulated at the level of physical objects. These are able to communicate themselves such that information between handling as well as handled units can be exchanged directly. This creates an intertwined communication network with highly increased number of participants. Complexity aspects will be crucial to restrain the communication effort incurred. The network expansions need to be integrated into the algorithmic operations using suitable methods and concepts to foster system efficiency and further enhance industrial processes.

All in all, this dissertation represents one important step in the field of autonomous material handling systems. A variety of unresolved questions remain and we are excited to see what the scientific communities will achieve in the future.

A Weighting Factors of the Objective Functions

We define the multi-criteria weighted objective functions (7.2) and (7.16) of the developed optimization models when assessing the quality of decentralized solutions (cf. Chapter 7). Their weights are set such that the optimization prioritizes a primary objective using a secondary objective as further decision criterion. Section A.1 details the underlying calculations concerning the single stage optimization model, while in Section A.2 the iterative segmented optimization model is discussed.

A.1 Single Stage Optimization Model

The multi-criteria objective function (7.2) of the single stage optimization model (cf. Section 7.1.2) aims at minimizing

- the sequencing time of the given set of transport units and
- the movements required for sequencing the set of transport units to their assigned output modules.

The main objective is to minimize the sequencing time using a minimum number of movements as secondary objective. Therefore, we need to ensure that the

amount of sequencing time within the objective function value always exceeds that of the necessary movements. This implies

$$\left| -\sum_{t \in T} \sum_{b \in B} x_{o^b}^{bt} \right| \overset{!}{\geq} \sum_{t \in T} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt}. \tag{A.1}$$

We specify a lower and upper bound for the terms of sequencing time and movement minimization, respectively, to generally satisfy equation (A.1).

Assuming a sufficiently large time frame T to sequence the given set of transport units from their starting positions at the assigned input modules to the required output modules, all transport units are positioned at their output modules at least for the last time point $t \in T$. Thus,

$$\left| -\sum_{t \in T} \sum_{b \in B} x_{o^b}^{bt} \right| \geq |B| \tag{A.2}$$

holds for the lower bound of the sequencing time minimization term.

To determine an upper bound of the movement minimization term, we consider the requirements for moving transport units (cf. constraints (7.9) and (7.10)). Generally, transferring a transport unit between two adjacent modules m and $\acute{m}$ takes $t_{conv} = 2 \cdot \Delta t$ and consists of the following steps (cf. Figure 7.1):

- occupying module m at time t,
- moving from module m to module $\acute{m}$ at time $t + \Delta t$, and
- occupying module $\acute{m}$ at time $t + 2 \cdot \Delta t$.

Assuming that a transport unit is continuously moving, the maximum number of movements it can perform within the given time frame T cannot exceed $\left\lceil \frac{|T|}{2} \right\rceil$. From this follows

$$\sum_{t \in T} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt} \leq \left\lceil \frac{|T|}{2} \right\rceil \cdot |B| \,. \tag{A.3}$$

Combining equations (A.1), (A.2) and (A.3) we obtain

$$\begin{aligned} |B| &\overset{!}{\geq} \left\lceil \frac{|T|}{2} \right\rceil \cdot |B| \\ \Leftrightarrow \; 1 &\overset{!}{\geq} \left\lceil \frac{|T|}{2} \right\rceil \end{aligned} \tag{A.4}$$

which is a contradiction, as $|T| > 2$ is required for any transport unit movement. Thus, we apply a weighting factor $\omega \in \mathbb{N}$ ensuring that sequencing time minimization is generally prioritized. Based on equations (A.1) and (A.4), this gives

$$\omega \cdot \left| - \sum_{t \in T} \sum_{b \in B} x_{o^b}^{bt} \right| \overset{!}{\geq} \sum_{t \in T} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} y_{m\acute{m}}^{bt} \tag{A.5}$$

and therefore,

$$\begin{aligned} \omega \cdot |B| &\overset{!}{\geq} \left\lceil \frac{|T|}{2} \right\rceil \cdot |B| \\ \Leftrightarrow \; \omega &\overset{!}{\geq} \left\lceil \frac{|T|}{2} \right\rceil . \end{aligned} \tag{A.6}$$

Setting $\omega = |T|$ generally satisfies equation (A.6) and results in the formulation given in the objective function of the single stage optimization model (7.2).

A.2 Iterative Segmented Optimization Model

The multi-criteria objective function (7.16) of the iterative segmented optimization model (cf. Section 7.1.3) minimizes in each iteration

- the pre-sequenced arrangement of the system at the end of the given time frame and
- the required movements to reach the corresponding positions of the transport units.

The main objective relates to pre-sequencing using a minimum number of movements as secondary objective. Therefore, we need to ensure that the amount of pre-sequencing within the objective function value always exceeds that of the necessary movements. This implies

$$\sum_{b \in B} z^{b\left(t_0^k + (|T_k|-1)\cdot \Delta t\right)} \overset{!}{\geq} \sum_{t \in T_k} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} \left(t - t_0^k\right) \cdot y_{m\acute{m}}^{bt}. \tag{A.7}$$

We specify a lower and upper bound for the terms of pre-sequencing and movement minimization, respectively, to generally satisfy equation (A.7).

Based on constraints (7.17), (7.18) and (7.22), $\sum_{b \in B} z^{bt} = 0$ holds if and only if at time t all transport units of set B are positioned at their assigned output modules. Thus, a valid lower bound for the pre-sequencing term is given by

$$\sum_{b \in B} z^{b\left(t_0^k + (|T_k|-1)\cdot \Delta t\right)} \geq 0. \tag{A.8}$$

By analogy with equation (A.3), an upper bound of the movement minimization term is based on the the requirements for moving transport units (cf.

