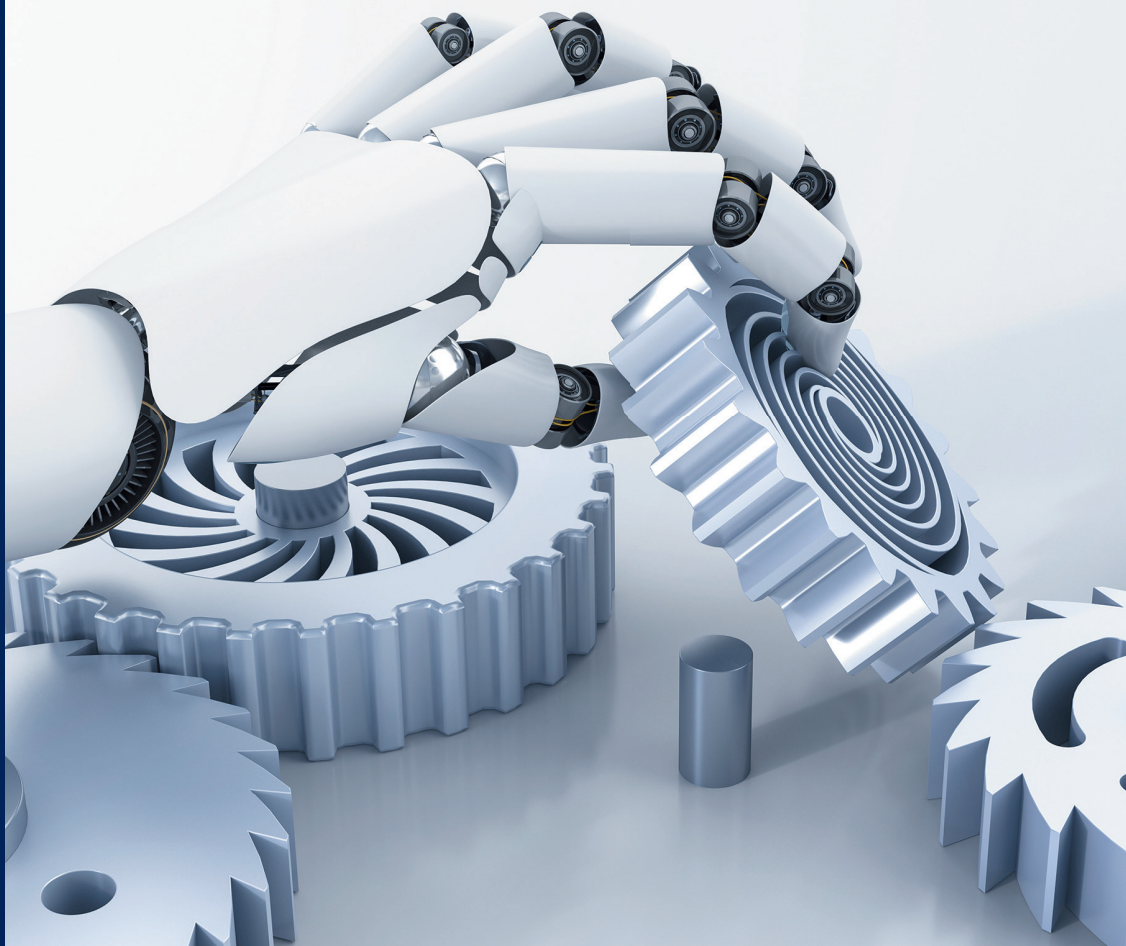
TIM LAMBALLAIS TESSENSOHN

Optimizing the Performance of Robotic Mobile Fulfillment Systems



Optimizing the Performance of Robotic Mobile Fulfillment Systems

Optimizing the Performance of Robotic Mobile Fulfillment Systems

Prestaties optimalizeren van mobiele robot-orderverzamelingssystemen

Thesis

to obtain the degree of Doctor from

Erasmus University Rotterdam

by command of the

rector magnificus

Prof.dr. R.C.M.E. Engels

and in accordance with the decision of the Doctorate Board.

The public defence shall be held on

Thursday 16 May 2019 at 9:30 hours

by

Tim Lamballais Tessensohn

born in Capelle aan den IJssel, The Netherlands

Erasmus University Rotterdam

Doctoral Committee

Promoters:	Prof.Dr. M.B.M. de Koster
	Prof.Dr.Ir. R. Dekker
	Dr. D. Roy
Other members:	Prof.Dr. R.A. Zuidwijk
	Prof.Dr. L. Suhl
	Prof.Dr.Ir. I.J.B.F. Adan

Erasmus Research Institute of Management – ERIM

The joint research institute of the Rotterdam School of Management (RSM)
and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam
Internet: `http://www.erim.eur.nl`

ERIM Electronic Series Portal: `http://repub.eur.nl/pub/`

ERIM PhD Series in Research in Management, 411

ERIM reference number: EPS-2019-411-LIS

ISBN 978-90-5892-538-1

Design: PanArt, `www.panart.nl`

This publication (cover and interior) is printed by Tuijtel on recycled paper, BalanceSilk®. The ink used is produced from renewable resources and alcohol free fountain solution. Certifications for the paper and the printing production process: Recycle, EU Ecolabel, FSC®, ISO14001. More info: `www.tuijtel.com`

Acknowledgments

Decentralized systems have something fascinating and exciting. The components make decisions for themselves, and yet, the result is one of harmonious movements, like ants carrying out tasks without a central decision maker. So when René showed me the Kiva system, with all those robots moving around semi-autonomously, I was hooked. Now at the end of the journey, I can look back on four years where I was given great freedom to learn new skills and to explore the scientific world. I would like to thank my supervisors Debjit Roy and René de Koster, who showed great enthusiasm during our weekly meetings, and who critically assessed my ideas, and inspired me with theirs. Their infinite patience made it possible for me to finish my PhD, not only during the PhD itself but also after its official end. René, thank you for quickly grasping what I meant to say even when I communicated it vaguely and chaotically. Debjit, thank you for having faith in me and for showing me around in Ahmedabad, it was a great pleasure to visit. I would also like to thank Roy Thurik, Adriana Gabor and Remy Spliet for encouraging me to start a PhD in the first place.

I had the good fortune to meet Marius Merschformann during a conference and the great honor to write two papers together with him. Marius, I had an absolutely wonderful time staying in Paderborn and hosting you in Rotterdam. It was a great joy to collaborate with you, and to have many and very long discussions about our little robots, thank you for your support and for being an awesome friend. My research visit in Paderborn was very enjoyable, thanks to the many wonderful people who made for great company, nice talks and who quickly integrated me into the department. I would like to express

my deep gratitude to Leena and Uwe Suhl. Thank you for hosting me, for inviting me to social events, for making me feel welcome and for showing me great hospitality and generosity. I would also like to thank everyone in the DSOR group, I will never forget the chocolate wednesdays, from popcorn chocolate to chili chocolate with gold pieces and everything in between, it was just awesome. Besides visiting Paderborn, I also had the great pleasure of staying at the IIM Ahmedabad in India. Thank you very much for hosting me Debjit, it was a wonderful experience. I would also like to thank Prabu for helping me out when I arrived in Ahmedabad and Govind for guiding me and René around the city. Leaving RSM, I will miss many people who have come to be great friends. Thanks Joris for our daily coffee and our many conversations, and for showing me that iron throats do exists. I will forever be impressed at your ability to drink coffee that is far too hot for me to even begin sipping. Elisa, thanks for being a great friend and support, and for showing me how awesome Lama's and Alpaca's are. Jelmer, thank you for the many suggestions you gave me and for lending me a critical ear when I had another crazy idea. Hereby, especially for you Jelmer: "queuing". Carmen, thank you for the many nice conversations and for putting flowers in all the offices. As everyone knows, over the years office mates develop a special bond, and after three and a half years together Masoud has learned to read my mind and I his. Masoud I hope we will continue to do so in the future. I would also like to thank my other office mates, Dong Li, Ruiqi Hou, and Alberto Giudici, I very much enjoyed sharing an office with you and I will miss our conversations. Arpan, thanks for all the amazing stories about Nepal and for helping me out when I went to visit, I hope we will be friend for many years to come. The department of Technology and Operations Management has a very social and friendly atmosphere with many events and outings and I had many nice conversations with colleagues over coffee and in the hallways, over cake, before and after lectures, when going out after work hours, etc. I would like to thank all my colleagues at TOM, current and former, for giving me a great time in the department, thanks for seeking me out and shaking my hand when you heard that I left, I will miss you all very

much. My family always supported me and stood by my side and for that I am deeply grateful. Pa, Ma, Sannie, you are simply awesome. Last but certainly not least: Smita, thank you for filling my world with joy, laughter, and love.

Tim Lamballais Tessensohn
Rotterdam, 2019

Contents

1 Introduction

The rise of e-commerce has changed the role of the warehouse in the supply chain. Traditionally, the role of the warehouse has been to store inventory to fulfill orders from other businesses. These business to business (B2B) orders at a warehouse are typically for multiple products with relatively large quantities per product, because the orders are meant to refill inventories at another point in the supply chain. Business customers, for example retailers and brick-and-mortar shops, typically place their orders before their inventories have depleted and order in bulk. They maintain a safety stock that provides some tolerance for replenishment delays and errors.

With e-commerce, products can be sent directly from a warehouse to an individual consumer, circumventing other points in the supply chain. Consumers purchase online in great numbers. According to Walker Sands Communications (2016), an estimated 31 percent of USA consumers shop online weekly and nearly 75 percent did so at least monthly during 2016. A majority prefers to buy books, consumer electronics and office supplies online over going to a brick-and-mortar shop, and about a quarter of consumers shop online for luxury items. In Europe, e-commerce sales have seen double digit growth year over year, with a majority of European internet users shopping online (E-commerce Europe, 2016).

In other words, the warehouse has become a so-called "customer order point" (COP) (Olender, 2012). For individual consumers, ordering online is an alternative to buying a product in an ordinary brick-and-mortar store. This has four implications for e-commerce operations. First, customers want to receive their order as fast as possible, because they pay in advance. At a

store they also would have the product immediately after their purchase. Second, so-called value added services (VAS) may be needed. A customer may want the product wrapped as a gift, or marketing may want to include a brochure (Olender, 2012). Third, customers have a low tolerance for errors in the e-commerce order they receive, because delivery errors do not occur when purchasing at an ordinary store. They also dislike the hassle of returning a product due to a mistake at the supplier's end. Fourth, protected by law, customers are allowed to return products to the warehouse. This leads to high return rates for certain types of goods. Customers cannot try out goods such as clothes, shoes or other apparel when ordering online, whereas they can when purchasing at an ordinary store. They solve this by simply ordering all the items they would like to try out and then returning the ones they do not wish to actually purchase. The return of goods is a legal requirement in the EU and the USA, which forces e-commerce companies to accomodate the high number of returns in their warehouse operations.

These four implications have a direct impact on warehouse operations. Orders need to be processed faster, as customer impatience exerts pressure to reduce lead times, while the error rate must be kept as close as possible to zero. In addition, certain value added services that were not part of operations before may need to be included, as well as processes to handle the return of goods. The pressures on warehouse operations drive a shift towards more automation. Automated material handling systems tend to have far lower error rates than manual picking and increase the accuracy of inventory management and tracking. Automated systems may also automatically handle some of the value added services, such as packaging or adding brochures. Another driver towards automation are labor shortages (Gue et al. (2014), Morris (2015), MHI & Deloitte (2014), MHI & Deloitte (2015), MHI & Deloitte (2016)), which is exacerbated due to high turnover of personnel (Min, 2007). For automated parts-to-picker systems, pick rates tend to be quite high, because pickers are positioned at stations while the products come to them via conveyor devices. High pick rates alleviate some of the impact of labor shortages and allow reductions in customer waiting time.

