Order-picking workstations for automated warehouses

Citation for published version (APA):
Andriansyah, R. (2011). *Order-picking workstations for automated warehouses*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Mechanical Engineering]. Technische Universiteit Eindhoven. https://doi.org/10.6100/IR715619

DOI:
10.6100/IR715619

Document status and date:
Published: 01/01/2011

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Order-picking workstations for automated warehouses

Ricky Andriansyah

This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the carrying industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

This work has been carried out under the auspices of the Engineering Mechanics research school.

A catalogue record is available from the Eindhoven University of Technology Library

ISBN: 978-90-386-2539-3

Reproduction: Universiteitsdrukkerij Technische Universiteit Eindhoven

Cover
Design by Ricky Andriansyah and Aditya Sonihaya (http://ayamsuhayam.deviantart.com).
Background: storage of automobile spare parts. Image courtesy of Viastore systems.

Order-picking workstations for automated warehouses

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 29 augustus 2011 om 16.00 uur

door

Ricky Andriansyah

geboren te Bengkalis, Indonesië

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. J.E. Rooda
en
prof.dr.ir. I.J.B.F. Adan

Copromotor:
dr.ir. L.F.P. Etman

Karena sesungguhnya sesudah kesulitan itu ada kemudahan.
Sesungguhnya sesudah kesulitan itu ada kemudahan.

So, verily, with every difficulty there is relief.
Verily, with every difficulty there is relief.

Voorwaar, zo komt gemak naast ongemak.
Voorwaar, gemak komt naast ongemak.

Al-Quran, 94:5–6.

Preface

Earning a PhD in the Netherlands. That turns out to be my final answer to the typical question people used to ask me in my senior year of high school, "what would you do in ten years from now?" I have to admit that it was a challenging transition from studying industrial engineering to doing research in systems engineering (which is part of mechanical engineering). Nevertheless, looking back to what I have done to come to this point, I feel a great sense of accomplishment.

This dissertation is yet another milestone and will forever remind me of the great times of doing research. The times when I collaborated with outstanding minds and presented my work in Monopoli, Porto, Baltimore, and Noordwijkerhout. In the course of my research I have received help and support from many people. Now it is my chance to acknowledge their contributions.

First of all I would like to thank Koos Rooda as my first promotor. He was the one who gave me the opportunity to perform research in the Systems Engineering Group and provided a comfortable working environment. Thank you for your excellent supervision and support from the first day I joined your group.

Pascal Etman, my copromotor, for the discussions, advices, support, and constructive criticisms you have provided me with. I have learned so much and I really enjoyed working with you. Thanks to you, I have come a long way in the last four years.

Ivo Adan, my second promotor, for the many valuable discussions that often lead to fresh ideas in difficult times. Particularly his comments and remarks have significantly improve the quality of Chapter 5 of this dissertation.

Roelof Hamberg, Jacques Resing and Liqiang Liu, for the insight and ideas they provided me at the regular meetings we had in ESI or EURANDOM. Roelof also provided valuable comments on this dissertation.

Bruno van Wijngaarden, Toine Ketelaars, and Roy van Putten from Vanderlande Industries. Bruno has provided the necessary data for the case studies in Chapters 3 and 4. Toine has reviewed the confidentiality issues for each of my publication. Roy provided a description of the system addressed in Chapter 5.

The external Doctorate Committee members René de Koster, Kees Jan Roodbergen and Onno Boxma, whose valuable suggestions have improved the quality of this dissertation.

Albert Hofkamp for his help with all programming-related issues in χ 1.0 and Python.

The bachelor and master students whom I have worked with: Richard Jordan, Willem de Koning, Maarten van Maanen, Roel de Natris, Marco Paese, Jorine Heling, and Diederik Stel. Thank you all for your contributions.

Mieke Lousberg and Nicole Palmer, for their personal interest and secretarial support.

All colleagues at the Manufacturing Networks Group and the Systems Engineering Group for the pleasant working environment they have created. Special thanks go to my former office mates: Ad, Casper, Dirk, Maarten, Michiel, Qin, Remco, and Simon. Thank you guys for - among others - not letting me speak in English anymore since my first day in the basement, the crash course of Dutch proverbs, and the endless insanity that kept me alert all day.

My families in Sunter and Duren Sawit, at 5 hours time difference away from here, for their unconditional support and prayers. Knowing that everything is okay back there always means a lot to me.

Nolan, it's unbelievable how you could recharge my energy by that contagious smile of yours. This is as far as your dad gets as he turns 28 years old. And it's merely one example of the many things you can achieve in your days to come.

And finally, the one that makes it all complete. She who always believes that I can pull this off. She who understands me inside out. For my lovely wife, Dina Wiyasti, no words will suffice. Dina, you are simply wonderful. Thank you for everything.

They say time flies when you're having fun. And indeed, the last four years passed in a glimpse.

Ricky Andriansyah

Eindhoven, August 2011

Summary

Order-picking workstations for automated warehouses

The FALCON (Flexible Automated Logistic CONcept) project aims at the development of a new generation of warehouses and distribution centers with a maximum degree of automation. As part of the FALCON project, this dissertation addresses the design and analysis of (automated) workstations in warehouses with an end-of-aisle order-picking system (OPS). Methods are proposed for architecting, quantifying performance, and controlling such a system. Four main topics are discussed in this dissertation.

First, a modular architecture for an end-of-aisle OPS with remotely located workstations is presented. This architecture is structured into areas and operational layers. A hierarchical decentralized control structure is applied. A case of an industrial-scale distribution center is presented to demonstrate the applicability of the proposed architecture for performance analysis using the process algebra-based simulation language χ (Chi). Additionally, it is demonstrated how the architecture allows straightforward modification of the systems configurations, design parameters, and control heuristics.

Second, a method to quantify the operational performance of order-picking workstations has been developed. The method is based on an aggregate modeling representation of the workstation using the EPT (Effective Process Time) concept. A workstation is considered in which a human picker is present to process one customer order at a time while products for multiple orders arrive simultaneously at the workstation. The EPT parameters are calculated from arrival and departure times of products using a sample path equation. Two model variants have been developed, namely for workstations with FCFS (First-Come-First-Serve) and for workstations with non-FCFS processing of products and orders. Both models have been validated using data from a real, operating workstation. The results show that the proposed aggregate modeling methodology gives good accuracy in predicting product and order flow time distributions.

Third, the dissertation studies the design and control of an automated, remotely located order-picking workstation that is capable of processing multiple orders simultaneously. Products for multiple orders typically arrive out-of-sequence at the

workstation as they are retrieved from dispersed locations in the storage area. The design problem concerns the structuring of product/order buffer lanes and the development of a mechanism that overcomes out-of-sequence arrivals of products. The control problem concerns the picking sequence at the workstation, as throughput deteriorates when a poor picking sequence is applied. An efficient control policy has been developed. Its performance is compared to a number of other picking policies including nearest-to-the-head, nearest neighbor, and dynamic programming. Subsequently, the resulting throughput and queue length distribution are evaluated under different settings. Insights for design considerations of such a system are summarized.

Finally, the dissertation reflects on the findings from the proposed methods and uses them to come up with comprehensive design principles of end-of-aisle OPS with remotely located workstations. The various issues influencing the performance of such a system are highlighted. Moreover, the contribution of each proposed method with regards to these issues is delineated.

Contents

1 Introduction

Abstract | This dissertation addresses performance analysis of order-picking workstations for automated warehouses. The first chapter provides a general overview of warehousing and order-picking systems (OPS). A classification of OPS is discussed. Literature regarding end-of-aisle OPS, which is the type of system considered in this dissertation, is reviewed in greater detail. Typical issues for this system are summarized and existing performance analysis methods are elaborated. Based on this review, the research objectives are formulated. The research is performed within the context of the FALCON project.

1.1 Warehousing

Warehouses are indispensable entities of modern supply chains. They typically form the critical link between the bulk production facilities and the customers (e.g., retail stores, end customers). A warehouse provides temporary storage for excess production, thus supporting an uninterrupted flow of distribution across the entire supply chain. It represents significant capital investments tied up in goods, labor, equipments, and space. Within a limited area, thousands of different products are being stored and handled using various material handling technologies, to be dispatched to customers later on. Such activities are performed to meet customer demand conforming to the agreed delivery date.

New challenges are continuously being introduced to warehouses. Retailers are setting tighter delivery schedules to ensure a high service level and to avoid out-of-stock situations. Furthermore, noticeable changes in the order profile are occurring. Internet orders are stimulating a large volume of small orders, that is, orders requiring few items. Moreover, due to the innovations in the supply chain, the number of

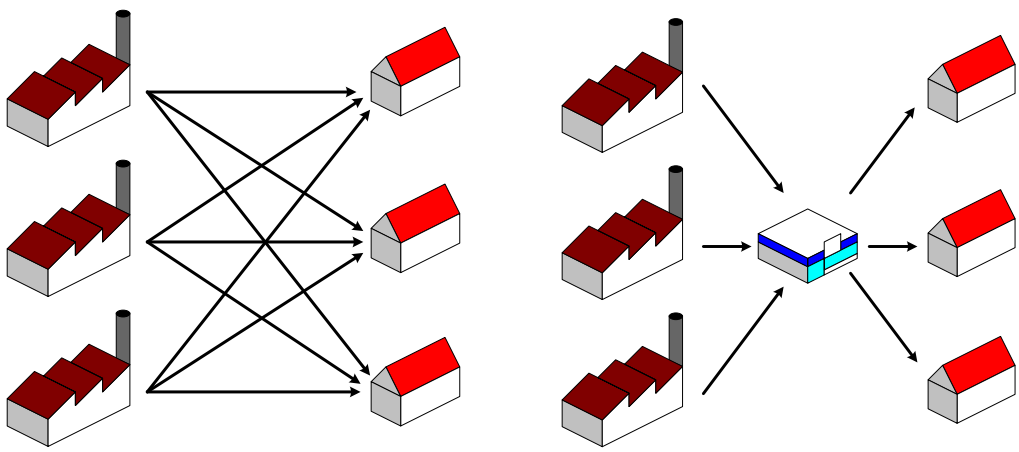
Stock Keeping Units (SKU) in some industries is rapidly increasing - a phenomenon known as SKU proliferation (Twist, 2005). These challenges force warehouses to continuously improve performance by adjusting their operation to the ever-changing requirements.

Warehouse functions

Warehouses sustain supporting functions for a streamlined operation throughout the entire supply chain. They act as the primary buffers in the supply chain such that manufacturers are able to respond timely to changes in supply and demand. This is a critical function particularly when products are characterized with a seasonal demand. Demand for Christmas decorations, for example, rises significantly in December. To cope with the sharp increase in demand, manufacturers of such products may already produce in bulk quantities months in advance, to be then stored in the many warehouses across the country.

Warehouses reduce customer lead time in a supply chain. This is because production lead time can be excluded from the customer lead time by fulfilling customer orders with products buffered in a warehouse. It is therefore important that warehouses have sufficient supply of products for a certain demand profile. This topic, namely warehouse replenishment strategies, has been investigated extensively in numerous studies (see a review by e.g. Minner (2003); Khouja and Goyal (2008)).

The existence of warehouses also cuts the transportation costs in the supply chain. Without warehouses, manufacturers have to deliver products directly to the customers. Consequently, deliveries with less-than-truckload quantities are frequent. This is undesirable because a fixed transportation cost typically exists for each delivery. Owing to warehouses, it is possible for manufacturers to reduce the frequency of deliveries and to send full truckloads to warehouses at each delivery. Furthermore, warehouses are able to schedule deliveries in full truckloads to numerous end customers. As more full truckload deliveries are performed, significant savings in



(a) Without a warehouse. (b) With a warehouse.

Figure 1.1: A warehouse reduces transportation costs and frequency of delivery.

transportation costs can be realized. Figures 1.1(a) and 1.1(b) illustrate the above two situations.

Finally, warehouses are increasingly used to provide value-adding activities. Some examples include customized assembling, labeling, and kitting. In this sense, warehouses serve the main role in the postponement strategy - an organizational concept of not performing some activities in the supply chain until customer orders are received (van Hoek, 2001).

Warehouse operations

A number of operations are performed in a warehouse. These operations can be categorized into *inbound processes, storage,* and *outbound processes*. Inbound processes include receiving and putting-away products from various production facilities. Subsequently, these products are stored temporarily in a storage area. Outbound processes are comprised of order-picking, checking, packing, and shipping products to meet customer orders. Of all operations in a warehouse, *order-picking* has been identified by far as the most costly and time-consuming operation. Order-picking is defined as the process of retrieving products from the storage area to meet customer demand. It accounts for approximately 55% of the total warehouse operating cost (Drury, 1988; Bartholdi III and Hackman, 2010). Traveling, which is both non value-adding and tiresome, is found to be the single dominant activity (Frazelle, 1996) within order-picking. Other activities involved in order-picking, e.g. searching and extracting, are highly repetitive in nature and also consume vast labor hours. Figure 1.2 depicts the typical warehouse operations.

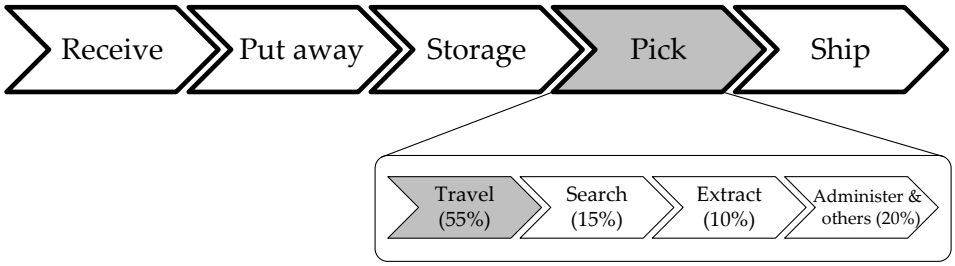


Figure 1.2: Warehouse operations.

The cumbersome order-picking operation combined with high labor cost and advanced material handling technology has turned warehouse automation into an increasingly common practice in the industry. Automated warehouses are expected to provide higher throughput and reliability with lower cost compared to that of their manual counterparts. Automation should allow warehouses to work continuously 7 days a week while eliminating human-related errors. Moreover, a significant cut in labor cost is possible as less labor is present on the shop-floor. A high startup investment, however, is usually required depending on the type of automation. Nevertheless, there has been a noticeable increase in the number of automated warehouses, for example in the United States (Roodbergen and Vis, 2009). One may expect this trend to increase further in the future as warehouse automation provides a solution to the arduous order-picking activities (including traveling, searching, and extract-

ing) and that the latest order-picking technology allows for a high picking rate (up to 1000 picks per worker hour (de Koster et al., 2007)).

Classification of order-picking system

Different types of order-picking systems (OPS) for warehouses exist in the literature. Following Dallari et al. (2009) and de Koster et al. (2007), a classification of OPS is proposed in Figure 1.3. Five types of OPS are distinguished, namely *manual/automated picker-to-parts* systems, *manual/automated parts-to-picker* systems, and *pickerless* systems.

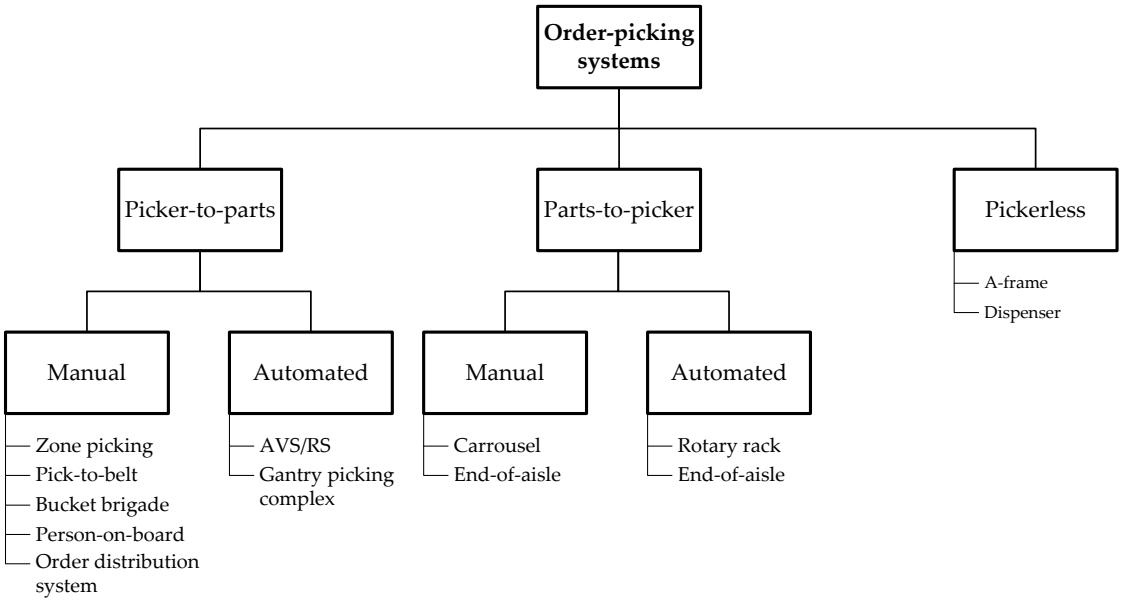


Figure 1.3: Classification of OPS.

A picker-to-parts system is characterized by pickers that are moving from one location to another within the storage area in search of products that are required by customer orders. Some manual variants of this system include zone picking (Petersen, 2002), pick-to-belt (de Koster, 1994), bucket brigades (Koo, 2009; Bartholdi III et al., 2001), person-on-board (Dallari et al., 2000), and order distribution system. Their automated counterparts include Autonomous Vehicle Storage/Retrieval Systems (Fukunari and Malmborg, 2009; Ekren et al., 2010) and a gantry picking complex (Kim et al., 2002, 2003). Some typical issues of picker-to-parts OPS discussed in the literature are optimal layout design and dimensioning of the storage area, picker routing, storage assignment, zoning, and order batching. A review by de Koster et al. (2007) provides a thorough overview on these issues.

On the contrary, a parts-to-picker system is distinguished by products moving from the storage area to pickers that subsequently collect the products to meet customer orders. Examples of manual variants of this system are vertical lift modules (also known as carrousels) (Litvak and Vlasiou, 2010) and end-of-aisle OPS with remotely located manual workstations. Their automated variants include rotary rack (Li and Bozer, 2010) and end-of-aisle OPS with remotely located automated workstations. Literature on parts-to-picker systems focuses mainly on system configuration, storage

assignment, batching, sequencing of storage/retrieval, and dwell-point strategy. An extensive review on these issues is given by Roodbergen and Vis (2009).

The last category is the pickerless system, in which products are accumulated by the system without any picker intervention. Human operators are needed only during the replenishment process. Examples of such systems are A-frames and dispensers.

Given the different types of OPS, determining which OPS to be used in a warehouse is not trivial. For this purpose, Dallari et al. (2009) proposed an OPS selection methodology to be used in the warehouse design phase by taking into account the number of SKU and the order profile involved.

1.2 End-of-aisle order-picking systems

The current research focuses on one instance of parts-to-picker OPS, namely an *end-of-aisle* OPS. As the name suggests, an end-of-aisle OPS refers to a system where products are delivered by Storage/Retrieval (S/R) cranes from the storage racks to the order picker located at the end of the aisle between the storage racks. If products are stored as a unit-load in bins, then the storage area is commonly referred to as *miniloads*. In this dissertation, a miniload is defined as a single aisle formed by two parallel storage racks with one automated S/R crane.

There are three main types of end-of-aisle OPS with miniloads, namely *conventional*, *horse-shoe*, and *closed-loop conveyor* (Park et al., 1999). A conventional miniload system has two pick positions (right and left) attached to the end of each aisle. When the bin at one of the two pick positions has been picked, the S/R crane takes the bin, stores it back in the storage racks, and returns with a new bin. A horse-shoe miniload has inbound and outbound buffers at the end of each aisle. Product bins to be picked are buffered in the inbound buffer, while products bins that have been picked are buffered in the outbound buffer to be stored back to the storage racks. A picker is present between the two buffers to pick the product bins. A miniload with a closed-loop conveyor typically has remotely located order-picking workstations. The closed-loop conveyor transports product bins from the miniloads to the workstations. At the workstation, a (robot) picker collects the products required for one order altogether.

This dissertation focuses on an end-of-aisle OPS consisting of miniloads, remotely located order-picking workstations, and a closed loop conveyor. This system is illustrated in Figure 1.4.

An end-of-aisle OPS with remotely located workstations offers a wide range of benefits. One of the main advantages of having multiple storage racks detached from the workstations is that products for multiple customer orders can be retrieved simultaneously. The workstations will then be able to process multiple orders at the same time. Also, in case the S/R crane in one of the storage racks fails, products may still be retrieved from the other storage racks. This increases system robustness against failures. Having multiple storage racks also allows more SKUs to be stored. Such a setting provides a solution for SKU proliferation. Moreover, the use of high-bay storage racks served by automated cranes saves significant amount of required floor space, hence better space utilization compared to systems with low and wide

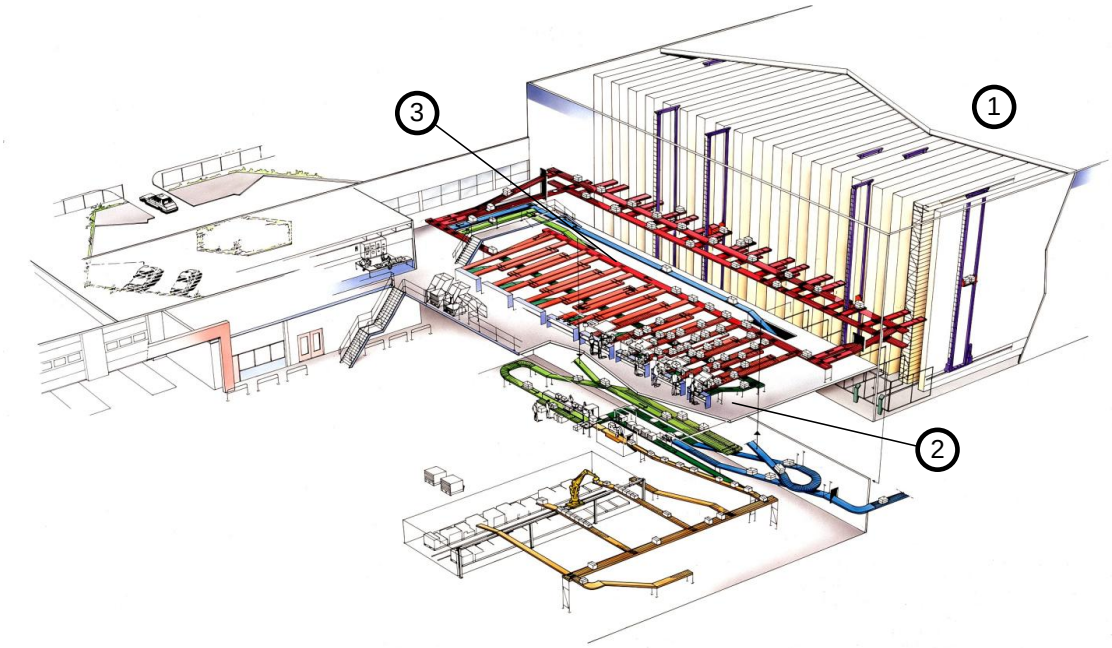


Figure 1.4: An end-of-aisle OPS with miniloads (1), order-picking workstations (2), and a closed-loop conveyor (3). Figure courtesy of Vanderlande Industries.

storage racks. This type of system is also less prone to misplacement or theft than manual systems involving traveling pickers. These advantages advocate the use of end-of-aisle OPS with remotely located workstations in automated warehouses.

Design considerations

Along with its configuration, an end-of-aisle OPS also brings some typical issues. These issues include:

- *Storage assignment*. A typical storage area for an end-of-aisle order-picking workstation comprises a large number of storage locations. The storage assignment problem essentially concerns the assignment of storage locations to various SKUs, taking into account the fact that some SKUs are requested more frequently than others. Storage assignment strategies include e.g., dedicated storage, random storage, class-based storage, and turnover-based storage.

- *Retrieval sequencing*. A customer order may consist of more than one product. In this case, retrieving products in a first-come-first-serve sequence may require the S/R crane to travel a large distance, which eventually deteriorates throughput. This happens when the next product to be retrieved is located far from the current location of the S/R crane following a storage operation.

- *Out-of-sequence arrivals*. When products for multiple orders are simultaneously retrieved from various locations in different storage racks, these products are likely to arrive out-of-sequence at the workstation. That is, the sequence of product arrival at the workstation is not similar to the sequence in which the

products were initially requested. This is because products may overtake one another on their way to the workstation. Depending on the workstation setting, this situation may hinder workstation throughput.

- *Pipeline filling*. A pipeline capacity is defined as the maximum number of products that may simultaneously move from the storage area to the order-picking workstation. A high pipeline capacity allows products to arrive faster at the workstation. However, increasing the pipeline capacity may result in heavier out-of-sequence arrivals.

- *Picking sequence*. The picking sequence influences the throughput when pickers are allowed to process more than one order at a time. Given a number of products and orders to be picked, the problem is to determine the sequence in which products are processed so as to keep the throughput as high as possible.

Note that the issues of out-of-sequence arrivals, pipeline filling, and picking sequence are not relevant for conventional and horse-shoe miniloads. Both of these systems have a picker located immediately at the end of the aisle and each picker is dedicated to one miniload only. The S/R crane always retrieves products for one order at a time. Once all products for an order have been picked, the S/R crane retrieves products for the next order and presents them to the picker. In this sense, out-of-sequence arrivals and picking sequence are not an issue. Furthermore, since the picker is attached to the miniload, the pipeline filling also becomes irrelevant. These three issues are only relevant in the system considered in this dissertation, namely an end-of-aisle OPS with remotely located workstations.

Literature on end-of-aisle OPS

Bozer and White (1990) considered an end-of-aisle OPS with miniload where each picker is assigned to one miniload only. Hence, each picker processes one order at a time. The miniload operates in a dual-command cycle, i.e., the miniload crane performs a retrieval operation immediately following a storage operation. Given a predetermined storage space, they proposed a heuristic design algorithm to determine the minimum number of storage aisles required to attain a specific throughput. They considered the case where two and four pick positions are available for the pickers. The proposed heuristic is practical to provide insight early in the design phase of such OPS. A subsequent study by Bozer and White (1996) analyzed a more general setting by assigning a picker to more than one miniload.

A retrieval sequencing problem for a miniload in an end-of-aisle OPS was studied by Mahajan et al. (1998). They considered a system similar to that of Bozer and White (1990), where each miniload with a dual-command cycle is assigned to one picker. With this setting, orders must be picked one at a time. They assumed that the S/R machine is the bottleneck and proposed a nearest-neighbor retrieval sequencing heuristics to improve the throughput. The heuristic is shown to increase the throughput by 5-15% as compared to that of the traditional first-come-first-serve retrieval sequence.

Park et al. (1999) discussed a buffer sizing problem in an end-of-aisle OPS with a miniload having a horse-shoe front-end configuration. Here, a picker is assigned to one specific miniload as in the previous studies. Picking time is assumed to be exponentially distributed and storage/retrieval time is assumed to follow a general distribution. Under these assumptions, they developed an analytical model based on a two-stage closed queueing system to investigate the effect of buffer size on the system throughput.

Storage assignment strategies for an end-of-aisle OPS are also subject to several studies. These studies are motivated by the fact that a minority of products stored in the storage racks may be required by most of the retrievals. As such, one may distinguish a specific location for these frequently requested products to reduce the retrieval time. Studies on this issue were performed by Eynan and Rosenblatt (1994); Park et al. (2003, 2006).

Some other studies are directed towards approximation of throughput bounds for an end-of-aisle OPS. These studies typically consider conventional miniloads with two pick positions and one picker per miniload. Literature on this problem includes e.g., Foley and Frazelle (1991); Foley et al. (2002, 2004). As yet, most literature on end-of-aisle OPS has been focusing on conventional miniloads with two pick positions.

1.3 Methods for performance analysis

The complex and expensive nature of order-picking in an end-of-aisle OPS suggests the importance of performance analysis of such a system. Performance analysis provides not only feedback on the quality of a proposed design and/or operational policy, but also insights into how they can be improved. Two commonly used approaches in the literature are analytical models and simulation models. The type of questions that can be answered using these approaches is different, and thus they are generally used at different design phases and for different purposes.

Analytical models

Analytical models are particularly useful during the system design phase, when one is mainly interested in having a quick overview on the performance of different designs. Analytical models serve this purpose under necessary assumptions for mathematical tractability. This being said, analytical models may not capture all details of the system.

Many analytical performance analyses of end-of-aisle OPS are based on closed queueing network models. Specifically, these are systems with conventional miniloads that perform a dual-command cycle. In such cases, the number of jobs in the system is constant as the S/R machine only performs a retrieval operation immediately following a storage operation. For example, Bozer and White (1990) developed a two-server closed queueing model of an end-of-aisle OPS with conventional miniloads. The number of pick positions represents the fixed number of jobs while the picker and the S/R machine are the two servers. Two types of pick time distributions were considered, namely exponential and deterministic. Using this model they were able

to quantify the expected throughput and utilization. This model was extended by Bozer and White (1996) by considering multiple pick positions per aisle and multiple aisles per picker. The closed queueing network now contains multiple loops, where each loop consists of a picker and an S/R machine.

Park et al. (1999) proposed a two-stage cyclic queueing model with a limited capacity for an end-of-aisle OPS with a horse-shoe miniload. The picker is modeled as having an exponentially distributed pick time, while no specific distribution was assumed for the travel time of the S/R machine. From this model they were able to derive a closed-form expression for performance measures including the steady-state probability, system throughput, utilization, mean number of jobs at each queue, mean residence time at each queue, and mean cycle time. The model was then used in designing the buffer capacity for a certain system throughput. An extension of this study was done by Koh et al. (2005), where they assumed that one picker may serve multiple aisles in an end-of-aisle OPS with a horse-shoe miniload. They were able to find the steady-state behavior by modeling the system as a two-stage cyclic queueing model. Additionally, they proposed an optimization model to find the optimal buffer size.

Another approach is to derive the analytical expression of performance measures directly. Foley and Frazelle (1991) derived an exact, closed-form expression for the system throughput of an end-of-aisle OPS with conventional miniloads having uniformly distributed retrieval locations and general pick time distributions. The resulting expression can be used to compute the minimum number of aisles given the pick time distribution, the required system throughput, and storage capacity. Foley et al. (2002) derived tight upper and lower bounds on the throughput of an end-of-aisle OPS with conventional miniloads given different scenarios of partial information about the pick time distribution. Park et al. (2003) developed closed-form expressions for the mean and variance of cycle times of an S/R machine for a conventional miniload. Such performance measures are valuable to be used as input for the closed queueing model of an end-of-aisle OPS from previously mentioned studies.

Simulation models

Simulation models are widely used alternatives to analytical models for performance analysis at the system utilization phase. They are especially useful in evaluating numerous what-if scenarios of (detailed) operational policies or parameters on a specific design. They also allow more details of the system to be captured than in analytical models. As such, these models are practical in identifying specific operational settings that improve the system performance. However, building a valid, credible, sufficiently detailed simulation model and generating appropriate outputs may be very time consuming. Moreover, detailed simulation models may require extensive computational capability. Nevertheless, simulation is still the most widely used technique for warehouse performance analysis in practice (Gu et al., 2010).

Numerous simulation models of end-of-aisle OPS have been developed so far. Perry et al. (1984) proposed an optimum-seeking approach based on a discrete-event simulation model for designing an end-of-aisle OPS with a conveyor-loop and remotely

located workstations. Using a so-called expected value model, they created a heuristic to identify the optimal configuration of the physical system design for such OPS. A modular simulation model with an interactive user interface was created by Raghunath et al. (1986) for end-of-aisle OPS. The modularity allows the user to model different types of miniload including conventional, horse-shoe, and closed-loop conveyor from the user interface. Medeiros et al. (1986) developed a simulation model for an end-of-aisle OPS with a conventional miniload. Their model can be used to design a miniload system that is capable of meeting or exceeding a given number of dual cycles per hour. Pulat and Pulat (1989) analyzed the performance of a miniload with horse-shoe configuration by using a simulation, an open queueing network model, and an intensive sampling approach. They were mainly interested in the percentage idleness of the crane and the picker. Takakuwa (1996) created a module-based simulation model for an end-of-aisle OPS with automated guided vehicles delivering products from the miniload to the remotely located workstations. They argued that the modular approach reduces model development time. A number of design alternatives, including storing/retrieving policy, overall layout of conveyors, and arrangement of racks inside the miniloads were evaluated.

The data issue in modeling for performance analysis

Regardless of the type of model used, data availability is the key for prediction accuracy in performance analysis. The absence of measurable data compromises reasonable inputs required for the model. In turn, performance analysis based on misleading assumptions is harmful for decision-making. It is therefore crucial that all model parameters can be limited to data that is obtainable from the shop-floor.

In the context of manufacturing, Wu et al. (2008) proposed a classification of interruptions for a single machine manufacturing system. They proposed two main types of interruptions, namely run-based events and time-based events, both of which are further categorized into preemptive events and non-preemptive events. For each category of interruptions, they proposed an analytical queueing model to predict the typical performance measures such as flow time and queue length. The various interruptions mentioned in their study are also relevant in the context of end-of-aisle OPS. Some examples are power outage, preventive maintenance, setup, warm-up, and out-of-spec input.

Unfortunately, data collection in an operational logistic system is not straightforward. This also applies for an end-of-aisle OPS due to the numerous stochastic behaviors involved in such a system. Picking time and stochasticities due to interruptions are typically difficult to quantify. Alternatively, arrival and departure times of products/orders at the order-picking system are often available from the WMS (Warehouse Management System). For this reason, an aggregate modeling technique using the concept of EPT (Effective Process Time) is considered. The EPT represents the aggregation of all process time components involved in an order-picking workstation (see Figure 1.5). This way, there is no need to explicitly assign a separate value to each stochastic behavior at the order-picking workstation. More importantly, EPT is calculated directly based on the available data of arrival and departure time of products at the order-picking workstation.

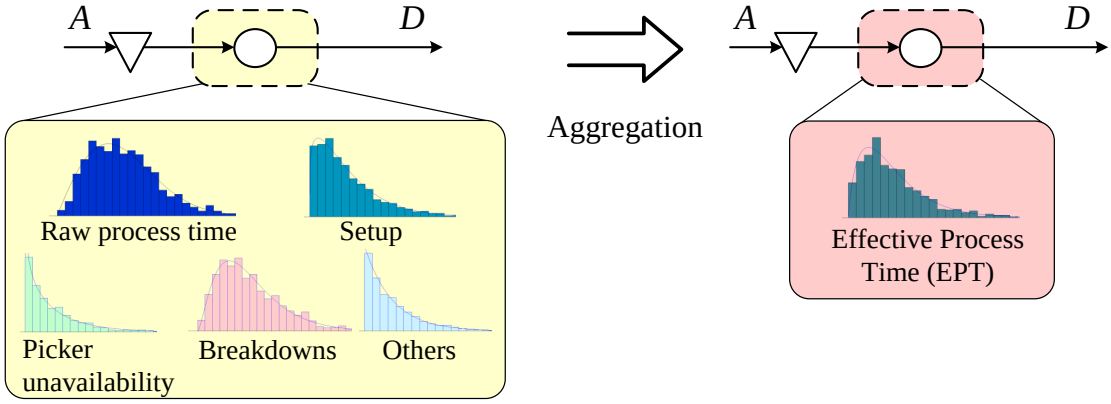


Figure 1.5: EPT as an aggregation of all process time components calculated from arrival (A) and departure (D) time of products.

Methods to quantify EPT directly from arrival and departure data and to use the EPT for performance analysis have been developed in earlier research work at Eindhoven University of Technology. Jacobs et al. (2003) defined EPT as the total amount of time a job could have been, or actually was, processed on a machine. They developed an algorithm to quantify EPT from arrival and departure times of jobs for workstations with single and multiple machines. The algorithm was then extended by Jacobs et al. (2006) for workstations with batch machines. Another approach to quantify EPT directly from arrival and departure data of jobs is by using a so-called sample path equation. This approach was used by Kock (2008); Kock et al. (2008a,b) to analyze the performance of an assembly line, a finitely buffered manufacturing flow lines with multiple servers, and single server, respectively. The latest work on EPT-based aggregate modeling was done in the area of semiconductor manufacturing. An aggregate model has been developed for generating cycle time-throughput-product mix curves (Veeger et al., 2010) and predicting the cycle time distributions of integrated processing workstations (Veeger et al., 2011). The model was then extended to predict the cycle time distributions of networks of such workstations (Veeger, 2010). We refer to Veeger (2010) for a thorough overview of EPT-based aggregate modeling.

Note that Hopp and Spearman (2000, 2008) coined the term EPT referring to the effective process time of a machine taking into account the variability in process including setups, rework, and random failures. Assuming some given values for each disturbance, one can calculate the mean and squared coefficient of variation of the EPT using closed-form formulas (Hopp and Spearman, 2008). The EPT formula can be used for both preemptive and nonpreemptive disturbances. This way of working is the exact opposite of the aggregate modeling techniques developed in the literature mentioned previously and considered in this dissertation.

1.4 FALCON project

This dissertation is the result of research performed as a part of the FALCON (Flexible Automated Logistics CONcept) project. The project is a joint endeavor of a consor-

tium of industrial and academic partners under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the carrying industrial partner. The aim of the FALCON project is to find and develop efficient means to analyze, design, and implement layered systems that shall comply with stringent performance requirements. The project considers the development of a new generation of warehouses and distribution centers with maximum degree of automation. Three main topics in the FALCON project are decentralized control engineering, system modeling, and automated item handling. This dissertation contributes to the system modeling topic.

Developing a new warehouse automation concept often means pushing the boundaries of feasibility of material handling technology. Even when this is successfully achieved, market uncertainty still exists regarding the customer acceptance of the new concept. Moreover, developing and realizing the new concept must be done quickly to achieve a short time-to-market. All of these typically cause high development cost for such a complex system. With this regard, models that allow quick adjustments and analysis while still giving satisfactory accuracy to reality are crucial to provide insights on system performance early in the design phase. Such models can be used to compare the added value of the new concept relative to the development cost, which eventually supports the decision-making process.

Robustness of system performance is also a relevant issue in developing highly automated warehouses. For such systems with large investment of money and resources, it is no longer sufficient to have a working system under a predefined, specific setting. The next generation warehouses are 'flexible platforms' that provide technically feasible stepping stones for further development to anticipate changes in the future. That is, the warehouses are robust in performing under continuously changing customer requirements. Reusability, exchangeability and flexibility are the key requirements for the next generation of highly automated warehouses. With this regard, new performance analysis methods that can support the design and operation of these automated warehouses are needed.

1.5 Research objective

This dissertation considers an end-of-aisle OPS with remotely located manual and automated order-picking workstations. The research aims to develop methods for the design, modeling, and control of such a system so as to quantify and to improve its throughput and flow time performance. Specifically, this dissertation focuses on the performance of manual and automated order-picking workstations. The objective of this dissertation is threefold.