constraints (7.9) and (7.10)) while additionally including the time-dependent weighting factor $\left(t - t_0^k\right)$. Let

$$\Psi = \max \left(\sum_{t \in T_k} \sum_{m \in M} \sum_{\acute{m} \in N_m} \left(t - t_0^k\right) \cdot y_{m\acute{m}}^{bt} \right) \tag{A.9}$$

be the maximum possible sum of weighted movements any transport unit is able to perform within the given time frame T_k, then

$$\sum_{t \in T_k} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} \left(t - t_0^k\right) \cdot y_{m\acute{m}}^{bt} \leq \Psi \cdot |B| \tag{A.10}$$

holds. Assuming a transport unit is continuously moving within the given time frame T_k, Ψ results from the sum of every second element within the integer sequence $\{0, \Delta t, 2 \cdot \Delta t, \ldots, (|T_k| - 1) \cdot \Delta t\}$. Thus,

$$\Psi = \begin{cases} \Psi_{even} & \text{if } |T_k| \bmod 2 = 1 \\ \Psi_{odd} & \text{if } |T_k| \bmod 2 = 0, \end{cases} \tag{A.11}$$

where

$$\begin{aligned} \Psi_{even} &= \sum_{i=0}^{\left\lfloor \frac{|T_k|-1}{2} \right\rfloor} 2 \cdot i \cdot \Delta t \\ &= 2 \cdot \Delta t \cdot \sum_{i=0}^{\left\lfloor \frac{|T_k|-1}{2} \right\rfloor} i \end{aligned} \tag{A.12}$$

represents the case of summing all even-indexed elements $2 \cdot i \cdot \Delta t$ and

$$\begin{aligned} \Psi_{odd} &= \sum_{i=0}^{\left\lceil \frac{|T_k|-1}{2} \right\rceil - 1} (2 \cdot i + 1) \cdot \Delta t \\ &= 2 \cdot \Delta t \cdot \sum_{i=0}^{\left\lceil \frac{|T_k|-1}{2} \right\rceil - 1} i + \left\lceil \frac{|T_k|-1}{2} \right\rceil \cdot \Delta t \end{aligned} \tag{A.13}$$

represents the case of summing all odd-indexed elements $(2 \cdot i + 1) \cdot \Delta t$.

Using the Gaussian summation theorem,

$$\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}, \tag{A.14}$$

yields

$$\Psi_{even} = \left\lfloor \frac{|T_k|-1}{2} \right\rfloor \cdot \left(\left\lfloor \frac{|T_k|-1}{2} \right\rfloor + 1 \right) \cdot \Delta t \tag{A.15}$$

and

$$\Psi_{odd} = \left(\left\lceil \frac{|T_k|-1}{2} \right\rceil - 1 \right) \cdot \left\lceil \frac{|T_k|-1}{2} \right\rceil \cdot \Delta t + \left\lceil \frac{|T_k|-1}{2} \right\rceil \cdot \Delta t. \tag{A.16}$$

With $\left\lfloor \frac{|T_k|-1}{2} \right\rfloor \leq \frac{|T_k|-1}{2}$ and $\left\lceil \frac{|T_k|-1}{2} \right\rceil \leq \frac{|T_k|-1}{2} + 1$, we obtain

$$\Psi_{even} \leq \frac{|T_k|-1}{2} \cdot \left(\frac{|T_k|-1}{2} + 1 \right) \cdot \Delta t \tag{A.17}$$

and

$$\Psi_{odd} \leq \frac{|T_k|-1}{2} \cdot \left(\frac{|T_k|-1}{2} + 1 \right) \cdot \Delta t + \left(\frac{|T_k|-1}{2} + 1 \right) \cdot \Delta t. \tag{A.18}$$

This gives

$$\Psi \leq \left(\frac{\left(|T_k| - 1\right)^2}{4} + |T_k| \right) \cdot \Delta t. \tag{A.19}$$

Therefore, we can deduce an upper bound of the term of movement minimization as

$$\sum_{t \in T_k} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} \left(t - t_0^k\right) \cdot y_{m\acute{m}}^{bt} \leq \left(\frac{\left(|T_k| - 1\right)^2}{4} + |T_k| \right) \cdot \Delta t \cdot |B| \,. \tag{A.20}$$

From equations (A.7), (A.8) and (A.20) follows

$$0 \overset{!}{\geq} \left(\frac{\left(|T_k| - 1\right)^2}{4} + |T_k| \right) \cdot \Delta t \cdot |B| \tag{A.21}$$

which is a contradiction. To generally satisfy equation (A.7), the pre-sequencing term of the objective function additionally requires a weighting factor $\omega \in \mathbb{N}$ as well as a constant $\varsigma \in \mathbb{N}$ such that equation (A.7) becomes

$$\omega \cdot \sum_{b \in B} \left(z^{b\left(t_0^k + (|T_k| - 1) \cdot \Delta t\right)} + \varsigma \right) \overset{!}{\geq} \sum_{t \in T_k} \sum_{b \in B} \sum_{m \in M} \sum_{\acute{m} \in N_m} \left(t - t_0^k\right) \cdot y_{m\acute{m}}^{bt}. \tag{A.22}$$

Using the upper and lower bound of equations (A.8) and (A.20) yields

$$\begin{aligned} \omega \cdot \varsigma \cdot |B| &\overset{!}{\geq} \left(\frac{\left(|T_k| - 1\right)^2}{4} + |T_k| \right) \cdot \Delta t \cdot |B| \\ \Leftrightarrow \; \omega \cdot \varsigma &\overset{!}{\geq} \left(\frac{\left(|T_k| - 1\right)^2}{4} + |T_k| \right) \cdot \Delta t \end{aligned} \tag{A.23}$$

As $|T_k| \neq \emptyset$, setting $\varsigma = \Delta t$ and $\omega = |T_k|^2$ generally satisfies equation (A.23) and results in the formulation given in the objective function of the iterative segmented optimization model (7.16).

B Simulation Study

As associated terminology is not used unambiguously in the literature (cf. Lorig 2019, Law 2015, Lawson 2015), we define the following terms to describe our simulative evaluations. A *simulation study* comprises different parameter settings to deduce the influence of relevant input parameters on system behavior. Each *parameter setting* represents a set of input parameters for which the system is simulated to evaluate specified performance indicators. Investigating stochastic systems requires several *replications* of a simulation run with different random seeds per parameter setting to obtain reliable results. A *simulation run* captures system behavior during an uninterrupted period of time. All replications referring to the same parameter setting are aggregated to a *simulation experiment*. The average values of all performance indicators of all replications of a simulation experiment represent the *simulation result* for the corresponding parameter setting. The *simulation time* is measured in seconds, i.e., recording system behavior occurs at each integer multiple of 1 s.

Within the simulation studies of Chapter 8, we aim to achieve practically relevant results relating to stable system operation. We determine the warm-up period and stopping criteria for each simulation run as described in Sections B.1 and B.2, respectively. The number of replications per parameter setting follows from the requirements given in Section B.3.