E-commerce combined with a shift towards more automation has led to a variety of responses, such as changes in warehouse operations, changes in distribution networks and more flexible automation. Changes in operations include the co-existence of different material handling systems in a warehouse dedicated to e-commerce operations. For example, one area may contain an automated system for products that fit within a tote or box, whereas larger products may be stored in a pallet area. In addition, companies with both an online channel and brick-and-mortar stores, i.e., an omni-channel company, often separate the warehouse operations for these channels at different locations. These warehouses may use different systems. Changes in distribution networks may result from a desire to reduce customer service times. An e-commerce company can complement a few large, centrally located warehouses with additional smaller warehouses located close to urban areas. Customer waiting times can then be reduced by serving customers in these urban areas from the smaller, nearby warehouses. Alternatively e-commerce companies may reduce the number of warehouse by outsourcing part or all warehouse operations to a third party logistic provider (3PL). 3PL companies store the inventory of multiple companies within their warehouse, taking care of all warehouse operations for the customers of these companies (Bond, 2016). As warehouse operations is their core business, 3PLs be better positioned to invest in expensive automated systems than their clients.
E-commerce not only drives a shift towards more automation, but more specifically it drives a shift towards automated systems with higher degrees of flexibility. The type of flexibility required of the automated material handling systems depends on the type of e-commerce company. Flexible automation is particularly important for two types of e-commerce companies. The first type is e-commerce companies with a large assortment of products, each with low turnovers and facing strong demand fluctuations. Manual picker-to-parts systems tend to lead to high average walking distances in these companies, because large assortments require large warehouses. In addition, low turnovers mean fewer picks per storage location, and strong demand fluctuations mean that sorting inventory to reduce travel distances is

difficult, which increases the average distance between picks. An automated parts-to-picker system that eliminates picker walking time may therefore improve performance, but only if that system has the flexibility necessary for handling strong demand fluctuations. The second type of e-commerce company that benefits from flexible automation are e-commerce companies that experience double or triple digit annual growth. New forms of flexible automation have recently allowed this type of companies to also benefit from automation, whereas in the past they would have been deemed to small or their ordering process to unstable for automation. This type of company requires automated systems that can scale rapidly in response to demand growth. Since this type of rapid growth is difficult to forecast, the up- or downscaling of the system should be possible at any time and in small as well as large increments. Moreover, such companies may need to move from one location to another or add a second location to their operations to accommodate their growth. Therefore, the automated system should be deployable in existing buildings, and if possible be movable to another location.
Table 1.1 shows several automated material handling systems and their ability to exhibit different forms of flexibility. One of these systems is the Robotic Mobile Fulfillment System (RMFS): a robotized parts-to-picker automated sorting and picking material handling system conceived with the aim of efficiently fulfilling e-commerce orders. This system is the prime subject of study in this thesis. As Table 1.1 describes, the RMFS is capable of displaying different forms of flexibility.

1.1 Robotic Mobile Fulfillment Systems

The Robotic Mobile Fulfillment System was originally conceived by Reinhardt Jünemann (Jünemann, 1989). The main component is a small, mobile robot that can move underneath a movable shelf, called a pod, lift the pod, and transport it within the warehouse. The first to implement an operational version was Kiva Systems. Kiva Systems created what they called a “mobile

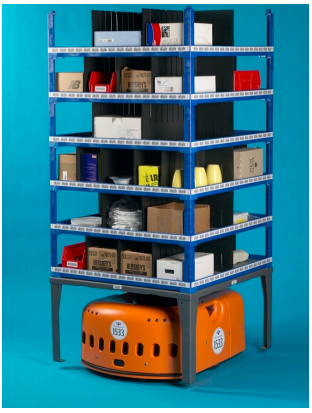
Table 1.1: Automated warehousing systems and different forms of flexibility

	Scaling inventory capacity	Scaling throughput capacity
Miniload Crane system	−− impossible in short term, difficult in the long term	−− impossible in short term, difficult in the long term
Miniload AVSR	− to some extent possible by adding or removing new aisles and elevators	−/+ possible by adding or removing AGVs, but elevators may be bottleneck
Autostore	−/+ possible by expanding or shrinking storage structure	+ relatively easily done by adding or removing robots
RMFS	++ easily done by adding or removing pods	+ relatively easily done by adding or removing robots
	Suitability for deployment in existing buildings	**Ability to move entire system to new location**
Miniload Crane system	− only possible in tall buildings	− possible, but expensive, and requires long time
Miniload AVSR	−/+ possible if no obstacles like roof supporting columns block construction	− possible, but expensive, and requires long time
Autostore	−/+ possible if no obstacles like roof supporting columns block construction	−/+ possible, but expensive, and requires medium time
RMFS	+ possible to build around obstacles	++ straightforward, cheap and requires little time
	Ability to store products of different shapes and sizes	**Inventory sorting in response to demand fluctuations**
Miniload Crane system	−/+ product must fit in a tote	−/+ possible during idle time
Miniload AVSR	−/+ product must fit in a tote	−/+ possible during idle time
Autostore	−/+ product must fit in a tote	+ possible during operations and idle time
RMFS	++ pods can be customized for wide range of shapes and sizes	++ occurs automatically during operations

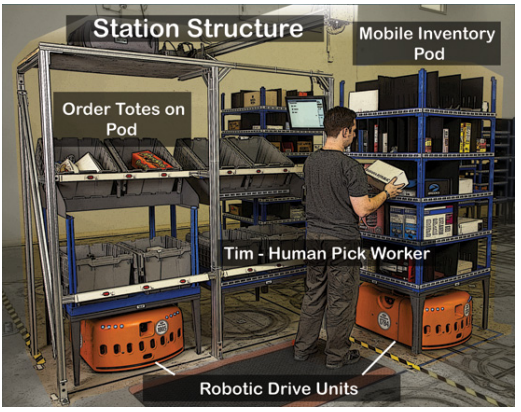
fulfillment system". Amazon bought Kiva Systems in 2012, renamed it "Amazon Robotics" and deployed RMFSs in ten of its warehouses in November 2014 (Business Wire, 2015).

The RMFS is a parts-to-picker system and works as follows. An order arrives at the system. Each pick station has capacity for a number of unfinished orders. If one or multiple pick stations have capacity left, the order is assigned to a pick station, otherwise it is added to a pool of unassigned orders. The products required by the order are allocated to one or more storage pods. These are then brought to the pick station of the order by the robots. The picker at a pick station picks products from a pod and thus fulfills order lines from the unfinished orders at the pick station. After the picker is done with a pod, the pod is transported back to the storage area and the next pod in the queue at the pick station is presented to the picker. Instead of returning to the storage area, it is also possible that the pod is brought to another pick station, where it enters the queue of that pick station. The robots handle the transportation of the pods between storage area and the pick stations. When an idle robot is instructed to bring a pod to a pick station, it drives to that pod, lifts it, and transports it to the pick station, where the robot and pod enter the queue of that pick station. The transportation of pods to replenishment stations for replenishing the inventory on the pods is identical to the transport of pods to the pick stations. The design of the pods is such that goods rarely fall off during transit (Wulfraat, 2012). The robots are equipped with sensors to avoid collision and use barcode stickers on the ground to navigate. From time to time, in between tasks, a robot travels to a recharge station for a short recharge. At a workstation, a pod queues until a worker either picks products from it, or replenishes products on the pod. The picker is usually guided by a pick-to-light system, showing which products to pick in what quantities and where to drop off (in which customer bin) the picked products. A pick station provides support for picking 6 to 12 orders simultaneously and can also be equipped for additional activities such as packaging, special labeling and other value adding services. Usually some workstations are occupied with picking while the others are

occupied with replenishment. If necessary, a station can switch from picking to replenishment activities and vice versa. Goods delivered to the warehouse from a supplier can in principle be inserted directly into the system, but are usually first put into a reserve area elsewhere in the warehouse. Figure 1.1a shows a robot carrying a pod (Wurman & Enright, 2011) and Figure 1.1b shows a workstation in detail (Kiva Systems, 2010).



(a) Robot carrying a pod (Wurman & Enright, 2011)



(b) Workstation in detail (Kiva Systems, 2010)

Automated parts-to-picker systems are nothing new in the material handling world, but the system has a few key advantages in the area of e-commerce order fulfillment. The key advantages that set the system apart are flexibility, modularity and scalability.

Pods can be modified to accommodate many different types and shapes of products, from a pod with shelves for books to a pod that can store hanging garments. A pod typically contains multiple SKUs simultaneously. For instance, a pod equipped with shelves may contain a wide selection of different books. If necessary the robots can also carry pallets and cases rather than pods.

The system is modular, since it consists of pods, robots and workstations, which are all identical, movable elements. For each of these resources it is

relatively straightforward to increase or decrease their number, which means that the system can be scaled up or down in a short period of time. The modularity of the system also means that the system is robust against failure: if one robot fails, operations can simply continue, as the robot only blocks a small amount of space that the other robots can bypass. In addition, the layout of the warehouse is flexible and can be altered in a short period of time. The barcode stickers used for navigation, as well as the workstations, can be relocated easily and the robots can simply relocate the pods. This also means that the system can be integrated into an existing building relatively quickly, and that the entire system can be moved to another building elsewhere. This process typically takes several months but, interestingly, one company apparently managed to move the entire system to a new location in one weekend, see Mountz (2012) and Wulfraat (2012).

These characteristics of the system make it attractive for e-commerce companies, as they tend to experience quite volatile and unpredictable demand growth. Deployment is relatively quick, with for example Acumen deploying the system in a mere 14 weeks (Wulfraat, 2012). The ease of integrating it in an existing structure means that companies can create a separate section for e-commerce operations next to the other warehouse operations within an existing warehouse. The ease of moving the entire system means that a company can relocate to a bigger site as it grows, or can move closer to the customer base. For peak periods, such as Christmas, the system can be scaled up by temporarily hiring more workers and adding more pods, robots and workstations. The main disadvantage is the cost: a typical system with around 50 to a 100 robots costs around two to four million dollars, most of which is spent on the robots (Wulfraat, 2012).

The first implementation of the RMFS was at Staples. It reported that picking rates are twice as high as with their previous system. According to Wulfraat (2012), picker productivity tends to be two to three times higher compared to pick-to-conveyor systems, and about 5 to 6 times higher compared to manual picking. Figure 1.2b shows an Amazon Robotics robot and Figure 1.2a the original Kiva robot. Amazon Robotics does not sell its

system to third parties and hence competitors have emerged. Presently these competitors are Swisslog in Europe with their CarryPick system, see Figure 1.2c, GreyOrange in India with their Butler robot system, see Figure 1.2d, Mobile Industrial Robots in Denmark with the MiR100, see Figure 1.2e and Scallog in France with their mobile robot, see Figure 1.2f.

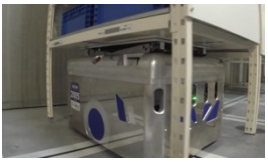
(a) Kiva Systems, source: Singularity Hub (2016)

(b) Amazon Robotics, source: Business Insider (2016)

(c) CarryPick, source: Swisslog (2016)

(d) GreyOrange, source: Grey Orange (2016)

(e) MiR, source: MiR (2016)



(f) Scallog, source: Scallog (2016)

Figure 1.2: RMFS robots

1.2 Research Opportunities

The Robotic Mobile Fulfillment System provides new research opportunities. Algorithms solve decision problems in real time, either via a central computer or via decentralized control by the involved entities themselves. For example, the robots themselves can decide on path planning, and pick stations can decide on order assignment. In the latter problem, complications specific for

RMFSs are that a pod can fulfill multiple order lines across different orders at a pick station and can visit multiple pick stations sequentially. As Wurman & Enright (2011) and Wurman et al. (2008) remark, many unsolved decision problems remain to be researched and optimized. These problems involve selecting the inventory pods to be brought to the workstations; selecting the storage locations to return pods to; assigning orders to pick stations; allocating tasks to robots; and planning paths or routes for the robots. Path planning has been examined in numerous other applications, but takes an interesting twist here: unloaded robots, i.e., robots not carrying pods, can move underneath the stationary pods and take different routes than loaded robots.

The decision problems are interconnected. For example, when a robot returns a pod to storage, at which storage location should the pod be stored? On the one hand, this depends on whether the pod will be needed in the near future by another pick station and on the popularity of the products on the pod. If the pod is needed soon or if it contains popular products, it may be better to store it close to the pick stations, to reduce travel time in the future. On the other hand, the travel time of the robot carrying the pod may be reduced if the pod can simply be stored at a location along the route to the next task of the robot, instead of the robot having to make a detour. However, which task should the robot do next? One option is to choose the robot's next task before the storage location of the pod is determined, and then a storage location can be chosen in the vicinity of the robot's route to the next task. Another option would be to first decide where to store the pod and then allocate a next task to the robot. The problem in this example is further compounded by the sheer size of the solution space and the short computation time available, as these operational decisions must be made in real-time.