First, a flexible architecture of such end-of-aisle OPS is necessary for detailed modeling. Creating a simulation model of a complete end-of-aisle OPS is not trivial as numerous entities interact with one another in parallel. Customer orders are received, distributed, and subsequently processed using a number of control strategies that trigger material flow at shop-floor level. A flexible structure is desired that allows an easy and straightforward implementation of necessary changes to investigate various what-if scenarios. Preferably the same architecture can be implemented

in the operation of such OPS.

Second, a performance analysis method that overcomes the data availability issues and gives good accuracy is required for the order-picking workstations. Since acquiring necessary parameters in practice is not trivial, a key requirement of the method is that one does not need to model in detail the various stochasticities involved in the order-picking workstation. Preferably, the performance analysis method operates only on little, yet measurable data that is available from the WMS. The method would be able to predict the workstation performance under different settings.

Third, the order-picking workstations have to be able to treat out-of-sequence arrivals of products. For automated workstations this is particularly challenging because they have to process multiple orders simultaneously to arrive at a sufficiently high throughput. The out-of-sequence arrivals must be dealt with at the workstation, allowing the miniload to work at full capacity in retrieving and sending products to the workstation. To this end, the automated order-picking workstation has to be designed such that it provides the highest possible picking capacity while at the same time addresses the out-of-sequence arrival of products. Avoiding deadlock and developing efficient picking strategies are the main issues in this respect. A favorable picking policy is the one that is robust to different extent of out-of-sequence arrivals.

1.6 Contributions and outline

This dissertation consists of three parts, which are contained in Chapters 2 to 5. Each part contributes in addressing the research objectives posed in the previous section.

Chapter 2 presents a modeling architecture for simulation of an end-of-aisle OPS with remotely located workstations. The model architecture is structured into areas and operational layers. A hierarchical decentralized control structure is applied. Using an industrial scale warehouse as a reference case, it is shown that the proposed architecture is implementable for performance analysis using a process algebra based simulation language χ (Chi). The architecture allows for easy implementation of different system structures, design parameters, and control heuristics. As an example, the throughput performance from using two different order release strategies and adding/removing miniloads is analyzed.

Chapters 3 and 4 discuss the EPT-based aggregate modeling for performance analysis of manual order-picking workstations. The picker at the workstation can only process one order at a time while products for multiple orders can be present at the same time. A sample path equation is used to calculate the EPTs based on product arrival and departure times that can be obtained from the WMS. The EPTs of the first products of an order are distinguished from the EPTs of the remaining products of the order, which is referred to as the 1st tote difference approach. Two model variants have been developed, namely in Chapter 3 for workstations with first-come-first-serve (FCFS) and in Chapter 4 for workstations with non-FCFS processing sequence of products and orders. EPT distributions are used as input in both model variants. An overtaking distribution and a so-called decision probability are used as additional inputs in the model with non-FCFS processing. In both chapters a case study is provided to validate the EPT-based aggregate modeling technique. The technique gives

good accuracy in predicting product and order flow time distributions using product arrival and departure times from an operating order-picking workstation.

Chapter 5 addresses the design and control of an automated order-picking workstation. The workstation is capable of processing multiple orders simultaneously by means of a gantry robot. A simple overtaking function is proposed to model the various extent of out-of-sequence arrivals. An architecture with a built-in carrousel is proposed. The resulting throughput and queue length distribution from four picking policies are compared. Noteworthy insights for design considerations of such a system are drawn.

In the conclusion presented in Chapter 6 we reflect on the findings from the previous chapters to come up with comprehensive design principles of an end-of-aisle OPS with remotely located workstations. The various issues influencing the performance of such a system are highlighted. The contribution of each method proposed in the previous chapters with regards to these issues is delineated.

1.7 Reader's guideline

Chapters 2 to 5 are self-contained articles that can be read independently. These are articles that have been published or accepted to conferences and/or journals. It is recommended, however, to read Chapter 3 before Chapter 4 as the latter chapter is a follow-up on the former.

2

Architecture of an end-of-aisle order-picking system

R. Andriansyah, W.W.H. de Koning, R.M.E. Jordan, L.F.P. Etman, and J.E. Rooda,
A process algebra based simulation model of a miniload - workstation order picking system,
Computers in Industry (2011), 62: 292-300.

Abstract | A modular discrete-event simulation model for an end-of-aisle order-picking system has been developed using a process algebra based simulation language. The proposed model architecture is structured systematically such that distinctions between areas and operational layers can be clearly identified. Furthermore, subsystems and decentralized controls are applied in the model architecture. The modularity of the model is demonstrated by experiments, in which some control heuristics and the number of miniloads are altered. A realistic, industrial scale distribution center is used as the reference case for the simulation study. The resulting model architecture allows easy implementation of various system structures, design parameters, and control heuristics.

2.1 Introduction

The state-of-the-art technology in material handling systems has turned AS/RS (Automated Storage/Retrieval Systems) into common practice for distribution centers. An AS/RS is a comprehensive material handling system that typically comprises storage racks, storage/retrieval cranes, input/output (I/O) locations, and conveyors. The system is able to handle the storage, retrieval, and transportation of unit loads without interference of human operators. A large variety of system options for

AS/RS is currently available in the industry. Recently, Roodbergen and Vis (2009) provided a thorough overview of AS/RS systems.

We consider a special class of AS/RS namely the *end-of-aisle systems* with *totes* as the unit loads. In such an end-of-aisle AS/RS, *product totes*, which contain items belonging to the same SKU (Stock Keeping Unit), are retrieved from the storage racks and are sent to the order-picking workstation. At the workstation, an operator picks the required amount of items from a product tote and puts them into another tote, known as the *order tote*. Afterwards, the product tote is sent back to the storage rack if the tote is not empty. This system is also referred to as the miniload-workstation order picking system.

Most of the literature on AS/RS simulation is directed towards performance analysis. One of the earliest of such studies was done by Houshyar and Chung (1991) who analyzed the performance of a small AS/RS warehouse under different scenarios. Lee et al. (1996) conducted a simulation study in a larger scale AS/RS warehouse with RGVs (Rail-Guided Vehicles) to determine the strategy that yields the optimal number of RGVs, the utilization of the crane, and the maximum throughput of the system. Potrč et al. (2004) considered the performance analysis of a multi-shuttle AS/RS. Meller and Mungwattana (2005) investigated the effect of dwell-point strategy for a highly utilized AS/RS.

Typically, simulation models are exclusively created for specific, pre-defined system configurations. In this case, altering the system structure (for example adding or subtracting the number of aisles, cranes, order-picking workstations, *etc*) or design parameters (for example control heuristics, order pattern, replenishment policy, *etc*) may require much time and effort before the model is fully functional. Also, many simulation models are simplified such that model architecture and control structure become less of an issue.

Studies discussing the development of model and control structures for AS/RS are very limited. We argue, however, that model architecture and control structure are crucial for simulation studies of industrial scale AS/RS. After all, one of the main strengths of simulation in AS/RS research is to compare numerous designs, taking into account more design aspects in combination with control policies so as to obtain more information on good design practice (Roodbergen and Vis, 2009). For this purpose, a profound model architecture and control structure are needed.

We propose a novel approach towards building a simulation model for a comprehensive end-of-aisle OPS based on *process algebra*. The contribution of this study is twofold. First, we show the applicability of a process algebra based simulation language, χ (Chi), in modeling a realistic, industrial scale end-of-aisle OPS. Second, we propose a modular model architecture with regards to system structure and design parameters.

The remainder of this chapter is organized as follows. The end-of-aisle OPS under study is described in Section 2.2. In Section 2.3 we provide some related works on detailed modeling of AS/RS using Petri nets. Section 2.4 provides a brief overview of process algebra and the language χ. Subsequently, the overal architecture of the proposed simulation model is presented in Section 2.5. An example of modeling using χ is explained in Section 2.6. Section 2.7 provides experiments to show the modularity of our model. Finally, Section 2.8 concludes the chapter.

2.2 System description

The end-of-aisle OPS elaborated in this study is based on an existing distribution center. In Section 2.2.1 we present the physical structure, while in Sections 2.2.2 and 2.2.3 we describe respectively the storage and retrieval, and the item-picking. Figure 2.1 shows the overal structure of the system.

2.2.1 Physical structure

The end-of-aisle OPS can be divided into three areas, namely *miniloads, workstations,* and *conveyors*. Miniloads provide temporary storage spaces for product totes. At the workstation, items are picked from product totes and are put into order totes. Conveyors connect the miniload area to the workstation area, and the other way around, for moving the product totes.

Miniloads are automated storage racks equipped with cranes to serve two functions, namely the storage and retrieval of product totes. Each miniload consists of two single-deep racks with a single crane in the middle to access product totes. Each crane is capable of holding up to four product totes simultaneously. The cranes move horizontally along the aisle between the racks, while the holder of product totes move vertically to store or retrieve the totes. There are five miniloads present in the system and a total of 31250 storage locations for product totes.

Each of the three workstations in the system consists of three input buffers and one output buffer (see Figure 2.1). There are maximal three orders active at the same time at a workstation, and thus maximal nine orders are active in the whole workstation area. An operator works on one order at a time, putting all items picked

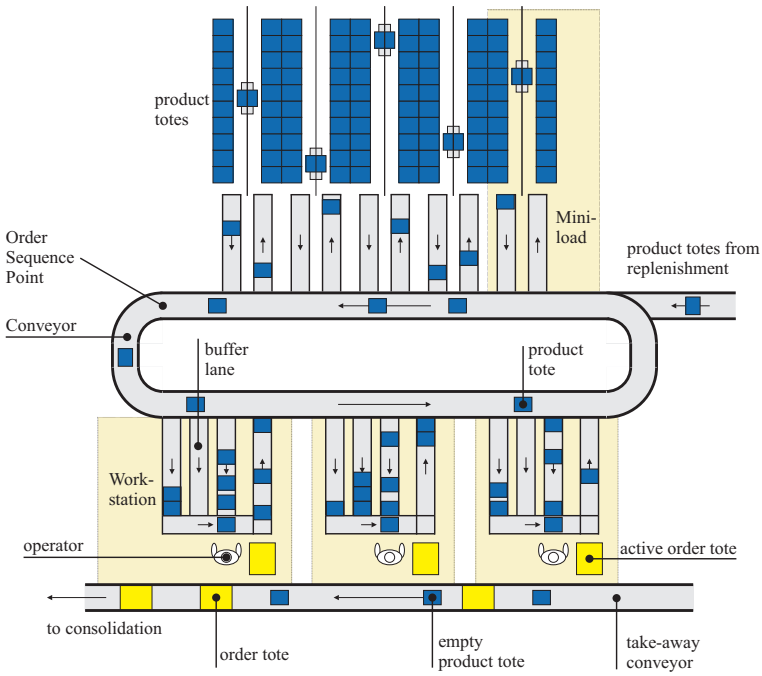


Figure 2.1: End-of-aisle order-picking system.

from the product totes belonging to one order into the order tote(s). The operator is not allowed to start working on the next order when not all items for the current order have been picked.

The central conveyor loop transports product totes from the miniload area to the workstation area, and the other way around. As there is only a limited number of positions on the conveyor, only product totes that have successfully reserved a position are allowed to enter the conveyor.

The operation of the OPS is triggered by customer orders that enter the system at any time. Each order can contain up to 316 order lines. An order line represents an SKU type and the required amount of items for that SKU. Note that it is possible that not all items required by an order can fit in one order tote. In this case, multiple order totes are dedicated for one order. The distribution of the number of order lines per order is shown in Figure 2.2. The mean and standard deviation of this distribution are 9.824 and 22.471, respectively. In total, 1624 SKUs are handled in this OPS. Three main operations in the system are *storage, retrieval*, and *item-picking*.

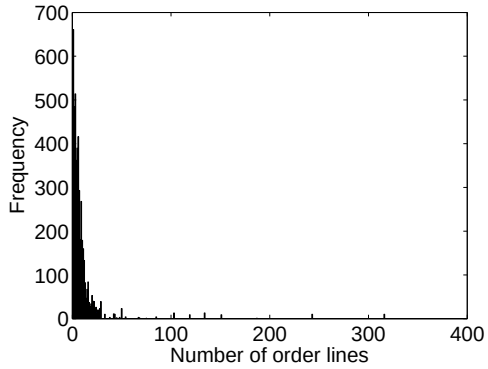


Figure 2.2: Distribution of number of order lines per order.

2.2.2 Storage and retrieval

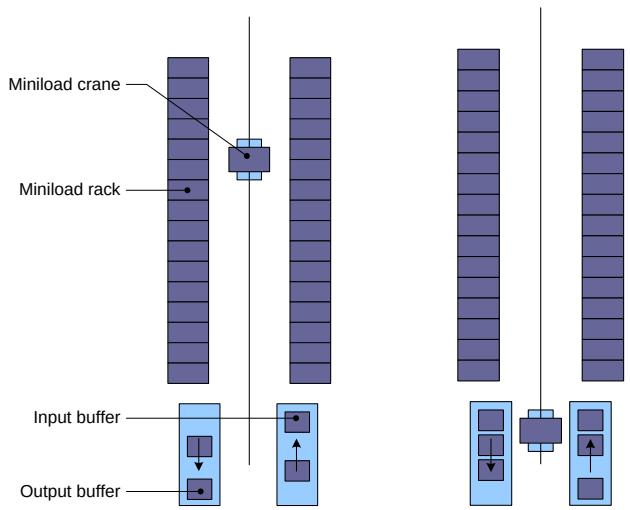
Storage happens when a product tote needs to be kept temporarily in the miniload until it is required to fulfill an order. Two types of product tote exist, namely *replenishment* and *returning* product totes. A replenishment product tote is a recently arriving tote that is full of items. A returning product tote is a tote that has just finished being picked at the workstation but still contains some items left. This type of tote has a higher priority for fulfilling an order than a replenishment tote. An incoming product tote is stored in the miniload that has the least amount of item of the SKU contained in that tote. If more than one miniload has the least number of items of an SKU, the destination miniload is chosen randomly. This is the storage strategy of the miniload.

Retrievals take place at the miniload and start when the miniload controller has chosen the next order to be completed from a list of all arriving orders. The chosen order is further divided into jobs, which specify the SKU type and the required number of items to be picked. These jobs are then assigned to the five miniloads. When a miniload is assigned with a retrieval job, it reserves a number of product totes until

the required quantity of items is covered by the items in the reserved tote(s). Once a product tote is reserved for a job, items in that tote can only be used to fulfill that particular job and may not be used for other jobs. The reserved totes are retrieved by the miniload cranes and are put on the output buffer of the miniload. The totes wait until they get access to the central conveyor loop to be sent to one of the workstations. Note that the inventory position of each SKU is continuously updated. The inventory position serves as the base for the replenishment process, that is, ordering additional items from the outside suppliers. In this system, an order-up-to level replenishment policy (s, S) is used (see Silver et al. (1998)).

Two queues can be distinguished for the miniload operation. *Storage* queue q_s is a physical queue at the miniload input buffer, while *retrieval* queue q_r is a virtual queue of totes at the miniload controller. The trigger for storage or retrieval action is either $q_s \geq 4$ or $q_r \geq 4$. That is, the miniload crane waits until a batch of four totes is formed. However, if after a delay of 120 seconds (a so-called *time out*) no batch of four totes has been formed either in q_s or q_r, then a storage or retrieval action will still be triggered.

The decision on which of the two actions is executed depends on the position of the miniload crane at that moment. Two positions are possible, namely *inside* and *outside* the miniload racks. When the miniload crane is inside the racks, then it is ready to retrieve (Figure 2.3(a)). Otherwise, if the miniload crane is outside the racks, then it is positioned next to the input/output buffer, ready to store (Figure 2.3(b)). At the start of a work day, the miniload crane is outside the racks.



(a) Crane inside. (b) Crane outside.

Figure 2.3: Positions of a miniload crane.

The miniload crane operates in such a way that the time out occurrence is minimized. For example, if the miniload crane is inside the miniload rack (ready to retrieve, Figure 2.3(a)) while $0 \leq q_r < 4$ and $q_s \geq 4$, then a storage will be triggered. In this case, the miniload crane immediately retrieves the available totes in q_r before moving to the input buffer to take the four totes to be stored from q_s. As such, time out is avoided, albeit less than four totes are retrieved by the crane.

2.2.3 Item-picking

Once a product tote has reached its destination workstation, an operator picks the required amount of items and puts the item(s) into an order tote. An order tote corresponds to one order and there can be more than one order tote for an order. When all items required for an order are already picked into the order tote(s), the totes are moved to the take-away conveyor (see Figure 2.1).

Following item-picking, the operator checks whether the product tote has become empty. If this is the case, the empty product tote is put on the take-away conveyor along with the finished order totes to be sent to a consolidation area. Alternatively, if the product tote still contains any items left, the tote is put on the central conveyor loop to be stored again in one of the miniloads.

The destination miniload for a returning product tote is not necessarily the same miniload from which it has been retrieved. The returning product totes are distributed over the five miniloads in such a way that the contents of all miniloads become as equal as possible. After the destination miniload is determined, the product totes travel to the input buffer of the destination miniload, waiting for the miniload crane to store them into the miniload racks.

2.3 Modeling an AS/RS using Petri nets

Petri nets have often been used in the detailed modeling of AS/RS. Petri nets are powerful for modeling systems with concurrent and asynchronous activities. The Petri-net formalism has an intuitive graphical representation. Both states and actions in a system are explicitly described using this formalism.

The use of Petri nets in detailed modeling of AS/RS for performance analysis is illustrated for instance by Knapp and Wang (1992); Dotoli and Fanti (2005); Lin and Wang (1995); Chincholkar and Chetty (1996); Lee et al. (2006). Knapp and Wang (1992) created a simplistic model of an AS/RS with 4 SKUs using timed Petri nets, assuming exponential distributions of interarrival and service times. Dotoli and Fanti (2005) proposed a modular, colored timed Petri-net based model for an AS/RS serviced by RGVs. Modularity is obtained by modeling each of the physical structures separately. A two-layer hierarchical control structure is developed, involving a scheduler and a resource controller. The control structure is, however, not embedded explicitly in the model architecture. Their simulation model represented an AS/RS with two aisles, 16 storage locations, and two unidirectional storage/retrieval conveyors. Other works on material flow modeling and performance analysis of AS/RS using Petri nets were performed by Lin and Wang (1995); Chincholkar and Chetty (1996); Lee et al. (2006).

Another study by Hsieh et al. (1998) focused on the use of Petri nets in developing a generic AS/RS model structure. They distinguished two components in their model, namely information flow and crane operation. An AS/RS is regarded as a number of unit operation modules, each of which consists of a crane, side racks, buffer stations, and subsystem controller. Using this approach, they argue that a complete AS/RS model of any size can be constructed.

A different formalism to describe concurrent systems is process algebra. Process algebra and Petri nets share two important characteristics (Basten, 1998); first, they have a precise mathematical definition; second, they are both designed for reasoning about concurrent systems. Contrary to Petri nets, process algebra is a pure symbolic formalism, with no explicit representation of system states. Instead, the process-algebraic description focuses on the dynamic behavior of the concurrent system. The small collection of process terms and operators allows for a compact specification.

There is barely any literature describing the modeling of AS/RS using process algebra. We show the application of a process algebra based simulation language χ for creating a modular model architecture of an operating, industrial automated warehouse consisting of miniloads (i.e., AS/RS) and workstations.

2.4 Process algebra and χ

With process algebra the behavior of parallel systems can be described by algebraic means. It provides a framework for formal reasoning about processes and data, where emphasis is given on processes that are executed concurrently (Fokkink, 2000).

Process algebra is invaluable for the study of systems composed of several subsystems or processes that operate in parallel. Each subsystem or process can influence the execution of other subsystems or processes. One may regard a system as a collection of processes connected by numerous communication channels to form the complete system.

The language χ (Chi) is a process algebra dialect. χ was developed for modeling, simulation, and control of concurrent manufacturing systems (Hofkamp and Rooda, 2008). χ provides means to specify so-called process terms and operators on these process terms. The following basic process terms are available:

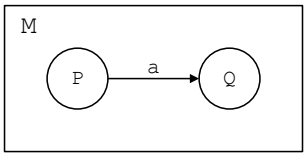
- Assignment
  With `x:= e` the value of expression `e` is assigned to variable `x`.
- Send and receive
  With `a!e` the value of expression `e` is sent over channel `a`. With `a?x` the value of the received object over channel `a` is assigned to variable `x`. Communication over a channel only takes place if the send and receive term can be executed simultaneously (synchronous communication).
- Delay
  With `delay t`, a process delays for `t` time units.
- Print
  With `!!"hello"` the text `hello` is printed to the screen.

There are also the so-called guarded process terms. A guarded process term `b -> p` consists of condition `b` and basic process term `p`. Such a guarded process term is executed provided that condition `b` holds and basic proces term `p` can be executed. This is elucidated in the example of Section 2.6.

Aside of the process terms, the following operators are used:

- Loop
  With `* p` process term `p` is repeated forever.
- While
  With `b *> p` process term `p` is repeated as long as condition `b` holds.
- Sequential composition
  With `p ; q` process term `p` is executed before process term `q`.
- Alternative composition
  With `p | q` process term `p` or process term `q` is executed.
- Parallel composition
  With `p || q` process term `p` and process term `q` are both executed in parallel.

The binding strength of the operators is given from high to low by 1. `*`, `*>` (loop, while); 2. `;` (sequential composition); 3. `|`, `||` (alternative, parallel composition).




```
model M() =
|[ chan a: nat :: P(a) || Q(a) ]|

proc P(chan a!: nat) =
|[ var i: nat = 0
:: *( a!i; delay 2.0; i:= i + 1 )
]|

proc Q(chan a?: nat) =
|[ var j: nat
:: *( a?j; !!time, "\t", j, "\n" )
]|
```

Figure 2.4: Processes `P`, `Q`, channel `a`, model `M`.

Figure 2.4 depicts graphically model `M` with two parallel processes `P` and `Q`. The specifications of the model and the two processes are also provided. In the specification of model `M` it is shown that process `P` and process `Q` are connected via channel `a`. The system works as follows. Every 2.0 time units process `P` sends the value of variable `i` via channel `a`. The value of `i` is incremented afterwards. Process `Q` waits for communication with process `P` via channel `a`. After receiving a value of type `nat` (natural), this value is assigned to variable `j`. The current time and the value of variable `j` is then printed to the screen. Execution of model `M` by using a simulator gives the following result:

```
0.0    0
2.0    1
4.0    2
...    ...
```

We refer to van Beek et al. (2006) for a formal description of the operational semantics of χ. The language χ contains a rich collection of basic data types and container data types. Examples of basic types are boolean (denoted by `bool`), natural (denoted by `nat`, with values `0,1,...`), and real. Examples of container data types are array, record tuple, set and list. In Section 2.6 these data types are used. For a definition of the language including the data types, expressions, and statements, we refer to Hofkamp and Rooda (2008). A tutorial of the language is provided by Vervoort and Rooda (2007). A lecture note on analysis of manufacturing systems using this process algebra language is provided by Rooda and Vervoort (2007).

2.5 Model architecture

The model architecture is developed in such a way that modularity is supported. The goal is to create a model where changes to design specifications relating to control heuristics and model structure (for example the number of miniloads and/or workstations) can be made locally with as little influence as possible on the other parts of the model. An additional advantage of this model architecture is that the model is easy to comprehend intuitively.

2.5.1 Areas and layers

In the model architecture we define areas and operational layers. We distinguish three areas and four layers, as shown in Figure 2.5. Here, circles represent processes and arrows represent channels between (two) processes.

Similar to the physical structure of the system, the three areas are miniload, workstation, and conveyor area, respectively. The four operational layers are *order layer, global control layer, local control layer,* and *material flow layer* (see Figure 2.5).

The *order* layer encompasses all operations that are related to the administration

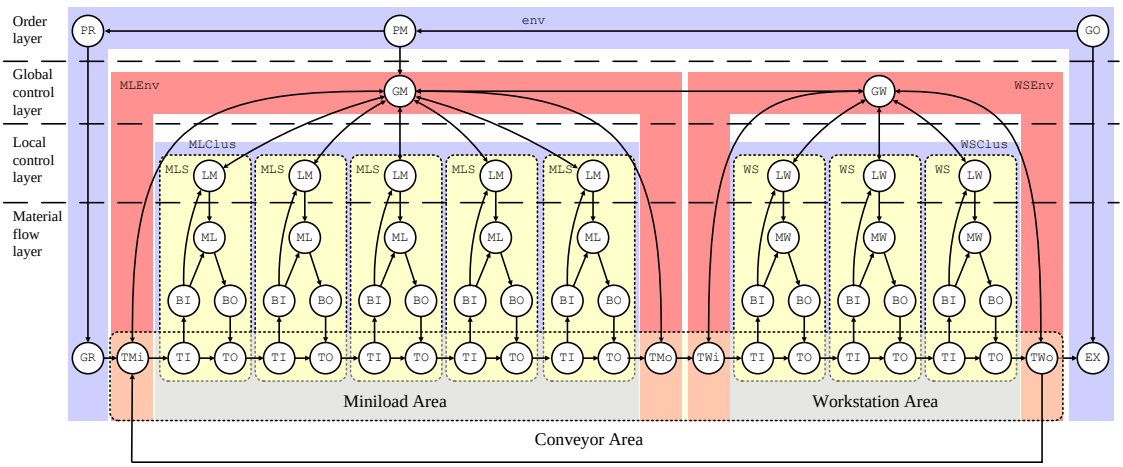


Figure 2.5: Model architecture.

of demand and supply. These operations include the creation of new customer orders by order generator GO and the placement of inventory replenishment orders by replenishment planner PR. To do this, GO maintains historical data regarding the number and composition of order lines, while PR keeps up-to-date data on inventory positions and reorder points of all SKUs. The arriving customer orders are delivered to the miniload area by miniload planner PM.

The *control* layer contains processes that record all relevant information used for decision-making in each area within the system. This layer is further divided into *global control* and *local control* layers. The difference between the two layers is the scope of information that is accessible in each layer.

The *global controller* holds information over *all* subsystems beneath its supervision. That is, global miniload controller GM possesses up-to-date data on the availability of each SKU across all five miniload subsystems MLS. Similarly, global workstation controller GW contains data on the number of orders present at each of workstation subsystems WS.

The *local controller* contains information pertaining to the specific subsystem within its scope. Local miniload controller LM, for example, has access to information *only* from physical miniload ML under its supervision. The data contained in LM includes the number of items available for each SKU in its corresponding physical miniload. As such, a local controller is not aware of the presence of other local controllers in the system. The same holds for local workstation controllers LW. Here, LW maintains data of product totes that are present at the buffer of its physical workstation MW.

The *material flow* layer represents the physical material (product totes) movement. Processes that belong to this layer include the input and output (I/O) buffers BI and BO, I/O conveyor TI and TO, and the physical miniload and workstation ML and MW, respectively. Note that the I/O buffers and I/O conveyor are present both in the miniload and workstation areas.

Processes TMi, TMo, TWi, TWo, TI and TO in the model altogether form the conveyor area. Note that the conveyor area is treated differently from the other two areas. Controllers for the conveyor area are integrated in the controller for the miniload and workstation areas. The conveyor requires information about the destination miniload or workstation for the totes. This information is contained in the totes themselves. As such, there is no need to model a separate controller for the conveyor.

2.5.2 Decentralized control structure

According to Sandell Jr. et al. (1978), the presupposition of centrality fails to hold in large systems due to either lack of centralized information or lack of centralized capability. An appealing alternative is to utilize *decentralized control* instead. As outlined by Anderson and Bartholdi III (2000) from industrial case studies, advantages of utilizing decentralized control include cheap processing units, local information in the processing units, cheap data collection, simple data processing algorithms, quick data processing, robustness to system failures, and real-time operation of the processing units. In our model, a decentralized control is implemented in a hierarchical multi-layer architecture. Each controller is responsible for making decisions within

its own scope based on the communication with the surrounding processes.

Different types of decisions are made in each layer of the hierarchical control structure. For instance, global miniload controller GM makes a decision about which order will be completed next. To make such a decision, GM maintains a list of available orders, a list of available SKUs from all five LMs, and a list of available workstations from GW. GM also decides which of the five miniloads is assigned with jobs from the new order. This decision is made based on information provided by all five LMs about which miniload contains the oldest tote of the SKU required by the jobs. The jobs are then assigned to that miniload. Finally, GM decides in which miniload a returning tote is stored.

For one particular miniload, local controller LM decides which action (storage or retrieval) is taken by the physical miniload. It also decides when the action is executed. The decision is based on real-time data about the length of storage and retrieval queues (contained in BI and LM, respectively).

Similarly in the workstation area, global workstation controller GW decides whether a new order can be released to the workstations. This decision is based on information about the total number of orders currently active in the workstations. Local workstation controller LW determines to which of the three buffer lanes an arriving tote will be put. A stacking algorithm has been created for this purpose (see Jordan (2008)).

We argue that information can be utilized efficiently in such decentralized, autonomous control. Only relevant information for decision making is communicated between processes. Communication events happen exactly at the decision moments with as few communications as possible.

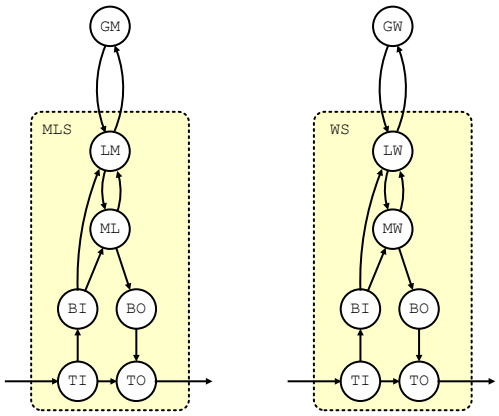
2.5.3 Subsystems

Increased modularity is also gained from creating miniload and workstation subsystems. MLS and WS consist of a number of processes that together represent respectively one miniload and one workstation (see Figure 2.6). The modularity of the model as reflected by the subsystems provides scalability. The number of subsystems such as MLS and WS can be easily increased or decreased, and the respective global controllers GM and GW can be easily adjusted.

Note that Hsieh et al. (1998) have also proposed a modular model structure based on Petri nets by creating a so-called unit operation modules. A unit operation module consists of a crane, side racks, buffer stations, and subsystem controller. They argued that the operation module allows them to develop an AS/RS in any size. However, their model structure did not incorporate time. As such, performance analysis could not be conducted for the AS/RS.

2.5.4 Concurrent processes

The underlying concept of concurrent processes can also be seen from Figure 2.6(a). Here, seven processes operate in parallel for a miniload subsystem. Each process



(a) Miniload. (b) Workstation.

Figure 2.6: Miniload and workstation subsystems.

contains its own local variables, which are not shared with other processes. Only relevant data for decision making is communicated between processes.

As an example, GM and LM are located in different layers. Each of these processes bears unique control heuristics. Since the processes are concurrent, altering a control heuristic implemented in, for example GM will not affect LM and vice versa, provided of course that the communicated data types remain unchanged. We elaborate further on this issue in Section 2.6.

2.5.5 Alternative model architecture

An implicit assumption has been made for the proposed model architecture in Figure 2.5, namely that there is a single route on the conveyor which stops at every miniload and workstation. Totes routed to the furthest miniload/workstation always travel through the other miniloads/workstations. This assumption is valid for the system under study.

There are also other configurations where totes heading to one workstation have (partly) a different route than totes heading to other workstations. The routing and traveling of totes from the miniloads to the workstations and vice versa are modeled by the conveyor area in the architecture of Figure 2.5. Depending on the case at hand, the architecture of the material flow through the conveyor may need to be adjusted.

Figure 2.7 shows an example of an alternative model architecture for the conveyor area. This figure can be seen as an excerpt of Figure 2.5 for processes between TWi and TWo in the workstation area. We now account for totes having specific routings depending on the destination workstation. The delay at TWi and each of the TO can be adjusted to include the travel distance. Similar modification may arise for the miniload area if there are also different routes to and from various miniloads. Note that the control structure remains exactly the same. Since each tote carries its own destination information, no changes to the control structure are necessary.

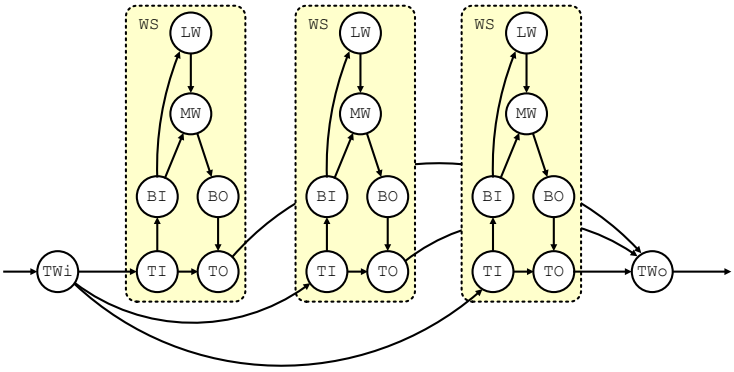


Figure 2.7: Alternative model architecture: conveyor area.

2.6 Modeling of local miniload controller LM

In this section we present a χ model of a part of the miniload system, namely the local miniload controller LM. For an overview of the entire model in χ we refer to Andriansyah et al. (2008). As shown in Figure 2.8, LM is connected to global miniload controller GM, input buffer BI, and physical miniload ML. LM receives data via six channels: one connected to GM, one to BI, and four to ML; LM sends data via four channels: three connected to GM and one to ML.

Figure 2.9 shows the specification of process LM in χ. The specification consists of three parts, namely declaration of (channels and variables) parameters (Lines 1 - 6), declaration of variables used locally in the process (Lines 7 - 11) and the body of the process containing the process terms (Lines 12 - 25).

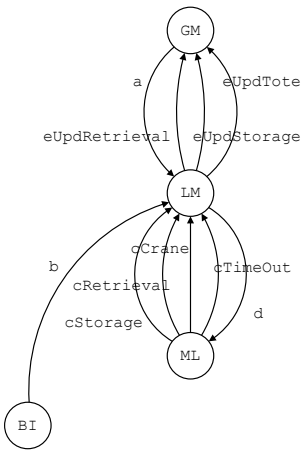


Figure 2.8: Communications of LM.

Parameters and variables (Lines 1 - 11) have a type. We give a few examples. Channel a is a receiving channel (Line 1). A tuple with four fields (4-tuple) are received via channel a. The meaning of the four nat fields is respectively SKU identifier, requested quantity, order identifier and sequence number in the order. Channel b is of type tTote (Line 1). A tTote is a 6-tuple containing a field of type pTote, an order number, a suborder number, a source, a destination, and the number of items to pick. A pTote consists of a 4-tuple, with a tote identifier, a start time of a tote, an SKU identifier and a quantity. Variable skuV is an array of NSKU elements (Line

```
 1 proc LM ( chan a?: (nat,nat,nat,nat), b?: tTote, cTimeOut?: void
 2              , cStorage?: [tTote], cCrane?: bool, cRetrieval?: nat
 3              , d!: (bool,[tTote]), eUpdTote!: (real,nat)
 4              , eUpdStorage!: [(real,nat,nat)], eRetrieval!: nat
 5              , val k: nat
 6         ) =
 7 |[ var skuV: NSKU * [pTote] = initskuV()
 8      , sVec: NSKU * nat = initNatVec()
 9      , x: (nat,nat,nat,nat), m,n: nat, t: real
10      , ys: [tTote] = [], zs : [tTote] = [], z: tTote
11      , rs: [tTote], ps: [(real,nat,nat)], crIn: bool = false
12 :: *( a?x; m:= hd(skuV.(x.0)).qty
13       ; (skuV,sVec,ys,t):= updJobs(skuV,sVec,x,ys,k)
14       ; eUpdTote!(t,m)
15     | b?z; zs:= zs ++ [z]
16     | cStorage?rs; (skuV,sVec,ps):= updTotes(skuV,sVec,rs)
17       ; eUpdStorage!ps
18     | cRetrieval?n; eUpdRetrieval!n
19     | cCrane?crIn
20     | cTimeOut?; d!(crIn,select(ys,zs,crIn))
21       ; (ys,zs):= updQueue(ys,zs,crIn)
22     | operate(len(ys), len(zs)) -> d!(crIn, select(ys,zs,crIn))
23       ; (ys,zs):= updQueue(ys,zs,crIn)
24     )
25 ]|
```

Figure 2.9: Process definition of LM.

7). Every element consists of a list of pTote. Variable skuV is initialized by function initskuV, which is not further explained here (see Andriansyah et al. (2008) for more details). Variable x (Line 9) is of 4-tuple type similar to the type of channel a. Variable zs is a list containing objects of type tTote. This is denoted by [tTote], meaning a list of objects of type tTote (Line 10). Variable z is of type tTote (Line 10). Objects can be added to a list using concatenate operator ++ (Line 15).

Process LM can execute seven different subtasks as described in the body of the process. These subtasks are represented by an alternative composition operator (Lines 12 - 23). This alternative composition is repeated forever by a loop operator (Line 12). An informal explanation of the seven subtasks in the alternative composition operator is:

1. LM receives a retrieval job from GM via channel a. Then LM reserves one product tote for retrieval, and subsequently updates GM via channel eUpdTote how many items of this SKU are available in this tote (Lines 12 - 14). (If the number of items is less than the requested number, GM generates a new retrieval job with the remaining number of items).

2. LM receives data from BI via channel b indicating a tote that needs storage has just arrived at input buffer BI. LM adds this tote to the list of totes to be stored (Line 15).

3. LM receives from ML a list with maximal four totes that recently have been stored in ML. These totes are now available for retrieval. LM updates its status accordingly and informs GM (Lines 16 - 17).

4. LM receives from ML the number of totes that ML has retrieved. LM informs GM that new storage locations now have become available in ML (Line 18).

5. LM receives from ML an update of the position of the miniload crane (Line 19). The crane position is used in the subsequent subtasks for the decision between storage and retrieval action.

6. LM receives from ML a time-out signal. This signal indicates that ML has been idle for a certain time interval. Subsequently LM sends a list of totes to be stored or retrieved to ML and updates the storage and retrieval queues (Lines 20 - 21).

7. LM sends to ML a list of totes to be stored or retrieved once a complete batch with totes has become available. Accordingly LM updates the storage and retrieval queues (Lines 22 - 23).

This example shows how communication takes place between concurrent processes as modeled in χ and how decisions are made based on local information. This feature is particularly relevant for systems with a hierarchical, decentralized control structure, where information are stored locally in each process.

2.7 Experiments

A number of experiments using the proposed model architecture have been performed to analyze the effect of different control heuristics on the order throughput and the mean order flow time.

2.7.1 Control heuristics

In the first experiment, we show how different control heuristics can be implemented locally and we show what effects the new control heuristic has on the system performance. Two scenarios regarding the retrieval of product totes are considered in this experiment.