B.1 Warm-up Period

Every simulation run suffers from the problem of initial transient behavior, as it starts at time $t = 0$ with an empty system. Recorded performance indicators at the beginning of the simulation run do not necessarily represent the expected system behavior during operation. The average value of a performance indicator based on the total run time of the simulation run corresponds to a biased estimator of its value in steady-state. The warm-up period of a simulation run is defined as the duration until it reaches a steady-state behavior. Therefore, all observations obtained during the warm-up period are neglected such that performance indicators are recorded based on the remaining steady-state observations. (Law 2015, p. 511f.)

Based on the extensive comparisons and recommendations of methods for determining the end of the warm-up period given in Hoad et al. (2010), we use the Marginal Standard Error Rule (MSER) introduced by White (1997). It aims to select the end of the warm-up period – the so-called truncation point τ – minimizing the length of the marginal confidence interval about the truncated sample mean. White et al. (2000) discuss a variant of MSER, which is called MSER-k, used more commonly as it relies on smoother data.

To determine the truncation point of each simulation run within our simulation studies, we refer to the absolute system throughput per second, as system throughput represents the key performance indicator of the sequencing system. Let t be the duration of a simulation run in seconds generating the set of $n = \lfloor t \rfloor$ throughput observations Γ, where γ_i represents the i-th of the n observations. For smoothing these observations, we aggregate each k successive observations which yields

$$\tilde{\gamma}_j = \frac{\sum_{i=1}^{k} \gamma_{(j-1)k+i}}{k} \qquad j = 1, 2, \dots \left\lfloor \frac{n}{k} \right\rfloor . \tag{B.1}$$

With $m = \left\lfloor \frac{n}{k} \right\rfloor$,

$$\tilde{\Gamma}(m, \theta) = \frac{\sum_{j=\theta+1}^{m} \tilde{\gamma}_j}{m - \theta} \tag{B.2}$$

represents the average of the last $(m - \theta)$ k-aggregated observations. Based on MSER-k, the truncation point τ is defined as

$$\tau = \underset{0 \leq \theta < m}{\arg\min} \left\{ \frac{1}{(m - \theta)^2} \cdot \sum_{j=\theta+1}^{m} \left(\tilde{\gamma}_j - \tilde{\Gamma}(m, \theta) \right)^2 \right\}, \tag{B.3}$$

where a valid truncation point is given if

$$\tau > \frac{m}{2} \tag{B.4}$$

holds (Hoad and Robinson 2011). The k-aggregated observations $1, \ldots, \tau$ are associated with the warm-up period, while the remaining ones $(\tau + 1), \ldots, m$ refer to the system in the steady-state. Thus, recording performance indicators starts at a simulation time of $(\tau + 1) \cdot k$ seconds. Based on White et al. (2000), k was consistently set to 5 when determining the warm-up period within all simulation runs of this dissertation.

B.2 Stopping Criterion

The stopping criterion determines the length of a simulation run for each replication ι. Within a defined minimum and maximum simulation duration t^s_{min} and t^s_{max}, we aim to terminate the simulation run as soon as the obtained results are stable. We calculate an admissible deviation range $\mathcal{I}_{\varepsilon^r}$ using the mean system throughput of the overall simulation run time in steady-state. The stopping criterion is satisfied if all observations of the currently last time interval of length ν of the simulation run are included in this range.

During simulation, the key figures of the system are steadily recorded. Let Γ_τ^ι be the set of current throughput observations of simulation replication ι in steady-state, i.e., after the truncation point τ. This gives $n_\tau^\iota = |\Gamma_\tau^\iota|$ steady-state throughput observations γ_τ^ι within this replication. Based on these currently available observations, we obtain the throughput result $\bar{\gamma}^\iota$ of replication ι as

$$\bar{\gamma}^\iota = \frac{\sum_{\gamma_\tau^\iota \in \Gamma_\tau^\iota} \gamma_\tau^\iota}{n_\tau^\iota}. \tag{B.5}$$

The admissible deviation range is specified by the required precision ε^r of a simulation replication, i.e.,

$$\mathcal{I}_{\varepsilon^r} = \left[\bar{\gamma}^\iota \cdot (1 - \varepsilon^r)\,; \bar{\gamma}^\iota \cdot (1 + \varepsilon^r)\right]. \tag{B.6}$$

Replication ι stops at a current simulation time t^ι with $t^s_{min} < t^\iota < t^s_{max}$ if, retrospectively, the absolute throughput result of all observations recorded within the simulation time from $(t^\iota - \nu)$ to t^ι respects the admissible deviation range.

We consistently set $\nu = 100\,s$, $t^s_{min} = 1,000\,s$, $t^s_{max} = 20,000\,s$, and $\varepsilon^r = 1‰$ within all simulation runs of this dissertation.

B.3 Number of Replications

We generate a set of seeds to initialize the pseudorandom number generators used in all simulation experiments of this dissertation. Thus, the replications of a simulation experiment are independent and identically distributed, as each simulation run relies on a new random seed (Law 2015, p. 489). We determine the number of replications $\mathcal{N}$ for a simulation experiment using the Replication/Deletion Approach (cf. Law 2015, p. 523f.). Additionally, we set a minimum and maximum number of replications $\mathcal{N}_{min}$ and $\mathcal{N}_{max}$.

During a simulation experiment, the results of the single replications differ due to stochastic parametrization. This variation can be described by the $(1-\alpha)$-confidence interval $\mathcal{K}_{\acute{\iota}}$ where $\acute{\iota}$ denotes the number of currently simulated replications. Let $t_{(\acute{\iota}-1),(1-\frac{\alpha}{2})}$ be the $(1-\frac{\alpha}{2})$-quantile of t-distribution with $(\acute{\iota}-1)$ degrees of freedom, then $\mathcal{K}_{\acute{\iota}}$ follows as

$$\mathcal{K}_{\acute{\iota}} = \left[\Phi_{\acute{\iota}} - t_{(\acute{\iota}-1),\left(1-\frac{\alpha}{2}\right)} \cdot \frac{\mathcal{S}_{\acute{\iota}}}{\sqrt{\acute{\iota}}} \; ; \; \Phi_{\acute{\iota}} + t_{(\acute{\iota}-1),\left(1-\frac{\alpha}{2}\right)} \cdot \frac{\mathcal{S}_{\acute{\iota}}}{\sqrt{\acute{\iota}}} \right], \tag{B.7}$$

where $\Phi_{\acute{\iota}}$ denotes the mean and $\mathcal{S}_{\acute{\iota}}$ the standard deviation of the throughput results of replications $1 \ldots \acute{\iota}$ with

$$\Phi_{\acute{\iota}} = \frac{1}{\acute{\iota}} \cdot \sum_{j=1}^{\acute{\iota}} \bar{\gamma}^j \tag{B.8}$$

and

$$\mathcal{S}_{\acute{\iota}} = \sqrt{\frac{1}{\acute{\iota}-1} \cdot \sum_{j=1}^{\acute{\iota}} \left(\bar{\gamma}^j - \Phi_{\acute{\iota}}\right)^2}. \tag{B.9}$$