The operational problems in a Robotic Mobile Fulfillment System give rise to five interesting concepts, namely (1) "pile-on", (2) "well-sortedness", (3) priority zoning, (4) dynamic resource allocation, and (5) the centralization-decentralization trade-off. (1) Pile-on is the number of units that can be

picked from a single pod that visits the pick station. The higher the average pile-on is, the fewer pods have to be brought to the pick stations and the fewer robots are needed for the same order throughput rate. Increasing pile-on requires improving the assignment of orders to pick stations, since assigning orders together to a pick station that need products from the same pod(s) improves the number of product picks per pod. Selecting an inventory pod that covers as many order lines as possible also increases pile-on. Another method for increasing pile-on is careful planning in the replenishment process. During replenishment, units of products are placed on a pod. The replenishment process thus determines which products are stored on each pod in which quantities, i.e., the replenishment process determines the composition of a pod. The wider the range of popular products on a given pod, the more order lines that pod is likely to be able to fulfill at a pick station. The main motive for wanting to increase pile-on is to reduce the number of robots needed in the system. An alternative method to reduce the required number of robots would be to reduce travel times. However, beyond a certain point, gains in travel time are offset by additional queueing time for the robots at the workstations. (2) Well-sortedness measures the distance between the pods and the pick stations and weighs these distances with the "pod popularity". The pod popularity is a weighed total across all products on a pod of the frequency with which product is ordered multiplied by the number of units of that product on the pod. The continual repositioning of pods during operations can be used to sort products. Executed correctly, the popular products will be close to the pick stations at all times. Well-sortedness can be used to understand to what degree continual pod repositioning supports sorting inventory according to popularity. It may be difficult to measure, as pods may contain many different SKUs and popularity may fluctuate due to volatile demand during operations. (3) Priority zoning is the allocation of storage locations into priority zones. A priority zone contains pods that are needed at the pick stations in the near future. By creating priority zones near the pick stations, travel times between the pick stations and the storage area are reduced. Furthermore, the variance of travel times between pick

stations and storage areas is reduced, as travel to more distant parts of the storage area is no longer included. Reduced travel time variance facilitates more careful planning and scheduling with regard to robot queueing at the pick station, which may result in less queueing and less idle time at the pick station. However, as pods also need to be transported between the priority zones and the other parts of the storage area and the replenishment stations, other types of travel time may increase. Priority zones can be dynamically changed in real time by changing the assignment of storage locations to the priority zones. (4) The RMFS allows quick reallocation of resources between picking and replenishment activities. Robots can be reallocated after every completed task, and workstations can be altered from pick to replenishment stations and vice versa in a short amount of time. Dynamic reallocation should enable a faster response to peak demand and peak supply. (5) In principle, all decisions can be taken by a central computer. In practice it may be better to let some decision problems be handled in a decentralized manner. For example, the robots are more agile and respond quicker to other robots when path planning is decentralized, see also Frazzoli et al. (2002) and Wang & Botea (2008). Decentralization decreases the time needed to recover from unexpected events and decreases the likelihood that such events derail the system. Distributing the decision process can be necessary in some cases. For operational decision problems solved in real-time, the available solution time is rather short, while the solution space is rather large.

The operational problems take place in real time and need to be solved as fast as possible, whereas the dynamic reallocation of resources such as robots and workstations may take place just a few times during a day or a week. The decisions of dynamic reallocation of resources should therefore be taken at the tactical level. Other decisions that should be taken at the tactical level include the setting of the replenishment level per product and determining the number of pods per product. The latter problem is also of great importance in other e-commerce settings and has so far not been answered. At the strategic level, warehouse design problems such as workstation placement and storage area dimensioning should be studied, as the solutions to these

problems have a large impact on subsequent activities and cannot easily be changed. The placement of recharge stations is an interesting topic but the recharging process may be quite different between implementations of RMFSs. Zou et al. (2016) compare different possible recharging methods. An overview of the decision problems, the parameters required for solving the decision problems, and the decision variables that are part of the solutions to the decision problems, are shown in Figure 1.3.

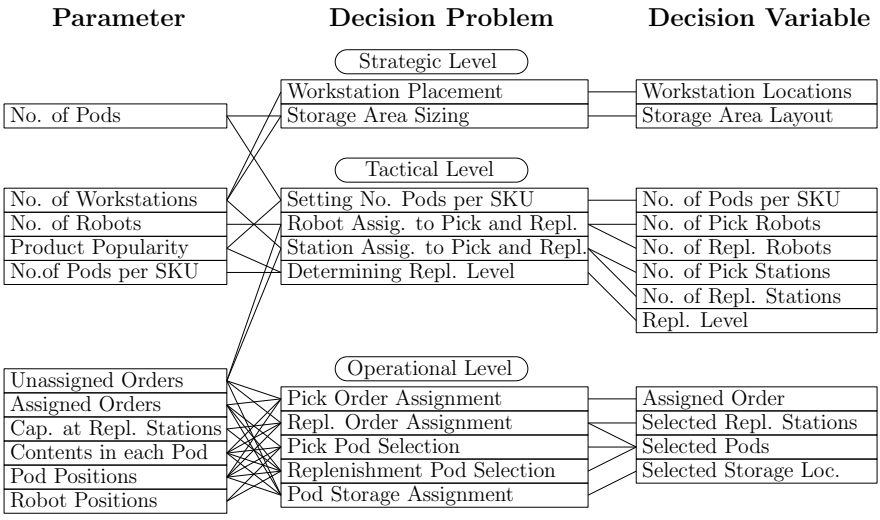


Figure 1.3: Overview of decision problems with parameters and decision variables

1.3 Contribution and thesis outline

This thesis focuses on several of the problems outlined in Figure 1.3 at the strategic, tactical and operational level. Table 1.2 shows the decision problems that are addressed in each chapter and the objective in solving the decision problems. Chapter 2 examines the decision problems of "Workstation

Placement" and "Storage Area Dimensioning". These decisions have to be taken at the strategic level and the objective is to find the placement of workstations and dimensioning of the storage area that maximize the order throughput capacity. Chapter 3 studies problems at the tactical level, namely "Setting the Number of Pods per SKU", "Station Assignment to Pick and Replenishment" and "Determining the Replenishment Level per Product". These three problems are related and the objective is to minimize the order throughput time. Chapter 4 researches the allocation of both robots and workstations across both pick and replenishment activities when demand fluctuates between periods of peak demand and low demand. The aim is to find a policy that reallocates robots and workstations such that order throughput time is minimized. Chapter 5 focuses on several operational decision problems: "Pick Order Assignment", "Replenishment Order Assignment", "Pick Pod Selection", "Replenishment Pod Selection", and "Pod Storage Assignment". These operational problems influence each other to some extent. The performance of the solutions is measured in the average pile-on and order throughput time achieved, where average pile-on should be maximized and order throughput time minimized.

Table 1.2: Decision problems per chapter

Chapter	Decision Problem	Objective
Ch. 2	Workstation placement Storage area sizing	Max. order throughput
Ch. 3	Number of pods per SKU decision Station assig. to pick and repl. Replenishment level decision	Min. order throughput time
Ch. 4	Robot assig. to pick and repl. Station assig. to pick and repl.	Min. costs
Ch. 5	Pick order assignment Replenishment order assignment Pick pod selection Replenishment pod selection Pod storage assignment	Max. unit throughput

Multiple methodologies are used to research these problems. In Chapters 2 and 3, Semi-Open Queueing Networks (SOQN) and Closed Queueing Networks (CQN) are used to examine the design of the warehouse layout and to analyze the replenishment process. Dynamic reallocation of resources across picking and replenishment activities is modeled with a combination of queueing networks and Markov Decision Processes (MDP) models in Chapter 4. Chapter 5 addresses problems at the operational level. Several decision mechanisms are defined per decision problem and their performance is evaluated in realistic simulations for a wide range of scenarios.

Chapter 2: Warehouse Design

This chapter models Robotic Mobile Fulfillment Systems and focuses on warehouse design aspects, such as workstation placement and storage area sizing. These two aspects are analyzed in order to maximize the performance of the pick process. As an RMFS tends to be a multi-million dollar investment, it is important for companies to have an estimation of the order throughput capacity of the system before they decide to invest in the system. Potential customers also need to know the necessary number of workers and robots to realize the required order throughput, and whether the system can handle peak demand situations. This chapter develops queueing network models for both single-line and multi-line orders, to analytically estimate maximum order throughput, average order cycle time, and robot utilization. The main benefit of this approach over simulation is a very short computation time, which allows the user to quickly explore many different layout configurations and robot zoning strategies and thus in a sense optimize the warehouse layout for the given throughput capacity requirements. Semi-Open Queueing Networks (SOQNs) are used to estimate robot utilization, workstation utilization, and order cycle time for each workstation. This approach is suitable for stationary demand. Closed Queueing Networks (CQNs) are developed to show the maximum order throughput, which is useful for examining which peak demand situations the warehouse would be able to handle and which not. By increasing the number of workstations it is possible to investigate whether

the system can handle a temporary peak demand situation such as Christmas by hiring additional temporary staff. The warehouse layouts examined have storage areas of different sizes and length-to-width ratios and have different configurations of workstation placements around the storage area. The models contribute to the literature by introducing the first analytical models built for analyzing RMFSs. More specifically, the models contribute by incorporating accurate driving behavior of robots, multi-line orders, robot zoning strategies, and robot acceleration and deceleration. The results of the experiments lead to the following insights: (1) the analytical models accurately estimate the average order cycle time, and robot utilization, (2) the maximum order throughput appears to be quite insensitive to the length-to-width ratio of the storage area, and (3) the maximum order throughput is affected by the location of the workstations around the storage area.

Chapter 3: Inventory Allocation

The replenishment process supports the picking process and in an RMFS replenishment and picking occur concurrently. The replenishment process should be given sufficient resources to prevent stock-outs of products and delays for the picking process. Replenishment can have a large impact on the picking process in other ways as well, since it determines which products are placed on which pods in what quantities. One interesting question in this regard is: across how many pods should the inventory of a product be spread? Spreading the inventory across multiple pods will increase the chances that a pod will be nearby when a pick station needs the product, which reduces travel times. Spreading the inventory too thin will make it less likely that larger orders can be fulfilled from a single pod. It also means that the pod with that product may need to be replenished more often. This also depends on the replenishment level.

This chapter provides insights for practice by showing how to optimize three key decision variables related to the replenishment process: (1) the number of pods per product, (2) the ratio of the number of pick stations to replenishment stations, and (3) the replenishment level per pod. These

three decision variables are examined together as they jointly affect the performance of the system.
Our results show that throughput performance improves substantially when inventory is spread across multiple pods, which is probably due to reduced travel times. Order throughput also increases when an optimum ratio between the number of pick stations to replenishment stations is achieved, which reduces unnecessary queueing time. In addition, the experiments give an interesting insight into the replenishment level. A standard practice may be to only replenish a pod with a product when the pod runs completely out of that product. This practice turns out to be far from optimal: replenishment should happen when there are still some units left. Chapter 3 contributes methodologically by introducing a new type of Semi-Open Queueing Networks (SOQN): the cross-class matching multi-class SOQN. For this new type of SOQN, additional and counterintuitive necessary stability conditions hold that are derived and explained in this chapter. Lastly, this chapter contributes by introducing a novel interpretation of the classes in the queueing network, namely as the number of units left on a pod. This gives modelers an additional technique to use, who can use storage locations or pallets instead of pods.

Chapter 4: Resource Reallocation

Demand fluctuations, and especially peak demand, have a large impact on warehouse operations. The Christmas season is a months long period of peak demand, which forces warehouses to hire temporary workers and expand capacity. Some other types of demand peaks are relatively short lived. For example, people tend to order online in the evenings and especially in the weekends.
Peak demand and dynamic resource reallocation are the focus of this chapter. As mentioned above, dynamic resource reallocation is one of the defining and distinguishing characteristics of an RMFS. Resources like workstations and robots can be reassigned from picking to replenishment activities, or vice versa, in a relatively short period of time. As relatively tranquil periods grow into peak demand periods, resources may be reallocated from replenishment

to picking. In the tranquil periods, the replenishment process may receive abundant resources to prepare the storage area for the peak demand to come. In addition, if peak demand periods are relatively short lived, than by dynamically and quickly reallocating resources, an RMFS may be able to handle the peak demand period, and any excess order may be handled in the subsequent, tranquil period.
This chapter models the picking and replenishment activities within an RMFS as a queueing network and then uses Markov Decision Processes to determine the optimal resource reallocation strategies. It contributes methodologically by modeling more than one resource (namely robots and workers), and more than one process (namely both picking and replenishment), and by including time-varying demand.
Dynamic resource reallocation gives the system the ability to quickly adapt as the number of unfinished orders grows too large or as the inventory on the pods dwindles too low, but what are the benefits compared to other policies? This chapter compares dynamic resource reallocation with four benchmark policies from practice. It shows that the benchmark policies perform relatively close to the optimal policy.
Moreover, the experiments indicate that the characteristics of the peak demand phase determine which policy performs best. Finally, the chapter shows that continual reallocation of resources across picking and replenishment can be beneficial and tends to outperform policies that do not do this, especially in situations with a limited number of robots. This provides an interesting direction for future research into continual resource reallocation. All results taken together are interesting for practitioners, since the benchmark policies come from practice, can thus be quickly implemented, and this chapter sheds light on which policy performs best in a given circumstance.

Chapter 5: Decision Rules

This chapter addresses the problems of order assignment, replenishment assignment, pod storage assignment, task allocation and path planning by formulating multiple decision mechanisms for each decision problem.

The decision mechanisms are implemented in a realistic simulation of the RMFS, which simulates in great detail every aspect of the system, from the acceleration and deceleration of the robots and traffic congestion to the queueing and handling at the workstations. In each simulation run, a decision mechanism is chosen for each problem component, and the decision mechanisms may influence each other. For example, whether an order should be assigned to a pick station depends on how far the pods necessary for that order are located from that pick station. However, which pod to select for transportation to the pick station and to which location to return a pod also depends on which orders are assigned to which pick station. In addition, in an e-commerce environment orders typically have short deadlines, so solutions have to be computed nearly instantly.
The main contribution of this chapter lies in the combination of three factors: (1) the simulation is realistic and detailed and includes many aspects that are abstracted away in the models in other chapters, for example traffic jams among the robots. The results should therefore be quite close to the results that a real RMFS offers, (2) we search through the search space quite exhaustively: all combinations of decision rules are simulated and a large number of warehouse scenarios are included, which required special High Performance Computing systems, and (3) we use eight different performance measures, and can therefore assess performance from different angles. Taken together, this chapter provides valuable insights for both practitioners and researchers, as this is one of the first studies were multiple decision problems are addressed simultaneously and thoroughly.