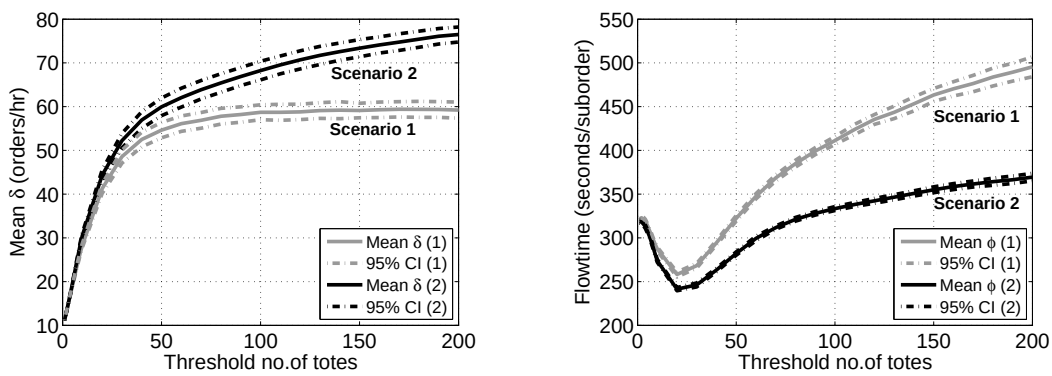
In the first scenario, product totes that belong to the same order are retrieved first. The retrieval queue contains all totes for one order followed by all totes for the next order, and so on. Since the miniload retrieves totes in FCFS sequence, the miniload retrieves all totes belonging to one order before retrieving totes from the next order.

In the second scenario, the miniload is able to retrieve totes from multiple orders simultaneously. This is advantageous because an order can contain up to 316 order lines, which means that a lot of totes might be needed just to fulfill one order. If all miniloads are busy retrieving totes only from this one order, then only one workstation will be busy with item-picking. This is due to the fact that all totes from an order must be sent to the same workstation. Hence, we argue that by allowing a miniload to retrieve totes from three orders simultaneously, the order throughput and the mean order flow time improve significantly.

Experiments are performed in the following manner. All five miniloads are empty each time a simulation run is started. These miniloads are then filled with totes from 1624 SKUs until the number of items for each SKU reaches the predetermined maximum level, which is the order-up-to level S in the (s, S) replenishment policy. This period is called the initialization phase. When the last tote during the initialization phase has entered the miniload, the processing of orders is started. Ten simulation runs have been performed where each run lasts until 10,000 orders have been processed.

We compute the order throughput and the mean order flow time to compare the performance of both scenarios. Order throughput is defined as the number of orders completed per hour. Order flow time (in seconds per order) is measured from the moment the order is released until the last product tote of that order leaves the workstation. Once the flow times of all orders in one simulation run are obtained, the mean order flow time is calculated. The results are depicted in Figures 2.10(a) and 2.10(b). In these figures, the threshold number of totes refers to the maximum number of product totes allowed to leave the miniload. If the number of totes reaches the maximum threshold, then no new order is allowed to be released. This threshold limits the traffic intensity of the system.

Figure 2.10 suggests that different retrieval heuristics indeed affect the system performance significantly. It is noteworthy to highlight that the changes only take place locally at process `LM`, while the rest of the model stays the same. Any other heuristics can be implemented locally as long as the data type communicated to the surrounding process is not altered.



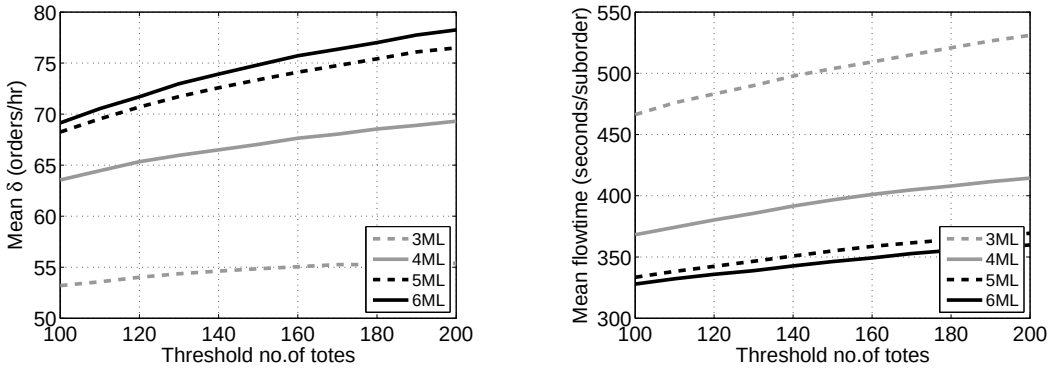
(a) Order throughput. (b) Mean order flow time.

Figure 2.10: Simulation results from systems with different control heuristics.

2.7.2 Number of miniloads

The second experiment involves changing the number of miniloads. In the model, constant identifiers are used to define the number of miniloads, the number of workstations, SKUs, miniload storage capacity, maximum traffic intensity, batch size for storage/retrieval, and the maximum number of active orders. Miniloads can be added/subtracted from the system by changing the value of the number of miniloads.

The results of changing the number of miniloads are depicted in Figures 2.11(a) and 2.11(b). It is clear that the largest improvement with regards to order throughput and mean order flow time is gained when increasing the number of miniloads from three to four when three workstations are present. The information derived from the figures is helpful during the design phase, for example to decide upon the number of miniloads required to achieve a certain throughput.



(a) Throughput various ML.

(b) Flow time various ML.

Figure 2.11: Simulation results from systems with various MLs.

2.8 Conclusions

A simulation model for an end-of-aisle OPS has been presented. The model is realized using the process algebra-based simulation language χ. The compactness of the χ specification has been illustrated. A decentralized hierarchical model architecture has been applied. The proposed modular architecture allows easy incorporation of different system structures, design parameters, and control heuristics.

The process algebra-based language χ is highly suitable for modeling AS/RS systems including a multi-layered control architecture. Information contained in processes is local, whereas the data exchange between the parallel processes is modeled using communication statements. Since a local controller is contained in each of the physical miniload/workstation subsystems, changes to the system structure can be made easily by adjusting the interface between the local controller of the particular miniload/workstation subsystem to the corresponding global controller. The structure also allows various control heuristics to be implemented locally. Finally the obtained model can be used for simulation studies. This has been shown in an example.

The following chapters focus on the performance analysis of order-picking workstations. To begin with, an aggregate model of an order-picking workstation with first-come-first-serve processing of products and orders is discussed.

3

Aggregate modeling of a single-order workstation

R. Andriansyah, L.F.P. Etman, and J.E. Rooda,
Flow Time Prediction for a Single-Server Order Picking Workstation using Aggregate Process Times,
International Journal on Advances of Systems and Measurements (2010), 3 (1&2): 35 - 47.

Also partly presented at SIMUL 2009 Conference, Porto, Portugal, September 20-25, 2009.*

Abstract | A simulation modeling approach based on aggregate process times is proposed for the performance analysis of order-picking workstations in automated warehouses with first-come-first-serve processing of orders. The aggregate process time distribution is calculated from tote arrival and departure times. The aggregate process time is referred to as the effective process time. An aggregate model uses the effective process time distributions as input to predict tote and order flow times. Results from experimental settings show that the aggregate model accurately predicts the mean and variability of tote and order flow times. As a case study, an aggregate model is developed to predict flow times for a real, operating order-picking workstation. The resulting flow time predictions give satisfactory accuracy for both tote and order flow times. Meaningful insights are obtained for performance improvement.

3.1 Introduction

Order picking has been identified as the most expensive process in a warehouse. It is estimated that 55% of the total warehouse operating expenses is caused by order-picking only (Tompkins et al., 2003). Even in automated warehouses, order-picking

*The conference paper received the SIMUL 2009 best paper award.

remains a very capital intensive operation (Goetschalckx and Ashayeri, 1989). This fact alone highlights the importance of performance analysis and improvement of order-picking systems.

In this chapter we consider a parts-to-picker, end-of-aisle, unit-load order-picking system (Roodbergen and Vis, 2009), with *totes* as unit-loads. An AS/RS (Automated Storage/Retrieval System) is used to retrieve product totes from a storage area. The totes are then transported using conveyors to an order-picking workstation. At the workstation, a picker takes the required amount of products from the totes. Afterwards, totes with remaining items are stored back by the AS/RS.

For automated warehouses, the existing literature mostly focuses on the AS/RS (Caputo and Pelagagge, 2006). Koh et al. (2005) developed an analytical model for a miniload with a horse-shoe style buffer. Park et al. (2006) analyzed the performance of a miniload with two-class storage. Bozer and Cho (2005) derived closed-form analytical results to evaluate the performance of an AS/RS under stochastic demand. Hur et al. (2004) developed an $M/G/1$ queueing model to estimate the performance of a unit-load AS/RS. Other references on performance analysis of similar AS/RS are available in the recent review by Roodbergen and Vis (2009).

The order-picking workstation under study can be regarded as a special type of *polling system*, where a number of queues are attended by a single server in a certain order. Several analytical queueing models of such systems exist, see e.g., Eisenberg (1972); Ferguson and Aminetzah (1985); Hirayama et al. (2004); Winands et al. (2006). Typically these methods consider gated or exhaustive service policies, or a combination of the two. Another variation is the $k-$limited polling system where the server continues to work at a queue until either a predefined number of customers k is served or until the queue becomes empty (see e.g., van Vuuren and Winands (2007)). These polling variants, however, do not fully correspond to the order-picking workstation we consider. In our case, a picker always completes an order before starting to pick items for the next order. That is, only one order is processed at a time. Hence, a picker may be idle at one queue (i.e., waiting for the remaining totes to arrive) while other queues are filled with totes.

We present a simulation model for quantifying the mean and variability of tote and order flow times for this type of order-picking workstation. A key aspect of our model is that we do not model in detail the various outages that contribute to the flow time performance. That is, the human pickers, picking faults, setup times, picking equipment failures, etc. are not modeled in every detail. In practice, these are typically difficult to quantify (Rouwenhorst et al., 2000). Instead, we model them by means of an aggregate process time distribution. The idea is that we want to obtain the aggregate process time distribution from tote arrival and departure events of the order-picking workstation in operation. Here we start from the concept of EPT (Effective Process Time) by Hopp and Spearman (2000, 2008) and the concept of measuring EPT distribution from arrival and departure events (Jacobs et al., 2003), using a sample path equation (Kock et al., 2008a).

Gu et al. (2010) concluded in their recent literature review that studies describing validated or applied design models, and practical case studies will give important contributions to warehouse research in the future. This chapter includes an extensive warehouse case study based on data obtained from a real, operating warehouse.

The remainder of this chapter is organized as follows. Section 3.2 describes the order-picking workstation. Section 3.3 describes the simulation model. Section 3.4 elaborates the aggregation method and the EPT measurement method. Section 3.5 discusses a number of validation experiments. Section 3.6 provides a case study to see the performance of the proposed method in a realistic setting. Section 3.7 concludes the chapter.

3.2 System description

Figure 3.1 shows the layout of the order-picking workstation under study. This system can be classified as a *parts-to-picker* system. Pickers work to fulfill *orders*. An order consists of a number of *order lines*. The number of order lines in an order is referred to as the *order length*. Order lengths may vary significantly. Internet orders, for example, typically have a short order length while orders from supermarkets may have a very long order length. An order line represents the required number of *items* from a certain SKU (Stock Keeping Unit).

Some key definitions are as follows. An order represents a customer demand for a number of items from certain SKUs. A *product tote* contains items of the same SKU type. An *order tote* contains all items that are required by an order. It is possible that more than one order tote is needed to fulfill an order, due to the number of required items for that order or the size of the items being picked. In that case we assume the order has been split into smaller orders accordingly, which means that in the remainder of this chapter we assume that every order corresponds to a single order tote. As such, from now on the term tote refers to a product tote, unless when the term order tote is explicitly used.

At the order-picking workstation, arriving product totes form queues on buffer conveyors. Once the picker and the required product tote are available, the product tote will be removed automatically from the buffer conveyor and transported to the pick position where the picker stands. The picker then picks a number of required items from the product tote and puts them in an order tote. The picker works on one order at a time until all lines of the order have been picked and the order is said to be

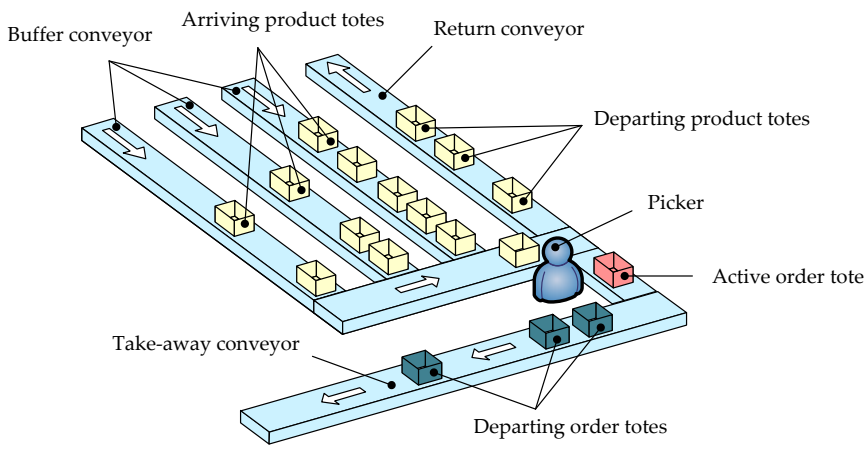


Figure 3.1: Order picking workstation.

finished. When an order is finished, the picker moves the finished order tote to a take-away conveyor that brings the order tote to a consolidation area. If a product tote is not yet empty after picking, the tote will be returned to the storage area using a *return conveyor*.

Order picking workstations have a typical characteristic that distinguishes them from ordinary manufacturing workstations. An order-picking workstation receives a number of product totes for different orders. In the type of system that we consider here, the picker can only pick items from the product totes that belong to the order currently being processed, known as the *active order*. As such, only product totes required to fulfill the active order are sent in a FCFS (First-Come-First-Serve) sequence from the buffer conveyor to the pick position, while all other product totes wait in the queue. If there are no product totes in the queue that belong to the active order, then the picker will be idle although totes for other orders may be present. In this system, totes of three orders may arrive simultaneously at the buffer conveyor. They are sorted such that the picker always has access to the totes of the active order.

Once all order lines of the active order are finished, the next order is processed following a FCFS sequence. Subsequently, product totes for this new order are sent to the pick position. Note that only one active order is allowed in the system under consideration as shown in Figure 3.1. In other order-picking systems it might be possible that more than one active order is processed, allowing the picker to pick items for multiple orders simultaneously. We do not consider this here.

Two performance measures are particularly of interest for this order-picking workstation, namely the (product) tote and order flow times. Tote flow time is defined as the total time spent by a product tote at the order-picking workstation, which starts when the tote arrives at the workstation and ends when it departs the workstation. Order flow time is defined as the time required to complete an order, which starts when the first product tote of an order arrives at the workstation and ends when the last product tote of the order has left the workstation. A complete order means that all items required for the order have been picked into the order tote.

3.3 Simulation model description

Figure 3.2 shows the simulation model representation of the order-picking workstation. The workstation is modeled as a polling system with a single server S and k infinite queues. The queues are denoted by $Q_i,\ i = 0, 1, 2, ..., k-1$. The number of queues k indicates the maximal number of orders for which product totes *simultaneously* arrive at the workstation; that is, order IDs of arriving product totes may be shuffled. Totes arrive with a rate of λ. Each tote has an *id* that denotes the order ID to which the tote belongs. All arriving totes with the same *id* are put into the same queue.

When the first tote of a new order enters a queue, a *gate* is immediately set for that queue. The gate indicates the number of totes required for the order, which equals the order length. The gate is kept open until all totes for the corresponding order have arrived at the queue. Once the last tote of the order has arrived, the gate is closed. In Figure 3.2 an open gate is represented by a dotted line and a closed gate

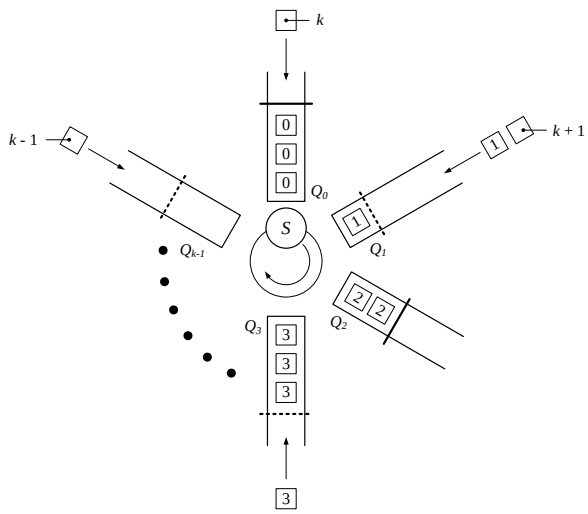


Figure 3.2: Order picking workstation as a polling system.

is represented by a solid line in the queue.

A new order is created each time the gate for another order has been closed. The variable *id* is increased by one and the totes arriving for new order are put in queue Q_i where $i = id$ modulo k. In Figure 3.2, for example, the gates of orders 0 and 2 have been closed and thus two new orders can be started. If the number of queues $k = 5$ (as in Figure 3.2), then the new orders 5 and 6 are put into queues Q_0 and Q_1.

The server attends the queues in a cyclic direction, causing the orders to be served in a FCFS sequence. The server will switch to the next queue only if the gate for the current queue has been closed and all totes in front of the gate have been served. If the server is done processing all totes in front of the gate but the gate is still open, then the server will become idle at the queue. In this case, the server waits until the remaining totes for the queue arrive.

3.4 Aggregation method and EPT measurement

The process time used in this chapter represents an aggregation of all components that contribute to the processing time at the order-picking workstation. We refer to the aggregate process time as the effective process time or EPT for short (see Hopp and Spearman (2008)). Jacobs et al. (2003) presented an algorithm to compute EPT realizations directly from arrival and departure events for infinitely buffered workstations with single-lot processing. Subsequent studies using this concept have been conducted for equipments in manufacturing lines with blocking (Kock et al., 2008a), equipments in assembly lines (Kock, 2008), and batch equipments (Jacobs et al., 2006). The former two studies employed sample path equations to calculate EPT realizations. We will do so here as well.

An order-picking workstation is characterized by several process time components (see Figure 3.3). At the core of the process is the time required for picking items, which is referred to as the raw pick time. In addition to the raw pick time, pickers may require some setup time (change-over time) between processing of orders.

Conveyor systems may break down, causing unavoidable delays. Picker availability is also an issue since it is likely that a picker is sometimes not present at the workstation. In our *aggregate model* (see Figure 3.3) these components are aggregated into a single EPT distribution. The idea is then to reconstruct the EPT distribution directly from tote arrival and departure times registered at the operating order-picking workstation under consideration, with the obvious advantage that one does not need to quantify each component contributing to the process time.

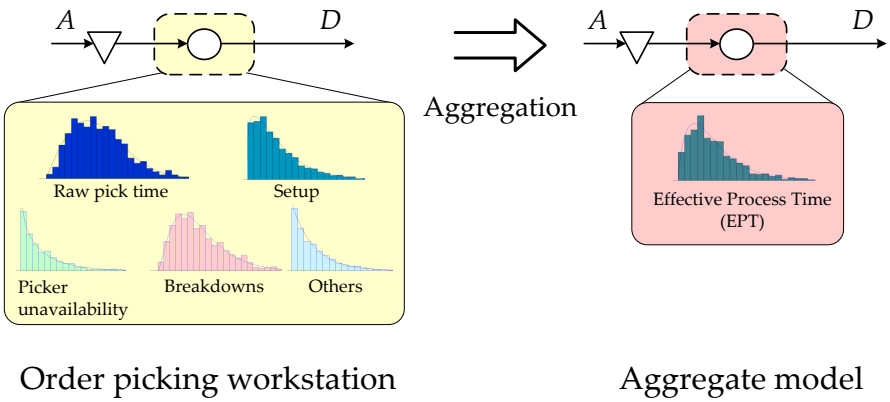


Figure 3.3: Aggregation method.

An EPT realization is calculated for each departing tote, which equals the total amount of time a tote *claims* capacity even if the tote is not yet in physical process. When EPT realizations for all departing totes have been obtained, an EPT distribution with mean t_e and squared coefficient of variation c_e^2 is created. We typically assume a gamma distribution, but other distributions may equally well be used. A gamma distribution is relatively easy to construct since the scale and shape parameters are readily obtainable from the mean and variance of the empirical EPT realizations.

Figure 3.4 shows an example of arrivals and departures of six totes at an order-picking workstation. Totes 1, 2, and 3 belong to order p, Tote 4 belongs to order q, and Totes 5 and 6 belong to order r. An arrival A_i occurs at the moment a product tote i enters the buffer conveyor of the order-picking workstation. A departure D_i occurs when item picking has been finished and the respective product tote i is moved to the return conveyor or to the take-away conveyor (see Figure 3.1).

EPT realizations are calculated using the following sample path equation:

$$\mathrm{EPT}_i = D_i - \max\{A_i, D_{i-1}\} \tag{3.1}$$

here D_i denotes the time epoch of ith departing tote. A_i denotes the arrival epoch of the corresponding i^{th} departing tote. The bottom part of Figure 3.4 illustrates how EPT realizations are obtained using Equation (3.1).

The first tote of an order typically has a different EPT distribution to the other totes in an order. The reason is that each time a picker starts working on a new order, a number of extra activities are performed. These activities include moving the active order tote to the take away conveyor, scanning the barcode of a new order tote to be used for the next order, and placing the order tote at the pick position. Furthermore,

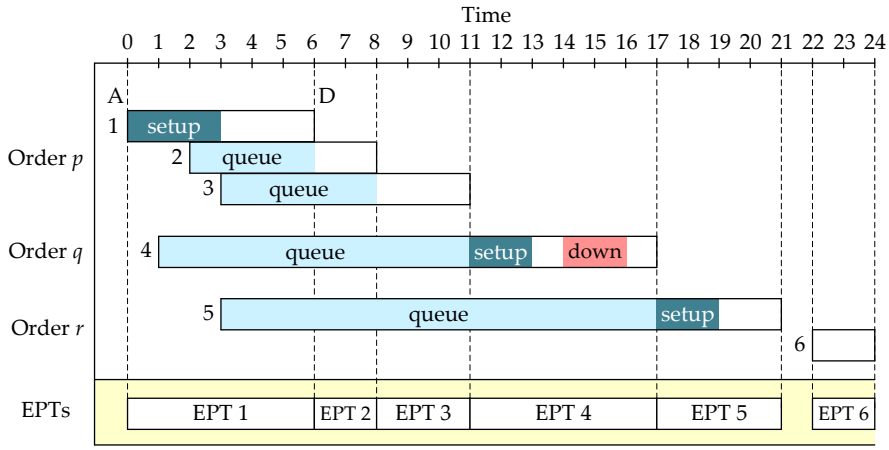


Figure 3.4: Gantt chart example.

pickers may leave their workstations for a break after finishing an order. These are setup activities, which usually only take place in preparation of picking items from the first product tote of a new order. Consequently, EPTs of the first tote typically include a setup time whereas those of the remaining totes do not. Therefore, we sort EPTs into EPTs for the first totes and EPTs for the remaining totes. So in our aggregate model we will use two distribution functions, accordingly, which we refer to as the 1st *tote difference* EPT approach.

3.5 Validation experiments

Validating the proposed aggregate model is done in two steps. The first step is comparing the predicted performance measures from the aggregate model with those of a *detailed model*. The detailed model is also used to validate the sample path equation for calculating the EPT realizations from tote arrival and departure events. The EPT realizations are calculated from simulated arrival and departure events. The main purpose of validation using a detailed model is to show the ability of the aggregate model to accurately predict the performance of an order-picking workstation at different utilization levels. The second step is comparing the predicted performance measures from the aggregate model with those of a real, operating order-picking workstation. In this case, EPT realizations are calculated from real arrival and departure events extracted from the logged data of the operating workstation. The purpose of this validation is to show the application of the aggregate modeling method in practice. This will be discussed as a case study in the next section.

Validation using a detailed simulation model is performed as follows. First, a detailed model is created to be used as a test case. We simulate this detailed model at a certain utilization level (referred to as the *training point*) to generate tote arrival and departure events. Subsequently, these events are used as input for the sample path equation to calculate EPT realizations. Two gamma EPT distributions are constructed namely for the first totes and the remaining totes. Next, we simulate the detailed model at various utilization levels to measure the mean and variability of tote and

order flow times.

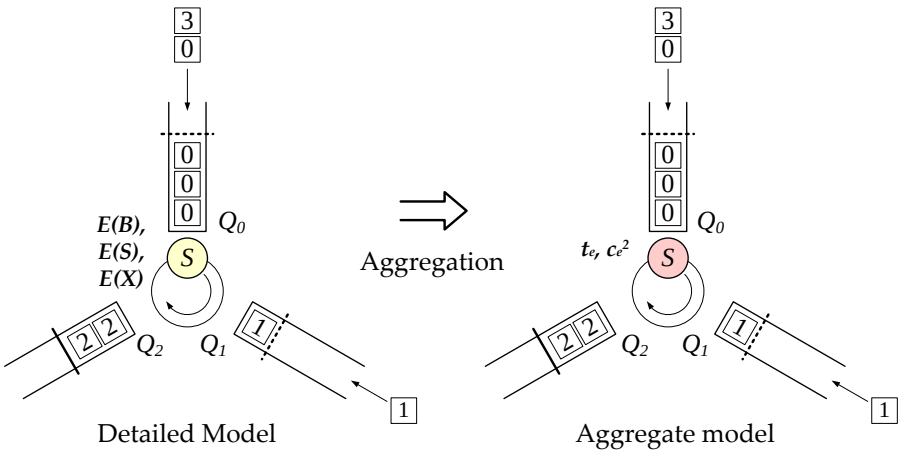


Figure 3.5: EPT-based aggregation.

In the aggregate model, we use the EPT distributions to sample the aggregate process time for the tote being processed. The aggregate model is then simulated at the same utilization levels as the detailed model. We compare the mean and variability of tote and order flow times from the aggregate model with those of the detailed model. In this way, we assess the accuracy of flow time predictions by the aggregate modeling method.

3.5.1 Detailed model

The detailed model represents an order-picking workstation with a number of process time components including raw picking time, setups and disturbances. This system is modeled as a polling system with three queues shown in Figure 3.5. As such, we assume that totes for three orders are generated simultaneously to the workstation, each with an exponential rate of $\frac{1}{3}\lambda$. An assumption is also made regarding the order lengths as provided in Table 3.1. This table shows the frequency of occurrence of orders requiring a certain number of product totes. The mean and standard deviation of the order length are 8.9778 and 9.7423, respectively.

Table 3.1: Data of order lengths and their frequencies.

Size	Freq.	Size	Freq.	Size	Freq.	Size	Freq.	Size	Freq.
1	331	11	80	21	20	31	11	41	11
2	243	12	67	22	20	32	6	42	10
3	257	13	41	23	12	33	5	43	11
4	181	14	24	24	13	34	12	44	12
5	195	15	34	25	7	35	11	45	7
6	208	16	42	26	10	36	12	46	2
7	147	17	19	27	6	37	7	47	3
8	91	18	14	28	12	38	9	48	5
9	134	19	17	29	20	39	8	49	3
10	90	20	27	30	5	40	6	50	1

The server S in Figure 3.5 represents a picker, which is characterized by the mean values of raw pick time $E(B)$, order tote setup $E(S)$, and other disturbances $E(X)$. The raw pick time is assumed to be gamma distributed with a mean of 17.5 seconds and an SCV (Squared Coefficient of Variation) of 0.8. The SCV is defined as the variance divided by the square of mean raw pick time. An order tote setup is performed each time a picker starts working on a new order. We assume that the order tote setup is uniformly distributed between 10.0 and 15.0 seconds. Other disturbances such as incorrect product tote administration, unreadable barcode on the product tote, distraction from other pickers, etc. occur during item picking. These disturbances are assumed to take place on average every 30 minutes, with a duration of on average 2 minutes. Both times are assumed to be exponentially distributed.

This model has been implemented using the process algebra based simulation language χ (Chi) 1.0 (Hofkamp and Rooda, 2008). χ uses a pseudo-random number generator based on Mersenne Twister (Matsumoto and Nishimura, 1998) to generate samples from distributions. But other simulation packages may of course be used as well.

The experimental setup used for the detailed model is as follows. The arrival and departure data is generated in a single simulation run of 1,000,000 totes. To measure the mean and variability of tote and order flow times we perform 30 simulation runs of 300,000 totes including a warmup period of 30,000 totes at utilization levels ranging from 0.30 to 0.95. Based on Welch's graphical procedure (Welch, 1983), such length of warmup period has been found sufficient to accommodate the problem of initial transient for utilization level $u = 0.95$, which is the most variable setting. Additionally, 30 replications are sufficient to generate half-widths of the 95% confidence intervals of the mean tote and order flow times less than 2% of the sample means, at all utilization levels.

3.5.2 Measured EPT

To measure EPT realizations we first generate arrival and departure events from the detailed model at a training point of $0.8\delta_{\max}$, where $\delta_{\max}$ is the maximum throughput of the detailed model. Through simulation we obtain $\delta_{\max} = 0.05$ totes per second. Arrival and departure events of 1,000,000 product totes are then generated. Subsequently, EPT realizations are calculated using Equation (3.1).

We apply the 1st tote difference as explained in Section 3.4. Two EPT distributions with parameters $t_{e,1} = 31.15$ seconds, $c^2_{e,1} = 0.59$ and $t_{e,2+} = 18.69$ seconds, $c^2_{e,2+} = 1.61$ are obtained for the first and remaining totes of orders, respectively. Suppose now we do not apply the 1st tote difference. That is, we do not distinguish between EPT realizations of the first totes and the remaining totes of orders. Without the 1st tote difference we obtain one EPT distribution with parameters $t_e = 20.08$ seconds and $c^2_{\mathrm{e}} = 1.44$.

Figure 3.6 shows the CDF (Cumulative Distribution Function) of EPT realizations with and without 1st tote difference. With the 1st tote difference we obtain two significantly different EPT distributions for the first and remaining totes of orders. Without the 1st tote difference, the EPT distribution of all totes is very similar to the EPT distribution of the remaining totes using the 1st tote difference. This is because

the number of EPT realizations of remaining totes is significantly larger than the first totes.

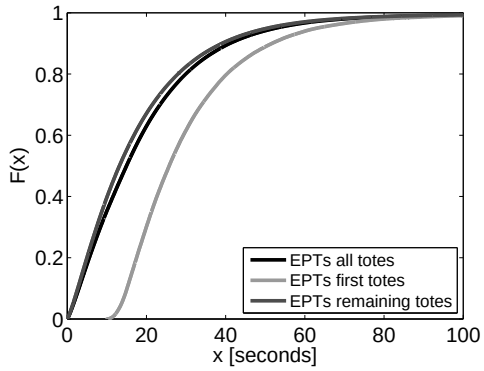


Figure 3.6: CDF of EPT realizations.

3.5.3 Analytical EPT

The mean EPT for this system can be obtained analytically by applying the EPT formula for preemptive and nonpreemptive outages consecutively (Hopp and Spearman, 2008). Other disturbances can be seen as a *preemptive outage* since they occur *during* picking. Order tote setup, on the contrary, is a *nonpreemptive outage* because the setup only occurs *between* picking. Since the formula provided by Hopp and Spearman (2008) assumes no distinction between job types, the resulting analytical mean EPT is as if the 1st tote difference is not applied.

Let t_0, σ_0^2, and c_0^2 be the mean raw pick time, its variance, and its SCV, respectively. The mean EPT t_e, effective variance σ_e^2, and effective SCV c_e^2 after including the preemptive outage are (Hopp and Spearman, 2008):

$$A = \frac{m_f}{m_f + m_r},\ t_e = \frac{t_0}{A},\ c_e^2 = c_0^2 + (1 + c_r^2)A(1 - A)\frac{m_r}{t_0} \tag{3.2}$$

where A is the workstation availability, m_f is the mean time between two consecutive disturbances, m_r is the mean repair time, and c_r^2 is the SCV of the repair times.

Next we include nonpreemptive outage in the EPT calculation. t_e, σ_e^2, and c_e^2 obtained previously become t_0, σ_0^2, and c_0^2. The order tote setup is characterized by the mean setup time t_s, its variance σ_s^2 and the number of jobs between setup N_s (or the mean order length from Table 3.1). Mean EPT t_e and effective SCV c_e^2 after including the nonpreemptive outage are (Hopp and Spearman, 2008):

$$t_e = t_0 + \frac{t_s}{N_s},\ \sigma_e^2 = \sigma_0^2 + \frac{\sigma_s^2}{N_s} + \frac{N_s - 1}{N_s^2} t_s^2,\ c_e^2 = \frac{\sigma_e^2}{t_e^2} \tag{3.3}$$

Applying the above formula with assumptions used in the detailed model, we obtain $t_e = 20.06$ seconds and $c_e^2 = 1.43$. Comparing these values with the measured EPT without 1st tote difference (see Section 3.5.2), we get errors of 0.11% and 0.81% for t_e and c_e^2, respectively. This result validates our method of measuring EPT realizations from tote arrival and departure events.

3.5.4 Aggregate model

The aggregate model comprises a single-server with aggregate process times sampled from EPT distributions. With the 1st tote difference, two EPT distributions with means and SCVs $t_{e,1}$, $c^2_{e,1}$, and $t_{e,2+}$, $c^2_{e,2+}$ are used for the first and remaining totes, respectively. Without 1st tote difference, the EPT distribution has mean t_{e} and SCV c^2_{e}. The sampled aggregate process time represents the duration in which the capacity is claimed by a tote.

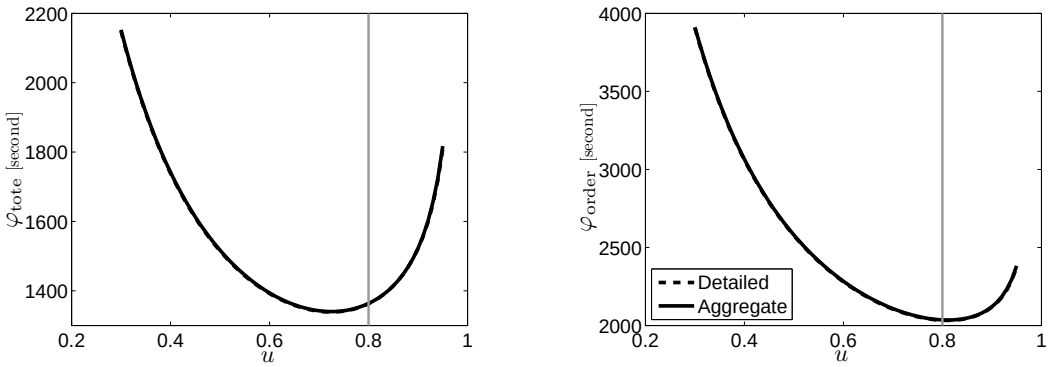
The aggregate model is simulated at the same utilization levels as the detailed model (see Section 3.5.1). At each utilization level, 30 simulation runs of 300,000 totes including a warmup period of 30,000 totes are performed. Based on Welch's procedure (Welch, 1983), this length of warmup period has been found to sufficiently accommodate the initial transient problem of the most variable setting, namely at utilization level $u = 0.95$. Additionally, the resulting half-widths of the 95% confidence intervals of the mean tote and order flow times are less than 2% of the sample means, at all utilization levels with 30 replications. We evaluate the flow time prediction accuracy of the aggregate model with and without 1st tote difference by comparing the flow times from the aggregate model with those of the detailed model.

3.5.5 Flow time prediction

Figure 3.7 shows the tote and order flow time predictions *with* 1st tote difference. The first tote of an order is assigned with an aggregate process time that is significantly larger than the remaining totes (see the values of $t_{e,1}$ and $t_{e,2+}$ in Section 3.5.2). This imposes a longer flow time for the first totes of orders. Therefore the remaining totes have to wait longer before they are processed. Consequently, the aggregate model correctly predicts both tote and order flow times. Errors for mean and variability of flow time prediction are less than 0.5% and 3.0%, respectively for both product tote and order flow times. Note that the vertical lines in Figure 3.7 and 3.8 denote the training point at which the arrival and departure events were generated using the detailed simulation model.

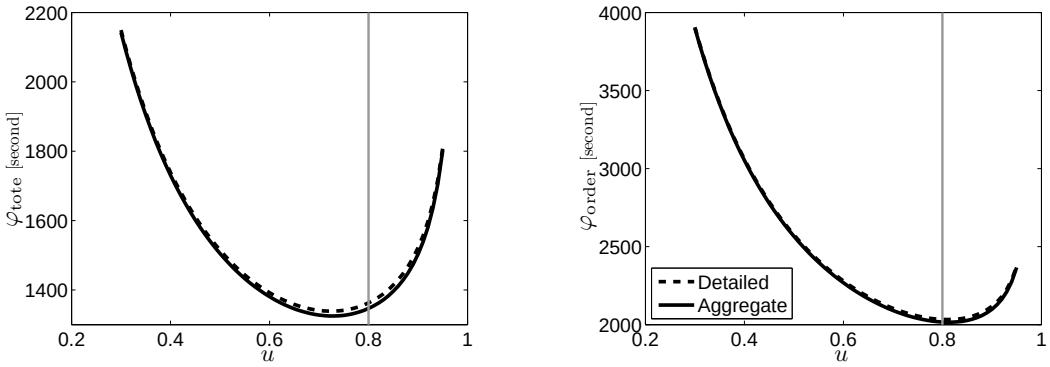
Figure 3.8 shows the tote and order flow time predictions *without* 1st tote difference. Flow time predictions by the aggregate model are consistently lower than the flow times from the detailed model. This observation can be explained as follows. The processing times for all totes in the aggregate model are sampled from an EPT distribution with parameters $t_e = 20.08$ seconds and $c^2_{\mathrm{e}} = 1.44$ (see Section 3.5.2). However, this t_{e} is significantly lower than the mean EPT of the first totes of orders $t_{e,1} = 31.15$ seconds when using 1st tote difference. This causes the aggregate model to underestimate the flow times of the first totes of orders because they are processed much faster in the aggregate model than in the detailed model. The flow times of the remaining totes are affected as well. These totes have shorter waiting time in the buffer and therefore their flow times become lower as well.

Figure 3.9 compares the percentage error in flow time predictions with and without 1st tote difference. It is clear that the 1st tote difference approach increases the flow time prediction accuracy.



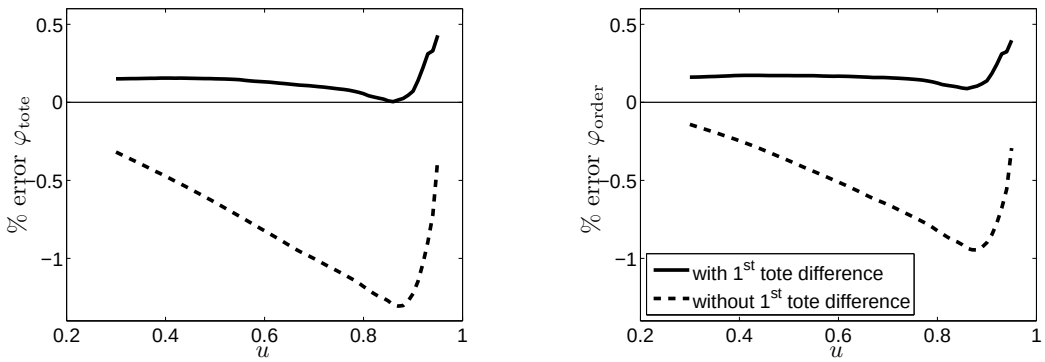
(a) Mean tote flow time. (b) Mean order flow time.