The simulation experiment terminates with a number of sufficient replications $\mathcal{N} \in \mathcal{N}_{min} \ldots \mathcal{N}_{max} - 1$ if the 95 %-confidence interval $\mathcal{K}_{\mathcal{N}}$ related to the mean throughput result $\Phi_{\mathcal{N}}$ satisfies a specified precision ε^e, i.e.,

$$\frac{t_{(\mathcal{N}-1),\left(1-\frac{\alpha}{2}\right)} \cdot \frac{\mathcal{S}_{\mathcal{N}}}{\sqrt{\mathcal{N}}}}{\Phi_{\mathcal{N}}} < \varepsilon^e. \tag{B.10}$$

Otherwise $\mathcal{N}$ equals $\mathcal{N}_{max}$.

We consistently set $\alpha = 0.05$, $\varepsilon^e = 0.025$, $\mathcal{N}_{min} = 10$, and $\mathcal{N}_{max} = 30$ when determining the number of replications of a simulation experiment within this dissertation.

Glossary of Notation

Sequencing Network

C	Set of sequencing nodes/modules
$d^*_{m\acute{m}}$	Length of a shortest path of modules from module m to module $\acute{m}$
$d^{\vec{n}}_{m\acute{m}}$	Directional distance from module m to module $\acute{m}$ via the adjacent module n
δ	Average density
E	Set of edges
e	Single edge
G	Graph
I	Set of input modules
i	Single input module
i^b	Input module transport unit b is assigned to
k	Connectivity of a network
κ_v	Capacity of node v
l_i	Loading delay at input module i
l_o	Unloading delay at output module o

λ	System arrival rate
λ^{γ}	System arrival rate of limiting throughput
λ_i	Arrival rate at input module i
$\check{\lambda}$	Delayed system arrival rate
$\check{\lambda}_i$	Delayed arrival rate at input module i
M	Set of conveyor modules
m	Single conveyor module
μ	System processing rate
$m : n$	Multiple input to multiple output relation
N_m	Set of modules adjacent to module m
$E(n^b)$	Expected number of transport units within the system
F	Number of movements
F_{dec}	Decentralized number of movements
F^*	Optimal number of movements
H	Sequencing time of overall problem
H_{dec}	Decentralized sequencing time of overall problem
H_{LB}	Lower bound of sequencing time of overall problem
H^*	Minimum sequencing time of overall problem
H^*_{real}	Minimum actual sequencing time of overall problem
O	Set of output modules
o	Single output module
o^b	Output module transport unit b is assigned to

$\mathcal{O}^m$	Order of algorithmic complexity at module level
$\mathcal{O}^n$	Order of algorithmic complexity at network level
φ_v	Space provided by a node
φ_o^u	Space required by physical object u
ρ	System utilization
t_{conv}	Transfer time of a transport unit between two adjacent modules
t_{msg}^{ij}	Message sending delay between module i and j
$t_{msg}^{m\acute{n}}$	Message sending delay between two adjacent modules
t_{switch}	Switching time of modules between orthogonal conveying directions
$E(t_a)$	Expected system interarrival time
$E(t_{a_i})$	Expected interarrival time at input module i
$E(t_d)$	Expected delayed system interarrival time
$E(t_{d_i})$	Expected delayed interarrival time at input module i
$E(t_p)$	Expected system processing time
ϑ	Positive integer greater than 1
U	Set of physical objects
U_r	Set of released physical objects
u	Single physical object
V	Set of nodes
V_e	Set of unoccupied nodes
v	Single node

w	Edge weight function

Physical Objects

a^b	Arrival time of transport unit b at the system
B	Set of transport units
B_i	Set of transport units arriving at input module i
B_o	Set of transport units unloaded at output module o
B^{eq}	Set of equally ranked transport units
$b^{eq^{(i)}}$	Ordering of equally ranked transport units
b	Single transport unit
e^b	Introducing time of transport unit b
e^b_{min}	Minimum introducing time of transport unit b
f^b_i	Direct successor of transport unit b within the queue of arriving transport units at input module i
k	Batch size
h^b	Sequencing time of transport unit b
h^{b^*}	Optimal sequencing time of transport unit b
h^b_{LB}	Lower bound of the sequencing time of transport unit b
P^b_o	Set of direct predecessors of transport unit b within the unloading sequence at output module o
p^b	Direct predecessor of transport unit b within its batch
π^b_i	Position of transport unit b within the queue of arriving transport units at input module i

Q_i^b	Set of predecessors of transport unit b within the queue of arriving transport units at input module i
q_i^b	Direct predecessor of transport unit b within the queue of arriving transport units at input module i
r^b	Rank of transport unit b within its batch
ϱ	$[0, 1)$-uniformly distributed random number
s	Sequencing rate
s_o^b	Direct successor of transport units b within the unloading sequence at output module o
σ^b	Position of transport unit b within the arrival sequence of its batch
t_{min}^b	Minimum transfer time of transport unit b from its input to its output module
t_p^b	Processing time of transport unit b
$E(t_p^b)$	Expected processing time of transport unit b
t_s^b	Sojourn time of a transport unit b
t_w^b	Waiting time of transport unit b
$t_{w^e}^b$	Waiting time of transport unit b on its input module
$E\left(t_{w^e}^b\right)$	Expected waiting time of transport unit b on its input module
$t_{w^q}^b$	Waiting time of transport unit b in front of its input module
u^b	Unloading time of transport unit b
u_{min}^b	Minimum unloading time of transport unit b

Decentralized Algorithm

C_i	Logical clock of module i
C^q	Logical clock of process q
$C^q\langle e_i \rangle$	Logical time of event e_i within process q
$\tilde{c}_m$	Estimated path cost of module m
$\tilde{c}^p$	Estimated cost for (sub-)path p
e_i	Single event
ϵ	Small positive number
d^p	Distance of (sub-)path p
n_b^p	Number of buffer modules on (sub-)path p
n_c^p	Number of directional changes on (sub-)path p
P_a	Process of transport unit a
p_b	Buffer penalty
p_c	Directional change penalty
p^*	Sub-path of an optimal solution
$\acute{p}$	Sub-path of a non-optimal solution
q	Single process
R_i	Resource, i.e., conveyor module i
s^a	Successor process of process P_a within their unloading sequence
T_{in}	Incoming transfer
T_{out}	Outgoing transfer