1.4 Authorship and Responsibilities

Chapters 1, 2, 3, and 6 were written by the author of this dissertation. The results were generated through simulations of queueing networks written in Java by the author. The formulation of the research questions, the reviews of literature, implementation and analysis of the models, and the conclusions

were carried out by the author of this dissertation. The promotor and co-promotor provided frequent and important feedback that greatly improved the work. Chapter 2 is published as Lamballais et al. (2017) and the revision benefited from comments by the reviewers and editors. These comments led to the creation of two additional appendices, one on determining the robot speed and another on simultaneous processing of multi-line orders. Chapters 3, 4, and 5 are currently under review.

Chapters 4 and 5 are the result of joint work with Marius Merschformann of the University of Paderborn. The research questions for both chapters were formulated jointly by Marius and the author of this dissertation. The author of this thesis was responsible for formulating the queueing networks and MDP model in Chapter 4 and for writing the text. The author of this thesis wrote the Java code for creating the results of chapter 4, while Marius Mershformann was responsible for executing the code on special computer infrastructure, the High Performance Computing (HPC) systems of the Paderborn Center for Parallel Computing. The promotor and co-promotor of the author of this dissertation provided important feedback on this chapter. Marius Merschformann wrote the realistic simulation framework of the RMFS that forms the backbone of Chapter 5. He furthermore ran the required simulations on the HPC systems of the Paderborn Center for Parallel Computing. The writing of the main text of chapter 5 has been carried out jointly. The promotor and co-promotor of the author of this dissertation as well as Professor Leena Suhl, the promotor of Marius Merschformann, gave important feedback on this chapter that substantially improved it.

2 Warehouse Design

2.1 Introduction

Understanding how order cycle time and robot utilization are influenced by warehouse layout and operating policies is important for practice. This chapter develops several models for estimating performance and robot utilization in an RMFS. These models address the most important process in an RMFS, namely the picking process. It is the most important, because it is responsible for picking the customer orders before their due time. One of the main benefits of an RMFS is that pick rates can reach between 200 and 300 lines per picker per hour (Wurman et al., 2008), (Wulfraat, 2012). The picking process works as follows. An order arrives and waits until it can be assigned to one of the workstations where the orders are picked (see Figure 2.1). Once the order is assigned to a workstation, robots can fetch products for it. Products are stored on inventory pods (i.e., movable shelf racks). A robot moves underneath a pod, lifts it, and brings the pod to a workstation, using the aisles and cross-aisles. The robot enters the workstation buffer and queues for its turn. Each workstation has one picker and once the picker has retrieved the required products from the pod, the robot transports the pod to a storage location and stores it there. The robot then drives to the next pod. As it is moving without a load it does not need to use the aisles but can move underneath the pods. Once all the required products of an order are collected, that order leaves the system and another order can be assigned to the workstation. For a complete description of an RMFS see Wurman & Enright (2011) and Wurman et al. (2008).

A top view of a typical warehouse layout with this system is shown in Figure 2.1. The pods are stored in blocks in the storage area with total width W and length L. The dark gray squares represent the pods and the light gray squares represent unoccupied storage locations. The workstations are situated at three sides of the warehouse in Figure 2.1. The aisles in the storage area all have a single travel direction to prevent deadlock and reduce aisle congestion. For the same reason, movement in the space between the workstations and storage area is also single directional. Figure 2.2 gives a close-up view of the system. The worker picks products from the pod in front of him or her and then adds them to the order totes on the left.

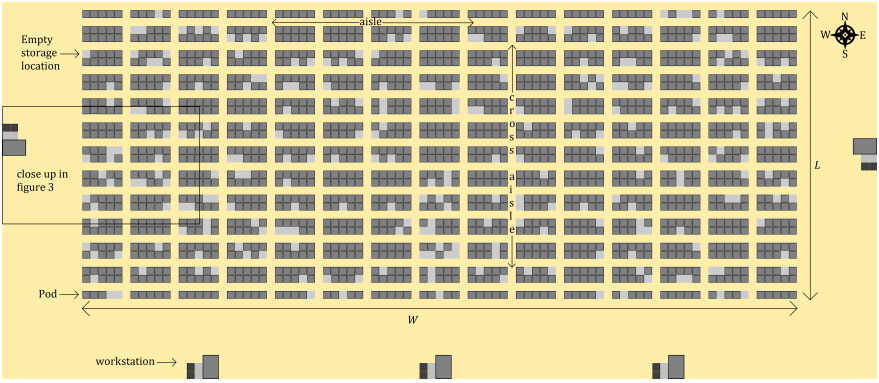
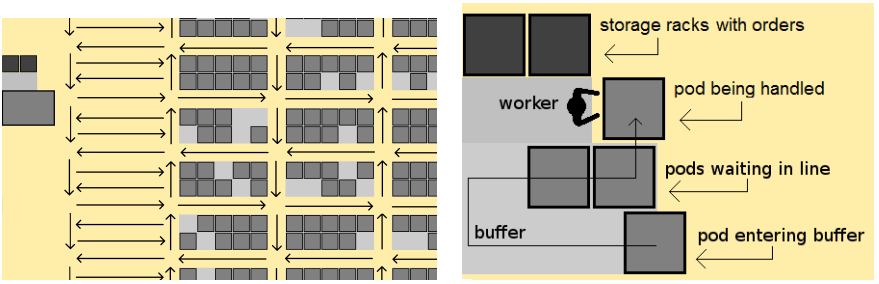


Figure 2.1: Top view of an RMFS with workstations on three sides

During the day, a pod is not required to maintain a fixed position, but can continually be repositioned. By changing the locations of the pods, the system can automatically sort inventory during operations and adapt to varying demand in the short run. The advantage is that the most popular products are usually located close to the workers, even during periods of strong demand fluctuation. Another advantage is that the layout of the warehouse can be rearranged relatively quickly. The number of work stations and their positions can easily be adapted to the changing numbers of workers in each shift. In addition, if storage capacity becomes insufficient, the layout



(a) Travel directions in the warehouse **(b)** Top view of a workstation

Figure 2.2: Close-ups of parts of the RMFS

can be adjusted by adding more pods and storage locations. In other words, the layout is not static, but can be changed to suit changing circumstances relatively quickly.

Typically, the storage area is quite compact, because it only contains products needed within the next few days. With enough robots, workers can be kept busy continuously. So far, few analytical models have been developed to estimate the performance or robot utilization of an RMFS. This chapter develops four queueing network models to estimate performance and robot utilization under different system parameters, warehouse layouts, and control policies. All models focus on the performance of a work station in isolation, but they differ in whether they allow only single-line orders or also multi-line orders, and in whether they divide the storage areas into zones or not. These analytical models require very little computation time and can therefore be used to rapidly optimize the warehouse design, which is not easily possible using simulation models. In addition, the development time needed to adapt these models to analyze a specific warehouse setting will be less than what is needed for a simulation. The queueing models can incorporate the stochasticity in the travel times of the robots and the time that orders have to wait before they can be released to the system, so that the robot utilization and performance metrics such as order throughput and order cycle time can be estimated. By measuring order throughput, order cycle time, and robot

utilization, these models enable warehouse managers and system developers to predict performance and optimize warehouse design. These models also enable researchers to rapidly compare the performance of the RMFS to other automation systems.
This chapter will answer the following design-related research questions. How many orders can be completed per hour given a certain number of robots and work stations? How does the length-to-width ratio of the storage area affect maximum order throughput performance? How does the location of the workstations in the storage area affect maximum order throughput performance? How many robots are needed to achieve a certain desired throughput level and order cycle time?
The remainder of this chapter is as follows: Section 2.2 reviews the literature, Section 2.3 explains the models, Section 2.4 describes the data and results, and Section 2.5 draws conclusions and provides directions for future research.

2.2 Literature

Lu et al. (2016) proposed dynamic order-picking strategies that allow for changes of pick-lists during a pick cycle, which have attracted attention with increase in e-commerce orders. Several modeling and performance analysis studies were also carried out on unit-load vehicle-based storage and retrieval systems. However, these studies consider strict rectilinear travel for storage and retrieval with one load/unload point only (see Tappia et al. (2016), Marchet et al. (2013), Roy et al. (2015a), Roy et al. (2015b), Roy et al. (2016)). Queueing models are popular for analyzing automated warehouse systems, because they can incorporate the stochasticity in the travel times of vehicles and in the speed of the workers, and can establish the effect on performance. Queueing networks have been developed for warehouse automation systems such as autonomous vehicle storage and retrieval systems (AVS/RS) and automated storage and retrieval systems (AS/RS). Kuo et al. (2007) use queueing models to focus on five key design variables in AVS/RS

systems for predicting vehicle utilization and service, waiting and cycle times. These five key design variables are the number of aisles, the number of storage columns per aisle, the number of storage tiers in the system, the number of vehicles in the system, and the number of lifts in the system. Their main conclusion is that these models are computationally effective for exploring the effect of these key variables. Fukunari & Malmborg (2009) develop a queueing model that can estimate the expected utilization of resources in an AVS/RS machine and that can incorporate both single and dual command cycles. Schleyer & Gue (2012) develop a queueing model to estimate the distribution of the order throughput time. This queueing model can handle both single-line and multi-line orders, and the model is based on discrete time to better capture arrival rates from empirical data. Heragu et al. (2011) model variants of both AVS/RS and AS/RS as Open Queueing Networks (OQN) and analyze the OQNs using a tool called the manufacturing system performance analyzer. Their conclusion is that this approach works better than simulation for rapidly evaluating different designs.

Besides OQNs, a number of papers use semi-open queueing networks (SOQN) for modeling a system because they can include the time an order has to wait before being processed. Roy et al. (2012) conduct a performance analysis for AVS/RS using a multi-class semi-open queueing network. This work explores the impact of system parameters, for example, the number of zones, the depth-to-width ratio, the number of vehicles and lifts, and the impact of operational decisions such as vehicle assignment rules on performance measures such as cycle times and vehicle utilization. As SOQNs do not have closed form expressions, they develop a decomposition approach to evaluate system performance. Roy et al. (2013) study blocking in AVS/RS and the effect on transaction cycle times and system throughput. They use a semi-open queueing model and vary system parameters to study the effect of blocking delays within the AVS/RS. Ekren et al. (2014) use a SOQN to analyze an AVS/RS and apply the matrix-geometric method to solve the model, and obtain quite accurate performance measures.

To the best of our knowledge, Nigam et al. (2014) is the only paper which

develops queueing networks for an RMFS. However, they estimate order throughput time only for single-line orders and do not include zoning. This chapter builds on this work by developing a queueing model that includes storage zoning and multi-line orders. In addition, it models robot travel underneath the pods and assumes a layout that is more realistic with multiple cross-aisles.

2.3 Models

2.3.1 Approach

The aim of this chapter is to construct an analytical model to study system performance. Performance is measured using three metrics, namely order throughput, average order cycle time, and robot utilization. Order throughput is the rate of orders leaving the system, the average order cycle time is the average time between order arrival at and departure from the warehouse, and robot utilization is the percentage of time that a robot is assigned to an order, averaged over all robots. This network should accurately estimate the three metrics, given system parameters such as the number of pods, robots and workers, and given different warehouse designs and different workstation locations. The network analyzes the performance of one workstation in isolation. The first, basic network assumes that all orders are single-line orders. The first extension to this model is to include storage zones. This means that the storage area is divided into multiple, non-overlapping regions called storage zones, where products are assigned to a storage zone depending on their demand frequency. Other forms of zoning are absent, so robots can work at any location and are not restricted to certain zones. The second extension can also handle multi-line orders.
This results in the following four models:

Model M_1: Single-line without storage zones

Model M_2: Single-line with storage zones

Model M_3: Multi-line without storage zones

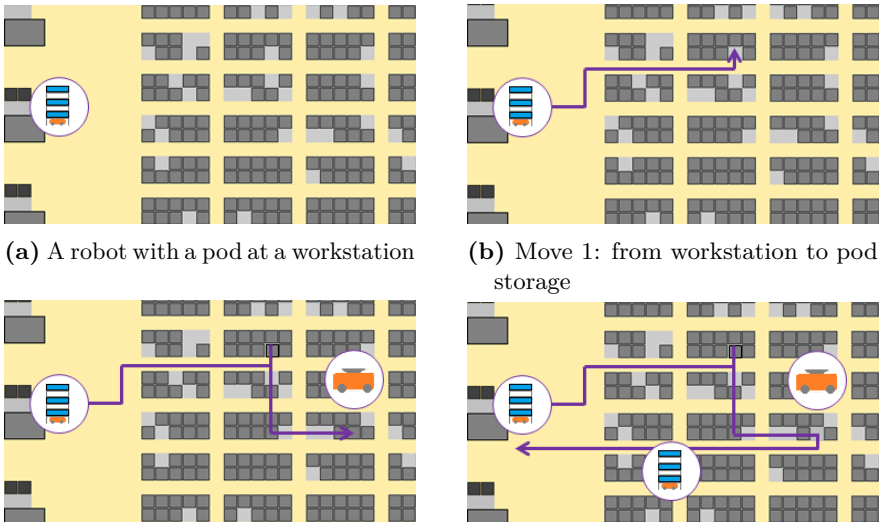
Model M_4: Multi-line with storage zones

The main assumptions are the following. (1) Storage and retrieval occurs at a random location. If storage zones are present, this location is random within the appropriate zone and otherwise it is a random location within the entire storage area. (2) Robots are dedicated to a workstation and are not used by another workstation if they are idle. (3) Aisles have single directional travel everywhere. (4) Delays at aisle intersections do not occur and neither do battery recharges or robot downtime. (5) Robot velocity is constant. (6) Robot congestion or blocking in aisles does not occur. This assumption is close to reality, since aisles are single directional and hence deadlock rarely occurs. (7) The storage area always contains a pod with sufficient units of a product to satisfy any incoming order line. (8) The Point Of Service Completion (POSC) is the dwell point policy for robots, which means that robots do not have to travel to a predetermined dwell point after a service completion. (9) Picking time is stochastic rather than deterministic, because the number of units needed to satisfy an order line vary. (10) The picking time follows a general distribution with mean $\frac{1}{\mu_p}$. (11) The order lines of an order are picked sequentially. (12) The order arrival process follows an exponential distribution.
The subsections below explain the four queueing networks. This is followed by an explanation of calculating the travel times and thereafter by the analysis of the queueing networks.