Figure 3.7: Flow time prediction with 1st tote difference at various utilization levels (u).



(a) Mean tote flow time. (b) Mean order flow time.

Figure 3.8: Flow time prediction without 1st tote difference at various utilization levels (u).



(a) Totes. (b) Orders.

Figure 3.9: Percentage error of flow time prediction at various utilization levels (u).

3.5.6 Effect of order length distribution

We investigate the effect of using different order length distributions on the accuracy of flow time prediction by the aggregate model. Geometric and uniform dis-

tributions are used for this purpose. A geometric distribution allows us to model an order pattern with many small orders (e.g., internet orders, slow-moving products) and an order pattern with many large orders (e.g., supermarket orders and fast-moving products). A uniform distribution allows us to model an order pattern with a predefined maximum order length.

The geometric distribution for order length is given by its probability mass function:

$$P\{X = n\} = (1 - p)^{(n-1)}p$$

where n is the order length and two values of p are used namely 0.8 and 0.2. With $p = 0.8$ the order length distribution is short-tailed and most orders will have a size of 1 tote. On the contrary, with $p = 0.2$ the order length distribution is long-tailed and most orders will have a size larger than 1 tote.

For the uniform distribution, we set the maximum order length $n_{\max} = 20$. As such, the probability that an order length takes any value between 1 and 20 is fixed at 0.05.

Each order length distribution is used in a simulation experiment with 30 replications of 300,000 totes and a warmup period of 30,000 totes. Again, we evaluate flow time predictions from two aggregate models namely with and without 1st tote difference. In each simulation replication, the mean and variability of individual flow times are calculated as φ and c^2_φ, respectively. We take the average of all 30 replications to get the mean values of both performance measures $\bar{\varphi}$ and $\bar{c}^2_\varphi$. Subsequently, a two sample t-test at significance level $\alpha = 0.05$ is conducted to compare the mean flow time from the detailed model $\bar{\varphi}_{\mathrm{D}}$ with that of the aggregate model $\bar{\varphi}_{\mathrm{A}}$. The following two-sided hypothesis is tested.

$$H_0 : \bar{\varphi}_{\mathrm{D}} = \bar{\varphi}_{\mathrm{A}}$$
$$H_1 : \bar{\varphi}_{\mathrm{D}} \neq \bar{\varphi}_{\mathrm{A}}$$

The results are shown in Tables 3.2 and 3.3. Prediction errors (in %) are indicated in the columns labeled with % e.

The aggregate model with 1st tote difference predicts the mean flow times of both tote and order significantly better than the one without 1st tote difference. This can be seen from the resulting p-values of the t-test. Without 1st tote difference, the p-values are significant at some utilization levels ($p < 0.05$). At those values we reject H_0 and conclude that the mean flow times from the detailed and aggregate model are different. However, all p-values are not significant for the aggregate model with 1st tote difference. Hence, we cannot reject H_0 and accept that the mean flow time of the detailed model is similar to the mean flow time predicted by the aggregate model.

The errors for flow time SCV are larger for the order length distribution that has high probability of small orders (see columns % e $\bar{c}^2_{\varphi_{\mathrm{A}}}$ and % e $\bar{c}^2_{\varphi_{\mathrm{A1}}}$ in Tables 3.2 and 3.3). For this type of order length distribution, setups between orders are performed more frequently. The error occurs because the gamma distributed EPTs do not correspond fully to the setup time, which is a convolution of a uniform and a gamma distribution. Consequently the errors of flow time variability increase as the EPT distribution is sampled more frequently. In this case, a more detailed fit for the EPT distribution of

the first totes is required. We refer to Blom (2007) for alternative EPT distribution fits. However, if the probability of having small orders is low, then using a gamma distribution is sufficient.

Table 3.2: Tote flow time (in seconds) and its variability.

u	Detailed (D)		Aggregate without 1st tote difference (A)					Aggregate with 1st tote difference (A1)				
	$\bar{\varphi}_{\mathrm{D}}$	$\bar{c}^2_{\varphi_{\mathrm{D}}}$	$\bar{\varphi}_{\mathrm{A}}$	$\bar{c}^2_{\varphi_{\mathrm{A}}}$	% e $\bar{\varphi}_{\mathrm{A}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A}}}$	p-value	$\bar{\varphi}_{\mathrm{A1}}$	$\bar{c}^2_{\varphi_{\mathrm{A1}}}$	% e $\bar{\varphi}_{\mathrm{A1}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A1}}}$	p-value
Geometric $p = 0.8$												
0.3	200.47	2.35	203.47	2.30	1.50	−2.32	**0.00**	200.42	2.34	−0.02	−0.73	0.92
0.4	179.69	1.81	182.82	1.76	1.74	−3.06	**0.00**	179.62	1.79	−0.04	−1.39	0.86
0.5	174.53	1.38	177.71	1.33	1.82	−4.09	**0.00**	174.37	1.35	−0.09	−2.54	0.67
0.6	180.91	1.05	184.06	0.99	1.74	−5.69	**0.00**	180.68	1.00	−0.12	−4.49	0.58
0.7	200.80	0.80	203.87	0.73	1.53	−8.07	**0.00**	200.57	0.74	−0.11	−7.21	0.63
0.8	245.98	0.63	248.88	0.57	1.18	−10.33	**0.00**	246.08	0.57	0.04	−9.24	0.89
0.9	379.11	0.60	381.60	0.54	0.66	−10.03	0.27	380.24	0.55	0.30	−8.44	0.62
0.95	625.63	0.66	630.62	0.62	0.80	−5.61	0.57	637.12	0.63	1.84	−3.70	0.25
Geometric $p = 0.2$												
0.3	1071.00	1.22	1064.40	1.23	−0.61	0.86	**0.03**	1070.40	1.22	−0.06	0.08	0.83
0.4	875.53	1.07	868.49	1.08	−0.80	1.16	**0.01**	874.89	1.07	−0.07	0.07	0.80
0.5	771.43	0.91	763.87	0.93	−0.98	1.43	**0.00**	770.88	0.91	−0.07	−0.02	0.81
0.6	719.18	0.76	710.85	0.77	−1.16	1.71	**0.00**	718.62	0.76	−0.08	−0.17	0.80
0.7	707.24	0.61	697.83	0.62	−1.33	1.87	**0.00**	706.43	0.60	−0.11	−0.43	0.72
0.8	744.15	0.46	732.88	0.46	−1.52	1.49	**0.00**	742.79	0.45	−0.18	−1.32	0.61
0.9	899.89	0.33	886.05	0.33	−1.54	−0.41	**0.01**	896.99	0.31	−0.32	−4.03	0.56
0.95	1184.60	0.31	1173.30	0.32	−0.96	2.79	0.46	1187.20	0.31	0.22	−0.67	0.87
Uniform $n_{\max} = 20$												
0.3	1161.10	0.90	1157.60	0.90	−0.30	0.22	**0.05**	1163.00	0.90	0.16	−0.21	0.28
0.4	938.48	0.80	934.43	0.80	−0.43	0.29	**0.01**	939.91	0.79	0.15	−0.26	0.31
0.5	815.37	0.69	810.86	0.69	−0.55	0.29	**0.00**	816.49	0.69	0.14	−0.38	0.36
0.6	746.40	0.58	741.42	0.58	−0.67	0.18	**0.00**	747.17	0.58	0.10	−0.59	0.51
0.7	716.53	0.48	710.72	0.48	−0.81	−0.09	**0.00**	716.55	0.47	0.00	−0.94	0.99
0.8	730.08	0.37	722.78	0.37	−1.00	−0.90	**0.00**	728.51	0.36	−0.22	−1.79	0.26
0.9	845.82	0.29	836.96	0.28	−1.05	−4.59	**0.01**	841.58	0.28	−0.50	−5.57	0.21
0.95	1089.00	0.31	1089.40	0.31	0.05	0.40	0.97	1087.90	0.30	−0.10	−3.29	0.94

Table 3.3: Order flow time (in seconds) and its variability.

u	Detailed (D)		Aggregate without 1st tote difference (A)					Aggregate with 1st tote difference (A1)				
	$\bar{\varphi}_{\mathrm{D}}$	$\bar{c}^2_{\varphi_{\mathrm{D}}}$	$\bar{\varphi}_{\mathrm{A}}$	$\bar{c}^2_{\varphi_{\mathrm{A}}}$	% e $\bar{\varphi}_{\mathrm{A}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A}}}$	p-value	$\bar{\varphi}_{\mathrm{A1}}$	$\bar{c}^2_{\varphi_{\mathrm{A1}}}$	% e $\bar{\varphi}_{\mathrm{A1}}$	% e $\bar{c}^2_{\varphi_{\mathrm{A1}}}$	p-value
Geometric $p = 0.8$												
0.3	272.23	1.65	275.21	1.64	1.09	−0.75	**0.00**	272.16	1.64	−0.02	−0.56	0.91
0.4	233.51	1.33	236.61	1.31	1.33	−1.17	**0.00**	233.42	1.31	−0.04	−1.14	0.84
0.5	217.59	1.05	220.76	1.03	1.46	−1.96	**0.00**	217.41	1.02	−0.08	−2.23	0.68
0.6	216.79	0.81	219.94	0.78	1.45	−3.44	**0.00**	216.55	0.78	−0.11	−4.20	0.59
0.7	231.56	0.63	234.62	0.59	1.32	−6.02	**0.00**	231.32	0.58	−0.10	−7.12	0.64
0.8	272.85	0.51	275.78	0.47	1.07	−8.99	**0.00**	272.98	0.46	0.05	−9.49	0.86
0.9	402.98	0.52	405.54	0.47	0.64	−9.52	0.26	404.19	0.47	0.30	−8.54	0.60
0.95	648.27	0.61	653.30	0.57	0.78	−5.37	0.57	659.89	0.58	1.79	−3.54	0.25
Geometric $p = 0.2$												
0.3	1917.30	0.42	1911.20	0.43	−0.32	1.29	0.12	1917.20	0.42	0.00	0.15	0.99
0.4	1510.30	0.38	1503.50	0.39	−0.45	1.83	**0.03**	1510.00	0.39	−0.02	0.17	0.92
0.5	1279.30	0.34	1271.80	0.35	−0.58	2.34	**0.01**	1278.90	0.34	−0.03	0.07	0.90
0.6	1142.40	0.30	1134.20	0.31	−0.72	2.91	**0.00**	1142.00	0.30	−0.04	−0.11	0.87
0.7	1070.10	0.25	1060.70	0.26	−0.88	3.46	**0.00**	1069.40	0.25	−0.07	−0.44	0.77
0.8	1061.80	0.20	1050.40	0.21	−1.07	3.36	**0.00**	1060.50	0.20	−0.13	−1.62	0.64
0.9	1182.30	0.16	1168.30	0.16	−1.18	0.91	**0.01**	1179.50	0.15	−0.24	−5.15	0.58
0.95	1452.10	0.19	1440.80	0.19	−0.78	4.00	0.46	1454.80	0.18	0.19	−0.86	0.87
Uniform $n_{\max} = 20$												
0.3	2728.70	0.15	2729.70	0.15	0.04	0.39	0.75	2732.20	0.15	0.13	−0.33	0.25
0.4	2106.20	0.14	2106.40	0.14	0.01	0.56	0.94	2108.90	0.14	0.13	−0.46	0.25
0.5	1742.50	0.13	1742.10	0.13	−0.02	0.69	0.84	1744.60	0.13	0.12	−0.69	0.26
0.6	1512.30	0.11	1511.30	0.12	−0.06	0.75	0.58	1513.90	0.11	0.11	−1.03	0.34
0.7	1366.50	0.10	1364.60	0.10	−0.13	0.60	0.26	1367.10	0.10	0.05	−1.65	0.66
0.8	1292.30	0.09	1288.90	0.09	−0.27	−0.27	**0.04**	1291.10	0.08	−0.09	−3.07	0.45
0.9	1339.30	0.09	1334.00	0.08	−0.40	−5.35	0.13	1335.10	0.08	−0.32	−8.11	0.20
0.95	1553.20	0.13	1556.90	0.13	0.23	1.90	0.81	1551.70	0.12	−0.10	−3.76	0.91

For geometrically distributed order length with $p = 0.8$, the errors for mean flow time prediction without the 1st tote difference % $e\ \bar{\varphi}_{\mathrm{A}}$ are all positive. That is, the predicted flow times from the aggregate model consistently overestimate the flow times from the detailed model. This can be explained as follows. Recall that without 1st tote difference all EPT realizations are collected into a single bucket. In the case of geometric distribution with $p = 0.8$, most orders have a size of 1 tote. Therefore, most EPT realizations are high because EPTs include setup time for orders with size of 1. The resulting EPT distribution has a high mean EPT t_{e}. Since only one EPT distribution is used for sampling the aggregate process time, totes that do not require setup time (remaining totes of an order) also have high aggregate process times. This causes extra waiting for totes in the buffer and consequently higher flow times.

As a statistical note, Tables 3.2 and 3.3 contain a total of 144 settings, each of which has been simulated with 30 replications. The outcome of each setting has been checked using the χ^2 goodness-of-fit test to see if the normality assumption is met at significance level $\alpha = 0.05$. There are only 9 settings that do not fulfill the normality assumption. For these settings, data transformation has been applied to accommodate nonnormality (Hair Jr. et al., 2006). Furthermore, the resulting half-widths of 95% confidence intervals of mean tote and order flow times based on 30 replications are less than 1% of the sample means, except for the settings at utilization $u = 0.95$. At this particularly high utilization level, the maximum half-width of 95% confidence interval is found to be 2.52% of the sample mean.

3.6 Case study

A case study with data obtained from an operating automated warehouse is used to illustrate the applicability of our method in a real warehouse setting. The warehouse shown in Figure 3.10 distributes slow-moving products to a number of supermarkets in the Netherlands. Three processing units are present in the warehouse, namely miniloads, a conveyor loop, and order-picking workstations. Miniloads provide temporary storage spaces for product totes. The conveyor loop transports product totes from the miniload to the order-picking workstations, and the other way around. Three order-picking workstations, with a similar structure as shown in Figure 3.1, are available to process customer orders. We predict the flow time of totes and orders at the order-picking workstation using an aggregate simulation model. These flow times exclude the time spent while retrieving the totes from the miniload and the time spent by the totes while traveling on the conveyor loop. That is, the tote and order flow times start when a tote and the first tote of an order arrive at the order-picking workstations, respectively.

3.6.1 Data processing

The data consists of event logs collected via Programmable Logic Controllers (PLC) from all processing units in the warehouse. From this PLC data we extract tote arrival and departure events at the order-picking workstations. EPT realizations are then calculated using Equation (3.1). Other parameters are also extracted from the

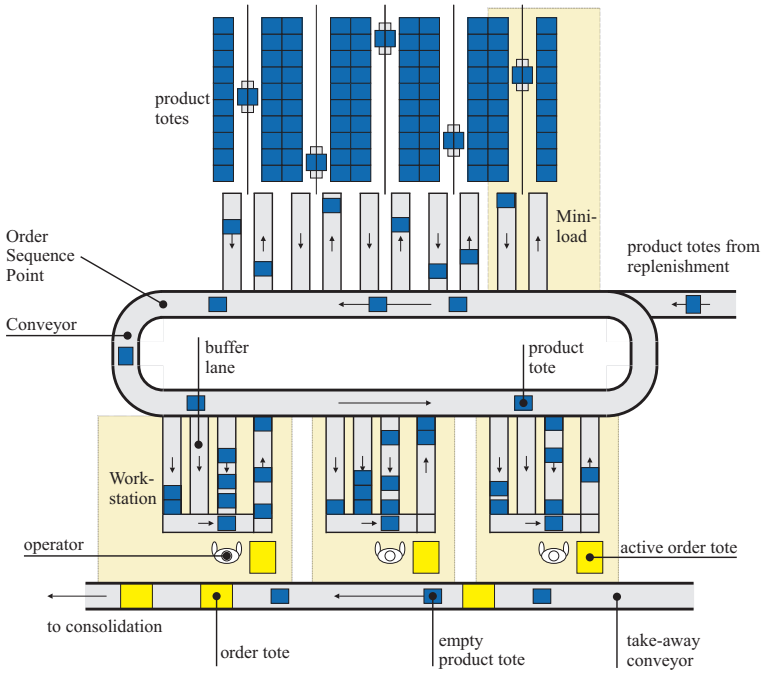


Figure 3.10: Layout of an automated warehouse.

event data, including the interarrival times of totes, order lengths, and the order release strategy. These parameters are the input for the aggregate simulation model to predict tote and order flow times. Subsequently, we compare the predicted flow times with the flow times measured from the data. This demonstrates the prediction accuracy of the method when applied to the data from a real, operating warehouse. Figure 3.11 depicts the flow chart of activities performed in this case study.

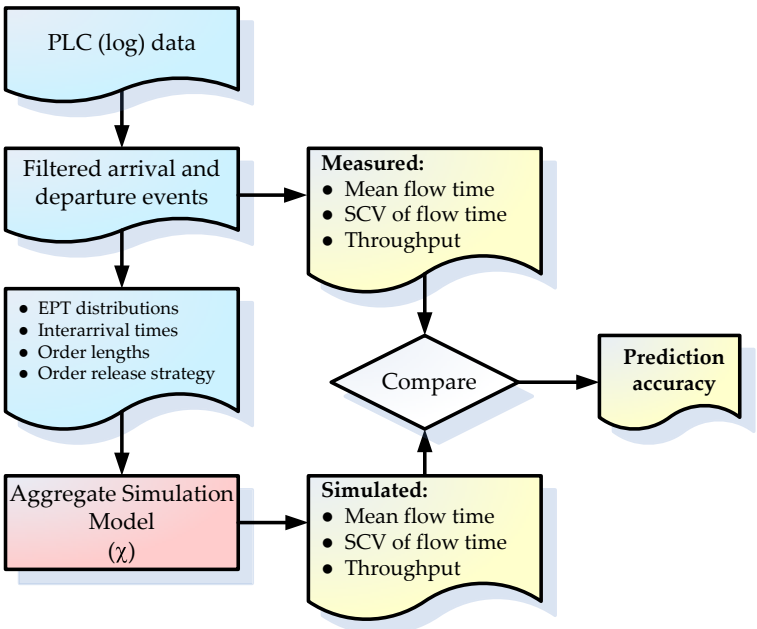


Figure 3.11: Case study flow chart.

Arrival and departure events from three working days are extracted from the PLC data for all three order-picking workstations. An event consists of type (arrival or

departure), time, order ID, order length, and tote ID. Some recorded events may be inconsistent or extreme outliers; e.g., a tote may have an arrival recorded without a departure, or the other way around. Extreme outliers are present when some exceptionally large delays occur between two events, for instance due to lunch breaks. These breaks occur also when there are some totes still waiting in the buffer. The outliers cause very large values for the EPT of the next required tote waiting to be served, the product tote flow times of all product totes in the buffer, and the order flow times pertaining to the totes in the buffer. Therefore, we filter the arrival and departure events to exclude inconsistent events and large delays between two events if they are longer than 60 seconds. This threshold has been chosen based on the observation that it is very unlikely that there is no arrival or departure event at all within 60 seconds from the previous event. Only 2.8% of all arrival and departure events are discarded due to data filtering. The remaining arrival and departure events (97.2%) are used to extract EPTs, interarrival times, order lengths, and order release strategy.

The calculated EPTs are sorted based on the 1st tote difference rule into EPTs of the first totes and EPTs of the remaining totes. We observed that many large EPTs occur when not all totes for the active order are present in the buffer as the picker starts picking. That is, the EPTs are smaller when the picker finds all totes for the active order present in the buffer. This observation holds for both the EPTs of the first totes and the EPTs of the remaining totes. A possible explanation is that pickers tend to leave the workstation when not all totes for the active order are present in the buffer. As such, we further sort the EPTs based on the *completeness* of totes in the buffer when the picker starts picking a tote. This results in four types of EPT as shown in Figure 3.12, namely EPTs first totes complete $(1^{st}, c)$, EPTs first totes incomplete $(1^{st}, i)$, EPTs remaining totes complete $(2+, c)$, and EPTs remaining totes incomplete $(2+, i)$. We use shifted gamma distributions to represent all four EPT distributions because the EPTs can never be smaller than a certain value. Hence, using a shifted gamma distribution with the minimum EPT value as offset should produce a better fit than using a gamma distribution as previously done in Section 3.5. Figure 3.13 visualizes the EPT distributions gathered from workstation 1. The EPTs have been normalized due to data confidentiality. We can see that there are significant differences between all four EPT distributions. It is therefore important to distinguish the EPTs based on the 1st tote difference approach and completeness.

The interarrival distribution can be easily extracted from the tote arrival times at

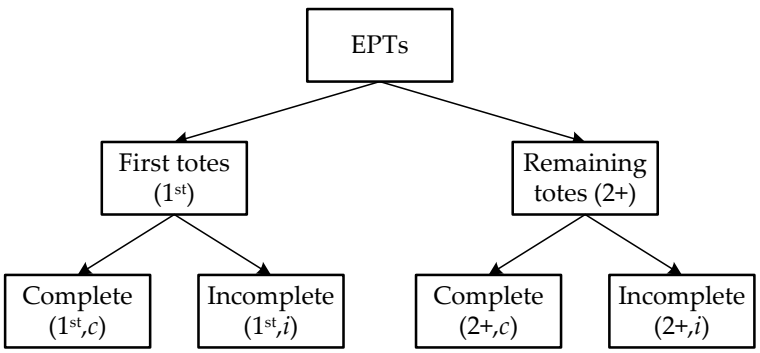


Figure 3.12: Sorting of EPTs.

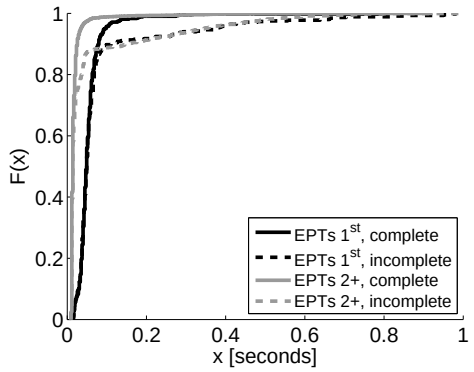


Figure 3.13: CDF of four types of EPT from workstation 1.

the workstation. However, these interarrival times alone are not sufficient. We also need to reconstruct the order release strategy used in the operating warehouse. Such strategy determines to which order the next arriving tote belongs. Since tote and order flow times are affected by the sequence in which totes arrive, the flow time prediction accuracy also depends on how accurate the order release strategy is modeled in the aggregate simulation model as compared to the reality. The order release strategy is reconstructed also from the arrival and departure events, as follows. We create several so-called *buckets* for each order length. When the first tote a new order arrives at the workstation, the order length of that order is registered. Afterwards, we count the number of totes arriving subsequently for this order until the arrival of the first tote of the next order. The resulting number of totes is then collected into the corresponding bucket based on the order length. Next, an empirical distribution function of the number of totes is created for each bucket. These distributions will be used in the aggregate simulation model to sample the number of totes to be generated for the active order before generating the first tote of the next order.

To assess the quality of the reconstructed order release strategy, we compare the interarrival time of orders from the real data with the simulation. The order interarrival time is defined as the time between the arrival of the first totes of new orders. The result is depicted in Figure 3.14. The interarrival times have been normalized due to data confidentiality. The reconstructed order release strategy resembles the order release strategy used in the operating warehouse.

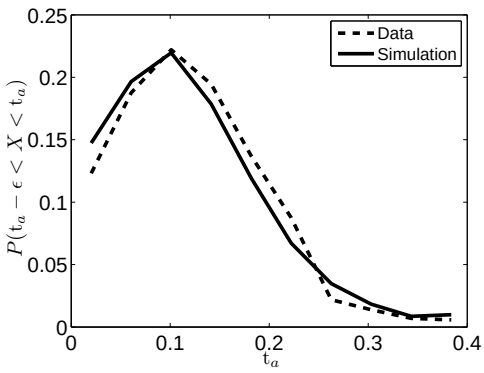


Figure 3.14: Distribution of interarrival times of orders.

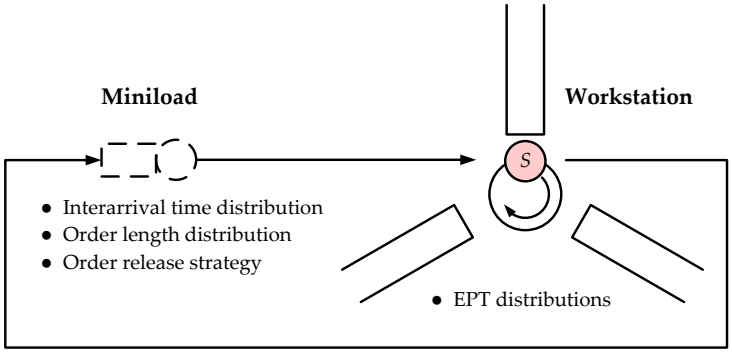


Figure 3.15: Aggregate simulation model of the operating warehouse.

3.6.2 Flow time prediction

The aggregate simulation model shown in Figure 3.15 is used to predict the tote and order flow time of the operating warehouse. We model the system as a closed queueing network with two sequential servers namely the miniload and the workstation. The input parameters used in each server are shown in the figure.

The miniload generates new totes for the workstation if there is a space available in the finite buffer of the workstation. For each tote generated, the miniload determines the interarrival time of the tote, the tote ID, the order ID, and the order length. Since totes from multiple orders are generated simultaneously, the order release strategy reconstructed previously is used to determine the order ID indicating the order to which a tote belongs. The order release strategy works as follows (see also Figure 3.16). Suppose the system is empty and the miniload generates the first tote of order X with an order length of 5 totes. At that moment we say that a new order, i.e., order X, has entered the system. We sample from the corresponding bucket, that is the bucket for order length 5, the number of totes N to be generated for order X before generating the next order. Suppose we sample $N = 3$, this means the next three generated totes will belong to order X. The fourth generated tote belongs to a new order, e.g., order Y. Now two orders are in the system and the miniload will decide based on a certain probability whether the tote generated next belongs to order X, order Y, or another new order. If all totes of order X have been generated, then the value N will be sampled again from the correct bucket based on the order length of order Y. The interarrival times t_a of totes are sampled from the interarrival time distribution.

The workstation is modeled as a polling system with a finite buffer. The EPT for each tote being processed is sampled from one of the four EPT distributions, depending on the type of the tote and the completeness of totes for the active order. For example, if the tote is the first tote of an order and not all totes for this order are present in the buffer at the start of picking, then the EPT for this tote will be sampled from the distribution of EPTs 1st incomplete.

The aggregate simulation model has been run with 50 replications each with a run length of 1,000,000 totes excluding a warm-up period of 300,000 totes for all three workstations. The predicted flow times from the aggregate simulation model are then compared with the real flow times from the data.

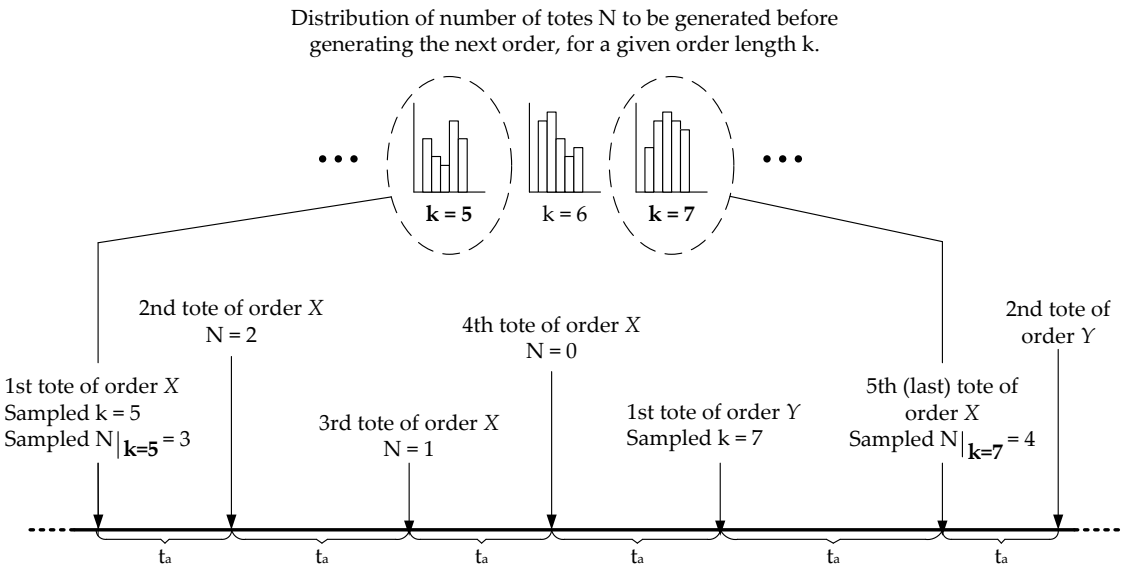
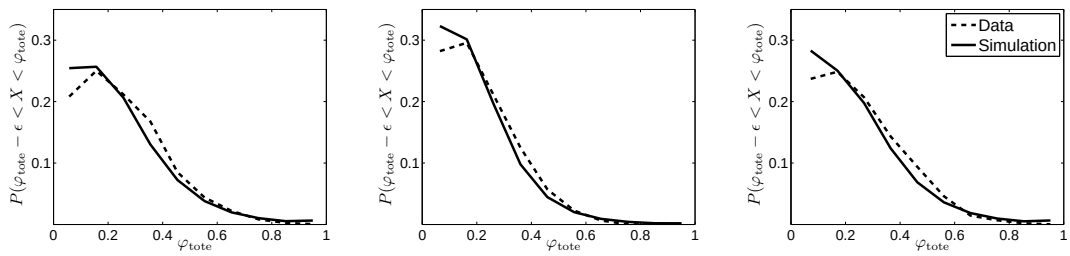


Figure 3.16: Illustration of the order release strategy.

The resulting flow time distributions shown in Figures 3.17 and 3.18 suggest that the aggregate simulation model accurately predicts the flow time of totes and orders. Indeed, the errors of mean tote and order flow time prediction are less than 5.5%. We also observe that the order flow time variability is better predicted than tote flow time variability. That is, in Figure 3.17 the aggregate model consistently overestimates the occurrence of small values of tote flow times. Note that Figures 3.17 and 3.18 show normalized values due to data confidentiality.

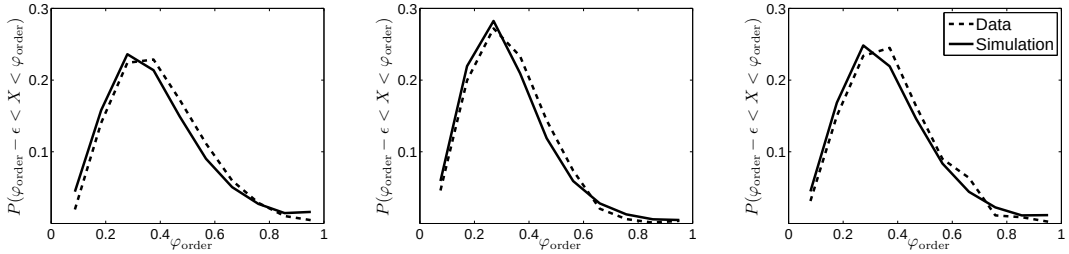


(a) Workstation 1. (b) Workstation 2. (c) Workstation 3.

Figure 3.17: Tote flow time distributions.

3.6.3 Discussion

We found that tote and order overtaking exist in the real warehouse. Overtaking of totes occur when the picker picks a product tote that did not arrive the earliest for the active order. Any time loss due to overtaking is not accounted for in the EPT calculation using Equation (3.1). Overtaking of orders occurs if upon completion of an order the picker processes the next order that is not the oldest in the buffer. We measured the average percentage of tote and order overtaking from the three workstations to be 18.3% and 4.6%, respectively. This may explain the underestimation



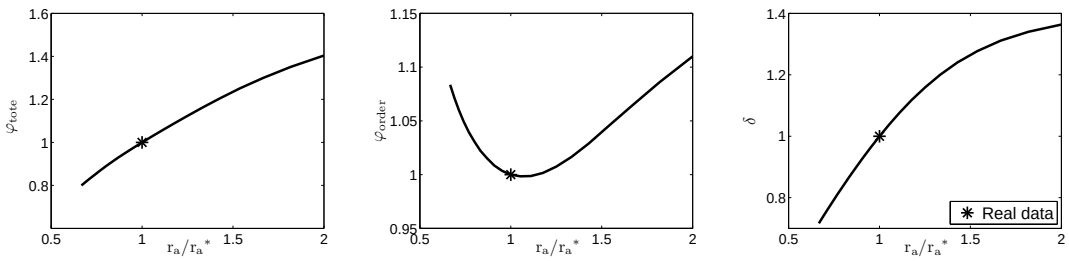
(a) Workstation 1. (b) Workstation 2. (c) Workstation 3.

Figure 3.18: Order flow time distributions.

of mean tote and order flow time prediction since overtaking is not considered in the aggregate simulation model.

Another insight from the case study is that pickers hesitate to wait for totes. If totes for the active order are not yet complete in the buffer by the time the picker starts picking, it is very likely that the picker will leave the workstation for a while. Suppose that a tote for the active order arrives shortly after the picker leaves, then the EPT for this tote will include the time when the picker was leaving the workstation. This will cause the EPT for this tote to become very large. To account for this phenomenon, we have sorted EPTs based on the completeness of totes in the buffer. In practice, one may want to improve the order release strategy such that the totes for the active order arrive more frequently than the totes for other orders. One may also consider allowing pickers to work on multiple orders simultaneously, hence reducing the likeliness that the picker becomes idle waiting for the required totes.

In this case study, the aggregate simulation model is able to predict the tote and order flow time with satisfactory accuracy based on real data of three working days. The aggregate model may further be used to analyze the effect of different interarrival rates, order length distributions, order release strategies, etc. on the system throughput and flow times. As an example, Figure 3.19 shows the predicted performance of workstation 1 under various interarrival rates. It is also possible to analyze tote and order flow time distributions at a certain utilization level, given different order profiles.



(a) Mean tote flow time. (b) Mean order flow time. (c) Tote throughput.

Figure 3.19: Predicted performance of workstation 1 under various interarrival rates.

3.7 Conclusions

In this chapter, a method to predict tote and order flow time distributions for a single-server order-picking workstation with FCFS processing has been proposed by means of a simulation model that is based on an aggregate process time distribution. Arrival and departure data of totes are the only input required to calculate the aggregate process time distribution. Inspired by Hopp and Spearman (2008), we refer to the aggregate process time as Effective Process Time. Two types of EPT realizations are distinguished namely for the first totes and for the remaining totes of orders, which we refer to as the 1st tote difference EPT method. We have demonstrated in the simulation validation study that such separation is important because the EPTs of the first totes are not identically distributed with the EPTs of the remaining totes. Therefore the EPTs are sorted into two EPT distributions. Gamma distributions are fitted to these empirical EPT distribution data, but in principle any other suitable distribution may be used. The two gamma EPT distributions are used to sample the aggregate process time in the aggregate simulation model. The proposed method accurately predicts the mean and variability of tote and order flow times.

The method is applied to data obtained from a real, operating warehouse. The flow time prediction by the aggregate simulation model has satisfactory accuracy, even with the relatively small amount of arrival and departure data. Practical insights for performance improvement are proposed based on the observation of EPTs from the real data. The resulting EPT distributions represent the actual pick rate of an order-picking workstation, which for performance analysis purposes can be compared to the expected pick rate. The aggregate simulation model can further be used to evaluate the order-picking workstation's performance under different settings.

The proposed method can also be used for manufacturing workstations processing a number of different product types, where switching from one product type to another requires a setup time (see e.g., Lefeber and Rooda (2006)). Here, the first job of a product type will have a different process time distribution than the remaining jobs. Hence, the 1st tote difference EPT approach used in this chapter is also valuable in other systems.

In the next chapter, an aggregate model of an order-picking workstation with non-FCFS processing of totes and orders will be addressed.

4

Aggregate modeling of a single-order workstation with overtaking

Presented at the 2010 Winter Simulation Conference, Baltimore, USA, December 5-8, 2010.
R. Andriansyah, L.F.P. Etman, and J.E. Rooda, Aggregate Modeling for Flow Time Prediction of an End-of-Aisle Order Picking Workstation with Overtaking,
Proceedings of the 2010 Winter Simulation Conference (2010), 2070-2081.

Abstract | An aggregate modeling methodology is proposed to predict flow time distributions of an end-of-aisle order-picking workstation in parts-to-picker automated warehouses with overtaking. That is, the products and orders are not processed on a first-come-first-serve basis. The proposed aggregate model uses as input an aggregated process time referred to as the effective process time in combination with overtaking distributions and so-called decision probabilities, which are all measured directly from product arrival and departure data. Experimental results show that the predicted flow time distributions are accurate, with prediction errors of the flow time mean and squared coefficient of variation less than 4% and 9%, respectively. As a case study, we use data collected from a real, operating warehouse and show that the predicted flow time distributions resemble those measured from the data.

4.1 Introduction

An automated warehouse is a network of processing units that store, transport, and consolidate vast amounts of products. Some typical processing units in such a warehouse are automated storage/retrieval systems, automated transport systems (e.g., conveyors, automated guided vehicles), and order-picking workstations. For such a system, performance evaluation is essential to provide feedback about how a specific

design or operational policy performs compared with the requirements, and how it can be improved (Gu et al., 2010).

The current study focuses on performance analysis of one particular processing unit of a parts-to-picker automated warehouse, namely the order-picking workstation. This workstation is a crucial value-adding processing unit that collects numerous products retrieved from the storage area to fulfill customer orders. We are mainly interested in predicting the mean and variability of product and order flow times at such a workstation. Flow time distribution gives an important insight into warehouse reliability in meeting customer due dates.

Some previous works on flow time prediction of different types of order-picking workstations are available. Koo (2009) compared the mean order flow time and the mean order pick rate of a zone picking system, a bucket brigade picking, and a combination of the two called zoned bucket brigade picking. Using a number of simulation experiments, he showed that under certain assumptions the zoned bucket brigade performs better than the other two systems. Yu and de Koster (2008, 2009) developed an approximation method based on a $G/G/m$ queueing network to evaluate the mean order flow time in a pick-and-pass order-picking system. Their method produces accurate prediction of mean order flow time, which is practical for quick evaluation of alternative system designs.

There is only limited literature on performance analysis that considers both the mean and the variability of flow times for an order-picking workstation. A performance analysis method that is able to quantify these two measures will provide better insight into the performance of the workstation. In the previous chapter we proposed a method based on aggregate process time for predicting the performance of an end-of-aisle, unit-load order-picking workstation. Assuming a FCFS (First-Come-First-Serve) processing at the workstation, we were able to predict the mean and the variability of product and order flow times. However, overtaking of products and orders, which often occurs in practice, was not taken into account.