$\rightarrow$	Happened before relation

Complexity Analysis

$\mathcal{C}$	Number of messages for a sequencing problem
$\mathcal{C}_a$	Average number of messages per active route
$\mathcal{C}_a^E$	Average number of messages per active route for execution
$\mathcal{C}_a^I$	Average number of messages per active route for initiation
$\mathcal{C}_a^P$	Average number of messages per active route for planning
$\mathcal{C}_b$	Average number of messages per transport unit
$\mathcal{C}_m$	Messaging effort at module level
$\hat{\mathcal{C}}$	Upper bound of the messaging effort for a sequencing problem
$\hat{\mathcal{C}}_a$	Upper bound of the messaging effort per active route
$\hat{\mathcal{C}}_a^A$	Upper bound of the messaging effort per active route for authorization
$\hat{\mathcal{C}}_a^B$	Upper bound of the messaging effort per active route for identifying non-buffering modules
$\hat{\mathcal{C}}_a^E$	Upper bound of the messaging effort per active route for transport execution
$\hat{\mathcal{C}}_a^I$	Upper bound of the messaging effort per active route for initiation
$\hat{\mathcal{C}}_a^P$	Upper bound of the messaging effort per active route for planning
$\hat{\mathcal{C}}_a^{R_a}$	Upper bound of the messaging effort per active route for negotiating reservations

$\hat{\mathcal{C}}_a^{R_r}$	Upper bound of the messaging effort per active route for planning relocations
$\hat{\mathcal{C}}_a^{S}$	Upper bound of the messaging effort per active route for path selection
$\hat{\mathcal{C}}_a^{U}$	Upper bound of the messaging effort per active route for updating output modules
n	System size
ξ	Positive linear scaling factor
ξ^X	Positive linear scaling factor for an algorithmic component

Optimization Models

K	Set of iterations
k	Single iteration
L	Large positive number
l	Time split factor
ω	Positive weighting factor
Ψ	Upper bound of weighted movement term per transport unit
Ψ_{even}	Upper bound of weighted movement term per transport unit restricted to even time points
Ψ_{odd}	Upper bound of weighted movement term per transport unit restricted to odd time points
ς	Positive constant
T	Set of time points of an overall sequencing problem

T_k	Set of time points for iteration k
t	Single time point
t_0^k	Initial time point of iteration k
Δt	Incremental length of successive time points
W_{it}	Objective function of the iterative segmented optimization model
W_{ss}	Objective function of the single stage optimization model
X_0^k	Set of non-zero x-variables defining the system arrangement iteration k starts from
x_m^{bt}	Binary decision variable defining the occupation of module m by transport unit b at time t
Y_0^k	Set of non-zero y-variables defining the system arrangement iteration k starts from
$y_{m\acute{m}}^{bt}$	Binary decision variable defining the movement of transport unit b from module m to an adjacent module $\acute{m}$ at time t
z^{bt}	Positive continuous decision variable defining the pre-sequenced arrangement of transport unit b at time t

Simulation Study

α	Confidence level
ε^e	Precision of a simulation experiment
ε^r	Precision of a replication
Γ	Set of throughput observations of a simulation run
Γ_τ^ι	Set of current steady-state throughput observations of replication ι

$\tilde{\Gamma}\,(m,\tau)$	Average of the last $(m-\tau)$ k-aggregated throughput observations
γ_i	i-th throughput observation
γ_τ^ι	Single steady-state throughput observation of replication ι
$\bar{\gamma}^\iota$	Throughput result of replication ι
$\tilde{\gamma}_j$	j-th k-aggregated throughput observation
$\mathcal{I}_{\varepsilon^r}$	Admissible deviation range
ι	Single simulation replication
$\acute{\iota}$	Number of simulated replications
$\mathcal{K}_{\acute{\iota}}$	95 %-confidence interval of throughput results at replication ι
k	Smoothing factor
m	Number of k-aggregated throughput observations
$\mathcal{N}$	Number of replications of a simulation experiment
$\mathcal{N}_{min}$	Minimum number of replications of a simulation experiment
$\mathcal{N}_{max}$	Maximum number of replications of a simulation experiment
n	Number of throughput observations of a simulation run
n_τ^ι	Number of steady-state throughput observations of replication ι
ν	Time duration of deviation range
$\Phi_{\acute{\iota}}$	Mean throughput result at replication $\acute{\iota}$
$\mathcal{S}_{\acute{\iota}}$	Standard deviation of throughput result at replication $\acute{\iota}$
t	Duration of a simulation run
t^ι	Current simulation time of replication ι
t_{min}^s	Minimum simulation duration

t^s_{max}	Maximum simulation duration
$t_{(i-1),(1-\frac{\alpha}{2})}$	$(1-\frac{\alpha}{2})$-quantile of t-distribution with $(i-1)$ degrees of freedom
τ	Truncation point
θ	Number of k-aggregated throughput observations within warm-up period

List of Figures

List of Tables

List of Publications

Fleischmann, J. and K. Furmans (2023). Sequencing with Decentralized Control using Modular Highest-Density Conveyor Systems. *IEEE Transactions on Automation Science and Engineering*, p. 1–11.

References

Alahmad, R. and K. Ishii (2021). A Puzzle-Based Sequencing System for Logistics Items. *Logistics 5*(4), p. 1–18.

Archer, A. (2007). The 15 Puzzle: How it Drove the World Crazy. *The Mathematical Intelligencer 29*(2), p. 83–85.

Archer, A. F. (1999). A Modern Treatment of the 15 Puzzle. *The American Mathematical Monthly 106*(9), p. 793–799.

Arnold, D. and K. Furmans (2019). *Materialfluss in Logistiksystemen* (7. ed.). Berlin: Springer Vieweg.

Arora, S. and B. Barak (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press.

Azadeh, K., R. de Koster and D. Roy (2019). Robotized and Automated Warehouse Systems: Review and Recent Developments. *Transportation Science 53*(4), p. 917–945.

Boysen, N., D. Briskorn, S. Fedtke and M. Schmickerath (2019). Automated sortation conveyors: A survey from an operational research perspective. *European Journal of Operational Research 276*(3), p. 796–815.