2.3.2 Model M_1: Single-line without storage zones

The first queueing network describes a single workstation. It is based on three basic robot movements as depicted in Figure 2.3. Suppose that the picker has completed picking products from a pod. This means that an

order line was filled using a product stored on that pod. The robot moves the pod from the workstation to a storage location and stores it. This is move 1. When it is matched with a request to retrieve a product for another order line, the robot will move from the storage location and move to a pod that contains that product. It then lifts this pod and takes it to the workstation. This is move 2. Upon arrival, it lifts the new pod and brings it to the workstation. This is move 3. Figure 2.4 shows the queueing model that corresponds to this process.



(a) A robot with a pod at a workstation

(b) Move 1: from workstation to pod storage

(c) Move 2: from pod storage to pod retrieval

(d) Move 3: from pod retrieval to workstation

Figure 2.3: Robot movements

At the workstation, robots queue until it is their turn, and the worker picks first-come first-serve with an average rate of μ_p lines per time unit. Each workstation has exactly one worker, therefore workstations are modeled as single server stations. After the picker has finished with the pod, the order line leaves the system. The robot with pod then enters a station that models move 1, namely traveling to a storage location and storing the pod. Once

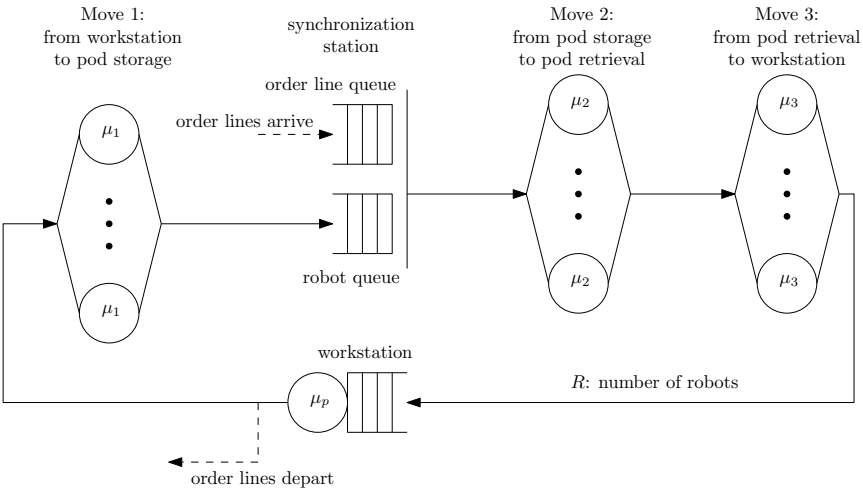


Figure 2.4: Model M_1: Single-line without storage zones

the pod is stored, the robot is unloaded and must be matched with a new order line at the synchronization station. The dwell policy is POSC, so the robot waits under the pod. Order lines arrive at the synchronization station and are synchronized with idle robots. After the synchronization station, two Infinite Server (IS) stations model the time it takes to execute move 2 and move 3, respectively.

Move 1 models travel from the workstation to the storage location where the pod needs to be stored. The storage policy is random storage, so the robot goes to any of the storage locations with equal probability. It is possible to obtain a distribution for the time a robot needs for move 1 by calculating the travel times between the workstation and each storage location (see Section 2.3.6) and weighing those travel times with these probabilities. The service rate is μ_1, as depicted in Figure 2.4. Service time distributions for move 2 and move 3 can be constructed in a similar way, where μ_2 and μ_3 are the service rates of the IS stations for move 2 and 3, respectively.

All the queues modeling travel are IS stations, because robots do not need to queue to begin traveling. The network is modeled as a Semi-Open Queueing

Network (SOQN) to capture the time that the order lines have to wait before being matched with a robot. Without the time needed for synchronization, the model would estimate the maximum order throughput possible rather than the actual throughput for a given order line arrival rate.

2.3.3 Model M_2: Single-line with storage zones

In an RMFS, pods with popular products tend to be stored near workstations and those with less popular products tend to be stored further away from workstations. The main purpose is to reduce travel time. The idea of storage zones incorporates this aspect into the model. In a zoned storage system, each storage zone corresponds to a particular part of the storage area and products are assigned to storage zones based on their demand frequency. The probability that an order line needs products on a pod belonging to a zone z is denoted by p_z. Robots are dedicated to workstations but not to storage zones; each robot can visit each storage zone. Storage zones do not overlap, so all storage zones together cover the entire storage area and multiple robots can be in the same storage zone simultaneously.
The model contains a total of Z zones. This means that the model contains Z stations modeling move 1, one station for each of the zones. Move 3 is modeled in a similar way, using Z stations. Move 2 is modeled using $Z \times Z$ stations, because the robot can be in any of the Z zones after storing a pod and may need to go to any of the Z zones to retrieve the next pod.
This model is shown in Figure 2.5. Storage and retrieval within the zones are random. Here μ_{z1}^{-1} is the average travel time from the workstation to a random storage location in zone z, with subscript 1 referring to move 1. In other words, μ_{z1} is the service rate of the IS station for zone z and move 1. μ_{ij2}^{-1} is the average travel time from a random storage location in zone i to a random storage location in zone j, with subscript 2 referring to move 2. μ_{z3}^{-1} denote the average travel time from a random storage in zone z to the workstation, with subscript 3 referring to move 3.
The routing probabilities shown in Figure 2.5 are based on the probabilities

p_z. For example, consider a scenario for move 2, where the robot stores a pod in zone 1 and needs to retrieve a pod in zone Z. The probability of this scenario occurring is $p_1 \times p_Z$, because the probability that the pod that was stored belongs to zone 1 is p_1 and the probability that the pod that needs to be retrieved belongs to zone Z is p_Z.

The division of the storage area into zones is workstation dependent, see the examples in Figure 2.6. In the these examples, the number of storage zones Z equals three, and zone 1 covers about 20% of the storage area, zone 2 about 30%, and zone 3 about 50%. For workstations that are located west or south of the storage area, the division is as indicated in Figure 2.6. The zoning is assumed to be workstation dependent. This implies that when the layout has one workstation located west of the storage area and another one east, then a storage location close to the one located west would be in zone A in the analysis of that workstation, but when analyzing the workstation located east, it would be a zone C location. The zones indicate the likelihood that a pod is retrieved from that area to the workstation (see Figure 2.6). This concurs with practice, since copies of fast movers can be stored on multiple pods and the system continues to reconfigure to keep the most popular products near the workstations (Wurman et al. (2008)).

2.3.4 Models M_3 and M_4: Multi-line without and with storage zones

This section extends both model 1 and 2 to multi-line orders. It assumes that the number of lines in an order follows a geometric distribution with parameter p. The average number of order lines is therefore $\frac{1}{p}$. Model 3 extends model 1 with multi-line orders and is shown in Figure 2.7.

During move 1, the robot transports the pod to a storage location and stores it. With probability $1 - p$, the order that was assigned to the robot needs more order lines and the robot goes to the IS station, modeling move 2. With probability p, the order that was assigned to the robot needs no more order

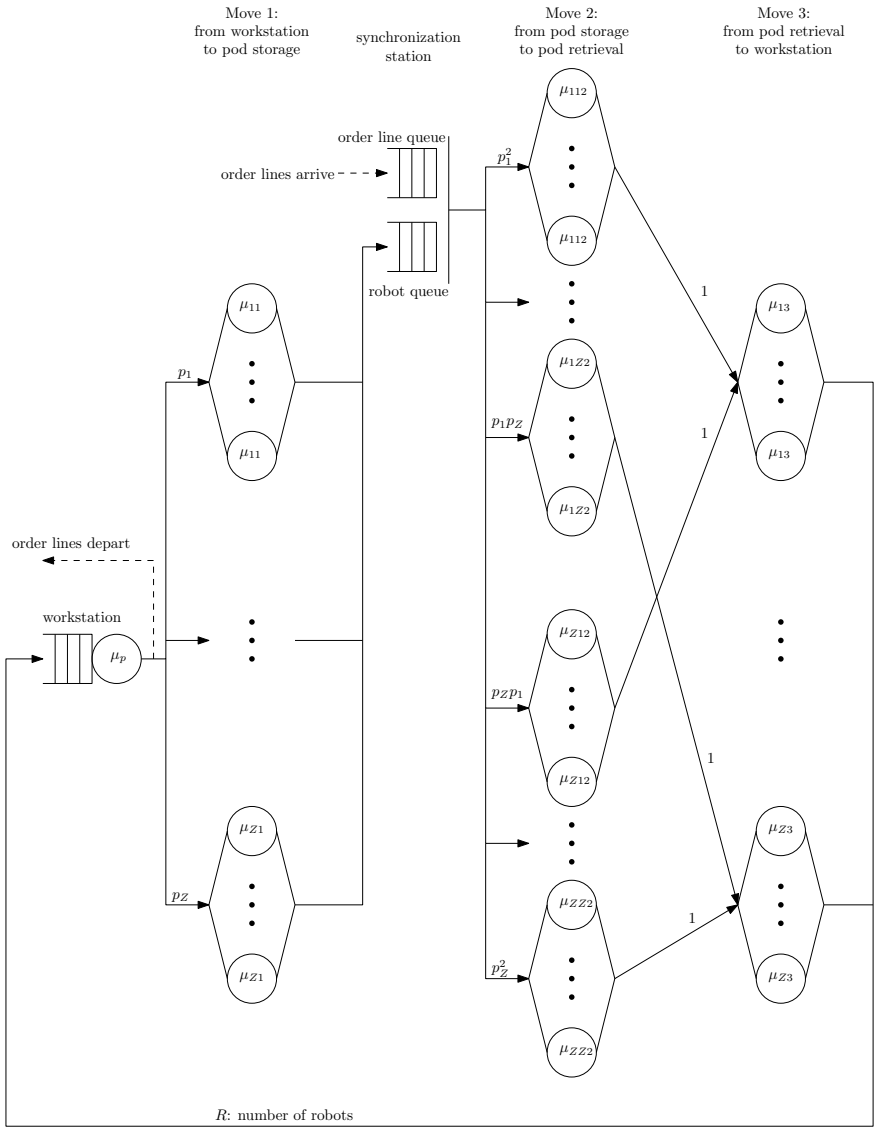
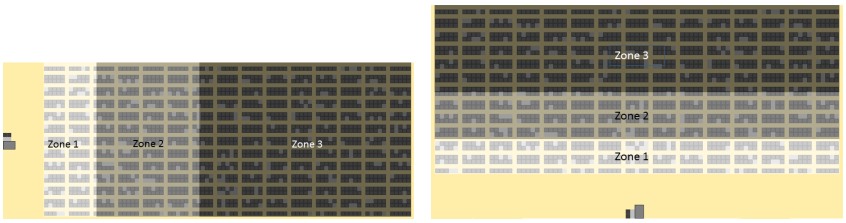


Figure 2.5: Model M_2: Single-line with storage zones



(a) Division for a workstation located west of storage area

(b) Division for a workstation located south of storage area

Figure 2.6: Division of the storage area in three storage zones

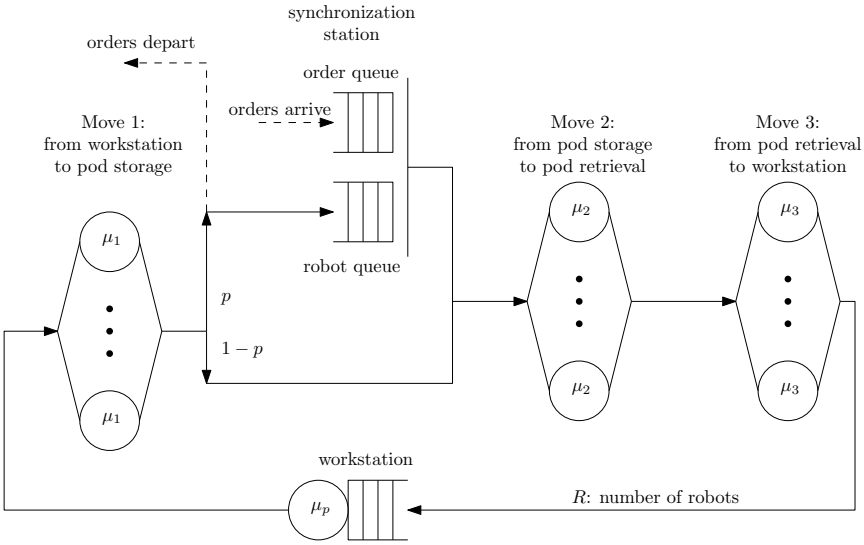


Figure 2.7: Model M_3: Multi-line without storage zones

lines, the order leaves the system and the robot goes to the synchronization station to wait for a new order. Model 2 can be extended in a similar way to arrive at model 4, a model with multi-line orders and storage zones.