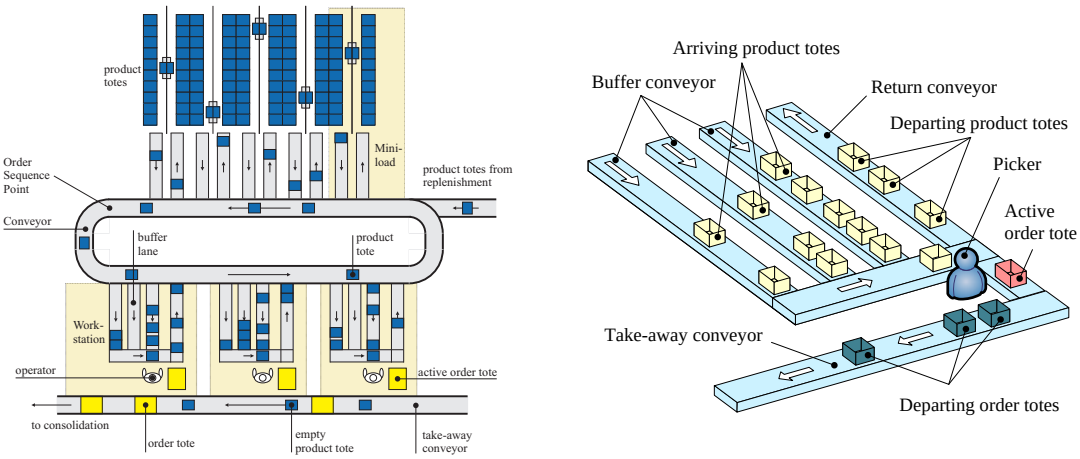
This chapter provides an extension to performance analysis of an end-of-aisle order-picking workstation as presented in the previous chapter. We relax the assumption of FCFS processing, allowing products and orders to overtake at the workstation. A simplified simulation model is proposed that only requires limited data obtainable from the shop-floor. This model is referred to as the aggregate model. The key aspect of this model is that we do not model in detail the stochastic behavior that is typically difficult to quantify (Rouwenhorst et al., 2000), e.g., the picking time, picking faults, setup times, equipment failures, etc. Instead, these are all aggregated into a so-called EPT (Effective Process Time) (Hopp and Spearman, 2008). The EPTs are measured directly from arrival and departure times of products using a sample path equation. Overtaking distributions and a so-called decision probability are also measured. The aggregate model uses the EPT distribution, the overtaking distributions, and the decision probability as input. The aggregate model predicts the mean and variability of product and order flow time with good accuracy.

The remainder of this chapter is organized as follows. Section 4.2 describes the end-of-aisle order-picking workstation. Section 4.3 describes the proposed aggregate model. Section 4.4 elaborates a number of simulation experiments to validate the aggregate model. Section 4.5 presents a case study in which the proposed aggre-

gate modeling methodology is applied to data from a real, operating order-picking workstation. Finally, Section 4.6 concludes the chapter.

4.2 End-of-aisle workstation

A typical layout of a parts-to-picker automated warehouse is shown in Figure 4.1(a). We consider an end-of-aisle, unit-load order-picking workstation as shown in Figure 4.1(b), which is a part of the automated warehouse. Products are delivered in totes via conveyors to the workstation. Product totes required by an order are not necessarily buffered in the same buffer conveyor. A product tote contains items of the same SKU (Stock Keeping Unit). An order consists of a number of SKUs to be picked, which is referred to as the order length. When a picker is idle and the required product tote is present, the tote is sent to the picker. The picker picks the number of required items out of the product tote and puts them in an order tote. Note that the picker processes one order at a time. The order being processed is called the active order, and the product totes belonging to this order are referred to as the *active totes*. If there are no active totes present in the buffer, then the picker will be idle even though totes for other orders may be available. Product totes that still contain items after picking are returned to the storage area. Note that in the remainder of this chapter the term tote always refers to a product tote, unless when the term order tote is explicitly used.



(a) System layout.

(b) order-picking workstation.

Figure 4.1: An automated warehouse with end-of-aisle order-picking workstations.

Tote and order overtaking may occur at this workstation. A tote overtaking takes place when the next tote processed by the picker is not the oldest active tote in the buffer. Two types of tote overtaking are distinguished. Overtaking by an *available* tote happens if the picker processes one of the active totes in the buffer, but not the oldest one. Overtaking by an *unavailable* tote happens when the picker does not pick any active tote, even though one or more active totes are present in the buffer. Instead, the picker waits for another active tote and picks this tote upon

its arrival. Similarly, an order overtaking occurs when upon finishing an order the picker processes another order that is not the oldest in the buffer.

4.3 Aggregate model

An aggregate model is proposed as shown in Figure 4.2. It is essentially a single-server queueing system with an infinite buffer. Within the buffer there are a number of infinite queues, each containing totes for the same order. When the server is idle, an active tote in the buffer may be processed based on a so-called decision probability and overtaking distribution. The decision probability gives the probability that an active tote will be processed or not by the idle picker. The overtaking distribution is used to determine which active tote in the buffer will be selected as the next tote to be processed. The server then processes the selected active tote with a processing time sampled from the EPT (Effective Process Time) distribution.

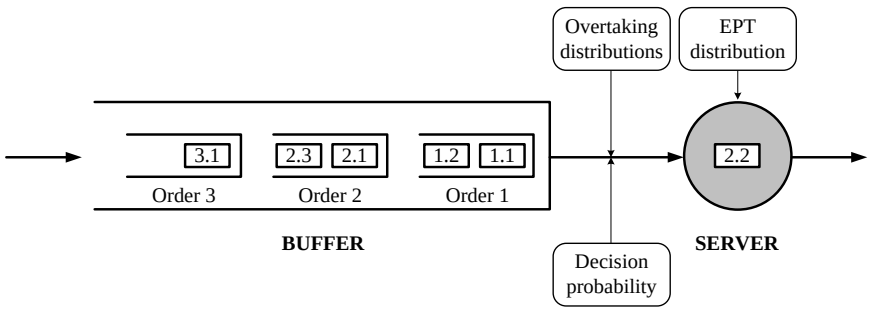


Figure 4.2: Aggregate model of an end-of-aisle order-picking workstation with overtaking.

Figure 4.2 shows an example of both tote and order overtaking. The server is currently processing tote 2.2, which is the 2nd arriving tote of order 2. However, order 1 arrived earlier than order 2 (that is, order 1 is positioned in front of order 2). Hence, order 2 overtakes order 1. Also, the 1st arriving tote of order 2 (tote 2.1) has not been processed yet. As such, the 2nd tote of order 2 overtakes the 1st tote of order 2.

The inputs for the aggregate model are measured directly from tote arrival and departure data of an operating order-picking workstation. Having the inputs, we then use the aggregate model to predict the mean and variability of tote and order flow times of the operating order-picking workstation.

4.3.1 Calculating EPTs

All process time components involved in an order-picking workstation, e.g., raw pick time, setup time, and outages, picker unavailability, breakdowns, etc., are aggregated into a single EPT distribution. The EPTs are obtained directly from arrival and departure data using either an EPT algorithm (e.g., Jacobs et al. (2003)) or using a sample path equation (e.g., Kock et al. (2008a)), depending on how the aggregate

model is defined. For the aggregate model proposed here, we will use a sample path equation to calculate the EPTs.

To illustrate how EPTs are calculated, Figure 4.3 shows an example of arrivals (A) and departures (D) of three active totes. There is no tote overtaking in this example; totes are processed in a FCFS sequence. The EPTs of the three totes are depicted at the bottom part of the figure. We use the following sample path equation to calculate the EPTs:

$$\mathrm{EPT}_i = D_i - \max\{A_i, D_{i-1}\}. \tag{4.1}$$

D_i denotes the departure time of i-th departing tote. A_i denotes the arrival time of the corresponding i-th departing tote. Note that the EPT of tote 1 (EPT 1) is comprised of a setup (time 0–3) and picking (time 3–7), while the EPT of tote 3 (EPT 3) is comprised of picking (time 11–13.5 and time 15.5–18) and a disruption (time 13.5–15.5).

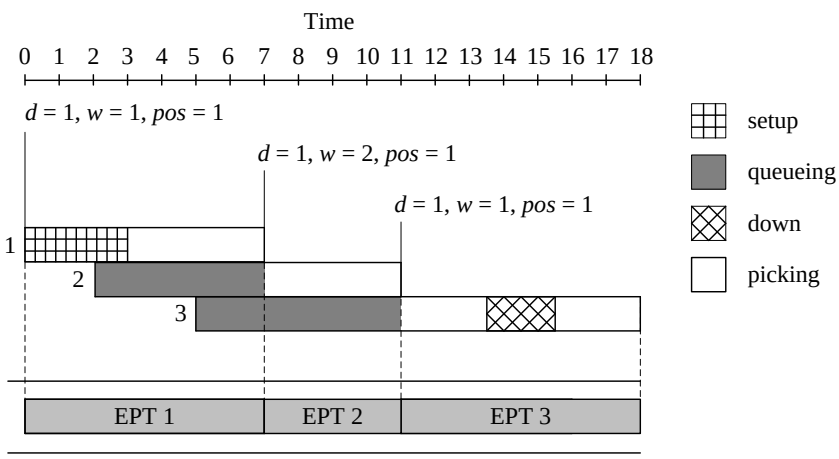


Figure 4.3: Tote-time diagram for active totes with order length $k = 3$, FCFS processing.

An EPT distribution is created when the EPTs of all totes have been calculated. A gamma distribution is used to represent the EPT distribution, but other suitable distributions may be used as well. Processing times of totes in the aggregate model are sampled from the EPT distribution.

4.3.2 Measuring overtaking

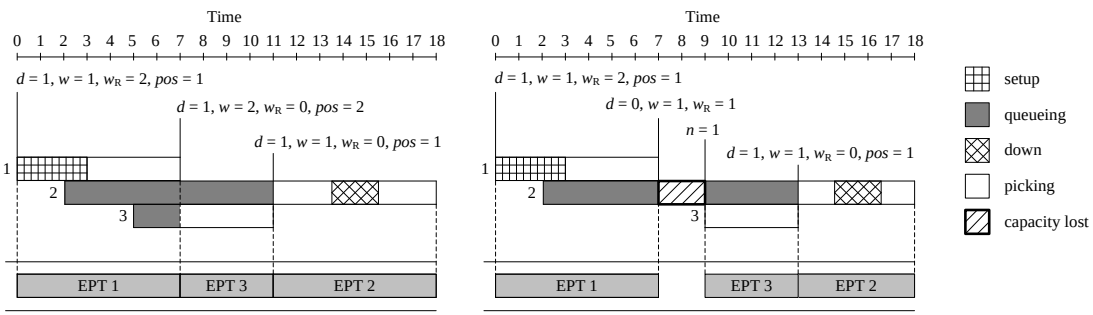
We model tote overtaking by means of WIP (Work-in-Progress)-dependent overtaking distributions. The following variables are used:

w = the number of active totes in the buffer.
w_{R} = the number of remaining active totes not yet arrived.
pos = the position of the active tote in the buffer that is processed next by the picker.

Let us observe the tote-time diagram in Figure 4.3. The picker finishes processing tote 1 at time 7. At that moment, totes 2 and 3 are in the buffer and the next tote

processed is the oldest tote (according to FCFS processing). Hence, at time 7 we have $w = 2$ and $pos = 1$. With the FCFS assumption, the value of pos will always be equal to 1 since the next tote to be processed is always the active tote located at the first position in the buffer.

Relaxing the FCFS assumption, we distinguish two types of tote overtaking, namely overtaking by an *available* tote and overtaking by an *unavailable* tote. Figure 4.4 depicts the two different types of tote overtaking.



(a) Overtaking by an available tote. (b) Overtaking by an unavailable tote.

Figure 4.4: Tote-time diagram for active totes with order length $k = 3$, with tote overtaking.

Let d be the decision of whether or not an active tote is immediately processed once the picker is idle and there is at least one active tote in the buffer, where:

$$d = \begin{cases} 1, & \text{A tote is processed.} \\ 0, & \text{A tote is not processed.} \end{cases}$$

In the case of overtaking by an available tote (Figure 4.4(a)), a tote is *always* processed if the picker is idle and there is at least one active tote present in the buffer. Therefore, we always have $d = 1$. The tote processed next is not the oldest active tote in the buffer. In Figure 4.4(a), tote 3 is processed earlier than tote 2, although tote 2 arrived earlier than tote 3. Hence, tote 3 overtakes tote 2. At time 7 we register $pos = 2$ since the active tote processed next (tote 3) is located at the second position in the buffer (behind tote 2). In general, $pos > 1$ indicates a tote overtaking.

In the case of overtaking by an unavailable tote (Figure 4.4(b)), it is possible that a tote is not processed even though the picker is idle and there is an active tote present in the buffer. In Figure 4.4(b), the picker is idle at time 7 and tote 2 is present in the buffer. However, the picker does not immediately process this tote. Thus, $d = 0$ is registered. Note that two out of three active totes have arrived (given order length $k = 3$). Hence, we register the number of remaining active totes not yet arrived $w_R = 1$ at time 7. Tote 3 arrives at time 9; it is processed immediately. At that moment we register $n = 1$, the number of new active totes for which the picker waits before he starts picking. That is, there is only one new active tote that arrives (tote 3) after $d = 0$ at time 7. Time 7–9 is considered as capacity loss.

Figure 4.5 illustrates order overtaking. There are three orders: P, Q, and R, which arrived at times 0, 1, and 8, respectively. Note that the arrival time of an order is equal to the arrival time of the first tote of that order. Each order consists of two totes. Order R overtakes order Q at time 11. To measure order overtaking, we use the same variables w and pos as in measuring tote overtaking. However, here w represents the number of *orders* in the buffer and pos represents the position of the *order* that is processed next by the picker. These two variables are evaluated every time a picker has finished an order.

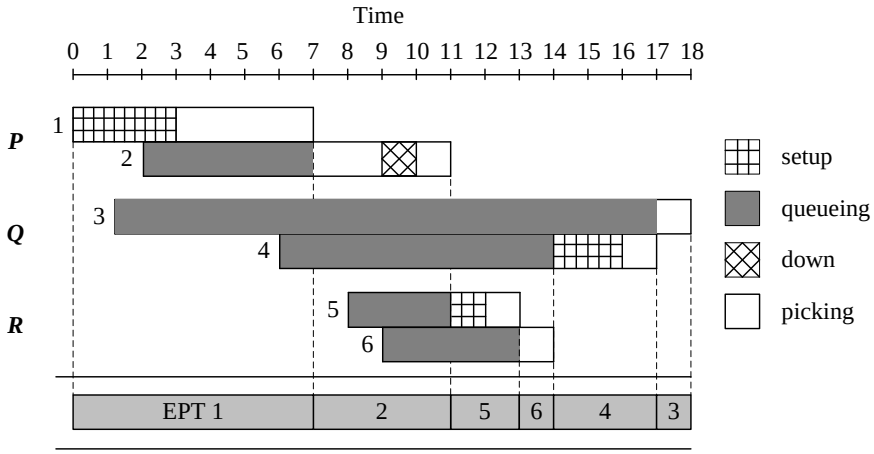


Figure 4.5: Tote-time diagram illustrating order overtaking.

A number of WIP-dependent overtaking distributions and WIP-dependent decision probabilities are created once all tote and order overtaking have been measured. For the overtaking distribution, the WIP level represents the number of active totes (tote overtaking) or orders (order overtaking) in the buffer. For the decision probabilities, the WIP level represents the number of remaining active totes not yet arrived.

4.3.3 Predicting flow times

We simulate the aggregate model with the EPT distribution, WIP-dependent overtaking distributions, and WIP-dependent decision probabilities as input. The mean and variability of tote and order flow times are then obtained.

The aggregate model works as follows. Totes are generated in the aggregate model with an arrival process representing the operating order-picking workstation. If the server is idle and the buffer contains one or more active totes, then a decision is made on whether or not one of the active totes is processed. This decision is based on the value sampled from the correct WIP-dependent decision probabilities. Here, the WIP corresponds to the number of remaining active totes not yet arrived. If it is decided that a tote should be processed, then a WIP-dependent tote overtaking distribution is sampled to determine which active tote will be processed. Here, the WIP corresponds to the number of active totes in the buffer. However, if it is decided that none of the active totes in the buffer should be processed, then we sample the number of active totes that should be waited for. The server stays idle until

the number of arriving active totes is equal to the sampled value. In this way, the capacity losses are explicitly modeled in the aggregate model. The selected tote will be sent to the server to be processed with a processing time sampled from the EPT distribution. If the picker has finished an order, the next order to be processed is determined using the WIP-dependent order overtaking distribution.

The aggregate model gives tote and order flow times as output. Tote flow time is defined as the total time spent by a tote at the order-picking workstation, which starts when a tote arrives at the workstation and ends when it departs from the workstation. Order flow time is defined as the time required to complete an order, which starts when the *first* tote of an order arrives at the workstation and ends when the *last* tote of the order departs from the workstation.

4.4 Model validation

A number of simulation experiments are conducted to validate the aggregate model. First, tote arrival and departure times are needed to calculate the inputs for the aggregate model. Ideally, these arrival and departure times are obtained from a real, operating order-picking workstation. However, for validation purposes we create a simulation model of the workstation in Figure 4.1(b) to generate tote arrival and departure times. These are generated at one utilization level called the *training point*. We refer to this “simulated" operating order-picking workstation as the *detailed* model. Subsequently, the EPT distributions, overtaking distributions, and the decision probabilities are determined from the arrival and departure data. The detailed and aggregate models are then simulated at various utilization levels to compare the flow times from both models.

Some parameters used in the detailed model are as follows. Totes from three orders arrive simultaneously at the workstation according to a Poisson process with a total arrival rate of δ. Once all totes of an order have arrived, the totes from a new order start arriving. Hence, totes from three orders are continuously arriving. We assume a uniformly distributed order length in the range $\{1, 2, ..., 20\}$. Four types of process time are modeled. Raw pick time is the time required to pick a tote. A setup, which includes activities such as moving the active order tote to the take away conveyor, scanning the barcode of a new order tote, and placing the new order tote at the pick position, is performed only when the picker processes the first tote of an order. Disruptions such as incorrect tote administration, unreadable barcode, or distraction from other pickers occur during order picking. Table 4.1 summarizes the values of parameters used in three experiments with the detailed model.

Table 4.1: Parameters for the detailed model.

	Distribution type	Parameter(s)
Raw pick time	Gamma	mean = 17.5 seconds, SCV = 0.8
Setup time	Uniform	min = 10.0 seconds, max = 15.0 seconds
Time between disruptions	Exponential	mean = 30 minutes
Disruptions length	Exponential	mean = 2 minutes

In the first two experiments we create only tote overtaking in the detailed model. It is assumed that upon departure of a tote, all active totes in the buffer and a predefined number of remaining active totes not yet arrived have the same probability to be selected as the next tote to be processed. That is, the next tote to be processed is the x-th arriving active tote, which is selected with the probability

$$p(x) = \frac{1}{N + \min\{n_{\mathrm{T}}, (k-m)\}}. \tag{4.2}$$

In this equation, N is the number of active totes in the buffer, k is the order length of the active order, m is the number of active totes that have arrived at the buffer so far, and n_{T} is a user-defined tote overtaking parameter representing the number of remaining active totes not yet arrived that can also be selected as the next tote to be processed. Note that if all active totes have arrived ($m = k$), the next tote to be processed will be one of the active totes in the buffer (N). Figure 4.6 illustrates tote overtaking in the detailed model with $n_{\mathrm{T}} = 3$ and order length $k = 10$.

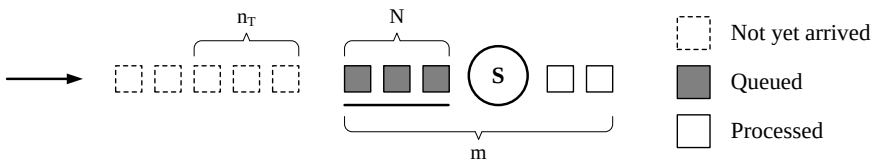


Figure 4.6: Tote overtaking in the detailed model: each active tote at positions $N + n_{\mathrm{T}}$ has the same probability given by Equation (4.2) to be selected as the next tote to be processed by the server S.

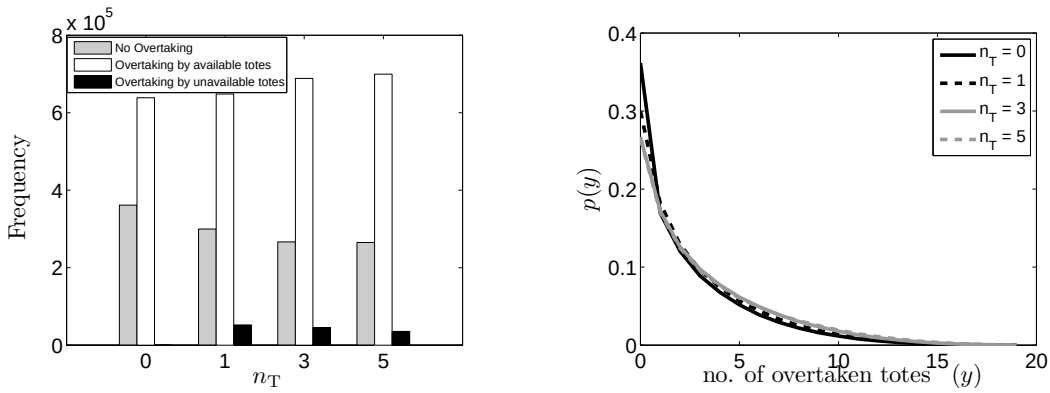
To model the two types of tote overtaking (see Section 4.3.2), we set $n_{\mathrm{T}} = \{0, 1, 3, 5\}$. Overtaking by an available tote is modeled with $n_{\mathrm{T}} = 0$, which suggests that the next tote to be processed may only be selected from the active totes present in the buffer. With $n_{\mathrm{T}} = \{1, 3, 5\}$, we model overtaking by an unavailable tote at different extents.

In the third experiment we create both tote and order overtaking in the detailed model. For the tote overtaking we set $n_{\mathrm{T}} = \{0, 3\}$. For the order overtaking we assume that when an order has been finished, each order in the buffer has the same probability to be selected as the next order.

The experimental setup is as follows. At each value of n_{T}, arrival and departure data of 1,000,000 totes are generated from one simulation run of the detailed model. This is performed at training point $u = 0.8$, where $u = \delta/\delta_{\max}$, δ is the total tote arrival rate, and $\delta_{\max}$ is the maximum throughput. To obtain the mean and variability of flow times, we run the detailed model and the aggregate model at utilization levels $u = \{0.30, 0.32, 0.34, ..., 0.98\}$. Fifty simulation runs are performed at each utilization level for both models. Each run length covers 300,000 totes excluding a warmup period of 30,000 totes.

Figure 4.7 depicts the frequency and probability of tote overtaking under different values of n_{T} from 1,000,000 totes. The frequency of overtaking by available totes increases with n_{T}, while the frequency of overtaking by unavailable totes decreases slightly with larger n_{T} (see Figure 4.7(a)). Yet, larger n_{T} leads to higher probability of overtaking many totes, as more remaining active totes that have not yet arrived

may also be selected as the next tote to be processed (see Figure 4.7(b)).



(a) Frequency. (b) Probability.

Figure 4.7: Tote overtaking under different values of n_T.

4.4.1 Tote overtaking by an available tote

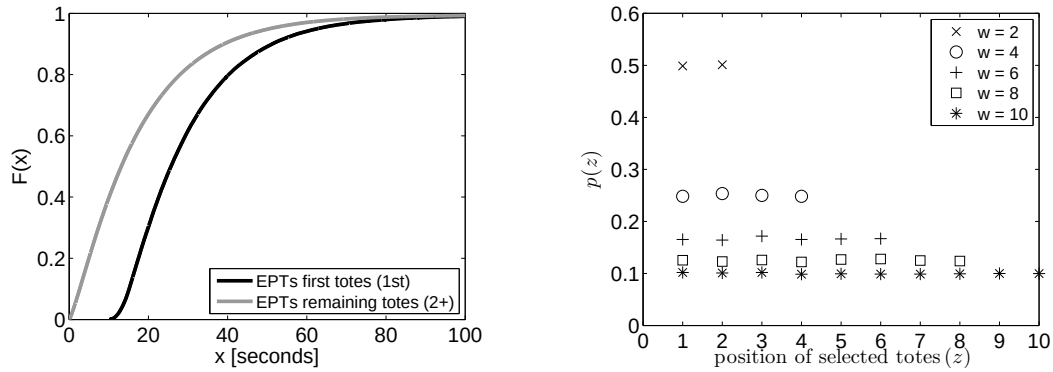
For tote overtaking by an available tote ($n_T = 0$), each active tote in the buffer has the same probability of $1/N$ to be selected as the next tote (see Equation (4.2)).

Figure 4.8 shows the EPT distributions and some of the WIP-dependent overtaking distributions measured from tote arrival and departure data. There are two EPT distributions as shown in Figure 4.8(a), namely the EPTs of the first tote of an order (EPTs 1st) and the EPTs of the remaining totes of an order (EPTs 2+). We sort the EPTs into two distributions because these two sorts of EPTs are not identically distributed. This is because EPTs 1st always contain setup times, whereas no setup time is involved in EPTs 2+. This approach is referred to as the *1st tote difference* EPT approach (see Chapter 3, Section 3.4). The significant difference between EPTs 1st and EPTs 2+ can be seen from their CDF (Cumulative Distribution Function) in Figure 4.8(a).

Figure 4.8(b) depicts tote overtaking distributions for the number of active totes in the buffer $w = \{2, 4, 6, 8, 10\}$. Since each active tote has the probability of $1/N$ to be selected as the next tote according to Equation (4.2), we expect to see a uniform distribution for all values of w. This is verified in Figure 4.8(b), where the active totes at the 1st until the w-th position in the buffer have equal probability to be selected.

In the case of tote overtaking by an available tote, an idle picker always processes an active tote as long as there is at least one active tote in the buffer (see Figure 4.4(a)). This is modeled explicitly in the aggregate model.

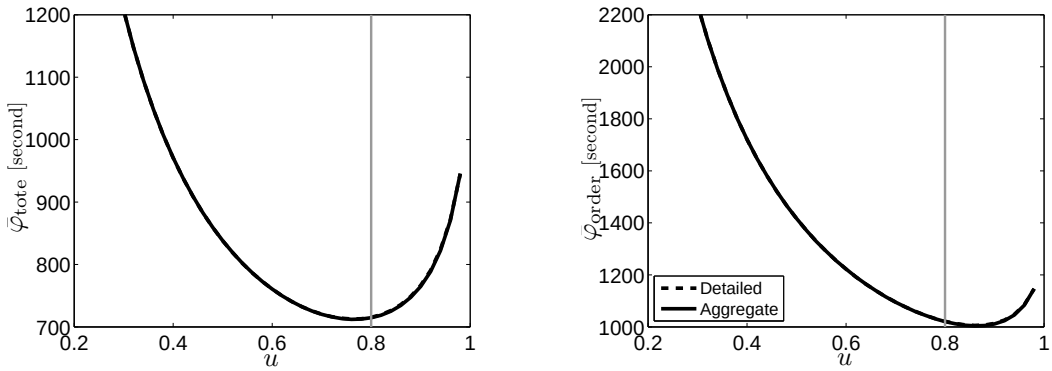
Figure 4.9 compares the mean flow times at various utilization levels u from the detailed model and the aggregate model. The flow time prediction by the aggregate model is very accurate as compared to the detailed model, with prediction errors of less than 0.25% for the mean flow times and 5.3% for the Squared Coefficient of



(a) CDF of EPTs.

(b) Overtaking distributions of WIP levels w.

Figure 4.8: Measured input for the aggregate model with tote overtaking by an available tote.



(a) Tote.

(b) Order.

Figure 4.9: Mean flow times in case of tote overtaking by an available tote.

Variation (SCV) of the flow times at all simulated utilization levels.

4.4.2 Tote overtaking by an unavailable tote

Tote overtaking by an unavailable tote is modeled in the detailed model by setting $n_{\text{T}} = \{1, 3, 5\}$. By having different values of n_{T}, we investigate the effect of intensity of overtaking by an unavailable tote on the flow time prediction accuracy by the aggregate model.

Figure 4.10 depicts the WIP-dependent decision probabilities $p(1)$ of processing an active tote in the buffer when the picker is idle. In this figure, the WIP level w_{R} indicates the number of remaining active totes that have not arrived. As expected, $p(1)$ decreases with increasing w_{R}. That is, the more remaining active totes not yet arrived, the lower the probability that one of the active totes in the buffer is processed. Larger n_{T} also leads to lower $p(1)$, given the same utilization level. This is because larger n_{T} allows for more remaining active totes not yet arrived to be selected, hence lower $p(1)$. Note that these decision probabilities are measured from the detailed model at utilization level $u = 0.8$ and are used in the aggregate model

to predict the flow times at various utilization levels $u = \{0.30, 0.32, 0.34, ..., 0.98\}$.

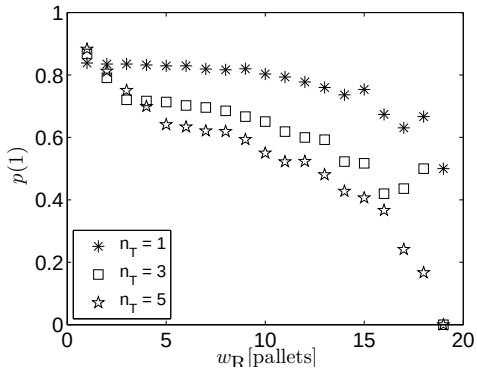
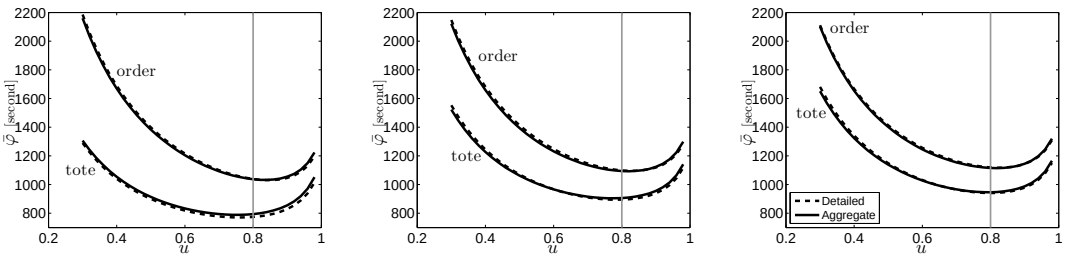


Figure 4.10: Measured decision probabilities from the detailed model at utilization level $u = 0.8$.

At all three settings, the aggregate model accurately predicts the mean and variability of tote and order flow times. The prediction errors for the mean tote and order flow times are less than 3.71% and 2.18%, respectively. The prediction errors for the SCV of tote and order flow times are less than 8.54% and 3.00%, respectively. Figure 4.11 compares the mean tote and order flow times from the detailed and the aggregate model.



(a) $n_T = 1$. (b) $n_T = 3$. (c) $n_T = 5$.

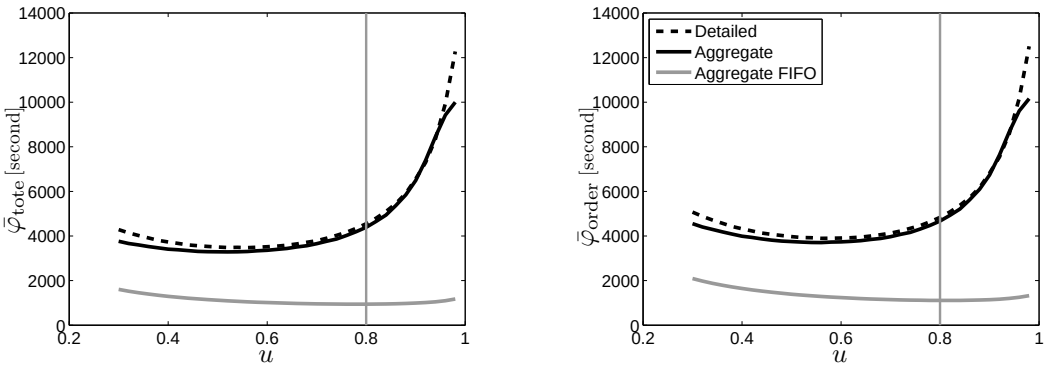
Figure 4.11: Mean tote flow times in case of tote overtaking by an unavailable tote.

4.4.3 Tote and order overtaking

We now consider the case where tote and order overtaking happen at the workstation. We allow tote overtaking by an available and an unavailable tote. Order overtaking occurs only between available orders; each order in the buffer has an equal probability to be selected as the next order.

Figure 4.12 compares the results from the aggregate model with and without order overtaking distribution to the results from the detailed model with $n_T = 3$. Without overtaking distribution, the aggregate model assumes a FCFS processing of orders. The figures suggest that the order overtaking distribution is significant to flow time prediction accuracy. This is because order overtaking happens very often in the detailed model. Furthermore, the flow time prediction by the aggregate model is less

accurate at very high and very low utilization levels. Recall that the WIP-dependent order overtaking distributions are obtained from the detailed model run at training point $u = 0.8$ and then used for all utilization levels in the aggregate model. However, when the aggregate model is run at a high utilization level (e.g., 0.98) it is very likely that an order overtaking distribution required for a high WIP level is not available, simply because this high WIP level was not encountered at the training point. In this case, we use the order overtaking distribution for the maximum available WIP level instead. This way of compromising for the missing order overtaking distribution decreases the prediction accuracy. Note that the aggregate model predicts the flow times more accurately given a detailed model with $n_T = 0$.



(a) Tote. (b) Order.

Figure 4.12: Mean flow times in case of tote and order overtaking, $n_T = 3$.

4.5 Case study

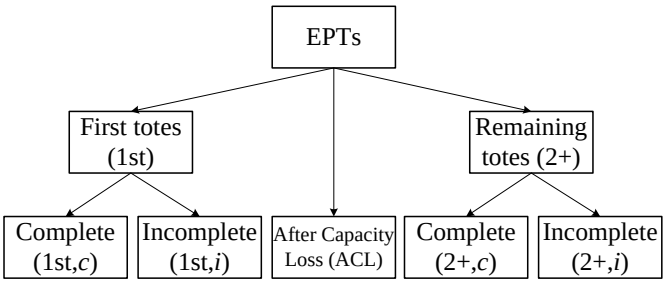
A real, operating automated warehouse is used as a case study to show the application of the proposed aggregate modeling methodology. We consider an automated warehouse that supplies a number of supermarket chains in the Netherlands. This warehouse consists of five miniloads, a conveyor loop, and three end-of-aisle order-picking workstations as shown previously in Figure 4.1(a).

Tote arrival and departure data of three working days are collected from all three workstations. The data is comparable to the one that was generated using the detailed model for validation purposes in the previous section. Each tote arrival or departure contains information about the time of occurrence, tote ID, order ID, and order length. A total of 18.3% tote overtaking has been measured, in which 15.9% is overtaking by available totes and the remaining 2.4% is overtaking by unavailable totes. There is also 4.6% order overtaking in the data.

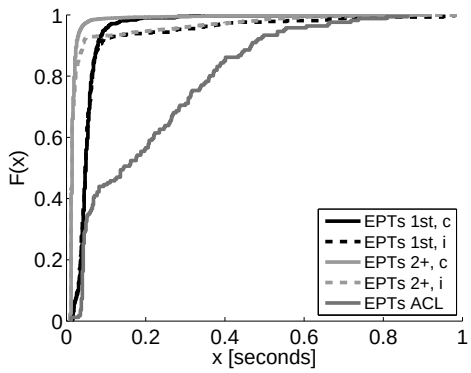
All parameters for the aggregate model are obtained only from tote arrival and departure data. We measure separately for each workstation the overtaking distributions, the decision probabilities, and the EPT distributions. Subsequently, we reconstruct the arrival process of totes at each workstation such that the interarrival time of totes and the interarrival time of orders are correctly represented.

A closer examination to the EPTs measured from the data reveals that there are

several sorts of EPTs. As expected, the EPTs of the first totes of an order are larger than the EPTs of the remaining totes of an order due to the setup applied to every first totes of an order. Furthermore, we observed that many large EPTs occur when not all totes for the active order are present in the buffer. A possible cause is that the pickers hesitate to wait for totes so they leave the workstations when the totes are not yet complete in the buffer. Since time during which the picker leaves the workstation is included in the EPT of the next tote to be processed, a large EPT occurs. As such, we sort the EPTs based on the completeness of totes in the buffer when the picker starts picking a tote. We also consistently found in the data that EPTs of the first totes after a capacity loss (e.g., EPT of tote 3 in Figure 4.4(b)) are substantially larger than other EPTs. The reason is because these EPTs include the time when the pickers actually have not returned yet after leaving the workstation following a capacity loss. Thus, we also treat these EPTs separately and label them as EPT After Capacity Loss (ACL). Figure 4.13 shows the different sorts of EPT and the Cumulative Distribution Functions (CDF) of EPTs for workstation 1.



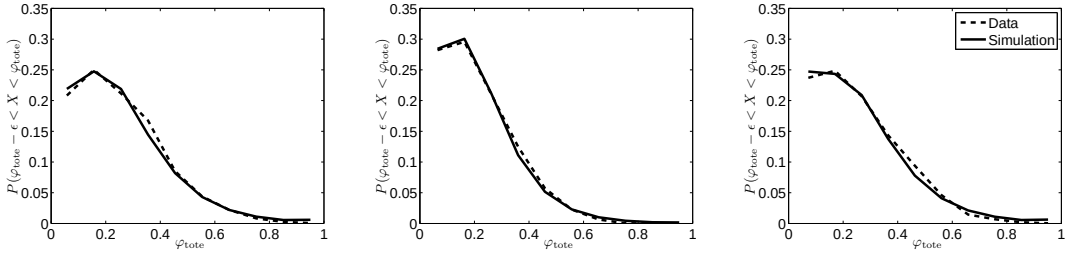
(a) Sorts.



(b) CDF.

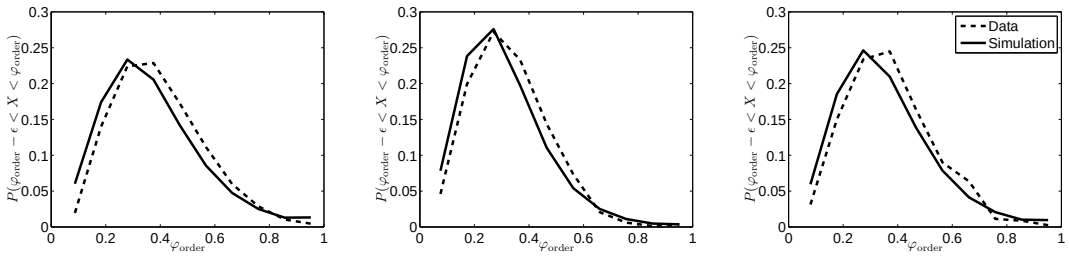
Figure 4.13: Different sorts of EPTs measured from the case study data.