Boysen, N., S. Emde, M. Hoeck and M. Kauderer (2015). Part logistics in the automotive industry: Decision problems, literature review and research agenda. *European Journal of Operational Research 242*(1), p. 107–120.

Boysen, N., M. Fliedner and A. Scholl (2009). Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research 192*(2), p. 349–373.

Brandes, U. and T. Erlebach (2005). Fundamentals. In: U. Brandes and T. Erlebach (eds.), *Network Analysis: Methodological Foundations*, Lecture Notes in Computer Science, p. 7–15. Berlin/Heidelberg: Springer.

Buakum, D. and W. Wisittipanich (2019). A Literature Review and Further Research Direction in Cross-docking. In: *Proceedings of the International Conference on Industrial Engineering and Operations Management*, Bangkok, Thailand, p. 471–481.

Bukchin, Y. and T. Raviv (2020). Optimal retrieval in puzzle-based storage systems with simultaneous load and block movement. *Preprint 10.13140/RG.2.2.29406.05442*.

Bukchin, Y. and T. Raviv (2022). A comprehensive toolbox for load retrieval in puzzle-based storage systems with simultaneous movements. *Transportation Research Part B: Methodological 166*, p. 348–373.

Chikán, A., E. Kovács, Z. Matyusz, M. Sass and P. Vakhal (2016). Long-term trends in inventory investment in traditional market and post-socialist economies. *International Journal of Production Economics 181*, p. 14–23.

Coffman, E. G., M. J. Elphick and A. Shoshani (1971). System Deadlocks. *ACM Computing Surveys 3*(2), p. 67–78.

Colling, D., Z. Seibold and K. Furmans (2016). GridSorter – Dezentral gesteuertes Materialflusssystem zum Transport unterschiedlich großer Ladungsträger. *Logistics Journal : Proceedings 2016*(5), p. 1–8.

Dayıoğlu, E. G., K. Karagül, Y. Şahin and M. G. Kay (2020). Route planning methods for a modular warehouse system. *International Journal of Optimization and Control: Theories & Applications 10*(1), p. 17–25.

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.

de Ryck, M., M. Versteyhe and F. Debrouwere (2020). Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems 54*, p. 152–173.

Diestel, R. (2017). *Graph Theory* (5. ed.), Volume 173 of *Graduate Texts in Mathematics*. Berlin/Heidelberg: Springer.

Dilts, D. M., N. P. Boyd and H. H. Whorms (1991). The Evolution of Control Architectures for Automated Manufacturing Systems. *Journal of Manufacturing Systems 10*(1), p. 79–93.

Duffie, N. A. (1990). Synthesis of Heterarchical Manufacturing Systems. *Computers in Industry 14*(1-3), p. 167–174.

Duffie, N. A., R. Chitturi and J.-I. Mou (1988). Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities. *Journal of Manufacturing Systems 7*(4), p. 315–328.

Erciyes, K. (2013). *Distributed Graph Algorithms for Computer Networks*. Computer Communications and Networks. London: Springer.

Firvida, M. B., H. Thamer, C. Uriarte and M. Freitag (2018). Decentralized omnidirectional route planning and reservation for highly flexible material flow systems with small-scaled conveyor modules. In: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Turin, p. 685–692. IEEE.

Fleischmann, J. (2023). *Research data to "Multi-Batch Sequencing with Decentralized Control using Highest-Density Conveyor Networks"*. Repository KITopen.

Fleischmann, J. and K. Furmans (2023). Sequencing with Decentralized Control using Modular Highest-Density Conveyor Systems. *IEEE Transactions on Automation Science and Engineering*, p. 1–11.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM 5*(6), p. 345.

Fottner, J., D. Clauer, F. Hormes, M. Freitag, T. Beinke, L. Overmeyer, S. N. Gottwald, R. Elbert, T. Sarnow, T. Schmidt, K.-B. Reith, H. Zadek and F. Thomas (2021). Autonomous Systems in Intralogistics - State of the Art and Future Research Challenges. *Logistics Research 14*(2), p. 1–41.

Furmans, K., C. Nobbe and M. Schwab (2011). Future of Material Handling – modular, flexible and efficient. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Furmans, K., F. Schonung and K. Gue (2010). Plug-and-Work Material Handling Systems. In: *11th IMHRC Proceedings*, Milwaukee, Wisconsin.

Furmans, K., Z. Seibold and A. Trenkle (2019). Future Technologies in Intralogistics and Material Handling. In: H. Zijm, M. Klumpp, A. Regattieri, and S. Heragu (eds.), *Operations, Logistics and Supply Chain Management*, Lecture Notes in Logistics, p. 545–574. Cham: Springer.

Gagliardi, J.-P., J. Renaud and A. Ruiz (2014). On sequencing policies for unit-load automated storage and retrieval systems. *International Journal of Production Research 52*(4), p. 1090–1099.

Gravel, M., C. Gagné and W. L. Price (2005). Review and comparison of three methods for the solution of the car sequencing problem. *Journal of the Operational Research Society 56*(11), p. 1287–1295.

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.

Gue, K. (2016). A High-Density, Puzzle-Based System for Rail-Rail Container Transfers. In: *14th IMHRC Proceedings*, Volume 14, Karlsruhe.

Gue, K. R. (2006). Very high density storage systems. *IIE Transactions 38*(1), p. 79–90.

Gue, K. R., K. Furmans, Z. Seibold and O. Uludağ (2014). GridStore: A Puzzle-Based Storage System With Decentralized Control. *IEEE Transactions on Automation Science and Engineering 11*(2), p. 429–438.

Gue, K. R. and B. S. Kim (2007). Puzzle-Based Storage Systems. *Naval Research Logistics 54*(5), p. 556–567.

Gue, K. R. and O. Uludağ (2012). A High-Density, Puzzle-Based Order Picking System. In: *12th IMHRC Proceedings*, Volume 19, Gardanne.

Gue, K. R., O. Uludağ and K. Furmans (2012). A High-Density System for Carton Sequencing. In: *Proceedings of the international material handling research colloquium.*

Günthner, W. and M. ten Hompel (2010). *Internet der Dinge in der Intralogistik*. Heidelberg: Springer.

Günthner, W. A., R. Kraul, P. Tenerowicz, R. Chisu and F. Kuzmany (2008). Vom Prozess zum Ereignis - ein neuer Denkansatz in der Logistik. In: *Jahrbuch Logistik*.