2.3.5 Analysis

The queueing networks of models 1 to 4 are analyzed as single class Semi-Open Queueing Networks (SOQN) and solved using the solution procedure from Section 2.2 of Buitenhek et al. (2000). This procedure solves a SOQN follows 3 steps. Step 1: A Closed Queueing Network (CQN) is created by removing the synchronization station from the SOQN. This CQN is analyzed with an Approximate Mean Value Analysis (AMVA) (see Appendix 2.A). The AMVA yields $\tau_{\text{CQN }1}$, the throughput of the CQN.
Step 2: A second CQN is created by replacing the synchronization station in the SOQN with a load-dependent exponential station. This station is denoted as station $S+1$, with S the number of stations in the first CQN. Station $S+1$ has service rate $\nu(r) = \lambda$ for $r > 1$, when r robots are at the station. Here λ denotes the arrival rate of the orders. The network is only stable if $\lambda < \tau_{\text{CQN }1}$. For $r = 1$ the service rate is $\nu(1) = (1 - \frac{\lambda}{\tau_{\text{CQN }1}})\lambda$. The same AMVA algorithm can then be used to analyze this second CQN, yielding the throughput $\tau_{\text{CQN }2}$. This AMVA algorithm also calculates $L_s(r)$ the queue length at station s when r robots are present. Step 3: the solution procedure analyzes station $S+1$ in isolation to calculate L_o, the mean length of the external queue of orders.
The other measures of interest are ρ_r, the utilization of the robots, t_{oc}, the order cycle time, and ρ_{ws}, the utilization of the workstation. Let L_r be the expected length of the robot queue at station $S+1$, as found by the AMVA algorithm for the second CQN. Then $\rho_r = 1 - \frac{L_r}{R}$, where R denotes the total number of robots in the system. Let L_i be the sum of the expected queue

lengths at the other stations, so $L_i = \sum_s L_s(R)$. Then the average order cycle time is as depicted in equation (2.1).

$$t_{oc} = \frac{L_o + L_i}{\lambda} \tag{2.1}$$

The workstation utilization is $\rho_{ws} = \tau_{\text{CQN }2} v_{\text{ws}} ES_{\text{ws}}$, with v_{ws} the visit ratio of the workstation and ES_{ws} the mean service time at the workstation, see also Appendix 2.A. This method allows each station in the network to have c_s parallel servers. The Infinite Server stations are modeled by setting c_s equals to the number of robots R. The AMVA is an approximation as it uses the first and second moments of the service time distributions as input, allowing the service times to follow a general distribution.

2.3.6 Travel Times

The service time of an IS station in model 1 depends on the travel times that this IS station models. This section explains how to calculate the travel times for each move. During move 1 and 3, the robot is carrying a pod (loaded travel) but in move 2, the robot is not carrying a pod (unloaded travel). The travel distance for unloaded travel is simply the Manhattan distance, but calculating the travel distance for loaded travel is more complicated. Therefore, this section will mostly focus on calculating travel distance for loaded travel.

Once the travel distances are known, they must be divided by the speed of the robot. For move 1, the time needed to store the pod is added and for move 3, the time needed for lifting is added. Storing time, lifting time, and robot speed are assumed to be constants and the robot does not need to accelerate or decelerate. The resulting travel times fully describe the service times of the IS stations in each of the models.

Move 1 and 3: Loaded Travel

Loaded travel is rectilinear and each aisle has only one travel direction. This means that the travel time to obtain queue length distributions in model 1 can be calculated using closed form expressions. The location entrance of a storage location is the point located in the aisle in front of that storage location. The robot uses the location entrance to enter the storage location. The workstation entrance is the point in the hall from where the robot can enter the buffer of a workstation. The start intersection is a more complicated concept. Suppose that a robot travels from a storage location that has a location entrance in an aisle with easterly travel direction and travels to a work station that is located west of the storage area. It then first has to move in easterly direction before it can move west to the workstation, so initially the distance between the robot and its destination will increase. The start intersection is the first point on the robot's route where the distance decreases. This point is always at an intersection of an aisle and a cross-aisle. More formally: on the shortest route between a storage location SL and a workstation WS, the start intersection is the first intersection with an outgoing arc that points towards the hall in which WS is located. For example, if SL is situated at an aisle with easterly travel direction while WS is located west of the storage area, the start intersection is the first intersection with direction west on the shortest route between SL and WS. Let the length of a block be denoted by l, the width by w, the unit distance by u, let the location entrance (abbreviated as le) of storage location SL be given by (x_{le}, y_{le}), the entrance of the buffer of workstation WS by (x_{ws}, y_{ws}), and the start intersection by (x_{si}, y_{si}). The unit distance is the width of one storage location and in the standard layout (see Figure 2.1) $l = 2u$ and $w = 5u$. The aisles are u wide. The distance $d_{ca,le}$ is the distance between (x_{le}, y_{le}) and the first cross-aisle while following the direction of the aisle in which the location entrance is situated. The distance $d_{le,si}$ is the distance between the location entrance and the start intersection. Aisles

and cross-aisles can only have one direction, see also Figure 2.8 and 2.9. In Figure 2.9, the start intersection is depicted as a big dot.

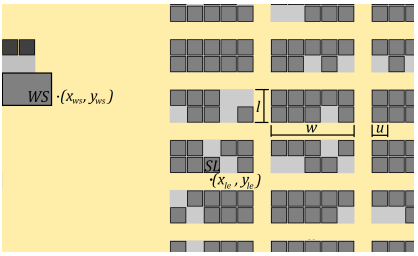


Figure 2.8: Notation explained graphically

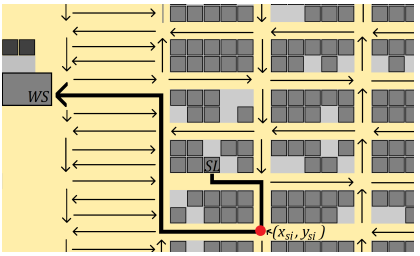


Figure 2.9: Shortest route from SL to WS

A shortest route from a storage location SL to a workstation WS can be divided into four parts. The first is the distance between SL and its location entrance, which is equal to u, since both storage locations and aisles are u wide. The second part is the distance between the location entrance and the start intersection. The third part is the Manhattan distance between the start intersection and the buffer entrance of WS, which equals $|x_{si} - x_{ws}| + |y_{si} - y_{ws}|$. The fourth part is a detour $\Delta_{le,ws}$ that may be necessary because of travel directions in the hall between WS and the storage area. This detour $\Delta_{le,ws}$ is either $2u$ or 0, depending on the location of the buffer entrance of WS. The conditions under which $\Delta_{le,ws} = 2$ are straightforward and simple, but too numerous to list here.
The distance of the shortest route for all storage locations and workstations can be derived from four fundamental cases:

Case 1: the workstation is located west (east) of the storage area and the location entrance is situated in an aisle with travel direction west (east)

Case 2: the workstation is located west (east) of the storage area and the location entrance is situated in an aisle with travel direction east (west)

Case 3: the workstation is located north (south) of the storage area and the first cross-aisle encountered has travel direction north (south)

Case 4: the workstation is located north (south) of the storage area and the first cross-aisle encountered has travel direction south (north)

In the first case, (x_{le}, y_{le}) is located in an aisle whose direction is towards WS. For example, (x_{le}, y_{le}) is located in an aisle with a westerly travel direction and WS is located west of the storage area. The distance D is as expressed in equation (2.2).

$$D = u + |x_{le} - x_{ws}| + |y_{le} - y_{ws}| + \Delta_{le,ws} \tag{2.2}$$

In the second case, (x_{le}, y_{le}) is located in an aisle whose direction is not towards WS. For example (x_{le}, y_{le}) is located in an aisle with a westerly travel direction and WS is located east of the storage area. In this example, the start intersection is the first intersection with travel direction east on the shortest route. If the first cross-aisle west of (x_{le}, y_{le}) has travel direction north, then the start intersection is the intersection to the northwest and if the cross-aisle has travel direction south, then the start intersection is the intersection to the southwest. In both cases $d_{le,si} = d_{ca,le} + l + 2u + w$. The distance D is as expressed in equation (2.3).

$$D = u + d_{le,si} + |x_{si} - x_{ws}| + |y_{si} - y_{ws}| + \Delta_{le,ws} \tag{2.3}$$

In the third case, WS lies to the north or south and the first cross-aisle has a direction towards WS. For example, suppose that WS lies north of SL and that (x_{le}, y_{le}) is located in an aisle with travel direction west, then the first cross-aisle west of SL has travel direction north and the start

intersection is the intersection directly west of (x_{le}, y_{le}). For this case in general, $d_{le,si} = d_{ca,le}$. The distance D is as expressed in equation (2.4).

$$D = u + d_{le,si} + |x_{si} - x_{ws}| + |y_{si} - y_{ws}| + \Delta_{le,ws} \tag{2.4}$$

In the fourth case, WS lies to the north or south and the first cross-aisle does not have a direction towards WS. For example, WS lies north of SL and (x_{le}, y_{le}) is located in an aisle with travel direction west, then the first cross-aisle west of SL has travel direction south. In general, one can choose between two possible start intersections, the first cross-aisle to the west with direction towards WS and the first to the east. Following the example, for the first option, the distance D_1 is as expressed in equation (2.5).

$$D_1 = u+d_{ca,le}+w+u+|(x_{le}-d_{ca,le}-w-u)-x_{ws}|+|y_{le}-y_{ws}|+\Delta_{le,ws} \tag{2.5}$$

For the second option, the distance D_2 is as expressed in equation (2.6).

$$D_2 = u+d_{ca,le}+2l+w+3u+|(x_{le}-d_{ca,le}+w+u)-x_{ws}|+|y_{le}-y_{ws}|+\Delta_{le,ws} \tag{2.6}$$

The distance of the shortest route is simply $D = \min(D_1, D_2)$. The first option is not available if the location entrance is situated in one of the westernmost blocks; the second option is not possible if it is situated in one of the easternmost blocks. The northernmost and southernmost blocks have only half the normal length and therefore, in the second and the fourth case, the term l becomes $\frac{1}{2}l$ if the path from location entrance to start intersection goes past these blocks.

All of the preceding formulas have been validated by comparing the results with the shortest routes found by the Dijkstra algorithm for a standard layout. These formulas are for routes from a storage location SL to a workstation WS, but are similar for routes from WS to SL. The formulas above therefore capture the routes for move 1 and 3.

2.4 Results

The results in this section come from three experiments. The first experiment shows the results for all four models and serves as validation. This experiment uses the standard layout as shown in Figure 2.1. The number of robots R, and the average number of order lines per order p are varied to understand the effect on the order cycle time, robot utilization, and workstation utilization. In the second experiment, the effect of the storage area's length-to-width ratio on maximum order throughput is explored by changing the number of aisles and cross-aisles while keeping the number of storage locations within 4% of 1800. In the third experiment, the effect that the locations of workstations along the sides of the storage area has on maximum order throughput is studied.
The traditional ABC categorization is used for zoning. This means that the storage area is divided into an A, B, and a C zone. According to Wulfraat (2012), robot speed is about three miles per hour, which is about 1.3 meters per second, and the average time for picking an order line is six seconds. This excludes the time needed to move the pod in front of the picker, for which no average length is mentioned. Additional experiments in Appendix 2.B show how to calculate the average robot speed given a maximum speed and an acceleration. These experiments indicate that 1.3 m/s corresponds to a maximum robot speed of 1.5 m/s and an acceleration of 0.75 m/s^2. According to Wulfraat (2012) and Wurman et al. (2008), pick rates are above 200 lines per hour, therefore the average time for picking is set at 15 seconds in total or 240 lines maximum per hour. The distribution of the picker time

is a C_k distribution, which is a Cox-k distribution as described in Bolch et al. (2006). The parameters are shown in Table 2.1, where cv^2 denotes the squared coefficient of variation.