We simulated the aggregate model with 50 replications, each with a run length of 1,000,000 totes excluding a warm-up period of 300,000 totes. We simulate each workstation separately. The resulting flow time distributions from the aggregate model are compared with those measured from the data. Figures 4.14 and 4.15 show the comparisons for all three workstations. Only normalized values are shown due to data confidentiality. Note that in reality, the three workstations operate within limited working hours. For non steady-state analysis, 50 simulation runs are also performed where each simulation is terminated once the number of product totes



(a) Workstation 1. (b) Workstation 2. (c) Workstation 3.

Figure 4.14: Tote flow time distributions.



(a) Workstation 1. (b) Workstation 2. (c) Workstation 3.

Figure 4.15: Order flow time distributions.

processed in one working day has been reached. The results are compared to the real data obtained from the three working days.

In general, the tote flow time distributions are accurately predicted by the aggregate model for all three workstations. The prediction accuracy of order flow time distributions is slightly less compared to that of tote flow time distributions. The prediction errors for the mean tote and order flow times are less than 1.35% and 6.31%, respectively. The absolute difference of SCV of tote and order flow times between the data and the aggregate model are less than 0.11 and 0.08, respectively. The predicted tote flow time distribution is improved as compared to that without applying the overtaking distribution and decision probability. This can be seen by comparing Figure 4.14 to Figure 3.17 of Chapter 3. These values applies to all three workstations. Results from the non steady-state simulation also show that the 95% confidence interval of the mean flow time from the aggregate model contains the flow times from the real data.

4.6 Conclusions

In this chapter an aggregate modeling method has been proposed to predict the flow time performance of an end-of-aisle order-picking workstation with non-FCFS processing. We require only limited, measurable data namely the arrival and departure data of totes. Based on this data, the effective process time distribution, overtaking distribution, and the so-called decision probability are reconstructed to be used in the aggregate model. Using a number of simulation experiments, it has been shown that the aggregate model predicts flow time of product and order accurately. The

method has been applied to data obtained from a real, operating automated warehouse. The resulting flow time prediction shows satisfactory accuracy. The proposed aggregate modeling method should be of value for analyzing system performance under, for example, different settings of order release strategies, product interarrival rates, or order length distributions.

The next chapter addresses the design and control of an automated order-picking workstation processing multiple orders simultaneously.

5

An automated multiple-order workstation

This chapter will be partly presented as:
R. Andriansyah, L.F.P. Etman, I.J.B.F. Adan and J.E. Rooda,
Automated order-picking workstation handling out-of-sequence product arrivals.
Accepted for presentation at the SIMULTECH 2011 conference, Noordwijkerhout, The Netherlands,
July 29-31, 2011.*

Abstract | A novel design of an automated order-picking workstation processing multiple orders simultaneously is proposed to be used in warehouses with an end-of-aisle order-picking system. A typical problem at this workstation is the out-of-sequence arrival of products, assuming the workstation receives products for multiple orders simultaneously. As multiple products are present, the picking sequence at the workstation affects the system throughput. The performance of four picking policies is compared in terms of order throughput and queue length distribution under different extents of out-of-sequence arrivals. Experimental results show the capability of the workstation to handle an arbitrary extent of out-of-sequence arrival of products. Noteworthy insights for design considerations of such systems are drawn.

5.1 Introduction

Warehouses nowadays are operating in a more-than-ever challenging environment. Internet orders are forcing warehouses to keep greater varieties of SKUs (Stock Keeping Units) and to deliver low-volume orders more frequently. Moreover, retailers are setting tighter order delivery schedules, so as to avoid out-of-stock situations. These

*The conference paper was nominated for the SIMULTECH 2011 best student paper award.

challenges combined with fierce market competition call for a more efficient order-picking operation. After all, order-picking, the process of retrieving products from the storage area to fulfill customer orders, is estimated to account for 55% of the total warehouse operating cost (Tompkins et al., 2003).

Warehouse automation is becoming a common practice to respond to these challenges. This can be seen from the notable growth in sales of automated material handling systems in recent years (Baker and Halim, 2007). Different sorts of automated picking technologies are continuously being introduced to the market. Each technology is typically designed by considering factors such as the number of SKUs and the expected picking volume, among others. A selection methodology, such as the one proposed by Dallari et al. (2009), can be used to determine the most suitable order-picking system for a given set of warehouse requirements.

The current study focuses on a particular class of order-picking systems, namely an end-of-aisle order-picking system. This system is typically composed of separate processing units including a storage area and an order-picking workstation, which are connected by a transportation unit such as a closed-loop conveyor. Such configuration is capable of processing a significantly large number of SKUs.

For such a system it is desirable to process multiple orders simultaneously to gain high throughput. To do so, products for multiple orders must be retrieved simultaneously from the storage area. This, however, poses a threat to the system performance. Ideally, products required for the earliest released order arrive earlier at the workstation than any other products. However, due to factors such as the number of storage racks, the composition of SKUs across the storage racks, and the retrieval time of products, products may not arrive completely in the same sequence as requested by the workstation. This situation is referred to as *out-of-sequence arrivals* of products. Combined with inefficient picking operations at the workstation, such a situation deteriorates the throughput performance of the order-picking system.

In this chapter we study an automated order-picking workstation that is able to deal with out-of-sequence arrivals. A novel workstation design with an integrated carrousel mechanism is proposed, where multiple orders can be processed simultaneously. The design is capable of handling arbitrary out-of-sequence arrival distributions. While there are a number of factors that affect the workstation performance, the focus of our study is on the picking policy for the proposed workstation design. We consider four picking policies and show that a significant gain in throughput can be realized by applying a proper picking policy.

The chapter is organized as follows. Section 5.2 is a literature review on some automated order-picking technologies. Section 5.3 elaborates the configuration of the order-picking workstation under study. Section 5.4 describes the four picking policies. In Section 5.5 a number of simulation experiments are performed to see the performance of the workstation under different settings. Finally, Section 5.6 concludes the chapter.

5.2 Literature review

A number of fully automated picking technologies have been discussed in the literature. Some of these technologies are an A-frame dispenser (Caputo and Pelagagge, 2006), a gantry picking complex (Kim et al., 2003), and a rotary rack (Li and Bozer, 2010).

An A-frame dispenser system (see Figure 5.1) consists of flow racks, in which products are placed, that are arranged side-by-side forming an A-shaped frame. Products are automatically dispensed onto a collection belt under the frame. An order box waits to be filled at the end of the belt. Once all items for an order have been collected, the system dispenses products required for the next order. This system is typically popular for drugs and pharmaceutical distribution facilities.

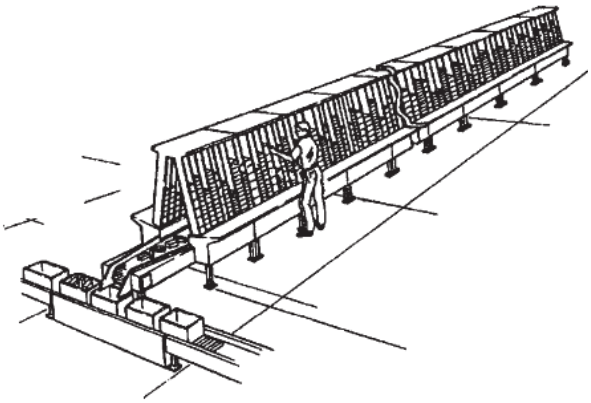


Figure 5.1: An A-frame dispenser (taken from Caputo and Pelagagge (2006)).

A gantry picking complex (see Figure 5.2) can be considered as a zone picking system with gantry robots as pickers. Picking takes place at a number of pick zones that are arranged in a serial configuration with a common conveyor among them such that order trays can pass sequentially from the first to the last pick zone. Each pick zone typically has one robot and a number of drop buffers. A robot in a particular zone picks an item required for an order and puts it into one of the drop buffers in its zone. Once the corresponding order tray passes along the drop buffer containing an item, the drop buffer then deposits its content to the order tray. A pick error occurs if an item is put into a drop buffer after the order tray designated to receive the item has passed that buffer. An order is finished once the order tray has passed through all zones and all items required for the order have been acquired from the drop buffers.

A rotary rack (see Figure 5.3) is a special type of carrousel that uses a dedicated drive for each level of the carrousel. This allows each level to rotate independently of and concurrently with the other levels. A number of robots are present to pick items. When an order is released, all levels of the rack rotate until the required items are positioned in front of the robots. The rack then stops and the robot picks the items. The robot puts the picked items onto a take-away conveyor located at the ground level of the rotary rack.

For all three automated picking systems mentioned above, order integrity is maintained since products are picked for one order at a time. We consider another system

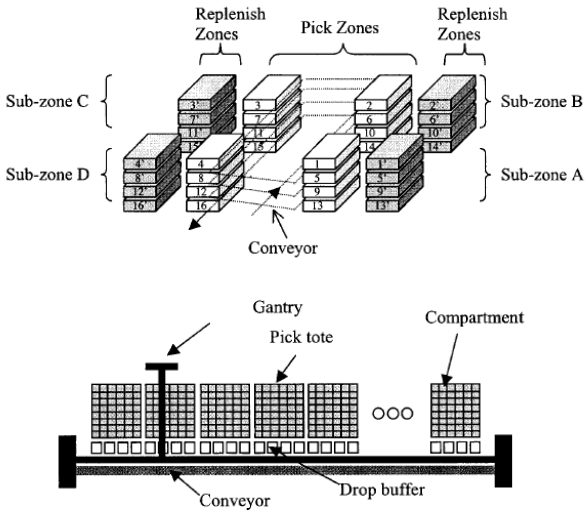


Figure 5.2: A gantry picking complex (taken from Kim et al. (2003)).

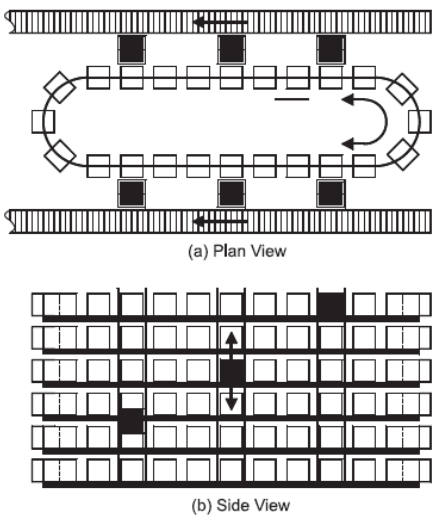


Figure 5.3: A rotary rack (taken from Li and Bozer (2010)).

where this is not always the case, namely an end-of-aisle order-picking system with remotely located workstations. Chapters 3 and 4 of this dissertation have investigated the performance of a manual end-of-aisle order-picking workstation where one order is processed at a time, but products for multiple orders are sent simultaneously to the workstation. For this type of system, out-of-sequence arrival of products becomes a relevant problem.

A number of patents that deal with the problem of out-of-sequence arrivals in an end-of-aisle order-picking system are available. Three main approaches can be identified from these patents. The first approach is to dispatch products in the correct sequence from the storage area using an AS/RS (Automated Storage/Retrieval System) by implementing an integrated sorting at the AS/RS as proposed by, e.g., Schäfer (2010). Products for multiple orders are retrieved simultaneously by the AS/RS and are subsequently put on the central conveyor in the sequence required by the workstation. This is done by dividing the conveyor into segments and then reserving the segments

for products in the correct sequence as needed. A product is released to the conveyor only if the reserved conveyor segment for the product has been found. In this way, a correct sequence of arrivals at the workstation can be achieved. The second approach is to rearrange the sequence of product arrivals directly at the workstation. For example, Winkler (1997) proposed a vertical buffer to be placed in front of an order-picking workstation. Products that arrive too early are temporarily stored in the vertical buffer and will be released to the workstation only by the time they are needed. Consequently, the workstation can always process products in the right sequence. The third approach to alleviate the problem of out-of-sequence arrivals is simply retrieving products for one order at a time, despite any potential losses in system performance. An example from this type of system is proposed by Guenzi and Feie (2003).

5.3 Workstation configuration

The automated order-picking workstation discussed in this chapter is part of a larger order-picking system, which typically also comprises a pallet storage area and conveyors. In this order-picking system, product pallets are retrieved from the pallet storage area using an AS/RS. Each product pallet contains a number of items from one SKU only. These pallets are transported using conveyors from the storage area to one of the picking workstations. At the workstation, items on product pallets are picked onto order pallets to fulfill orders. Product pallets that have been processed but still contain some items left are returned to the storage area. An order pallet corresponds to one order. Multiple orders are present and each order requires a number of SKUs, which is referred to as the order length.

The automated order-picking workstation must be able to handle different extents of out-of-sequence arrivals. This is because the AS/RS at the storage area is assumed to be the bottleneck of the entire order-picking system. Hence, the AS/RS should operate at its maximal capacity without taking into account the issue of out-of-sequence arrivals. The automated order-picking workstation should therefore be configured such that it is able to operate even under a high extent of out-of-sequence arrivals. Furthermore, a well-defined picking policy is required. This is because products for multiple orders are present at the workstation. A picking policy prescribes the sequence in which products should be picked to fulfill orders. The desired picking policy is the one that gives a high order throughput under arbitrary out-of-sequence arrival distribution.

The basic configuration of the automated order-picking workstation is shown in Figure 5.4. There are a number of conveyors that act as buffer lanes. The layout of Figure 5.4 has one order buffer and two product buffers. These buffers follow a first-in-first-out principle; only pallets located at the head of the buffers (foremost) can leave the workstation. A robot picks items from a product pallet and drops them onto the corresponding order pallet. This robot has a limited operating area referred to as the pickable area. Only pallets inside the pickable area are accessible to the robot. Once picked, a product pallet may leave the workstation, if possible. An order pallet may only leave if it already contains all SKUs required by the order.

Once a pallet leaves the workstation, all remaining pallets in the same buffer shift one position forward. It is possible, however, that a picked pallet cannot leave the workstation because there is another pallet in front of the picked pallet. The picked pallet is then referred to as a *blocked pallet*.

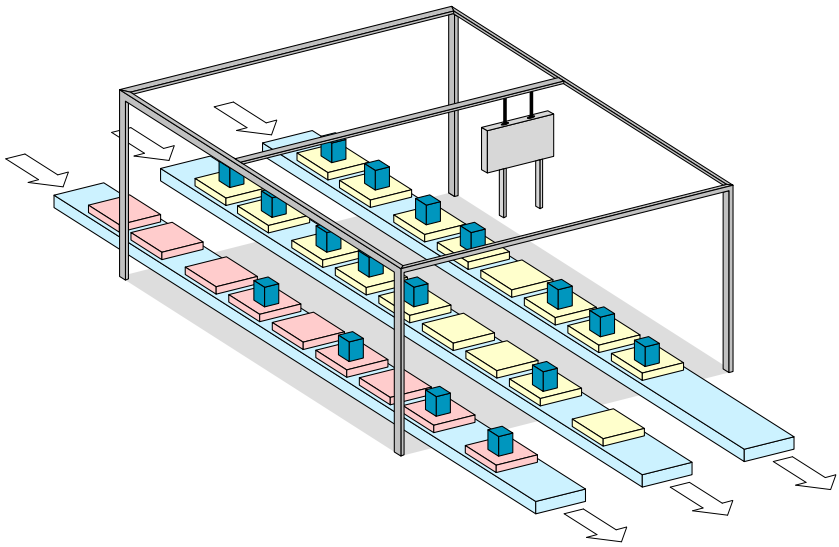


Figure 5.4: Layout of the automated order-picking workstation. One buffer lane is used for order pallets and two buffer lanes are used for product pallets. The pickable area is the area inside the frame. The arrows denote the direction of pallet movements.

Other issues relevant to this automated order-picking workstation are the pipeline capacity and system deadlock. The *pipeline capacity* is the maximum number of pallets that can be simultaneously on the way to the workstations. It is crucial that the pipeline is sufficiently filled. This reduces the interarrival time of product pallets at the workstation, hence providing the robot with enough work. Furthermore, the workstation needs an additional mechanism to avoid deadlock. Due to the out-of-sequence arrivals, system deadlock may occur namely when the buffers are full and the pallets at the heads of all product and order buffers cannot be picked. A proper design of such end-of-aisle workstation should therefore take this eminent problem into account.

A conceptual design of the workstation that avoids such a deadlock situation is proposed in Figure 5.5. The workstation consists of four buffers, namely order buffer *OP*, product buffer *PP*, recirculated product buffer *rPP*, and carrousel buffer *cPP*. The buffers *rPP* and *cPP* together provide for a carrousel. These two buffers can rotate independently in one direction, allowing product pallets to exchange places between the buffers. The robot can only pick/drop items from/to pallets within the pickable area, which is depicted by the striped line in Figure 5.5.

The system works as follows. Products retrieved from the storage area arrive at the product buffer *PP*. Each product has an identification number (ID) that is equal to the ID of the order to which the product belongs. The robot then selects a product pallet inside the pickable area of product buffers *PP* or *rPP*, picks an item from the product pallet, and drops the item onto the corresponding order pallet in *OP*. Only product pallets with IDs that belong to one of the *pickable orders* can be selected as

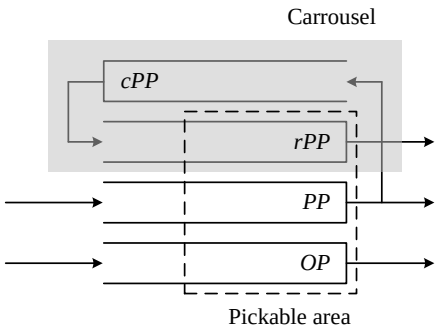


Figure 5.5: Conceptual design of the automated order-picking workstation.

the next pallet to be picked. A pickable order is an order whose order pallet and at least one of the corresponding product pallets are located inside the pickable area.

There are two operations involving the carrousel:

1. Inserting a product pallet into the carrousel.
   A product pallet is inserted if no product pallet in *rPP*, *cPP* and inside the pickable area of *PP* is required by the order pallets inside the pickable area of *OP*. A product pallet is inserted from *PP* to the carrousel buffer *cPP*. This also means that the product pallet has been recirculated. Subsequently, if there is a space available in *rPP*, the product pallet is immediately moved from *cPP* to *rPP*.

2. Rotating the carrousel.
   The carrousel is rotated if no product pallet inside the pickable area of *PP* and *rPP* is required by the order pallets inside the pickable area of *OP*, but such a required product pallet is present in *cPP* or outside the pickable area of *rPP*. During the rotation, one product pallet is moved from *rPP* to *cPP* and one product pallet is moved from *cPP* to *rPP*.

5.4 Picking policy

Having the workstation design as elaborated in the previous section, the problem now is to formulate a picking policy that is robust to out-of-sequence arrivals of pallets and efficient such that real-time implementation in practice is possible. We consider four picking policies namely *nearest-to-the-head, nearest neighbor, dynamic programming,* and *backward search* picking policies.

All picking policies use the information about the product and order pallets inside the pickable area of product buffers *PP*, *rPP*, and order buffer *OP*. First, the content of *rPP* is evaluated against *OP*. If there is a product pallet that can be picked in *rPP*, then the robot picks an item from the product pallet and puts it into its corresponding order pallet in *OP*. Otherwise, the content of *PP* is evaluated against *OP*. Similarly, if there is a product pallet that can be picked in *PP*, then the robot picks an item from the product pallet and puts it into its corresponding order pallet in *OP*. That is, priority is given to picking a product pallet from *rPP*. In case no product pallet can be picked from both *rPP* and *PP*, one of the two carrousel operations is performed. The

following section elaborates for each picking policy the criteria used in determining whether there is a product pallet that can be picked.

5.4.1 Nearest-to-the-head

The nearest-to-the-head picking policy aims at an uninterrupted flow of product pallets at the workstation. Recall that a product pallet can only leave the workstation once it has been picked and is located at the head of the product buffer. Once a product pallet leaves, all remaining product pallets in the buffer shift one position forward and a new product pallet can enter the workstation. Therefore, giving priority to picking the product pallet located at the head of the product buffer supports a continuous flow of product pallets at the workstation.

The nearest-to-the-head picking policy requires the robot to pick the product pallets according to their sequence in the buffer. It evaluates first whether the product pallet at the head of the product buffer can be picked. If this is not possible, then the product pallet located at the second position in the buffer is evaluated next. This evaluation is performed continuously until either a pickable order is found or the end of the pickable area is reached.

5.4.2 Nearest neighbor

One of the most serious pitfalls of the nearest-to-the-head picking policy is that it may cause the robot to travel without carrying any item from the current position of the robot to the head of the product buffer. Such travel consumes time, but is a non-value-added process; it is thus detrimental to the order throughput of the workstation. The nearest neighbor picking policy is proposed, which minimizes the distance in which the robot travels without carrying any item.

The nearest neighbor picking policy requires the robot to pick the product pallet located nearest to the current position of the robot. If there is more than one pickable order with the same distance to the robot, then the robot picks the pallet that is located closer to the head of the buffer.

5.4.3 Dynamic programming

Given an unlimited supply of product and order pallets at the workstation, the order throughput of the workstation depends on the robot processing time; a lower processing time leads to a higher throughput. The robot processing time consists of a travel time, pick time, and drop time. Assuming a relatively constant pick and drop time, one can increase the workstation throughput by reducing the travel time, which is a function of the robot travel distance.

The dynamic programming policy constructs a *picking trip* that minimizes the robot travel distance based on the current content of product and order buffers. The robot then picks a number of orders in a sequence as indicated in the picking trip. Once a product or an order pallet leaves the workstation, the buffer content changes. In this

case, a new picking trip is constructed using the dynamic programming approach based on the new content of the product and order buffers.

The robot travel distance depends on the sequence of picking a number of orders. When picking one order, the robot travels from its current position to a product pallet (picking an item) and finally to the corresponding order pallet (dropping the item). Following a picking process, the robot thus always starts a new picking process from the location of the previously served order pallet.

A dynamic programming policy is formulated to minimize travel distance D to reduce the travel time and consequently to increase the workstation throughput, where:

$$D = \sum_{i=1}^{m} (|y_{i-1} - x_i| + |x_i - y_i|), \tag{5.1}$$

with: (refer to Figure 5.6)

x_i = position of the product pallet of the order picked at the ith step.
y_i = position of the order pallet of the order picked at the ith step.
y_0 = current position of the robot.

In the travel distance only the movement along the buffer conveyor lanes is considered. The distance to move from one lane to another is neglected. In the above formulation, m is the number of orders picked within a picking trip, which is referred to as the *trip length*. Travel distance D increases with trip length m. Therefore, the optimal trip is the one that minimizes travel distance D per order, so the one minimizing D/m.

The first step in constructing a picking trip that minimizes travel distance D is evaluating the pallets contained in the product and order buffers. Recall that each product pallet has an identification number represented by the ID of the order to which the product belongs. Let

P be the set of IDs of product pallets within the pickable area
Q be the set of IDs of order pallets within the pickable area

Then $R = P \cap Q$ gives the IDs of *pickable orders*. These are the orders whose product and order pallets are both located inside the pickable area. Let $S \subseteq R$ be the IDs of *target orders*. These are the pickable orders whose product and/or order pallets are located at the head of the buffer. A target order is the last order to be picked in a picking trip because picking a target order causes a product and/or an order pallet to leave the workstation. With this regard, x_m and y_m in (5.1) are the positions of the product pallet and the order pallet for the target order, respectively. Picking a target order leads to a new content of either a product buffer or the order buffer, or both. Hence, a new picking trip needs to be constructed.

Having a target order is a requirement for constructing a picking trip. Recall that there are two product buffers namely *rPP* and *PP* evaluated against the order buffer *OP*. If both *rPP* and *PP* contain at least one target order, then a picking trip is constructed from *rPP*. That is, *rPP* has priority over *PP*. If only one product buffer contains a target order, then a picking trip is constructed from that product buffer.

Otherwise, if none of the two product buffers contains a target order, then one of the two carrousel operations explained previously is performed.

Figure 5.6 shows an example of the content of product buffer *PP* and order buffer *OP*. Assume now that product buffer *rPP* does not contain a target pallet and therefore is disregarded from the figure. Based on this figure, $P = \{11, 12, 14, 15, 18\}$, $Q = \{11, 12, 13, 14, 15\}$, $R = \{11, 12, 14, 15\}$, and $S = \{11\}$.

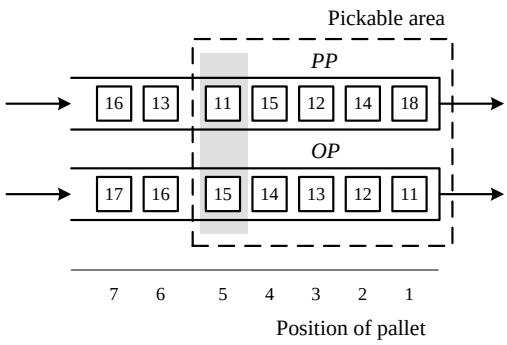


Figure 5.6: An example of the content of product (*PP*) and order (*OP*) buffers. The numbers represent the IDs of pallets. A product pallet at *PP* belongs to an order pallet at *OP* with the same ID. The robot is currently at position 5.

Constructing a picking trip can be regarded as a Traveling Salesman Problem (TSP). That is, given trip length m, pickable orders R, target orders S, and current position of robot y_0, determine the sequence of picking m pickable orders involving one and only one $s \in S$, which starts at y_0 and ends at s such that D as given in (5.1) is minimized.

Given R there may be TSPs of length m that contain one $s \in S$. Each TSP has to be solved separately. That is, for $m = 1, ..., m_{\max}$ the TSPs of length m are generated, and subsequently solved. Herein, the maximum trip length $m_{\max}$ follows from the number of elements in R excluding S and subsequently adding *one* element of S in the trip. Hence, the maximum trip length is $m_{\max} = |R| - |S| + 1$, where $|R|$ and $|S|$ denote the number of orders in R and S, respectively.

In the example of Figure 5.6 where $R = \{11, 12, 14, 15\}$ and $S = \{11\}$, the maximum trip length is 4, resulting in eight TSPs that contain one $s \in S$, each of which is a subset of R, namely for $m = 1$: {11}; for $m = 2$: {11,12}, {11,14}, {11,15}; for $m = 3$: {11,12,14}, {11,12,15}, {11,14,15}; and for $m = 4$: {11,12,14,15}. The number of TSPs increases exponentially with the number of pickable orders. With $|R| = 10$ and $|S| = 1$, as many as 512 TSPs need to be solved.

The optimal picking trip given R, S, and y_0 is the one that minimizes travel distance per order, i.e., the one minimizing D/m. Let r^*_m be the picking trip minimizing the travel distance D from *all* TSP of length m as a subset of R. Let $D_{r^*_m}$ be the resulting travel distance for r^*_m. The optimal picking trip is

$$r^* = \arg\min\{\frac{D_{r^*_1}}{1}, \frac{D_{r^*_2}}{2}, ..., \frac{D_{r^*_{m_{\max}}}}{m_{\max}}\}. \tag{5.2}$$

We use dynamic programming for TSP (Bellman, 1962) to solve each TSP. With dynamic programming, a TSP is handled in smaller parts by solving subproblems

sequentially. The solutions to these subproblems are then stored for future use. Larger subproblems are solved by a recursion formula from the smaller subproblems. The complete solution for the TSP is obtained through backtracking the solutions of the subproblems. A description of the dynamic programming for TSP is provided in the appendix. For the example in Figure 5.6, the optimal picking trip is found to be $y_0 \rightarrow 12 \rightarrow 14 \rightarrow 15 \rightarrow 11$. Note that a number of heuristics to solve TSPs exist in the literature. We refer to Johnson (1990) for a review of such heuristics.

A similar problem for a slightly different system has been addressed by Ascheuer et al. (1999). They considered an order sequencing problem for a miniload crane. The complete orders to be processed in one day are not known beforehand. As such, they formulate the problem as an online asymmetric TSP (ATSP) with the objective of minimizing the total unloaded travel time of the miniload crane. The optimal processing sequence for a set of orders is found by solving the ATSP using the branch-and-bound method. If a new order enters the system while the miniload crane is still processing orders according to the previous optimal sequence, then a new optimal sequence is calculated and the previous one is disregarded.

There is, however, a difference between the study by Ascheuer et al. (1999) and the current study. In our case, it is not possible to have a new buffer content (i.e., a new set of pallets to be picked) if the crane is not yet finished picking pallets according to the optimal sequence. In other words, a new optimal sequence is calculated only if all steps in the previous optimal sequence have been executed. Consequently, the dynamic programming policy gives an optimal sequence only for the static problem given a certain buffer content. It does not provide an optimal solution given the dynamic changes of the system, where it is not known a priori which order the next arriving pallet belongs to. Additionally, since there can be multiple trip lengths that lead to changes in the buffer content, our TSP formulation has the objective to minimize the travel distance per order.

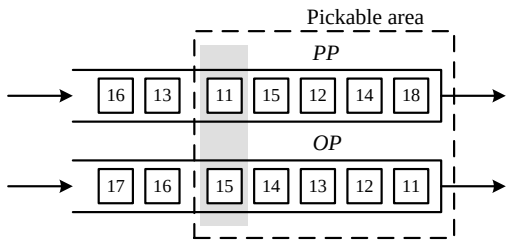
5.4.4 Backward search

The previously elaborated dynamic programming approach may require evaluation of a large number of TSPs. The number of TSPs that needs to be evaluated increases exponentially with the number of pickable orders. Consequently, the required computation time grows exponentially especially when multiple picks per product pallet are needed. Another heuristics-based policy called the backward search policy is proposed. This policy restricts the number of evaluations required, hence reducing the complexity of the problem.

Similar to the dynamic programming policy, the backward search policy also constructs a picking trip that starts from the current position of the robot and ends at a target order. The first step is evaluating the product buffers *rPP*, *PP* against order buffer *OP* to find a target order. Again, priority is given to constructing a picking trip based on the pallets in buffer rPP. Additionally, if more than one target order is found upon evaluating product buffer *rPP* or *PP* against order buffer *OP*, the target order located in the buffer that contains the greatest number of blocked pallets is chosen. Recall that a blocked pallet is a product or an order pallet that has been picked but cannot leave the workstation because another pallet in front of it has

not been picked yet. This formulation is chosen to minimize the number of blocked pallets in the buffer.

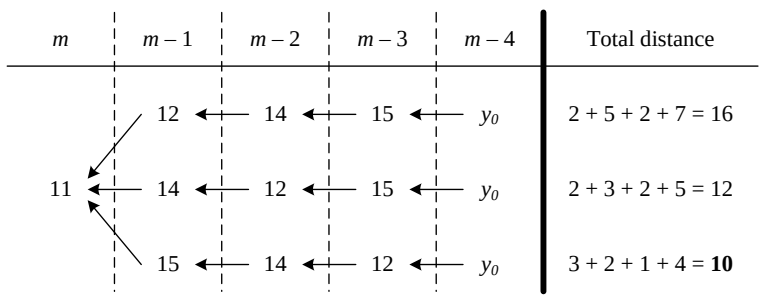
Given a trip length m, pickable orders R, target orders S, and current position of robot y_0, optimal picking trip r_m^* for trip length m is constructed by working backwards from target order $s \in S$ to the current position of the robot. At step m, the robot picks target order s in the buffer that contains the greatest number of blocked pallets. At step $m-1$, we *enumerate* on all pickable orders. This way, $|R|-1$ alternative trips are obtained, with order x_{m-1} as the order to be picked. From step $m-2$ backwards, for each alternative trip we search for order x_{m-2} that minimizes the distance to the previously picked order. This search is applied to all subsequent steps $\{m-3, m-4, ..., 1\}$ until a complete picking trip of length m has been constructed for all $|R|-1$ alternative trips. The optimal picking trip r_m^* is the picking trip that minimizes the total picking distance D given in (5.1). The whole procedure is performed for all trip lengths $1 \leq m \leq |R|$ and finally the picking trip r^* is selected as the one that minimizes travel distance D per order. Note that during the execution of r^* it is possible that the trip ends earlier if r^* includes more than one $s \in S$. The process for determining picking trip r^* is repeated each time the content of product or order buffers changes.



(a) Buffer content.

i \ j	11	12	14	15
12	7	-	2	3
14	5	2	-	1
15	4	3	5	-
y_0	4	3	5	2

(b) Distance matrix.



(c) Alternative trips.

Figure 5.7: The backward search policy.

Figure 5.7 illustrates the backward search heuristics to find the optimal picking trip for the content of product buffer *PP* and order buffer *OP* shown in Figure 5.7(a). Assume now that product buffer *rPP* does not contain a target pallet and therefore is disregarded from the figure. From *PP* and *OP* we obtain $R = \{11, 12, 14, 15\}$ and $S = \{11\}$, hence $s = 1$. The current robot position y_0 is depicted as the gray area in Figure 5.7(a). Distance d_{ij} is the distance traveled by the robot when picking one order, and is defined as

$$d_{ij} = |y_i - x_j| + |x_j - y_j| \tag{5.3}$$

for any $i, j \in U$ and $i \neq s$, where:

x_i = position of product pallet with ID i
y_i = position of order pallet with ID i

A distance matrix (Figure 5.7(b)) is calculated based on (5.3). Note that distance d_{ij} for $i = s$ is not calculated, since we require the target order s to be picked last. The distances from the initial robot position y_0 to all pickable orders are also calculated. Suppose now we want to find an optimal picking trip r_m^* with length $m = 4$. A number of alternative picking trips are generated as described earlier. At step m the picking of target order $s = 11$ is planned. At step $m - 1$ we enumerate all remaining pickable orders $R = \{12, 14, 15\}$ (see Figure 5.7(c)). At step $m - 2$ we search for the pickable order that minimizes the distance to the pickable order planned at step $m - 1$. For example, if at step $m - 1$ pickable order 12 is chosen, then at step $m - 2$ we search in the distance matrix for i that gives d_{i2} the minimum distance, namely $i = 14$. This approach is also applied at step $m - 3$. After step $m - 3$, all pickable orders have been planned and thus step $m - 4$ gives initial robot position y_0. Once the total distance for all alternative trips have been calculated, the optimal trip r_4^* of length 4 is identified. In this example, the optimal picking trip of length 4 is $y_0 \rightarrow 12 \rightarrow 14 \rightarrow 15 \rightarrow 11$.

Note that both dynamic programming and backward search policies only use information about the current buffer content when constructing the picking trip. This being said, it is possible that a higher order throughput is gained by other policies that take into account additional information namely about the products currently underway to the workstation.

5.5 Simulation experiments

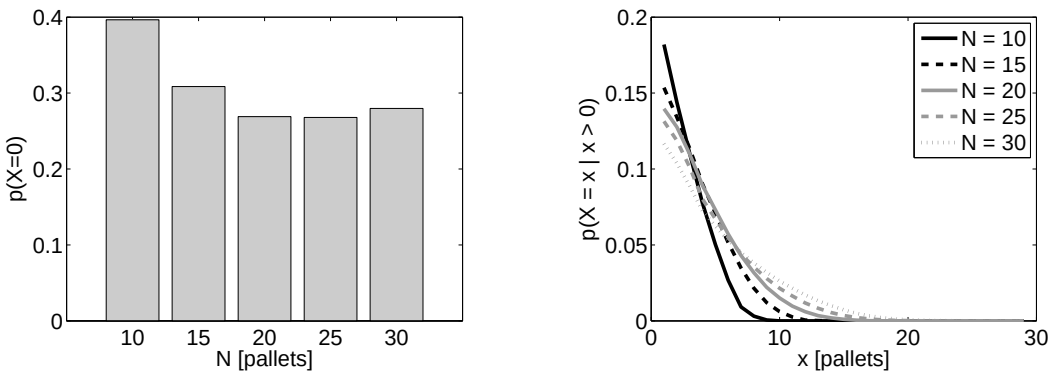
We investigate the performance of the automated order-picking workstation under all four picking policies for single and multiple picks per order. The performance measures of interest are the order throughput and the distribution of the queue length at the carrousel buffer. To do this, we first propose a method to model the out-of-sequence arrival of product pallets.

5.5.1 Modeling the out-of-sequence arrivals

An example of out-of-sequence arrival of product pallets at the workstation can also be seen from Figure 5.6. In this example we assume that each order pallet in buffer *OP* requires one product (that is, order length $k = 1$). Each order pallet has an ID, where lower IDs represent orders that are released earlier. A product pallet in *PP* with the same ID as an order pallet in *OP* means that the product pallet is required for the order pallet. Ideally, product pallets arrive in the same sequence as the order pallets, namely products for older orders arrive first. However, products arrive out-of-sequence, where e.g., product pallet 18 arrived earlier than the product pallets for orders 11,12,...,17. That is, product pallet 18 *overtakes* seven product pallets.

The number of overtaken products is calculated for each incoming product pallet. This gives an overtaking distribution that characterizes the out-of-sequence arrival of products.

An overtaking distribution is obtained by measuring the overtaking from a detailed simulation model of an AS/RS. The AS/RS system consisted of five storage racks from which products were retrieved. A total of 10,000 SKUs were contained in the storage racks; each SKU was contained only on one product pallet. This setting represents a warehouse that serves slow moving products. The cranes of the storage racks had a retrieval batch size of four pallets. That is, the cranes wait until there is a retrieval command for four SKUs before it starts retrieving the SKUs. The maximal pipeline capacity is N product pallets. These are the product pallets that have been retrieved by the AS/RS and are on their way to (but have not entered) the workstation buffer. Figure 5.8 shows the overtaking distributions measured from the detailed simulation model for different pipeline capacities $N = \{10, 15, 20, 25, 30\}$.



(a) Probability of no overtaking.

(b) Probability of overtaking x pallets.

Figure 5.8: Measured overtaking distribution from a detailed simulation model.

We used Figure 5.8(b) as inspiration to develop an analytical function to represent the overtaking. Let random variable X be the number of overtaken products, which can take a value of $\{0, 1, 2, 3, ..., N-1\}$. Let

$$\begin{aligned} P(X > 0) &= p, \\ P(X = 0) &= 1 - p. \end{aligned} \tag{5.4}$$

Here, $P(X = 0)$ is the probability that a certain product does not overtake other products. In the case of overtaking ($X > 0$) the possible number of overtaken products is $\{1, 2, 3, ..., N-1\}$. To obtain a shape of the overtaking distribution similar to Figure 5.8, we determine the probability of overtaking x products as:

$$P(X = x \mid x \in Y) = f(x) = (ax + b)\, xe^{-cx}, \tag{5.5}$$

where $Y = \{1, 2, 3, ..., N-1\}$. There are three parameters in the function, namely a, b, and c. The values of a and b are calculated given c. Since the maximal number of overtaken products is $N - 1$, it is known that:

$$P(X = N) = (aN + b)\, Ne^{-cN} = 0. \tag{5.6}$$

Furthermore, the sum of all probabilities of overtaken products is equal to 1. That is:

$$\sum_{x=1}^{N} f(x) = 1. \tag{5.7}$$

From (5.6) and applying (5.6) in (5.7), we obtain:

$$b = -aN.$$

$$a = \frac{1}{\sum_{x=1}^{N}(x-N)\, xe^{-cx}}.$$

Hence, parameters p and c determine the overtaking distribution given by (5.4) and (5.5).