Hao, G. (2020). *GridHub: a grid-based, high-density material handling system.* Electronic Theses and Dissertations. Paper 3408, University of Louisville, Louisville, Kentucky.

Hart, P. E., N. J. Nilsson and B. Raphael (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics 4*(2), p. 100–107.

Hoad, K. and S. Robinson (2011). Implementing MSER-5 in commercial simulation software and its wider implications. In: S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu (eds.), *Proceedings of the 2011 Winter Simulation Conference*, Phoenix, Arizona, p. 495–503. IEEE.

Hoad, K., S. Robinson and R. Davies (2010). Automating warm-up length estimation. *Journal of the Operational Research Society 61*(9), p. 1389–1403.

Holzmann, G. J. (1991). *Design and Validation of Computer Protocols*. Prentice Hall Software Series. Englewood Cliffs, New Jersey: Prentice-Hall.

Jayaraman, A., R. Narayanaswamy and A. K. Gunal (1997). A Sortation System Model. In: S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson (eds.), *Proceedings of the 29th Conference on Winter Simulation*, p. 866–871.

Johansen, K., S. Rao and M. Ashourpour (2021). The Role of Automation in Complexities of High-Mix in Low-Volume Production – A Literature Review. *Procedia CIRP 104*, p. 1452–1457.

Johnson, W. W. and W. E. Story (1879). Notes on the "15" Puzzle. *American Journal of Mathematics 2*(4), p. 397–404.

Karlsruhe Institute of Technology: Institute for Material Handling and Logistics (2009). FlexFörderer: Vollständig dezentrales Stetigfördersystem aus baugleichen Einzelmodulen: https://www.ifl.kit.edu/mitarbeiter_flexfoerderer.php. [Online] (Accessed on 13.06.2022).

Kartnig, G., B. Grösel and N. Zrnić (2012). Past, State-of-the-Art and Future of Intralogistics in Relation to Megatrends. *FME Transactions 40*(4), p. 193–200.

Kota, V. R., D. Taylor and K. R. Gue (2010). Retrieval Time Performance in Puzzle-Based Storage Systems. In: A. Johnson and A. Miller (eds.), *Proceedings of the 2010 Industrial Engineering Research Conference*, Norcross.

Kota, V. R., D. Taylor and K. R. Gue (2015). Retrieval time performance in puzzle-based storage systems. *Journal of Manufacturing Technology Management 26*(4), p. 582–602.

Krühn, T., M. Radosavac, N. Shchekutin and L. Overmeyer (2013). Decentralized and Dynamic Routing for a Cognitive Conveyor. In: *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Wollongong, Australia, p. 436–441. IEEE.

Kumar, A. (2007). From mass customization to mass personalization: a strategic transformation. *International Journal of Flexible Manufacturing Systems 19*(4), p. 533–547.

Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM 21*(7), p. 558–565.

Lanza, G., K. Ferdows, S. Kara, D. Mourtzis, G. Schuh, J. Váncza, L. Wang and H.-P. Wiendahl (2019). Global production networks: Design and operation. *CIRP Annals 68*(2), p. 823–841.

Law, A. M. (2015). *Simulation Modeling and Analysis* (5. ed.). New York: McGraw-Hill Education.

Lawson, J. (2015). *Design and Analysis of Experiments with R* (1. ed.). Texts in Statistical Science. Boca Raton, Florida: CRC Press.

Lieberoth-Leden, C., J. Fischer, J. Fottner and B. Vogel-Heuser (2018). Control Architecture and Transport Coordination for Autonomous Logistics Modules in Flexible Automated Material Flow Systems. In: *2018 IEEE 14th International Conference on Automation Science and Engineering*, Munich, Germany, p. 736–743. IEEE.

Lieberoth-Leden, C. and J. Fottner (2018). Deployment of an Distributed Strategic Material Flow Control for Automated Material Flow Systems Consisting of Autonomous Modules. In: *15th IMHRC Proceedings*, Savannah, Georgia, USA.

Lienert, T. and J. Fottner (2018). Routing-based Sequencing Applied to Shuttle Systems. In: *2018 21st International Conference on Intelligent Transportation Systems*, Maui, Hawaii, p. 2949–2954. IEEE.

Lorig, F. (2019). *Hypothesis-Driven Simulation Studies: Assistance for the Systematic Design and Conducting of Computer Simulation Experiments*. Dissertation, Universität Trier, Wiesbaden.

Ma, Y., H. Chen and Y. Yu (2022). An efficient heuristic for minimizing the number of moves for the retrieval of a single item in a puzzle-based storage system with multiple escorts. *European Journal of Operational Research 301*(1), p. 51–56.

Macal, C. and M. North (2014). Introductory tutorial: Agent-based modeling and simulation. In: A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller (eds.), *Proceedings of the 2014 Winter Simulation Conference*, p. 6–20. IEEE.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation 10*(2), p. 144–156.

Martins, L., M. L. R. Varela, N. O. Fernandes, S. Carmo-Silva and J. Machado (2020). Literature review on autonomous production control methods. *Enterprise Information Systems 14*(8), p. 1219–1231.

Mayer, S. (2009). *Development of a completely decentralized control system for modular continuous conveyors*. Dissertation, Universität Karlsruhe, Karlsruhe.

Mayer, S. and K. Furmans (2010). Deadlock prevention in a completely decentralized controlled materials flow systems. *Logistics Research 2*(3), p. 147–158.

Menger, K. (1927). Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae 10*, p. 96–115.

Mirzaei, M., R. B. M. de Koster and N. Zaerpour (2017). Modelling load retrievals in puzzle-based storage systems. *International Journal of Production Research 55*(21), p. 6423–6435.

Monostori, L., P. Valckenaers, A. Dolgui, H. Panetto, M. Brdys and B. C. Csáji (2015). Cooperative Control in Production and Logistics. *Annual Reviews in Control 39*, p. 12–29.

Parberry, I. (2015). A Memory-Efficient Method for Fast Computation of Short 15-Puzzle Solutions. *IEEE Transactions on Computational Intelligence and AI in Games 7*(2), p. 200–203.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems* (5. ed.). Cham: Springer.

Qiu, L., W.-J. Hsu, S.-Y. Huang and H. Wang (2002). Scheduling and routing algorithms for AGVs: a survey. *International Journal of Production Research 40*(3), p. 745–760.

Ramesh, S. V. (2010). *Principles of Operating Systems* (1. ed.). New Delhi: Laxmi Publications.