Table 2.1: Parameters used in the experiments

Parameter	Value
Number of aisles	12
Number of cross-aisles	14
Number of storage locations	1800
Number of zones	3: A, B, C zone
Number of storage location per zone	A: 20%, B: 30%, C: 50%
Probability pod comes from	A: 70%, B: 25%, C: 5%
Robot speed	1.3 (m/s)
R, number of robots	2, 8, 14
Time for pod lifting and storing	1 (s)
Distribution picker time	C_k, mean λ is 15 (s), cv^2 is 1.0

2.4.1 Experiment 1: A Single Work Station

In each of the four models a single workstation is analyzed. Since the robots are dedicated per workstation, the results can be calculated by analyzing each of the workstations separately and then taking the average across the workstations. In the tables below, R denotes the number of robots dedicated to the workstation, λ denotes the order arrival rate in orders per hour, ρ_r denotes the robot utilization, L_o is the mean length of the external order queue, t_{oc} denotes the average order cycle time in seconds, and ρ_{ws} denotes the utilization of the workstation. For the models with multi-line orders, models M_3 and M_4, the number of order lines in an order is geometrically distributed, $\sim \text{Geom}(p = k)$, where p refers to the parameter of the Geometric distribution and k is some number between zero and one. The parameter used is indicated by changing the notation to $M_3(p = k)$ and $M_4(p = k)$. Results are shown for R equal to 2, 8 and 14 and for a high and a low arrival

rate, leading to six tables in total (Tables 2.2 to 2.7). The arrival rates for 8 robots are 4 times the arrival rates for 2 robots and the arrival rates for 14 robots are 7 times the arrival rates for 2 robots. A discrete event simulation model was built to validate the results. It was built from scratch in Java and simulates the queueing models. The numbers in the tables are averaged over a hundred runs, where each run simulated the network for 168 hours, a full week. The width of the 95%-confidence intervals is usually less than 1 per cent of the number itself and at most a few per cent. For $p = 0.5$, the multi-line orders have an average of two order lines and they have an average of five order lines for $p = 0.2$. The order arrival rate was divided by 2 and 5 respectively to ease the comparison with the single-line models. The utilization of the robots and the workstation from the analytical method is nearly the same as for the simulation. The mean length of the external order queue does differ between the analytical method and the simulation and this affects the estimates of the order cycle time. The average order cycle time depends on the mean length of the external order queue and therefore also differs between the analytical method and the simulation. However, the differences are relatively small for the average order cycle time. The estimates of the analytical method typically stay below 10% of the estimates of the simulation, except for high arrival rate when $R = 2$. As is evident from the tables, using zones lowers the robot utilization and order cycle time.

Table 2.2: Results experiment 1: 2 robots and high arrival rate

$R = 2$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	31.68	64.6	0.95	255.2	13.2	64.7	0.52	206.1	13.2
M_2	31.68	49.1	0.34	150.5	13.2	49.1	0.20	133.7	13.2
$M_3(p = 0.5)$	15.84	64.6	0.95	510.4	13.2	64.7	0.74	460.5	13.2
$M_4(p = 0.5)$	15.84	49.1	0.34	300.9	13.2	49.1	0.27	283.8	13.2
$M_3(p = 0.2)$	6.34	64.6	0.95	1276.1	13.2	65.0	0.89	1232.0	13.3
$M_4(p = 0.2)$	6.34	49.1	0.34	752.3	13.2	49.2	0.31	734.8	13.2

Table 2.3: Results experiment 1: 2 robots and low arrival rate

$R = 2$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	14.40	29.3	0.06	160.4	6.0	29.3	0.03	154.4	6.0
M_2	14.40	22.2	0.02	117.1	6.0	22.2	0.02	114.6	6.0
$M_3(p = 0.5)$	7.20	29.3	0.06	320.7	6.0	29.4	0.04	314.2	6.0
$M_4(p = 0.5)$	7.20	22.2	0.02	234.2	6.0	22.3	0.02	232.0	6.0
$M_3(p = 0.2)$	2.88	29.3	0.06	801.8	6.0	29.6	0.05	799.4	6.1
$M_4(p = 0.2)$	2.88	22.2	0.02	585.4	6.0	22.4	0.02	582.4	6.1

Table 2.4: Results experiment 1: 8 robots and high arrival rate

$R = 8$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	126.72	70.0	0.80	181.8	52.8	69.9	0.51	173.3	52.8
M_2	126.72	54.9	0.22	130.9	52.8	54.8	0.16	129.1	52.8
$M_3(p = 0.5)$	63.36	70.0	0.80	363.7	52.8	70.0	0.64	354.1	52.8
$M_4(p = 0.5)$	63.36	54.9	0.22	261.9	52.8	54.9	0.19	260.0	52.8
$M_3(p = 0.2)$	25.34	70.0	0.80	909.2	52.8	70.0	0.73	898.2	52.8
$M_4(p = 0.2)$	25.34	54.9	0.22	654.6	52.8	54.8	0.21	652.4	52.7

Table 2.5: Results experiment 1: 8 robots and low arrival rate

$R = 8$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	57.60	30.1	0.00	150.5	24.0	30.1	0.00	150.4	24.0
M_2	57.60	23.0	0.00	114.9	24.0	23.0	0.00	114.9	24.0
$M_3(p = 0.5)$	28.80	30.1	0.00	301.0	24.0	30.1	0.00	301.0	24.0
$M_4(p = 0.5)$	28.80	23.0	0.00	229.9	24.0	23.0	0.00	229.9	24.1
$M_3(p = 0.2)$	11.52	30.1	0.00	752.6	24.0	30.1	0.00	753.7	24.0
$M_4(p = 0.2)$	11.52	23.0	0.00	574.7	24.0	23.0	0.00	575.1	24.0

Table 2.6: Results experiment 1: 14 robots and high arrival rate

$R = 14$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	221.76	91.5	14.80	448.1	92.4	91.3	12.63	412.7	92.4
M_2	221.76	82.6	7.25	305.4	92.4	82.7	7.19	304.5	92.4
$M_3(p = 0.5)$	110.88	91.5	14.80	896.2	92.4	91.4	13.49	852.2	92.4
$M_4(p = 0.5)$	110.88	82.6	7.25	610.9	92.4	82.7	7.35	614.1	92.4
$M_3(p = 0.2)$	44.35	91.5	14.80	2240.6	92.4	91.5	13.88	2159.8	92.4
$M_4(p = 0.2)$	44.35	82.6	7.25	1527.2	92.4	82.5	7.21	1520.7	92.3

Table 2.7: Results experiment 1: 14 robots and low arrival rate

$R = 14$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	100.80	31.3	0.00	156.5	42.0	31.4	0.00	156.6	42.1
M_2	100.80	24.2	0.00	121.0	42.0	24.2	0.00	121.1	42.0
$M_3(p = 0.5)$	50.40	31.3	0.00	313.1	42.0	31.3	0.00	312.9	42.0
$M_4(p = 0.5)$	50.40	24.2	0.00	242.1	42.0	24.2	0.00	242.0	42.0
$M_3(p = 0.2)$	20.16	31.3	0.00	782.6	42.0	31.3	0.00	782.5	42.0
$M_4(p = 0.2)$	20.16	24.2	0.00	605.2	42.0	24.2	0.00	604.4	42.0

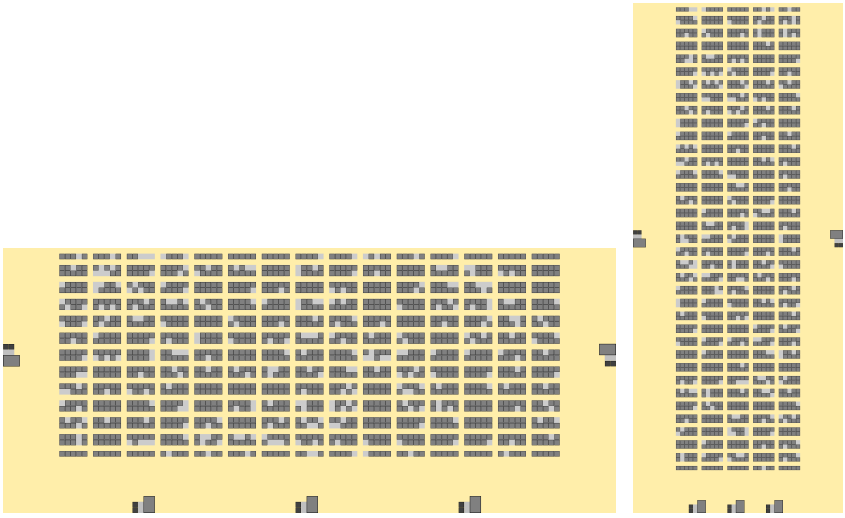
For the picker time, the model can also handle Cox-k distributions with cv^2 lower than 1, but then discrepancies arise between the analytical and the simulation results, even for robot utilization. Table 2.8 shows the results for 14 robots with a high arrival rate, where $cv^2 = 0.6$. In other words, choosing a cv^2 different from 1 is possible, but the analytical results will no longer be fully reliable. The models M_3 and M_4 assume that processing of multi-line orders happens sequentially, whereas in practice this may happen simultaneously. In Appendix 2.C simulations are shown where sequential and simultaneous processing of multi-line orders are compared.

Table 2.8: Results experiment 1: 14 robots, high arrival rate and $cv^2 = 0.6$

$R = 14$		analytical model				simulation model			
model	$\lambda(h^{-1})$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	L_o	$t_{oc}(s)$	$\rho_{ws}(\%)$
M_1	221.76	90.7	13.11	418.9	92.4	88.5	7.11	316.5	92.4
M_2	221.76	82.0	6.95	299.1	92.4	80.1	4.80	259.8	92.5
$M_3(p = 0.5)$	110.88	90.7	13.11	837.8	92.4	88.7	8.50	678.7	92.3
$M_4(p = 0.5)$	110.88	82.0	6.95	598.2	92.4	80.6	5.56	546.4	92.4
$M_3(p = 0.2)$	44.35	90.7	13.11	2094.6	92.4	88.7	9.14	1750.0	92.3
$M_4(p = 0.2)$	44.35	82.0	6.95	1495.5	92.4	80.7	5.80	1386.2	92.4

2.4.2 Experiment 2: Varying The Length-To-Width Ratio Of The Storage Area

Table 2.9 shows the maximum throughput in orders per hour of all the workstations together. In a system that achieves the maximum throughput, robots do not have to wait for an order and hence a synchronization queue is not needed. For the situation without zones, a CQN is created by removing the synchronization queue from model M_1 and similarly for the situation with zones, a CQN was created based on model M_2. The throughput per workstation was calculated by applying the single class AMVA method in Appendix 2.A to these CQNs. The number of aisles and of cross-aisles has to be even in each of the variants, because each (cross-)aisle has a single travel direction. For each aisle going west there must also be one going east and for each cross-aisle going north there must be one going south. The variants were chosen such that the number of storage locations was never more than 4% from 1800. As can be seen from Table 2.9, the result is relatively insensitive for the length-to-width ratio unless the ratio becomes unbalanced by a factor of more than 3 or 4. Using zones increases the maximum throughput by almost 50%. The length-to-width ratio can be measured in aisles or in meters. Since a block of storage locations measures 2 storage locations by 5 storage locations, a layout that has x aisles by y cross aisles has a storage area of $3x$ by $6y + 5$ meters.