To illustrate the proposed overtaking distribution, we have fitted function (5.5) to data measured from the detailed simulation model. The value of parameter c, also referred to as the shape parameter, is determined using the nonlinear regression fitting method in Matlab. Figure 5.9 shows the comparison of overtaking from simulation data and from the fitted function for $N = 10$.

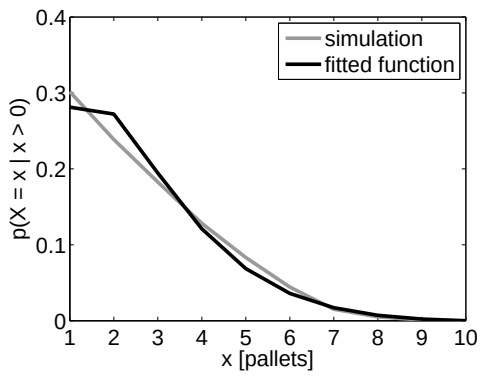


Figure 5.9: Comparison of overtaking from a detailed simulation and from the fitted overtaking function.

The shape parameter c can be changed for experimental purposes. Figure 5.10 shows several shapes of the overtaking distribution given various c values for $N = 10$. In general, larger c leads to smaller mean number of overtaken products, given that there is overtaking. By using different settings for overtaking probability p and shape parameter c, we are able to analyze the performance of the automated order-picking workstation under different extents of out-of-sequence product arrivals.

5.5.2 Assumptions and experimental settings

The following assumptions apply to all simulation experiments. The product buffer, the recirculated product buffer, and the order pallet buffer have a finite capacity of 10 pallets. A new product pallet is generated into the product buffer as soon as a space in the buffer becomes available. That is, there is no interarrival time involved in generating a new product pallet. The processing time of the robot is comprised of pick time, travel time, and drop time. The travel time depends on the travel

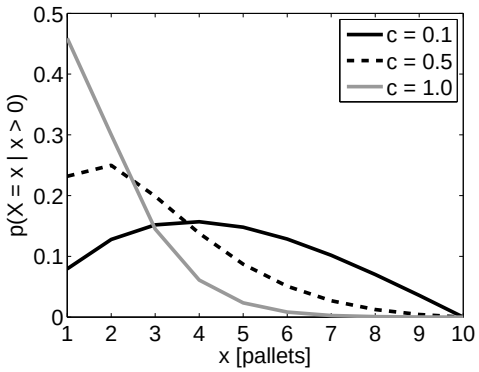


Figure 5.10: Comparison of overtaking distributions with different values of parameter c.

distance D (in pallets) that covers the cycle: current position - product pallet - order pallet. We assume that the robot requires 0.25 seconds to travel a distance of 1 pallet measured in horizontal direction only. That is, the movement from product to order pallet and vice versa is performed during the horizontal movement. The pick and drop time is assumed to be constant at 2 seconds each. Hence, the processing time t_e of the robot is (in seconds)

$$t_e = 0.25\ D + 4. \tag{5.8}$$

Further it is assumed that rotating the carrousel costs 2 seconds per rotation and that shifting the pallets forward in the buffer takes place while the robot is traveling.

Table 5.1 gives a list of parameters that are used in the experiments. Overtaking probability p and overtaking shape parameter c are used to create an overtaking distribution (see Section 5.5.1), representing the out-of-sequence arrivals of products. Order length k represents the number of SKUs required for an order, which equals the number of required product pallets given that a product pallet contains one SKU only. The maximum number of product pallets that are on their way to the workstation simultaneously is denoted by N, which is the pipeline capacity. The size of pickable area L gives the number of pallets that are within reach of the robot.

Table 5.1: List of parameters.

Parameter	Notation
Overtaking probability	p
Overtaking shape parameter	c
Order length	k
Size of pickable area	L
Pipeline capacity	N

All simulation experiments have been performed using process algebra based discrete-event simulation language χ (Chi) 1.0. We refer to Hofkamp and Rooda (2008) for a definition of this language. A tutorial of the language is provided by Rooda and Vervoort (2007).

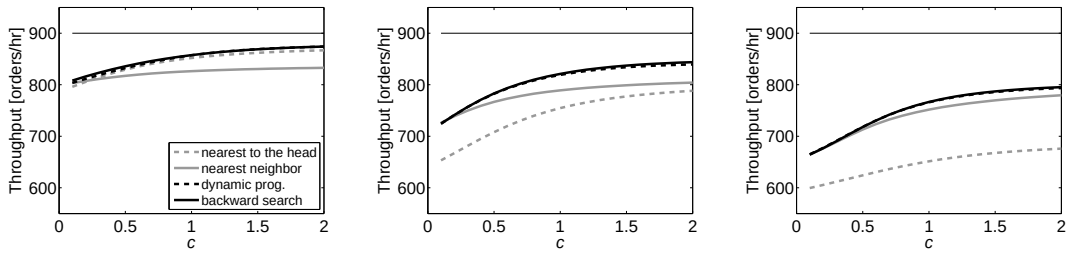
The following setup is used for all experiments. Each experiment consists of 50 simulation runs excluding a warm-up period of 30,000 time units. This warmup period

was determined using Welch's method (Welch, 1983) based on the combination of parameters that leads to the highest overtaking considered, namely $p = 0.8$, $c = 0.1$, $k = 1$, $L = 1$, and $N = 30$. The resulting warm-up period is then used for all experimental settings. A simulation run is terminated after 100,000 product pallets have been picked.

Out-of-sequence arrivals are generated in the simulation model as follows. A generator of product pallet holds a list of product pallets to be generated for orders in order buffer *OP*. For example, the list $[11, 12, 13, 14, 15, 16, 17]$ contains 7 product pallets each for a different order as denoted by the ID numbers. Each time a product pallet is generated, it is determined whether or not the new product pallet will overtake other product pallets based on the value of parameter p. In case of overtaking, the number of overtaken pallet x is sampled. Subsequently, the first x product pallets in the list is skipped and the product pallet at position $x + 1$ in the list is generated. In the previous list example, if $x = 2$ then product pallet 13 is generated. The updated list excludes this pallet.

5.5.3 Single product per order

First, we consider the case where each order consists of exactly one product ($k = 1$) and the pipeline capacity $N = 10$. Only one item is required for each product. We then investigate the effect of the overtaking distribution and the length of the pickable area (L) on the order throughput of the system and the queue length distribution at the carrousel. We initially set $L = 10$. Figure 5.11 depicts the order throughput of the workstation as a function of overtaking shape parameter c given overtaking probability $p = \{0.2, 0.5, 0.8\}$. The straight line in this figure is the maximal order throughput *without* out-of-sequence arrival ($p = 0.0$).



(a) $p = 0.2$. (b) $p = 0.5$. (c) $p = 0.8$.

Figure 5.11: Order throughput for $k = 1$, $L = 10$, $N = 10$.

The nearest-to-the-head picking policy performs well only when the product pallets arrive relatively in a good sequence, that is, with a low probability of overtaking. As products arrive more out-of-sequence, the performance of the nearest-to-the-head picking policy deteriorates compared to the other policies. With high overtaking, the resulting order throughput from the nearest-to-the-head policy may even be 17% lower than that of the backward search policy and that of the dynamic programming policy. The reason is that with high overtaking, the nearest-to-the-head policy requires the robot to return to the first position in the buffer after picking. This return

trip is performed without picking any other product pallets. Hence, the extra traveling without picking deteriorates order throughput. However, when products arrive at the buffer in a good sequence, most picking under nearest-to-the-head picking policy takes place at the foremost position in the buffer. No extra traveling occurs, which results in a high order throughput.

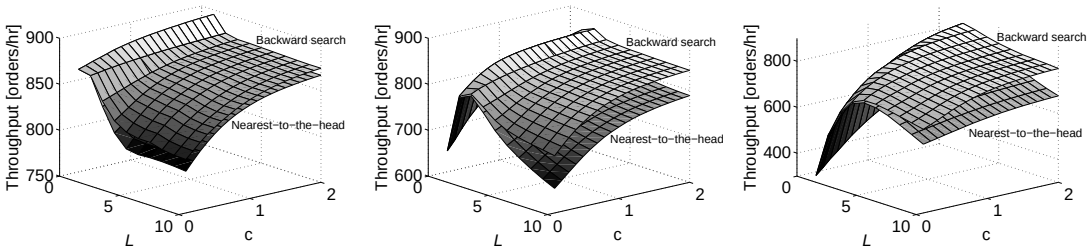
Contrary to the nearest-to-the-head policy, the nearest neighbor policy performs better at a high extent of out-of-sequence arrivals. This policy prevents the robot to travel empty by picking whichever product pallet nearest to the current location of the robot. Such way of working is advantageous particularly when products and orders are not sequenced neatly in the buffer. However, with low overtaking it may happen that the robot skips the product pallet at the foremost position of the product buffer and picks the product pallets in the rest of the buffer. The robot eventually returns to the foremost position of the product buffer. This return trip is performed without picking, causing extra traveling that deteriorates order throughput.

The dynamic programming and the backward search policies perform superior to the nearest-to-the-head and nearest neighbor policies. In general, the added value of these two smart picking policies is larger at high overtaking. The results suggest that constructing an optimal picking trip that minimizes the robot travel distance results in a high throughput under arbitrary extent of out-of-sequence arrivals. Additionally, the backward search policy is far less complex than the dynamic programming policy. For example, with $|R| = 10$ and $|S| = 1$, the number of trips evaluated is only 82 compared to the 512 TSPs that need to be solved using the dynamic programming approach. With this regard, the backward search policy may be more suitable for real-time application. Note that Figure 5.11 indicates that the throughput from the backward search policy is slightly higher than that of the dynamic programming policy. This is because the picking trip from the backward search policy may end before all products have been picked (see Section 5.4.4). Such an interrupted trip may, by coincidence, be advantageous for the order throughput if the new buffer content resulting from the interrupted trip is better sequenced. However, the throughput difference from the two policies is not significant. The simulation of the backward search policy is much faster (up to 85%) than that of the dynamic programming policy Therefore, from this point on we will continue our analysis using the backward search policy.

Figure 5.11 also indicates the effect of out-of-sequence arrivals. This can be seen by comparing the straight line (the maximal throughput when all pallets arrive in a good sequence) with the other lines in the figure. When 20% of all product pallets arrive out-of-sequence (Figure 5.11(a)), the decline in order throughput may already reach 11%. It is therefore beneficial to have a mechanism at the AS/RS that prevents out-of-sequence arrivals of products at the workstation. However, considering that the AS/RS is typically the bottleneck of the order-picking system, such mechanism should not compromise AS/RS throughput.

Next we investigate the effect of the pickable area size on the order throughput. We set $k = 1$ and $N = 10$ as before, but this time we vary the size of pickable area $L = \{1, 2, 3, ..., 10\}$. We also experiment on different overtaking probabilities $p \in \{0.2, 0.5, 0.8\}$. Figure 5.12 shows a 3-dimensional view of the order throughput as a function of pickable area size L and overtaking shape parameter c. Again, the

backward search policy gives higher order throughput than the nearest-to-the-head policy and the nearest neighbor policy (not shown in the figure).



(a) $p = 0.2$. (b) $p = 0.5$. (c) $p = 0.8$.

Figure 5.12: Order throughput for $k = 1$, $N = 10$, and variable L for nearest-to-the-head and backward search policies.

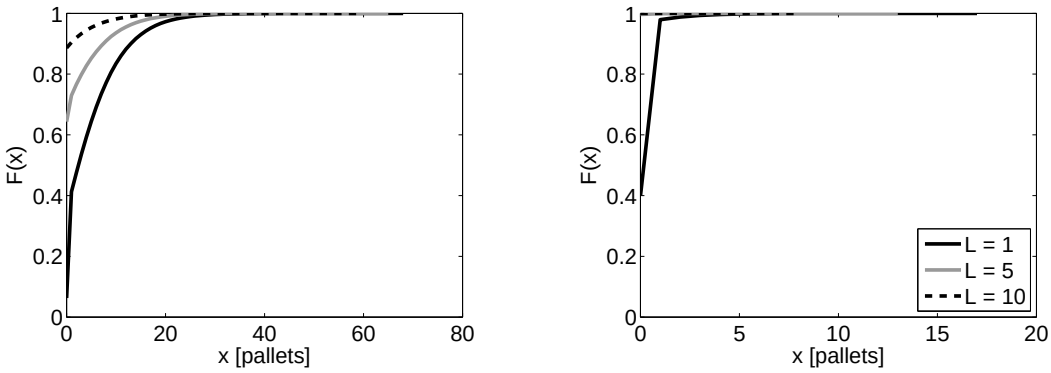
There appears to be an optimal size for pickable area L^* that results in the highest order throughput. The order throughput is influenced by the robot processing time as given in (5.8) and the carrousel rotation time. A large pickable area increases the travel time, which is a component of the robot processing time. In this sense, having a large pickable area may be detrimental to the throughput. On the contrary, a small pickable area causes more pallets to be sent to the carrousel buffer to avoid deadlock. Consequently, the carrousel rotates more frequently before a required pallet can enter the pickable area. This increases the carrousel rotation time, which leads to a lower throughput. As such, an optimal size of pickable area L^* exists that gives the highest order throughput under a certain overtaking distribution.

Figure 5.13 depicts the cumulative probability of the queue length at the carrousel beyond the pickable area using the backward search policy. This measure indicates the required carrousel capacity to avoid a deadlock situation. For an extremely high overtaking probability $p = 0.8$, the required carrousel buffer (cPP) capacity is between 10 and 30. For a more realistic overtaking probability $p = 0.5$ the required carrousel capacity is far less, even with the smallest possible size of pickable area $L = 1$.

5.5.4 Multiple products per order

Now we assume that each order requires more than one product. That is, multiple product pallets are needed for an order, but each product pallet is only picked once (one item per product). As a worst-case scenario we study the system performance under high overtaking $p = 0.8$ with a large pipeline capacity $N = 30$. Two cases are distinguished, namely fixed and variable order length.

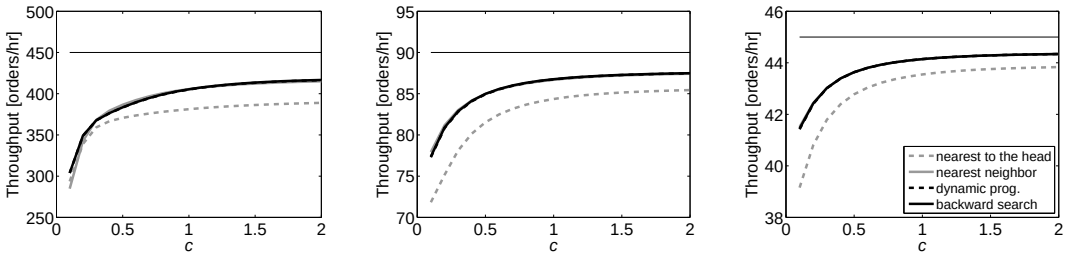
Figure 5.14 shows the resulting order throughput when the order lengths are fixed at $k = \{2, 10, 20\}$ for all four policies. The dynamic programming and backward search picking policies consistently give a high order throughput, which is not the case for the other two policies. This is also true given different sizes of pickable area $L = \{2, 6, 10\}$, as depicted in Figure 5.15. That is, dynamic programming and backward search picking policies are robust to the order length and the pickable area



(a) $p = 0.8$. (b) $p = 0.5$.

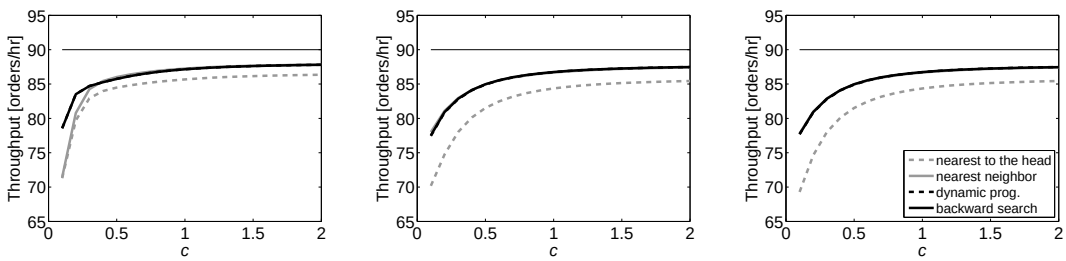
Figure 5.13: Cumulative probability of queue length at the carrousel beyond the pickable area for $c = 0.1$, $k = 1$, $N = 10$, with the backward search policy.

size. Note that in Figures 5.14 and 5.15 the straight lines give the maximal order throughput when all product pallets arrive in a good sequence ($p = 0.0$). In this case, the maximal throughput is constant at $\frac{1}{4} \times \frac{3600}{k}$ orders/hr.



(a) $k = 2$. (b) $k = 10$. (c) $k = 20$.

Figure 5.14: Order throughput for variable order length k, $p = 0.8$, $L = 5$, $N = 30$.



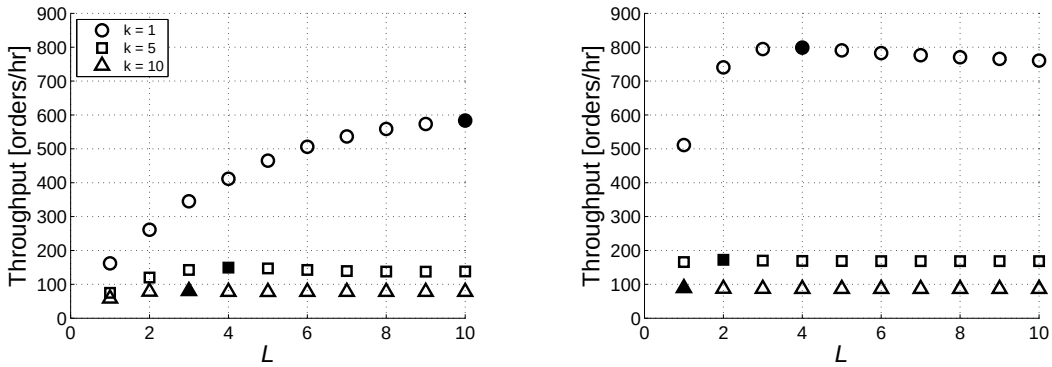
(a) $L = 2$. (b) $L = 6$. (c) $L = 10$.

Figure 5.15: Order throughput for variable pickable area size L, $p = 0.8$, $k = 10$, $N = 30$.

Next we investigate the optimal size of the pickable area with the backward search picking policy under different, fixed order lengths. Again, we consider the worst-case scenario of overtaking by setting $p = 0.8$ and $c = \{0.1, 1.0\}$. Subsequently the

order throughput is evaluated for various sizes of pickable area $L = \{1, 2, ..., 10\}$ and order length $k = \{1, 5, 10\}$. The results are shown in Figure 5.16.

There is an optimal size of pickable area L^* for a given order and overtaking profile. Increasing the pickable area does not necessarily increase the order throughput of the system. A large pickable area reduces the required carrousel capacity as shown in Figure 5.17. Consequently, if extending the pickable area is not expensive then it is favorable to make the pickable area sufficiently large to avoid excessive capacity requirement at the carrousel, taking into account the reduction of the order throughput. It should be noted that, in practice, having a large pickable area may increase the probability of picking errors by the robot. This is because the chance that the robot accidentally drops an item while traveling is higher when the travel distance is larger. If extending the pickable area is expensive, then keeping a small pickable area by considering the order profile is favorable. In this case, more capacity is needed at the carrousel because a small pickable area leads to more recirculation of product pallets.



(a) $c = 0.1$. (b) $c = 1.0$.

Figure 5.16: Order throughput for variable order length k and pickable area size L, $p = 0.8$, $N = 30$, with the backward search picking policy. The optimal size of pickable area L^* is shown in black.

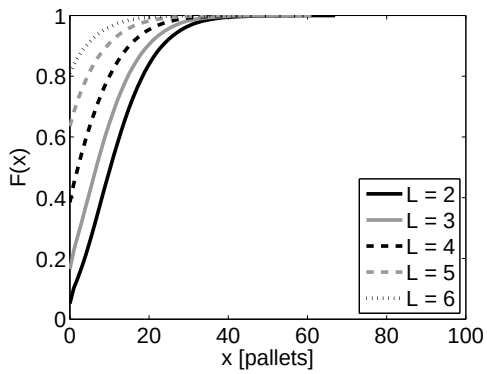


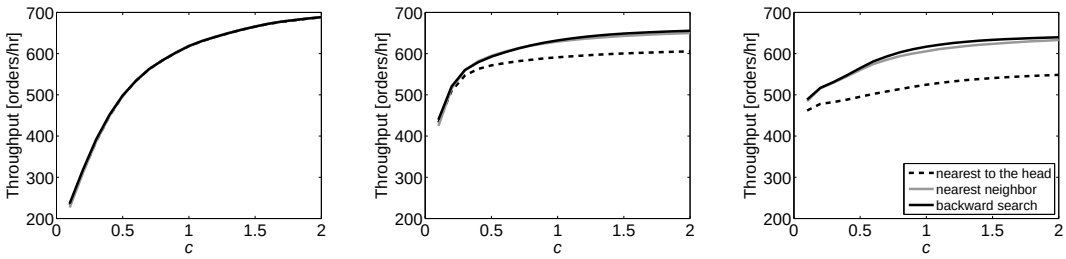
Figure 5.17: Cumulative probability of queue length at the carrousel with the backward search picking policy for $c = 0.1$, $k = 5$, $N = 30$, and different sizes of pickable area L.

Finally, we experiment with a variable order length. We assume order lengths that

follow a geometric distribution with a probability mass function

$$P\{K = k\} = (1-q)^{(k-1)}q, \;\; k = 1, 2, 3, ...$$

where k is the order length and $q = 0.8$. The mean order length is $E(K) = \frac{1}{q} = 1.25$. Using this value of q, the order length distribution is short-tailed and most orders will have a length of 1 SKU. Figure 5.18 depicts the order throughput from all three picking policies. The backward search picking policy gives the highest order throughput. The results again suggest that the importance of a smart picking policy increases with increasing size of the pickable area.



(a) $L = 2$. (b) $L = 6$. (c) $L = 10$.

Figure 5.18: Order throughput for geometrically distributed order length with variable pickable area size L, $p = 0.8$, and $N = 30$.

5.6 Conclusions

The performance of a novel conceptual design for an automated order-picking workstation processing multiple orders has been studied. We highlighted the typical problem the system has to deal with namely the out-of-sequence arrival of products at the workstation. That is, the sequence in which products arrive at the workstation is not the same as the sequence in which the products were requested. This has been modeled using a so-called overtaking function. Since there are a large number of possible picking trips for the robot, in particular if the size of the pickable area becomes longer, four picking policies have been investigated. The picking policy based on dynamic programming gives the highest order throughput. However, the approach is computationally expensive especially with increasing pickable area size, order length, and number of picks per product pallet. A so-called backward search policy is therefore proposed that is significantly cheaper but gives equally good performance. These two policies are preferred to simple policies such as nearest-to-the-head and nearest neighbor.

The conceptual design of the workstation includes a carrousel. This is actually a built-in solution of the workstation to overcome deadlock situation due to the out-of-sequence arrival of products. Having this carrousel, we do not require products to be delivered in a good sequence, nor do we require a sequencer (e.g., vertical buffer) in front of the workstation to rearrange product arrivals at the workstation. Such an

extra hardware is typically expensive. We also avoid sending unpickable products back to the conveyor loop or to the storage area.

The proposed overtaking function is a simple function that requires only two parameters. This function is capable of modeling different extents of out-of-sequence arrivals. Such a function is practical for performance evaluation when factors influencing out-of-sequence arrivals change.

A significant improvement in throughput can be gained by applying a smart picking policy at the workstation processing multiple orders. This is particularly the case when the number of pickable orders in the buffer at any time is large and there is a high extent of out-of-sequence arrival of products. Experimental results showed that the dynamic programming and the backward search picking policies, which strive to minimize the travel distance of the robot during a sequence of picking, perform well under low, medium, and high extent of out-of-sequence arrivals.

The final chapter concludes the dissertation, provides a vision towards design and operation of an end-of-aisle order-picking system, and suggests possible future research directions.

Appendix: dynamic programming for TSP

Let R be the set of pickable orders, $S \subseteq R$ denotes set of target orders. These are pickable orders whose product and/or order pallets are located at the head of the buffer. Let U be a subset of R containing m orders and one $s \in S$; this is a TSP of length m. The robot initial position is y_0. A distance matrix is constructed based on the position of product and order pallets whose IDs are contained in U. Distance d_{ij} is the distance traveled by the robot when picking one order, and is defined as

$$d_{ij} = |y_i - x_j| + |x_j - y_j| \tag{5.9}$$

for any $i, j \in U$ and $i \neq s$, where:

x_i = position of product pallet with ID i
y_i = position of order pallet with ID i

This formulation covers the distance from the previous order pallet i (current robot position) to the current product pallet j (pick item) and finally to the current order pallet j (drop item; new robot position). Additionally, we set an artificial distance $d_{sj} > 2 \times L$, where L is the size of the pickable area. This way, we prevent target order s to be picked in the middle of the picking trip and ensure that the target order is the last order picked in the picking trip. The following recursion is used to solve U, a TSP of length m.

```
For all i ∈ U:
    g(i, ∅) = 0
For n = 1 to m − 1:
    For all subsets V ⊆ U containing n orders:
        For all i ∉ V and i ∈ U:
            g(i, V) = min_{j∈V}{d_ij + g(j, V \ {j})}
```

where:

$g(i,V)$ = the shortest total distance for picking all orders in V starting from order $i \notin V$.

Once $g(i,V)$ is obtained, $p(i,V)$ is determined as the first order to pick after picking order i that gives the shortest total distance for picking all orders in V. This will be used when constructing the optimal picking trip.

Picking one order is defined as moving the robot from the order pallet of the previously picked order (current robot position), to the product pallet of the current order, and finally to its corresponding order pallet.

The recursion works as follows. First, the smallest subsets $V \subseteq U$ containing one order ($n = 1$ at line 3) are considered. For each V, we calculate the shortest distance to pick the only order in V after picking order i; $i \notin V$, $i \in U$; which is $g(i,V)$ with V containing one order. The first order to be picked following order i that gives the shortest distance is denoted as $p(i,V)$. Since at this point there is only one order in V, $p(i,V)$ will always be equal to that one order $v \in V$. Next, larger subsets $V \subseteq U$ containing two orders ($n = 2$ at line 3) are considered. Now we calculate for each V the shortest distance to pick two orders in V after picking order i; $i \notin V$, $i \in U$; which is $g(i,V)$ with V containing two orders. Again, we obtain $p(i,V)$, now with V containing two orders. This recursion is continued until all subsets $V \subseteq U$ containing $m-1$ orders have been evaluated and $g(i,V)$ and $p(i,V)$ for these largest subsets have been obtained. Note that in line 2 we set $g(i,\emptyset) = 0$ because we do not require the robot to return to its initial position after finishing the trip. This is different from the classical TSP formulation Bellman (1962) where $g(i,\emptyset)$ is equal to the distance from the order pallet i to the initial position of the robot. Also, this recursion is only used for solving a TSP of length $m > 1$, that is, $U \subseteq R$ and $|U| > 1$. For $m = 1$ the solution is straightforward namely picking target order $s \in U$.

Finally, the optimal picking trip r_m given m orders in U requires a travel distance of

$$D_{r_m} = g(y_0, U) = \min_{j \in U}\{d_{y_0 j} + g(j, U \setminus \{j\})\} \tag{5.10}$$

The minimum travel distance is the distance from the current robot position plus the shortest total distance for picking all orders in U starting from order j.

The optimal picking trip r_m for set U is obtained from sequencing the values of p. The first order to be picked from the current robot position y_0 is $p(y_0, U) = k_1$. This order k_1 is actually order j that gives D_{r_m} in (5.10). The second order to be picked is $p(y_0, U \setminus \{k_1\}) = k_2$, where $U \setminus \{k_1\}$ contains $m-1$ orders. The third order to be picked is $p(y_0, U \setminus \{k_1, k_2\}) = k_3$, where $U \setminus \{k_1, k_2\}$ contains $m-2$ orders. This way of working is continued until a picking trip $r_m = \{k_1, k_2, k_3, ..., s\}$ has been constructed, which gives the sequence of order picking for the robot.

The above description of dynamic programming is based on the assumption that each order requires exactly one product. In this case, each product pallet is uniquely identified by the order ID to which the product belongs. If an order consists of multiple products, each product pallet can be identified uniquely by the order ID and the position of each product pallet for that order at the buffer.

6 Conclusions

Abstract | This chapter presents the main results of the dissertation, a vision towards designing and operating end-of-aisle order-picking systems, and some possible future research directions.

The objective of this dissertation is to develop methods for the design, modeling, and control of an end-of-aisle order-picking system (OPS) with remotely located order-picking workstations. Specifically, the focus lies on quantifying and improving the throughput and flow time performance of manual and automated order-picking workstations. Several methods have been developed to address the three research questions posed in the introduction chapter. Summarizing, the contributions of this dissertation are: 1) a multi-level hierarchical architecture for system modeling and control, 2) an aggregate modeling method for performance analysis of manual order-picking workstations, 3) picking strategies capable of overcoming out-of-sequence arrival of products at an automated order-picking workstation.

6.1 Conclusions

Architecture of an end-of-aisle order-picking system

An end-of-aisle OPS consists of various processing units, each having its own control policy that contributes to the overall performance of the OPS. For example, a miniload has a product retrieval sequencing policy and a workstation has a picking

policy. Since one of the main strengths of simulation is to evaluate what-if scenarios, a simulation model for performance analysis should allow easy adjustment and implementation of different policies and configurations. For this purpose a modular framework for simulation modeling of an end-of-aisle OPS is required.

An architecture for modeling a complete end-of-aisle OPS with remotely located workstations has been developed in Chapter 2. It provides a comprehensive framework to investigate the effect of the system configuration on the throughput and flow time performance. Different areas and operational layers can be clearly distinguished in the architecture, which makes it intuitively easy to comprehend. A decentralized, two-layer control structure has been applied. This hierarchical control structure allows easy incorporation of various control heuristics for both miniloads and order-picking workstations. In such a control structure, information is contained locally and is exchanged only when it is needed.

The process algebra-based language χ (Chi) is suitable for modeling such a multi-layered architecture consisting of concurrent processes. This has been shown in an example where the architecture is used in a simulation study of an end-of-aisle OPS to investigate the effect of different control heuristics and system configurations on the mean flow time of products and orders. Owing to the subsystems and the decentralized control structure used in the architecture, necessary changes are kept to a minimum.

Aggregate queueing models for performance analysis

A manual order-picking workstation is characterized by a number of stochasticities that are difficult to quantify, including pick time, setup time, picker behavior, and breakdown. Additionally, only limited operational data is typically available from such a workstation. A valid model, however, requires model parameterizations based on the available data.

An aggregate modeling technique based on the concept of EPT (Effective Process Time) has been developed for a manual order-picking workstation processing one order at a time. The aggregate queueing model is used to simulate the product and order flow time distribution of an operating workstation. The key aspect of the aggregate model is that it does not require quantification of each stochastic component that affects the flow time performance. The modeling technique uses measurable data while aggregating the contributors of the various stochasticities present at an order-picking workstation. An EPT distribution, which represents the aggregation of all process time components of an order-picking workstation, is obtained directly from arrival and departure times of products at the workstation using a sample path equation. With this aggregation approach, the model parameters are estimated from data that can be easily obtained from the shop-floor. Two model variants have been developed namely aggregate models with FCFS (First-Come-First-Serve) and non-FCFS processing of products and orders.

In Chapter 3 the aggregate model with FCFS processing is presented. The aggregate model is a polling queueing system that uses EPT distributions as input. The EPTs of the first (product) totes of an order are distinguished from the EPTs of the remaining totes of an order. This is referred to as the 1st tote difference approach. Such a

differentiation is necessary because setup activities (e.g., preparing a new order tote, scanning the barcode of the order tote) are only performed while processing the first tote of an order. Consequently, the EPT distribution of the 1st tote of an order is significantly different to that of the remaining totes of an order. Experimental results indicate that the aggregate model with the 1st tote difference approach is significantly more accurate than the one without this approach in predicting the mean flow time of products and orders under different order length distributions.

In Chapter 4 the aggregate model with non-FCFS processing is presented. It is a single-server queueing system with an infinite buffer. Within the buffer there are a number of infinite queues, each containing products for the same order. EPT distributions with 1st tote difference are used as input for the model. Additionally, the non-FCFS processing of products and orders is modeled using an overtaking distribution and a so-called decision probability. Experimental results show the accuracy of this aggregate model in predicting the mean product and order flow time from an order-picking workstation with non-FCFS processing.

Both aggregate model variants have been validated using data from a real, operating order-picking workstation. The data indicates that 18.3% of all products and 4.6% of all orders are processed in a non-FCFS sequence. Consequently, the aggregate model with non-FCFS processing is more suitable for this data set. The prediction errors for the mean product and order flow times are less than 1.35% and 6.31%, respectively. Experimental results have also shown that the product and order flow time distributions of the operating order-picking workstation are well-predicted by the aggregate model.

System configuration and picking strategies for an automated order-picking workstation

A typical problem of an end-of-aisle OPS with remotely located workstations is the out-of-sequence arrivals of products. That is, products for multiple orders are retrieved from the miniload but they do not arrive at the order-picking workstation in the same sequence as requested. Combined with inefficient picking operations at the workstation, this situation deteriorates the throughput performance of the OPS. Moreover, out-of-sequence arrivals may cause a system deadlock at the workstation when left untreated.

In Chapter 5 a design of an automated order-picking workstation with an integrated carrousel mechanism has been proposed, where multiple orders can be processed simultaneously. The design is capable of handling an arbitrary extent of out-of-sequence arrivals. The out-of-sequence arrivals have been modeled using a simple function with two parameters. This function can be used to create various distributions of out-of-sequence arrivals. Moreover, four picking policies have been considered for the proposed workstation design. It has been shown that a significant improvement in throughput can be gained by applying a smart picking policy at the workstation, especially when there are many products in the buffer and there is a high extent of out-of-sequence arrivals. A picking policy that minimizes the picker travel distance in a sequence of picking a number of products performs well under a low, medium, and high extent of out-of-sequence arrivals.

6.2 End-of-aisle OPS: a vision

When designed and operated properly, an end-of-aisle OPS with remotely located workstations is an appealing alternative for an automated warehouse. However, there are a number of issues that may undermine the performance of such an OPS. Throughout this dissertation, some of these issues have been addressed and methods have been proposed to overcome them. Reflecting on the findings in the previous chapters of this dissertation, a vision of an end-of-aisle OPS with remotely located workstations is proposed.

One comprehensive framework

Configuring miniloads and workstations properly is essential to achieving the expected performance. However, numerous factors exist that affect the overall performance of an end-of-aisle OPS. Models capable of providing insight into the OPS performance under different configurations are therefore crucial. With this regard, the modeling architecture proposed in Chapter 2 offers a wide range of functionality in exploring many of these factors. Consequently, one does not need to build a separate simulation model to analyze each factor. This is made possible by the decentralized control structure and concurrent processes applied in the model architecture (see Sections 2.5.2 and 2.5.4).

Some of the strategies that can be integrated into the model architecture are as follows.

- Storage assignment strategy
  A storage assignment strategy determines the assignment of storage locations to various SKUs in a miniload. Examples of such a strategy are dedicated storage, random storage, class-based storage, and turnover-based storage. A storage assignment strategy can be incorporated in the local controller of a miniload. Since each miniload has its own local controller, it is also possible to implement different storage assignment strategies across the miniloads.

- Retrieval sequencing strategy
  Given a number of products to be retrieved from a miniload in a single retrieval trip of the miniload storage/retrieval (S/R) crane, the retrieval sequencing strategy gives the sequence in which the products should be retrieved. A smart sequencing strategy reduces the total retrieval time of products within a retrieval trip, hence increases the throughput of the miniload. Similar to the storage assignment strategy, a retrieval sequencing strategy can be implemented in the local controller of a miniload and thus each miniload may have its own retrieval sequencing strategy.

- Replenishment strategy
  A replenishment strategy is required to maintain the availability of each SKU in the warehouse. Such a strategy determines when replenishment products for a certain SKU should be ordered and in what quantity. Four common replenishment strategies available in the literature are (Silver et al., 1998):

1. Order-point, order-quantity (s,Q) system.
2. Order-point, order-up-to-level (s,S) system.
3. Periodic-review, order-up-to-level (R,S) system.
4. Combined periodic-review and order-point, order-up-to-level (R,s,S) system.

In principle, it is possible to apply these and other replenishment strategies at the replenishment planner within the order layer of the proposed architecture. In Chapter 2, for example, the (s,Q) replenishment strategy has been implemented. With this regard, the architecture can also be used to investigate e.g., the warehouse service level as a result of applying a specific replenishment strategy.

- Order release strategy
  An order release strategy determines the starting time of order processing at the workstation. A new order is released once all products required for the order are present in the miniloads and there is a workstation available to process the order. Once an order is released, products required for the order are retrieved from the miniloads and are sent to the workstation.
  An order release strategy affects the utilization of the pipeline capacity. Recall that the pipeline capacity gives the maximum number of products that can be simultaneously on the way to the workstations. An order release strategy should be designed such that the pipeline capacity is highly utilized, resulting in a continuous flow of products to the workstation. This reduces the interarrival time of products at the workstation, hence providing the pickers with sufficient work. Such an order release strategy can be implemented in the global controller of the miniloads, which exchanges information with the global controller of the workstations to check the availability of the workstations.

- Job allocation strategy
  A job allocation strategy distributes retrieval jobs of products across all available miniloads. Recall that numerous products need to be retrieved from the miniloads to fulfill orders at the workstations. Since there are multiple miniloads available, a job allocation strategy selects the miniload that has to retrieve certain products required for an order. Within the proposed architecture, a job allocation strategy can be implemented in the global controller of the miniloads. This global controller has the information about the number of available items per SKU in each miniload.

The proposed modeling architecture also supports integration with EPT-based aggregate models of order-picking workstations proposed in Chapters 3 and 4. Incorporating an aggregate model into the architecture for design/redesign purposes is advantageous because data measured directly from an operating order-picking workstation is used. That is, the aggregate model is used as a black box approach that gives the effective rather than the expected performance of a workstation.

For system redesign purposes, one may use real data to validate the resulting throughput/flow time performance from a simulation using the proposed architecture. Once

validated, simulations involving more miniloads and/or workstations than the current design can be performed to evaluate the added value of extra investments. Such an evaluation is essential because requirements for warehouses are increasingly changing and significant investment on material handling technologies is involved in an end-of-aisle OPS.

Emphasis on out-of-sequence arrivals

A high-performing end-of-aisle OPS with remotely located workstations seeks an efficient way of dealing with out-of-sequence arrivals, a problem unique to this type of OPS. In such an OPS, multiple miniloads feed products to multiple workstations, which in turn may process multiple orders simultaneously. It is important to understand the causes of out-of-sequence arrivals and the alternatives to overcome them.