Rana, S. P. and S. K. Taneja (1988). A Distributed Architecture for Automated Manufacturing Systems. *The International Journal of Advanced Manufacturing Technology 3*(5), p. 81–98.

Ratner, D. and M. Warmuth (1990). The (n2−1)-Puzzle and Related Relocation Problems. *Journal of Symbolic Computation 10*(2), p. 111–137.

Rosenfeld, A. (2022). Optimal retrieval in puzzle–based storage with heuristic search and tabulation. *Networks 79*(3), p. 390–402.

Seibold, Z. (2016). *Logical Time for Decentralized Control of Material Handling Systems*. Dissertation, Karlsruhe Institute of Technology, Karlsruhe.

Seibold, Z., K. Furmans and K. R. Gue (2022). Using Logical Time to Ensure Liveness in Material Handling Systems With Decentralized Control. *IEEE Transactions on Automation Science and Engineering 19*(1), p. 545–552.

Seibold, Z., T. Stoll and K. Furmans (2013). Layout-Optimized Sorting of Goods with Decentralized Controlled Conveying Modules. In: *2013 IEEE International Systems Conference*, Orlando, Florida, p. 628–633. IEEE.

Shekari Ashgzari, M. and K. R. Gue (2021). A puzzle–based material handling system for order picking. *International Transactions in Operational Research 28*(4), p. 1821–1846.

Shiller, Z. (2015). Off-Line and On-Line Trajectory Planning. In: G. Carbone and F. Gomez-Bravo (eds.), *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, Mechanisms and Machine Science, p. 29–62. Cham: Springer.

Sittivijan, P. (2015). *Modular Warehouse Control: Simultaneous Rectilinear Movement of Multiple Objects within a Limited Free Space Environment*. Dissertation, North Carolina State University, Raleigh, North Carolina.

Skiena, S. S. (2008). *The Algorithm Design Manual* (2. ed.). London: Springer.

Sohrt, S. and L. Overmeyer (2020). Decentralized Routing Algorithm with Physical Time Windows for Modular Conveyors. *Logistics Research 13*(8), p. 1–16.

Sohrt, S., Z. Seibold, T. Krühn, L. Prössdorf, L. Overmeyer and K. Furmans (2014). Buffering Algorithms for Modular, Decentralized Controlled Material Handling Systems. In: *1st Symposium on Automated Systems and Technologies*, Berichte aus dem ITA, p. 29–36. Garbsen.

Spath, D., O. Ganschar, S. Gerlach, M. Hämmerle, T. Krause and S. Schlund (2013). *Produktionsarbeit der Zukunft - Industrie 4.0*, Volume 150. Stuttgart: Fraunhofer Verlag.

Spieckermann, S., K. Gutenschwager and S. Voß (2004). A sequential ordering problem in automotive paint shops. *International Journal of Production Research 42*(9), p. 1865–1878.

Tai, K. C. (1994). Definitions and Detection of Deadlock, Livelock, and Starvation in Concurrent Programs. In: *Proceedings of the International Conference on Parallel Processing*, North Carolina, USA, p. 69–72. IEEE.

Tanenbaum, A. S. and H. Bos (2015). *Modern Operating Systems* (4. ed.). Boston, Massachusetts: Pearson Education.

Tarjan, R. E. (1974). A note on finding the bridges of a graph. *Information Processing Letters 2*(6), p. 160–161.

Taylor, D. and K. R. Gue (2008). The Effects of Empty Storage Locations in Puzzle-Based Storage Systems. In: J. Fowler and S. Mason (eds.), *Proceedings of the 2008 Industrial Engineering Research Conference*, Norcross, Georgia, p. 519–523.

Thun, J.-H., R. P. Marble and V. Silveira-Camargos (2007). A Conceptual Framework and Empirical Results of the Risk and Potential of Just In Sequence - A Study of the German Automotive Industry. *Journal of Operations and Logistics 1*(2), p. 1–13.

Trentesaux, D. (2009). Distributed control of production systems. *Engineering Applications of Artificial Intelligence 22*(7), p. 971–978.

Uludağ, O. (2014). *GridPick: A High Density Puzzle Based Order Picking System with Decentralized Control*. Dissertation, Auburn University, Auburn, Alabama.

van Steen, M. and A. S. Tanenbaum (2017). *Distributed Systems* (3. ed.). London: Pearson Education.

Wang, K. (2016). Logistics 4.0 Solution - New Challenges and Opportunities. In: *Proceedings of the 6th International Workshop of Advanced Manufacturing and Automation*, p. 68–74. Atlantis Press.

Wannenwetsch, H. (2014). *Integrierte Materialwirtschaft, Logistik und Beschaffung* (5. ed.). Berlin, Heidelberg: Springer Vieweg.

Warshall, S. (1962). A Theorem on Boolean Matrices. *Journal of the ACM 9*(1), p. 11–12.

White, K. P. (1997). An Effective Truncation Heuristic for Bias Reduction in Simulation Output. *Simulation 69*(6), p. 323–334.

White, K. P., M. J. Cobb and S. C. Spratt (2000). A comparison of five steady-state truncation heuristics for simulation. In: J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick (eds.), *Proceedings of the 2000 Winter Simulation Conference*, Orlando, Florida, USA, p. 755–760. IEEE.

Yalcin, A., A. Koberstein and K.-O. Schocke (2019a). An optimal and a heuristic algorithm for the single-item retrieval problem in puzzle-based storage systems with multiple escorts. *International Journal of Production Research 57*(1), p. 143–165.

Yalcin, A., A. Koberstein and K.-O. Schocke (2019b). Motion and layout planning in a grid-based early baggage storage system: Heuristic algorithms and a simulation study. *OR Spectrum 41*(3), p. 683–725.

Yu, Y., H. Yu, R. de Koster and N. Zaerpour (2017). Optimal algorithm for minimizing retrieval time in puzzle-based storage system with multiple simultaneously movable empty cells. *Working Paper*.

Zaerpour, N., Y. Yu and R. de Koster (2017a). Small is Beautiful: A Framework for Evaluating and Optimizing Live-Cube Compact Storage Systems. *Transportation Science 51*(1), p. 34–51.

Zaerpour, N., Y. Yu and R. B. M. de Koster (2017b). Response time analysis of a live-cube compact storage system with two storage classes. *IISE Transactions 49*(5), p. 461–480.

Zou, Y. and M. Qi (2021). A Heuristic Method for Load Retrievals Route Programming in Puzzle-based Storage Systems. *Preprint arXiv:2102.09274*.