There are numerous factors that cause products to arrive out-of-sequence at the workstation, including:

- The number of miniloads
  The primary factor affecting the out-of-sequence arrivals is the number of miniloads. In the simplest case of only one miniload feeding products to an order-picking workstation via a fixed routing, products will *always* arrive in a correct sequence as requested by the workstation. The extent of out-of-sequence arrival increases as more miniloads are present to serve the workstation. Products retrieved from different miniloads may overtake one another on their way to the workstation. However, having multiple miniloads increases the system robustness against failure. That is, products can still be retrieved from other miniloads when a miniload S/R crane fails. Additionally, multiple miniloads allow for a higher throughput of retrieved products.

- Retrieval batch size
  Retrieval batch size affects the retrieval time of products by the miniload S/R crane and therefore contributes to out-of-sequence arrivals. The retrieval time of a batch of products increases with the retrieval batch size. Consequently, the first (last) product retrieved by the miniload crane has the longest (shortest) waiting time before it is deployed onto the output buffer of the miniload. It could happen that a product for an earlier order has been retrieved but this product has to wait for other products in the same batch to be retrieved before they are deployed altogether onto the output buffer. Meanwhile, products for later orders retrieved from other miniloads may already on their way to the workstations, which eventually leads to out-of-sequence arrivals.
  A small batch size causes a higher mean retrieval time per product than a large batch size. This is because a small batch size requires the miniload S/R crane to travel more frequently from/to the drop-off point at the miniload output buffer. However, a small batch size allows products to be immediately deployed in the output buffer once they have been retrieved. As such, one should consider the trade-off between the throughput and the extent of out-of-sequence arrivals due to the retrieval batch size.

- SKU availability across miniloads
  The availability of SKUs is reflected by two characteristics namely the number of product totes per SKU and the distribution of SKUs across the miniloads. Ideally, each miniload contains all SKUs in the warehouse. This allows each miniload to fulfill incoming orders. However, this setting is more likely for warehouses serving fast-moving products than those serving slow-moving products. For slow-moving products it may not be economical to store an SKU in multiple product totes due to a high holding cost and a low turnover rate.
  In the extreme case where only one product tote is available for each SKU, there is no choice so as from which miniload the product will be retrieved. If one wants to reduce out-of-sequence arrivals in this case, the only way is to retrieve a product once it is no longer possible for this product to overtake products retrieved earlier by other miniloads. However, if multiple product totes for an SKU are available and these totes are distributed equally across the miniloads, then it is possible to formulate a job allocation strategy that reduces out-of-sequence arrivals early at the miniload.

- Order profile
  Order profile refers to the distribution of the number of SKUs typically required by an order in the warehouse. Short orders requires few SKUs, while long orders require many SKUs. The order profile affects the order release strategy, which eventually influences the out-of-sequence arrival pattern.
  If the warehouse primarily serves short orders, products for many orders need to be retrieved simultaneously to ensure a sufficient number of products being fed to the workstation. In this case, there will be a high chance that products for different orders overtake one another on their way to the workstation, assuming that the products are retrieved from different miniloads.
  However, if the warehouse primarily serves large orders, retrieving products for a few orders simultaneously already leads to sufficient products being fed to the workstation. In this case, there will be more overtaking of products within an order than between orders. Since out-of-sequence arrivals refer to overtaking of products between orders, the extent of out-of-sequence arrivals should be lower than that in the case of short orders.

During the design phase, one can measure the extent of out-of-sequence arrivals from a detailed simulation model using the proposed architecture in Chapter 2. It is also possible to analyze the effect of various factors on out-of-sequence arrivals. That is, using a detailed simulation model one can understand which factors significantly affect the extent of out-of-sequence arrivals.

Out-of-sequence arrivals eventually need to be treated early at the miniload and/or late at the order-picking workstation. The choice of where to deal with out-of-sequence arrivals should be based upon understanding the direct consequences of each option. At one extreme, the problem can be completely eliminated by retrieving products and sending them in the correct sequence as required by the order-picking workstation. Having the products arriving in the correct sequence, there is no need for additional mechanisms at the order-picking workstation. That is, the required order-picking workstation is reduced to a simple, single-server workstation with first-come-first-serve processing. A robotized order-picking workstation in this

case will require a simpler technology as the robot can be made static at one location. This way of working, however, comes at the cost of the miniload throughput as extra delay in retrieving products is required to sequence the products. At the other extreme, one may completely neglect the out-of-sequence problem at the miniload and treat this later at the workstation. In this case, the miniloads can work at their full capacity. This has been done in Chapter 5 where the miniload S/R cranes were assumed to be the bottleneck of the entire OPS. However, this approach requires additional mechanisms at the order-picking workstation to handle the out-of-sequence arrivals. Furthermore, treating out-of-sequence arrivals at the workstation may still result in a capacity loss at the workstation. For the system discussed in Chapter 5, for example, it has been shown that a 20% out-of-sequence arrivals of products may lead to a capacity loss of the workstation up to 10%.

The proposed architecture may be used to identify ways to reduce the out-of-sequence arrivals early at the miniload. For example, recall that a hierarchical control structure has been applied in the architecture resulting in global and local control layers. The global control layer of the miniload area contains the list of available SKUs in each miniload and decides the allocation of product retrievals among the miniloads. A job allocation strategy can be applied at the global control layer such that products for earlier orders are retrieved from the miniload that contains the required SKU and is located nearest to the workstation. That is, the travel distance of products required by the order released earlier is kept to a minimum. This reduces the chance that a product for an order released later arrives earlier at the workstation. Such a strategy seeks to reduce overtaking of products on their way to the workstations assuming that all SKUs are contained in each miniload. However, one should take precautions when attempting to reduce out-of-sequence arrivals early at the miniload. The miniload crane is often considered as the bottleneck of the OPS due to the high storage/retrieval time. As such, any attempt to reduce out-of-sequence arrivals early at the miniload should not severely compromise the miniload throughput.

Alternatively, the out-of-sequence arrivals can be handled later at the order-picking workstation in many ways. A smart picking policy overcomes the throughput loss at the workstation due to out-of-sequence arrivals. Examples of such a picking policy have been addressed in Chapter 5 for an automated order-picking workstation processing multiple orders. Whichever picking policy is used, the execution time of the policy needs to be sufficiently small to avoid extra delay in real-time implementation. Note that sufficient buffer space and an additional mechanism may be required at the workstation to prevent products from being recirculated to the closed-loop conveyor. This could be, for example, a carrousel mechanism as discussed in Chapter 5. Also, product sorting mechanisms can be applied at the order-picking workstation. Some examples of product sorting is using a sorting device in front of the workstation or using the available buffer lanes to sort products as in Chapters 3 and 4.

High-performing order-picking workstation

A properly configured order-picking workstation is indispensable when dealing with out-of-sequence arrivals of products. Two types of workstations can be distinguished. In a *single-order* workstation one order is processed at a time, while in a *multiple-*

order workstation multiple orders are processed simultaneously. For both types, products for multiple orders are typically present at the workstation.

Out-of-sequence arrivals deteriorate order throughput in both types of order-picking workstation. In a single-order workstation, out-of-sequence arrivals may cause pickers to be idle waiting for the products required for the currently processed order, as discussed in Chapters 3 and 4. That is, many products may be available in the buffer but none of these are required for the currently processed order. In a multiple-order workstation, out-of-sequence arrivals cause a higher processing time due to extra traveling of the picker, as discussed in Chapter 5. That is, products are not queued in the correct sequence in the buffer.

The order profile affects the required workstation configuration. If orders are predominantly short, many orders will be needed to fill the pipeline capacity. Hence, there is a higher chance of throughput loss due to out-of-sequence arrivals. Assuming that no mechanism is applied at the miniload to reduce out-of-sequence arrivals, more buffer space may be required at the workstation.

For a manual single-order workstation with first-in-first-out buffer as discussed in Chapters 3 and 4, it is crucial that the workstation is designed such that the picker always has access to the products required for the order currently being processed. This is necessary to avoid deadlock, a situation where products for the current order are out-of-reach to the picker. For this reason, 'smart' buffer lanes are required that route products for the current order such that these products are always accessible to the picker, hence preventing deadlock. An example of such a buffer is discussed by Jordan (2008).

An automated multiple-order workstation discussed in Chapter 5 should be configured such that the travel distance of the picker can be kept small. Recall that in such a workstation the picker can only pick products within the so-called pickable area. Having a large pickable area may increase the travel distance of the picker resulting in a lower throughput. A too small pickable area may cause a deadlock situation or frequent recirculation of products onto the closed-loop conveyor, which also leads to a throughput loss. With this regard, there exist an optimum size of the pickable area that results in the minimum travel time. Additionally, a smart picking policy that minimizes the travel distance of the picker as discussed in Chapter 5 is desirable.

Visible real-time performance measures

Keeping the performance of the OPS visible at all time is the first step towards continuous improvement efforts. Monitoring the system performance in real-time allows one to detect unexpected behavior early and subsequently to perform corrective actions quickly. Of utmost importance for order-picking workstations is the effective process time and utilization. Having these performance measures, it is possible to evaluate whether or not the workstation operates according to the expected performance. Possible improvements to the current way of working can be formulated based on the observed performance.

The effective process time can be used for real-time performance monitoring of an order-picking workstation. Chapters 3 and 4 elaborate how EPT realizations at a

workstation are calculated using a simple formula. Since only arrival and departure times of products are required as input, the method should therefore be feasible for real-time implementation.

Aggregate models for performance analysis

Aggregate modeling is an appealing alternative for accurate performance prediction of (automated) order-picking workstations under limited data availability. One of the main advantages of aggregate models is that one does not need to quantify separately each stochastic behavior involved in the system. Instead, they are aggregated into a single aggregate process time distribution. The aggregate process time accumulates the contributors of all disturbances affecting the processing. In the EPT approach presented in Chapters 3 and 4, EPTs as input to the aggregate model are calculated from typically available data namely arrival and departure times of products. This highlights the applicability of the method in practice. It is also possible to integrate aggregate models into other (detailed) simulation models. For example, a detailed model of a miniload can be integrated with an aggregate model of an order-picking workstation. This way, one can investigate the performance of a specific policy at the miniload while keeping the workstation model simple yet accurate.

As a final note, the EPT distribution calculated using the sample path equation can also be used for an analytical queueing network model. Having the EPT distribution as the only input simplifies the model and thus improves mathematical tractability. Such an analytical model is particularly useful to have a quick overview on the performance of different designs.

6.3 Future research

Design of multiple-order workstations

The design proposed in Chapter 5 is merely one out of the many possible designs for automated workstations processing multiple orders and dealing with out-of-sequence arrivals. Given the physical constraints such as the maximum length and width of the workstation, one has to decide upon the number and the type of buffers to be used. For example, three buffers are present inside the pickable area of the proposed design, two of which are used as product buffers. The performance of a workstation having other buffer compositions under various order profiles and out-of-sequence arrival distributions has not been investigated. This can be, for example, a workstation with only one product buffer or shared product/order buffers. Mechanisms at the workstation other than a carrousel for overcoming out-of-sequence arrivals and preventing deadlocks are also subject to further investigation. Paese (2010) and Heling (2011) have addressed some of these issues.

A different design is required for a manual order-picking workstation processing multiple orders. Owing to the human pickers, such a workstation is more flexible in the picking process. However, traveling from one location to another is a highly tedious job. With this regard, a manual order-picking workstation processing multi-

ple orders requires a different layout from its automated counterpart as discussed in Chapter 5.

Aggregate models for multiple-order workstations

Aggregate models in Chapters 3 and 4 are developed for workstations processing one order at a time. For such a workstation it is known that only products for the current order are picked. This is not the case in a multiple-order workstation. Here, products for different orders are picked one after another. Additionally, the two aggregate model variants developed previously assume that EPT is not calculated if no products for the current order are present in the buffer. With this regard, they are not directly applicable for multiple-order workstations. Aggregate modeling of multiple-order workstations is therefore subject to future research.

Miniload operation

The way the miniloads are operated affects the out-of-sequence arrivals and may cause deadlocks at the order-picking workstation. Reduction of out-of-sequence arrivals, for example, can be performed early at the miniload. This is done, in principle, by only allowing products that are needed in the very near future to be sent to the workstation. If more than one miniload is available, out-of-sequence arrivals can be reduced by selecting the miniload such that the travel distance of products for the orders released earlier is minimized. For one particular miniload, a dynamic programming-based approach similar to the one used in the picking policy of Chapter 5 can be applied to determine the sequence of retrieving a number of products that minimizes the total retrieval time of a batch of products. Also, recall that a workstation has a finite capacity in buffering products and orders. Preventing deadlock can be done by setting rules on which products are allowed to be sent to the workstation at a certain time, given the buffer content at the workstation. As a final note, the first step towards aggregate modeling of miniloads has been addressed by de Koning (2008).

Bibliography

Anderson, C., Bartholdi III, J. J., 2000. Centralized versus decentralized control in manufacturing: lessons from social insects. In: *Proceedings Complexity and Complex Systems in Industry*. September 19-20, Warwick, UK, pp. 92–105.

Andriansyah, R., de Koning, W. W. H., Jordan, R. M. E., Etman, L. F. P., Rooda, J. E., 2008. Simulation study of miniload-workstation order picking system, *Systems Engineering Report 2008-07*, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven.
URL `http://se.wtb.tue.nl/sereports`

Ascheuer, N., Grötschel, M., Abdel-Hamid, A. A., 1999. Order picking in an automatic warehouse: solving online asymmetric TSPs. *Mathematical Methods of Operations Research* 49 (3), 501–515.

Baker, P., Halim, Z., 2007. An exploration of warehouse automation implementations: cost, service and flexibility issues. *Supply Chain Management: An International Journal* 12.

Bartholdi III, J. J., Eisenstein, D. D., Foley, R. D., 2001. Performance of bucket brigades when work is stochastic. *Operations Research* 49 (5), 710–719.

Bartholdi III, J. J., Hackman, S. T., 2010. *Warehouse and Distribution Science*. The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, release 0.93 Edition.

Basten, A. A., 1998. *In terms of nets: system design with Petri nets and process algebra*. Ph.D. thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology.

Bellman, R., 1962. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM* 9 (1), 61–63.

Blom, E. L. W., 2007. EPT and flowtime distribution. Master's thesis, SE 420513, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology.

Bozer, Y. A., Cho, M., 2005. Throughput performance of automated storage/retrieval systems under stochastic demand. *IIE Transactions* 37 (4), 367–378.

Bozer, Y. A., White, J. A., 1990. Design and performance models for end-of-aisle order picking systems. *Management Science* 36 (7), 852–866.

Bozer, Y. A., White, J. A., 1996. A generalized design and performance analysis model for end-of-aisle order-picking systems. *IIE Transactions* 28 (4), 271–280.

Caputo, A. C., Pelagagge, P. M., 2006. Management criteria of automated order picking systems in high-rotation high-volume distribution centers. *Industrial Management and Data Systems* 106 (9), 1359–1383.

Chincholkar, A. K., Chetty, O. V. K., 1996. Simultaneous optimisation of control factors in automated storage and retrieval systems and FMS using stochastic coloured Petri nets and the Taguchi method. *International Journal of Advanced Manufacturing Technology* 12 (2), 137–144.

Dallari, F., Marchet, G., Melacini, M., 2009. Design of order picking system. *International Journal of Advanced Manufacturing Technology* 42 (1-2), 1–12.

Dallari, F., Marchet, G., Ruggeri, R., 2000. Optimisation of man-on-board automated storage/retrieval systems. *Integrated Manufacturing Systems* 11 (2), 87–93.

de Koning, W. W. H., 2008. Modeling a storage and retrieval system: architecture and model aggregations. Master's thesis, SE 420562, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology.

de Koster, R., Le-Duc, T., Roodbergen, K. J., 2007. Design and control of warehouse order picking: a literature review. *European Journal of Operational Research* 182 (2), 481–501.

de Koster, R., 1994. Performance approximation of pick-to-belt orderpicking systems. *European Journal of Operational Research* 72 (3), 558–573.

Dotoli, M., Fanti, M. P., 2005. A coloured Petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: a control perspective. *International Journal of Computer Integrated Manufacturing* 18 (2-3), 122–136.

Drury, J., 1988. Towards more efficient order picking, *IMM Monograph 1*, Institute of Materials Management, Cranfield, UK.

Eisenberg, M., 1972. Queues with periodic service and changeover time. *Operations Research* 20 (2), 440–451.

Ekren, B. Y., Heragu, S. S., Krishnamurthy, A., Malmborg, C. J., 2010. Simulation based experimental design to identify factors affecting performance of AVS/RS. *Computers and Industrial Engineering* 58 (1), 175–185.

Eynan, A., Rosenblatt, M. J., 1994. Establishing zones in single-command class-based rectangular AS/RS. *IIE Transactions* 26 (1), 38–46.

Ferguson, M. J., Aminetzah, Y. J., 1985. Exact results for nonsymmetric token ring systems. *IEEE Transactions on Communications* 33 (3), 223–231.

Fokkink, W., 2000. *Introduction to Process Algebra*. Springer-Verlag, Berlin.

Foley, R. D., Frazelle, E. H., 1991. Analytical results for miniload throughput and the distribution of dual command travel time. *IIE Transactions* 23 (3), 273–281.

Foley, R. D., Frazelle, E. H., Park, B. C., 2002. Throughput bounds for miniload automated storage/retrieval systems. *IIE Transactions* 34 (10), 915–920.

Foley, R. D., Hackman, S., Park, B. C., 2004. Back-of-the-envelope miniload throughput bounds and approximations. *IIE Transactions* 36 (3), 279–285.

Frazelle, E. H., 1996. *World-Class Warehousing*. Logistics Resources International inc.

Fukunari, M., Malmborg, C. J., 2009. A network queueing approach for evaluation of performance measures in autonomous vehicle storage and retrieval systems. *European Journal of Operational Research* 193 (1), 152–167.

Goetschalckx, M., Ashayeri, J., 1989. Classification and design of order picking. *Logistics Information Management* 2 (2), 99–106.

Gu, J., Goetschalckx, M., McGinnis, L. F., 2010. Research on warehouse design and performance evaluation: a comprehensive review. *European Journal of Operational Research* 203 (3), 539–549.

Guenzi, R. D., Feie, M. S., 2003. Robotic order picking system. US Patent no. US 6,626,632 B2.

Hair Jr., J. F., Black, W. C., Babin, B. J., Anderson, R. E., Tatham, R. L., 2006. *Multivariate Data Analysis*, 6th Edition. Pearson Prentice Hall, New Jersey.

Heling, J. W. E., 2011. Design of an automated item picking workstation. Master's thesis, SE 420650, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology.

Hirayama, T., Hong, S. J., Krunz, M. M., 2004. A new approach to analysis of polling systems. *Queueing Systems* 48 (1-2), 135–158.

Hofkamp, A. T., Rooda, J. E., 2008. Chi 1.0 reference manual, *Systems Engineering Report 2008-04*, Eindhoven University of Technology, Eindhoven.
URL `http://se.wtb.tue.nl/sewiki/chi`

Hopp, J. W., Spearman, M. L., 2000. *Factory Physics: Foundations of Manufacturing Management*, 2nd Edition. McGraw Hill, New York.

Hopp, J. W., Spearman, M. L., 2008. *Factory Physics*, 3rd Edition. McGraw Hill, New York.

Houshyar, A., Chung, I., 1991. Using simulation to compare different automated storage/retrieval systems designs. *Computers and Industrial Engineering* 21 (1-4), 629–633.

Hsieh, S., Hwang, J. S., Chou, H. C., 1998. A Petri-net-based structure for AS/RS operation modelling. *International Journal of Production Research* 36 (12), 3323–3346.

Hur, S., Lee, Y. H., Lim, S. Y., Lee, M. H., 2004. A performance estimation model for AS/RS by M/G/1 queuing system. *Computers and Industrial Engineering* 46 (2), 233–241.

Jacobs, J. H., Etman, L. F. P., van Campen, E. J. J., Rooda, J. E., 2003. Characterization of operational time variability using effective processing times. *IEEE Transactions on Semiconductor Manufacturing* 16 (3), 511–520.

Jacobs, J. H., Etman, L. F. P., van Campen, E. J. J., Rooda, J. E., 2006. Quantifying variability of batching equipment using effective process times. *IEEE Transactions on Semiconductor Manufacturing* 19 (2), 269–275.

Johnson, D. S., 1990. Local optimization and the traveling salesman problem. In: *Automata, Languages and Programming*. Vol. 443 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 446–461.

Jordan, R. M. E., 2008. Modeling the item picking area of the plus retail compact picking system. Master's thesis, SE 420554, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology.

Khouja, M., Goyal, S., 2008. A review of the joint replenishment problem literature: 1989-2005. *European Journal of Operational Research* 186 (1), 1–16.

Kim, B., Graves, R. J., Heragu, S. S., Onge, A. S., 2002. Intelligent agent modeling of an industrial warehousing problem. *IIE Transactions* 34 (7), 601–612.

Kim, B.-I., Heragu, S. S., Graves, R. J., Onge, A. S., 2003. Realization of a short cycle time in warehouse replenishment and order picking. *International Journal of Production Research* 41 (2), 349–364.

Knapp, G. M., Wang, H. P., 1992. Modeling of automated storage/retrieval systems using Petri nets. *Journal of Manufacturing Systems* 11 (1), 20–29.

Kock, A. A. A., 2008. *Effective process times for aggregate modeling of manufacturing systems*. Ph.D. thesis, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven.

Kock, A. A. A., Etman, L. F. P., Rooda, J. E., 2008a. Effective process times for multi-server flowlines with finite buffers. *IIE Transactions* 40 (3), 177–186.

Kock, A. A. A., Wullems, F. J. J., Etman, L. F. P., Adan, I. J. B. F., Nijsse, F., Rooda, J. E., 2008b. Performance measurement and lumped parameter modeling of single server flow lines subject to blocking: An effective process time approach. *Computers and Industrial Engineering* 54 (4), 866–878.

Koh, S. G., Kwon, H. M., Kim, Y. J., 2005. An analysis of the end-of-aisle order picking system: multi-aisle served by a single order picker. *International Journal of Production Economics* 98 (2), 162–171.

Koo, P. H., 2009. The use of bucket brigades in zone order picking systems. *OR Spectrum* 31 (4), 759–774.

Lee, C., Huang, H. C., Liu, B., Xu, Z., 2006. Development of timed Colour Petri net simulation models for air cargo terminal operations. *Computers and Industrial Engineering* 51 (1), 102–110.

Lee, S. G., Souza, R. D., Ong, E. K., 1996. Simulation modelling of a narrow aisle automated storage and retrieval system (AS/RS) serviced by rail-guided vehicles. *Computers in Industry* 30 (3), 241–253.

Lefeber, E., Rooda, J. E., 2006. Controller design for switched linear systems with setups. *Physica A* 363 (1), 48–61.

Li, L., Bozer, Y. A., 2010. Retrieval strategies for multi-tier automated carousel conveyors with multiple robots. *Simulation* 86 (7), 395–404.

Lin, S. C., Wang, H. P. B., 1995. Modeling an automated storage and retrieval system using Petri nets. *International Journal of Production Research* 33 (1), 237–260.

Litvak, N., Vlasiou, M., 2010. A survey on performance analysis of warehouse carousel systems. *Statistica Neerlandica* 64 (4), 401–447.

Mahajan, S., Rao, B. V., Peters, B. A., 1998. A retrieval sequencing heuristic for miniload end-of-aisle automated storage/retrieval systems. *International Journal of Production Research* 36 (6), 1715–1731.

Matsumoto, M., Nishimura, T., 1998. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation* 8 (1), 3–30.

Medeiros, D. J., Enscore Jr., E. E., Smith, A., 1986. Performance analysis of miniload systems. In: Wilson, J., Henriksen, J., Roberts, S. (Eds.), *Proceedings of the 1986 Winter Simulation Conference*. December 8-10, Washington D.C., USA, pp. 606–612.

Meller, R. D., Mungwattana, A., 2005. AS/RS dwell-point strategy selection at high system utilization: a simulation study to investigate the magnitude of the benefit. *International Journal of Production Research* 43 (24), 5217–5227.

Minner, S., 2003. Multiple-supplier inventory models in supply chain management: a review. *International Journal of Production Economics* 81-82, 265–279.

Paese, M., 2010. Analysis of an automated item picking workstation. Final master project, SE 420619, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology.

Park, B. C., Foley, R. D., Frazelle, E. H., 2006. Performance of miniload systems with two-class storage. *European Journal of Operational Research* 170 (1), 144–155.

Park, B. C., Foley, R. D., White, J. A., Frazelle, E. H., 2003. Dual command travel times and miniload system throughput with turnover-based storage. *IIE Transactions* 35 (4), 343–355.

Park, B. C., Frazelle, E. H., White, J. A., 1999. Buffer sizing models for end-of-aisle order picking systems. *IIE Transactions* 31 (1), 31–38.

Perry, R. F., Hoover, S. V., Freeman, D. R., 1984. An optimum-seeking approach to the design of automated storage/retrieval systems. In: Sheppard, S., Pooch, U., Pegden, D. (Eds.), *Proceedings of the 1984 Winter Simulation Conference*. November 28-30, Dallas, USA, pp. 348–354.

Petersen, C. G., 2002. Considerations in order picking zone configuration. *International Journal of Operations and Production Management* 22 (7), 793–805.

Potrč, I., Lerher, T., Kramberger, J., Šraml, M., 2004. Simulation model of multi-shuttle automated storage and retrieval systems. *Journal of Materials Processing Technology* 157-158, 236–244.

Pulat, B. M., Pulat, P. S., 1989. Throughput analysis in an automated material handling system. *Simulation* 52 (5), 195–198.

Raghunath, S., Perry, R., Cullinane, T., 1986. Interactive simulation modeling of automated storage retrieval systems. In: Wilson, J., Henriksen, J., Roberts, S. (Eds.), *Proceedings of the 1986 Winter Simulation Conference*. December 8-10, Washington D.C., USA, pp. 613–620.

Rooda, J. E., Vervoort, J., 2007. *Analysis of Manufacturing Systems*. Eindhoven University of Technology, Eindhoven.
URL `http://se.wtb.tue.nl/sewiki/chi`

Roodbergen, K. J., Vis, I. F. A., 2009. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research* 194 (2), 343–362.

Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G. J., Mantel, R. J., Zijm, W. H. M., 2000. Warehouse design and control: framework and literature review. *European Journal of Operational Research* 122 (3), 515–533.

Sandell Jr., N., Varaiya, P., Athans, M., Safonov, M., 1978. Survey of decentralized control methods for large scale systems. *IEEE Transactions on Automatic Control* 23 (2), 108–128.

Schäfer, G., 2010. Automated order-picking system having an integrated sorting function and method for operating the system. US Patent no. US 2010/0036521 A1.

Silver, E. A., Pyke, D. F., Peterson, R., 1998. *Inventory Management and Production Planning and Scheduling*. John Wiley and Sons, New York.

Takakuwa, S., 1996. Efficient module-based modeling for a large-scale AS/RS-AGV system. In: Charnes, J. M., Morrice, D. J., Brunner, D. T., Swain, J. J. (Eds.), *Proceedings of the 1996 Winter Simulation Conference*. December 8-11, Coronado, USA, pp. 1141–1148.

Tompkins, J. A., White, J. A., Bozer, Y. A., Frazelle, E. H., Tanchoco, J. M. A., 2003. *Facilities Planning*, 2nd Edition. John Wiley and Sons, New Jersey.

Twist, D. C., 2005. The impact of radio frequency identification on supply chain facilities. *Journal of Facilities Management* 3 (3), 226–239.

van Beek, D. A., Man, K. L., Reniers, M. A., Rooda, J. E., Schiffelers, R. R. H., 2006. Syntax and consistent equation semantics of hybrid Chi. *Journal of Logic and Algebraic Programming* 68 (1-2), 129–210.

van Hoek, R. I., 2001. The rediscovery of postponement a literature review and directions for research. *Journal of Operations Management* 19 (2), 161–184.

van Vuuren, M., Winands, E. M. M., 2007. Iterative approximation of k-limited polling systems. *Queueing Systems* 55 (3), 161–178.

Veeger, C. P. L., 2010. *Aggregate modeling in semiconductor manufacturing using effective process times*. Ph.D. thesis, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology.

Veeger, C. P. L., Etman, L. F. P., Lefeber, A. A. J., Adan, I. J. B. F., van Herk, J., Rooda, J. E., 2011. Predicting cycle time distributions for integrated processing workstations: an aggregate modeling approach. *IEEE Transactions on Semiconductor Manufacturing* 24, 223–236.

Veeger, C. P. L., Etman, L. F. P., van Herk, J., Rooda, J. E., 2010. Generating CT-TH-PM surfaces using EPT-based aggregate modeling. *Journal of Simulation* 4, 242–254.

Vervoort, J., Rooda, J. E., 2007. *Learning Timed χ 1.0*. Eindhoven University of Technology, Eindhoven.
URL `http://se.wtb.tue.nl/sewiki/chi`

Welch, P., 1983. The statistical analysis of simulation results. In: *Computer Performance Modeling Handbook*. Academic Press.

Winands, E. M. M., Adan, I. J. B. F., van Houtum, G. J., 2006. Mean value analysis for polling systems. *Queueing Systems* 54 (1), 35–44.

Winkler, W., 1997. Kommissionierverfahren, kommissionieranlage zur durchführung des verfahrens sowie sortierpuffer hierfür. Offenlegungsschrift DE 197 12 839 A 1.

Wu, K., McGinnis, L. F., Zwart, B., 2008. Queueing models for single machine manufacturing systems with interruptions. In: Mason, S. J., Hill, R. R., Mönch, L., Rose, O., Jefferson, T., Fowler, J. W. (Eds.), *Proceedings of the 2008 Winter Simulation Conference*. December 7-10, Miami, USA, pp. 2083–2092.

Yu, M., de Koster, R., 2008. Performance approximation and design of pick-and-pass order picking systems. *IIE Transactions* 40, 1054–1069.

Yu, M., de Koster, R., 2009. The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research* 198 (2), 480–490.

Samenvatting

Het FALCON (Flexible Automated Logistic CONcept) project is gericht op de ontwikkeling van een nieuwe generatie magazijnen en distributiecentra die zo veel mogelijk automatisch werken. Als onderdeel van het FALCON project richt dit proefschrift zich op het ontwerpen en analyseren van (automatische) werkstations in magazijnen met een *end-of-aisle* orderverzamelsysteem. Methodes voor het ontwerpen, kwantificeren van de prestatie, en het besturen van dit soort systemen worden voorgesteld. In dit proefschrift worden vier onderwerpen behandeld.

Ten eerste is een modulaire architectuur voor een *end-of-aisle* orderverzamelsysteem met gescheiden werkstations gepresenteerd. Deze architectuur is opgebouwd uit verschillende deelsystemen en operationele lagen. Een hiërarchische gedecentraliseerde besturingsstructuur is hierop toegepast. Een casus van een distributiecentrum op industriële grootte is gepresenteerd om de toepassing van de voorgestelde architectuur voor prestatie analyse met een op proces-algebra gebaseerde simulatietaal χ (Chi) te demonstreren. Tevens wordt getoond hoe het ontwerp eenvoudige wijzigingen van de systeemconfiguraties, ontwerp parameters, en besturingsheuristieken toestaat.

Ten tweede is een methode ontwikkeld om de operationele prestatie van orderverzamelwerkstations te kwantificeren. Deze methode is gebaseerd op aggregaatmodellering op basis van het concept van de Effectieve Procestijd (Effective Process Time, EPT). Een werkstation is beschouwd waarin een menselijke picker aanwezig is om één order per keer te bedienen terwijl producten voor meerdere orders tegelijkertijd bij het werkstation arriveren. De EPT parameters zijn berekend uit de aankomst- en vertrektijden van producten met behulp van een *sample-path* vergelijking. Twee varianten van het model zijn ontwikkeld, namelijk voor werkstations met FCFS (*First-Come-First-Serve*) en zonder FCFS. Beide modellen zijn gevalideerd met gegevens van een bestaand werkstation. De resultaten tonen dat de voorgestelde methode van aggregeren een goede nauwkeurigheid geeft in het voorspellen van verdelingen van de doorlooptijden van de producten en orders.

Ten derde wordt in dit proefschrift het ontwerpen en besturen van een gescheiden, geautomatiseerd orderverzamelwerkstation bestudeerd. Dit werkstation kan meerdere orders tegelijkertijd verwerken. Producten voor meerdere orders arriveren bij het werkstation in een andere volgorde dan waarin ze besteld zijn, omdat ze bijvoorbeeld van verspreide locaties in het opslaggebied komen. Het ont-

werpprobleem betreft het structureren van de product- en order- bufferbanen en de ontwikkeling van een inrichting dat de verstoringen van de volgorde van de productaankomst ondervangt. Het besturingsprobleem betreft de formulering van de pickvolgorde bij het werkstation. Een efficiënte besturingsstrategie is ontwikkeld en de prestatie hiervan is vergeleken met een aantal andere pick-strategieën zoals sequentieel, dichtstbijzijnde-buren en dynamisch programmeren. Ook zijn de doorzet en de verdeling van de wachtrijlengte geëvalueerd met verschillende aannames.

Ten vierde behandelt dit proefschrift de resultaten om een *end-of-aisle* orderverzamelsysteem met gescheiden werkstations te ontwerpen. De verscheidene aspecten die de prestaties van deze systemen beïnvloeden worden toegelicht.

Rangkuman

Proyek FALCON (*Flexible Automated Logistic CONcept*) bertujuan untuk mengembangkan gudang dan pusat distribusi dengan tingkat otomasi maksimal. Sebagai bagian dari proyek FALCON, disertasi ini membahas desain dan analisis stasiun kerja (otomatis) yang digunakan di gudang dengan Sistem Pemenuhan Order (SPO) ujung lorong (*end-of-aisle*). Sejumlah metode diajukan untuk merancang, mengukur kinerja, dan mengendalikan sistem tersebut. Empat topik utama dibahas dalam disertasi ini.

Pertama, sebuah arsitektur modular diajukan untuk SPO ujung lorong yang memiliki stasiun kerja berlokasi terpisah. Arsitektur ini dibuat terstruktur sehingga membentuk beberapa area dan lapisan operasional. Struktur pengendalian terdesentralisasi berbentuk hirarki diterapkan dalam arsitektur tersebut. Sebuah pusat distribusi berskala industri diangkat sebagai contoh kasus guna menunjukkan bagaimana arsitektur yang diajukan dapat diterapkan untuk analisis kinerja menggunakan χ (Chi), sebuah bahasa simulasi berbasis aljabar proses. Arsitektur yang diajukan juga dapat dengan mudah menerima modifikasi terhadap konfigurasi sistem, parameter desain, dan heuristik pengendalian.

Kedua, sebuah metode dirancang untuk mengukur kinerja operasional dari stasiun kerja pemenuhan order. Metode ini didasari oleh pemodelan agregat stasiun kerja dengan menggunakan konsep Waktu Proses Efektif (WPE). Sebuah stasiun kerja manual dibahas, dimana seorang pekerja memenuhi order pelanggan satu per satu, sedangkan produk-produk yang dibutuhkan oleh beberapa order datang bersamaan di stasiun kerja tersebut. Parameter untuk WPE ditentukan dengan menggunakan persamaan lintasan sampel (*sample-path*) berdasarkan waktu kedatangan dan kepergian produk. Dua jenis model telah diciptakan yaitu model untuk stasiun kerja dengan metode proses berurutan (*First-Come-First-Serve*) dan model untuk stasiun kerja tanpa metode proses berurutan. Kedua model telah divalidasi menggunakan data yang diperoleh dari stasiun kerja asli yang sedang beroperasi. Hasil penelitian menunjukkan bahwa metode pemodelan agregat yang diajukan mampu memprediksi distribusi waktu alir produk dan order dengan tingkat akurasi yang memuaskan.

Ketiga, disertasi ini membahas desain dan kontrol atas sebuah stasiun kerja otomatis berlokasi terpisah yang mampu memproses beberapa order pelanggan dalam waktu yang bersamaan. Produk-produk untuk beberapa order pelanggan pada umumnya tiba dengan urutan yang keliru di stasiun kerja. Hal ini dikarenakan produk-

produk tersebut diambil dari berbagai lokasi yang tersebar di area penyimpanan. Permasalahan desain meliputi penstrukturan lini buffer produk/order serta perancangan mekanisme yang mampu mengatasi masalah kedatangan produk-produk dengan urutan yang keliru. Permasalahan kontrol meliputi penentuan urutan pemrosesan produk di stasiun kerja, mengingat *throughput* (tingkat pemenuhan order) akan berkurang jika urutan pemrosesan yang buruk digunakan. Untuk itu, dirancanglah sebuah kebijakan kontrol yang efektif. Kinerja kebijakan tersebut dibandingkan dengan beberapa kebijakan kontrol lainnya yaitu pemrosesan berurutan, metode tetangga terdekat, dan pemrograman dinamis. *Throughput* dan distribusi panjang antrian yang dihasilkan oleh setiap kebijakan kemudian dievaluasi. Berbagai pemahaman yang diperoleh kemudian dipaparkan agar dapat digunakan sebagai pertimbangan dalam proses desain sistem.

Terakhir, disertasi ini menggunakan berbagai temuan yang diperoleh dari metode-metode yang diajukan untuk menyusun prinsip desain yang menyeluruh atas SPO ujung lorong dengan stasiun kerja berlokasi terpisah. Berbagai macam hal yang mempengaruhi kinerja sistem ini kemudian dibahas satu per satu. Selain itu, kontribusi dari setiap metode pada hal-hal tersebut juga dipaparkan.

Curriculum vitae

Ricky Andriansyah was born on March 22nd, 1983 in Bengkalis, a small island east of Sumatra, Indonesia. He completed the Mathematics and Natural Science program in 2000 at 68 Senior High School in Jakarta, Indonesia. Afterwards, he studied Industrial Engineering at Universitas Indonesia in Depok, Indonesia. He received the Toyota Top Student Grant in his final year and graduated cum laude in 2004.

In 2005 he received the StuNed scholarship from the Dutch government via Nuffic Neso Indonesia to follow the master program Operations Management & Logistics at Eindhoven University of Technology (TU/e) in Eindhoven, the Netherlands. He graduated cum laude in 2007 after writing a master thesis on joint optimization of buffers and servers in general queueing networks.

From 2007 he started a PhD project at the Systems Engineering Group of TU/e. His research concerns order-picking workstations for automated warehouses, which is part of the FALCON (Flexible Automated Logistics CONcept) project. The results of his research have been written in this dissertation and in a number of journal papers and conference proceedings. One of his publications (Chapter 3 of this dissertation) received the best paper award at the 1st International Conference on Advances in System Simulation (SIMUL), 2009. Another publication (Chapter 5 of this dissertation) was nominated for the best student paper award at the 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH), 2011.

Parallel to his research, Ricky was also involved in educational activities such as becoming an instructor for the course Analysis of Manufacturing Systems and coaching bachelor and master students during their internship or graduation projects.