(a) 12 aisles by 14 cross-aisles

(b) 36 aisles by 4 cross-aisles

Figure 2.10: Variants with different length-to-width ratios

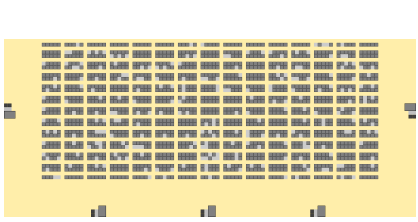
Table 2.9: experiment 2, maximum throughput per hour

		$R = 2$		$R = 8$		$R = 14$	
variant	# locations	no zones	zones	no zones	zones	no zones	zones
2 by 88 aisles	1780	73.5	105.3	291.6	412.5	504.3	693.8
4 by 44 aisles	1800	130.6	181.4	508.2	676.7	840.4	1014.6
6 by 30 aisles	1860	170.9	233.0	649.3	827.0	1011.2	1127.3
8 by 22 aisles	1840	204.2	274.4	755.9	928.3	1103.5	1172.4
12 by 14 aisles	1800	245.1	325.3	871.6	1029.0	1165.2	1193.9
14 by 12 aisles	1820	253.5	336.2	893.2	1048.9	1172.5	1196.3
16 by 10 aisles	1760	262.8	348.5	915.4	1068.1	1178.6	1197.7
20 by 8 aisles	1800	262.5	350.5	914.4	1072.5	1177.9	1198.1
26 by 6 aisles	1820	253.7	345.3	891.8	1066.4	1170.3	1197.9
36 by 4 aisles	1800	229.5	321.5	826.7	1029.9	1141.2	1195.4
60 by 2 aisles	1800	174.6	257.3	659.3	898.0	1013.0	1169.0

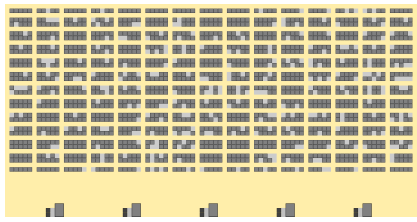
2.4.3 Experiment 3: Varying The Location of Workstations

Table 2.10 shows the maximum throughput in orders per hour of all the workstations together. The throughput per workstation was calculated by applying the single class AMVA method described in Appendix 2.A to the same CQNs as in experiment 2. The configurations are named as "Configuration x_1, x_2, x_3, x_4" where x_1 denotes the number of workstations located west of the storage area, x_2 the number of workstations located east of the storage area, x_3 the number of workstations located south of the storage area and x_4 the number of workstations located north of the storage area, see also Figure 2.11 for some examples. As can be seen from Table 2.10, the maximum throughput is sensitive to the location of the workstations. Interestingly, the results with zones are very different from the results without zones. Without zones, the maximum throughput tends to be higher if the workstations are located north and south of the storage area but if zones are present the maximum throughput tends to be higher if workstation are located west and east of the storage area. The difference between the best and worst workstation configuration is also higher if zones are present. The explanation is that without zones the average travel distance is shorter for

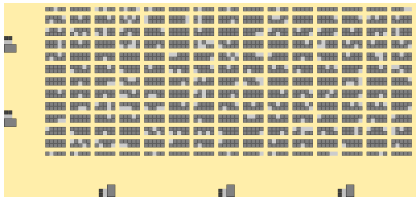
workstations north and south of the storage area, whereas with zones the average travel distance is shorter for zones west and east of the storage area.



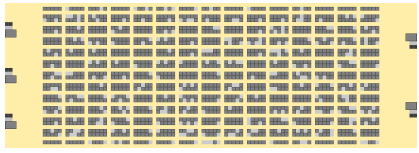
(a) Configuration 1,1,3,0



(b) Configuration 0,0,5,0



(c) Configuration 2,0,3,0



(d) Configuration 3,2,0,0

Figure 2.11: Work station configurations

Table 2.10: experiment 3, maximum throughput per hour

	$R = 2$		$R = 8$		$n = 14$	
variant	no zones	zones	no zones	zones	no zones	zones
configuration $0, 0, 3, 2$	252.0	296.3	889.2	986.1	1171.2	1191.1
configuration $0, 0, 4, 1$	254.2	299.5	893.8	990.6	1171.6	1191.0
configuration $0, 0, 5, 0$	251.1	295.3	886.6	983.0	1169.6	1190.2
configuration $1, 0, 2, 2$	249.8	312.5	884.0	1011.7	1169.8	1193.4
configuration $1, 0, 3, 1$	251.7	315.2	887.9	1015.3	1170.1	1193.3
configuration $1, 0, 4, 0$	247.6	309.6	877.6	1004.3	1166.7	1191.7
configuration $1, 1, 2, 1$	249.5	331.4	882.7	1040.9	1168.7	1195.6
configuration $1, 1, 3, 0$	245.1	325.3	871.6	1029.0	1165.2	1193.9
configuration $2, 0, 2, 1$	247.7	326.9	877.8	1035.9	1166.7	1195.4
configuration $2, 0, 3, 0$	243.3	320.9	866.6	1023.9	1163.1	1193.8
configuration $2, 1, 1, 1$	247.2	345.5	875.9	1064.5	1165.4	1197.5
configuration $2, 1, 2, 0$	241.1	337.0	861.5	1049.5	1161.8	1196.1
configuration $2, 2, 1, 0$	238.8	351.2	854.7	1073.1	1158.4	1198.0
configuration $3, 0, 1, 1$	245.4	341.4	871.0	1059.3	1163.2	1197.3
configuration $3, 0, 2, 0$	239.3	332.8	856.6	1044.3	1159.5	1195.9
configuration $3, 1, 1, 0$	238.8	351.5	854.8	1072.9	1158.3	1198.0
configuration $3, 2, 0, 0$	230.4	357.1	833.6	1081.5	1151.3	1198.5
configuration $4, 0, 1, 0$	237.6	348.6	851.5	1069.5	1156.8	1197.9
configuration $4, 1, 0, 0$	231.1	358.7	835.3	1083.1	1151.9	1198.5
configuration $5, 0, 0, 0$	229.9	355.9	832.0	1079.7	1150.5	1198.4

2.5 Conclusions

The main contribution of this chapter is that it is one of the first to model Robotic Mobile Fulfillment Systems, and includes accurate driving behavior of robots, and multi-line orders. This chapter develops queueing models to analyze an RMFS with and without zones and with single-line and multi-line orders, and it shows how to derive analytical expressions for distributions of the robot travel times. The aim was to gain insights for system design by measuring maximum order throughput, robot utilization, and order cycle time. The first experiment shows that the analytical method accurately estimates robot utilization, workstation utilization, and average order cycle

time. The second experiment indicates that the maximum throughput is insensitive to the length-to-width ratio of the storage area, except if this ratio becomes strongly skewed. The last experiment shows that the location of workstations around the storage area matters. If storage zones are used, maximum throughput tends to be higher if the workstations are located west and east of the storage area, whereas without storage zones, it tends to be higher if workstations are located north and south of the storage area.
Two limitations of this study are that congestion and robot switching between workstations have not been included in the model. In practice however, congestion should only have a small effect on the order throughput and workstation utilization, since the system is designed with the aim that the picker is kept busy to achieve high pick rates. Congestion may then cause a robot to enter the queue at the workstation a little later but the bottleneck in the system is the picker and not transportation. Also, congestion is unlikely to happen, as the number of robots is typically small relative to the space in which they drive and robots can travel underneath the racks when they are not carrying a pod. This makes it unlikely that the robots run into multiple other robots in the adjacent space around them. Robot switching could be beneficial if the workstations have a low utilization. However, this chapter focuses on design aspects, whereas robot switching is connected with operational decisions. For example, the robot may have stored the pod at a particular location, because it is then close to another pod that needs to go to that same workstation. The robot may also have stored the pod close to a workstation that needs products from it soon. In other words, robot switching may bring operational benefits and would be interesting to explore in future research on operational decisions. In addition, Wurman & Enright (2011) mention several operational problems that have not yet been solved for an RMFS. As they show, Robotic Mobile Fulfillment Systems still pose many challenging problems and contain interesting, unexplored avenues.

Appendix

2.A AMVA Algorithm

This appendix shows the single class AMVA algorithm used for evaluating a CQN. It is the AMVA algorithm from Appendix A.2 in Buitenhek et al. (2000), and it has not been adapted except for step 3g where the queue lengths are calculated including the robots in service. The Infinite Servers stations are modeled by setting the number of servers c_s equal to the number of robots R. The notation is explained in Table 2.A.1. Visit ratios are calculated as explained in Bolch et al. (2006).

Table 2.A.1: Notation used in the AMVA

Symbol	Meaning
S	the total number of stations
R	the total number of robots
$ES_{rem,s}$	the expected time remaining until the first departure at station s
ES_s	the first moment of the service time of station s
ES_s^2	the second moment of the service time of station s
$L_s(r)$	the expected robot queue length including robots in service at station s when the system contains r robots
$\tilde{L}_s(r)$	the expected queue length excluding robots in service at station s when the system contains r robots
$Q_s(r)$	the probability that all servers are busy at station s when the system contains r robots
$p_s(i \mid r)$	the probability that there are i robots at station s when the system contains r robots
$\tau(r)$	the throughput when the system contains r robots
$ET_s(r)$	the lead time at station s when the system contains r robots
c_s	the number of servers at station s
v_s	the visit ratio of station s

Step 1: Initialize:

$$p_s(0 \mid 0) = 1, \qquad s = 1, \ldots, S \tag{2.7}$$

$$Q_s(0) = 0, \qquad s = 1, \ldots, S \tag{2.8}$$

$$L_s(0) = 0, \qquad s = 1, \ldots, S \tag{2.9}$$

$$\tilde{L}_s(0) = 0, \qquad s = 1, \ldots, S \tag{2.10}$$

Step 2: Preprocessing. For $s = 1, \ldots, S$

$$ES_{rem,s} = \frac{c_s - 1}{c_s + 1} \frac{ES_s}{c_s} + \frac{2}{c_s + 1} \frac{1}{c_s} \frac{ES_s^2}{2ES_s} \tag{2.11}$$

Step 3: Iteration. For $r = 1, \ldots, R$

(a) For $s = 1, \ldots, S$

$$ET_s(r) = Q_s(r-1)ES_{rem,s} + \tilde{L}_s(r-1)\frac{ES_s}{c_s} + ES_s \tag{2.12}$$

(b)

$$\tau(r) = \frac{r}{\sum_{s=1}^{S} v_s ET_s(r)} \tag{2.13}$$

(c) For $s = 1, \ldots, S$ and for $b = 1, \ldots, \min(c_s - 1, r)$

$$p_s(b \mid r) = \frac{ES_s}{b} v_s \tau(r) p_s(b-1 \mid r-1) \tag{2.14}$$

(d) For $s = 1, \ldots, S$, if $r < c_s, Q_s(r) = 0$, otherwise,

$$Q_s(r) = \frac{ES_s}{c_s} v_s \tau(r)[Q_s(r-1) + p_s(c_s - 1 \mid r-1)] \tag{2.15}$$

(e) For $s = 1, \ldots, S$

$$p_s(0 \mid r) = 1 - \sum_{b=1}^{\min(c_s-1,r)} p_s(b \mid r) - Q_s(r) \tag{2.16}$$

(f) For $s = 1, \ldots, S$, if $r < c_s, \tilde{L}_s(r) = 0$, otherwise,

$$\tilde{L}_s(r) = \frac{ES_s}{c_s} v_s \tau(r)[\tilde{L}_s(r-1) + Q_s(r-1)] \tag{2.17}$$

(g) For $s = 1, \ldots, S$

$$L_s(r) = \tau(r) v_s E T_s(r) \tag{2.18}$$

2.B Determining Robot Speed

Given an acceleration and maximum speed, the average robot speed for a layout can be calculated as follows. Each route consists of straight linear segments that are connected by angles of 90 degrees. Each time a robot turns it starts with a speed of zero and increases speed until it hits the maximum speed or until it is halfway. Then the robot decreases the speed until it is zero again and it turns to go on the next segment. The overall average robot speed is then calculated by averaging across all routes. For the standard layout it was found that an average robot speed of 1.3 m/s, as mentioned in Wulfraat (2012), corresponds to an acceleration of 0.75 m/s^2 and a maximum speed of 1.5 m/s, which seems realistic. In Table 2.B.1 the average robot speed is shown for layouts with varying number of aisles and cross-aisles, using this acceleration and maximum speed. As can be seen, the average robot speed stays roughly between 1.3 and 1.4 m/s, even as the length-width ratio changes.

Table 2.B.1: Average robot speed for various layouts, with acceleration 0.75 m/s^2 and maximum speed 1.5 m/s

# aisles	# cross-aisles	average robot speed
2	88	1.427
4	44	1.380
6	30	1.362
8	22	1.336
12	14	1.316
14	12	1.321
16	10	1.310
20	8	1.314
26	6	1.330
36	4	1.332
60	2	1.365

2.C Simultaneous Processing of Multi-line Orders

Table 2.C.1 shows the results for the standard layout when multi-line orders are processed sequentially or simultaneously. Sequential processing means that only one robot retrieves all the pods for all the order lines of an order and is indicated in Table 2.C.1 with "seq.". Simultaneous, indicated with "sim.", means an order can be processed in parallel and that multiple robots can fetch pods for the same order. The parameter p is the parameter for the geometric distribution of the number of order lines, i.e. $p = 0.2$ means that orders have on average 5 order lines. Sequential processing can be modeled analytically by models M_3 and M_4, but for simultaneous processing there is no analytical model, therefore the results were generated by simulation. The results indicate that robot utilization is higher and order cycle time lower with simultaneous processing. However, the workstation utilization remains almost the same under both forms of processing.

Table 2.C.1: experiment with sequential and simultaneous processing of multi-line orders

		no zones			zones		
	$\lambda(h^{-1})$	$\rho_r(\%)$	$t_{oc}(s)$	$\rho_{ws}(\%)$	$\rho_r(\%)$	$t_{oc}(s)$	$\rho_{ws}(\%)$
seq. $R = 14, p = 0.5$	110.88	0.914	838.370	0.923	0.825	595.950	0.923
sim. $R = 14, p = 0.5$	110.88	0.933	787.952	0.927	0.864	502.530	0.923
seq. $R = 14, p = 0.2$	44.35	0.910	2006.742	0.921	0.821	1420.379	0.923
sim. $R = 14, p = 0.2$	44.35	0.935	1564.767	0.918	0.895	1079.375	0.922
seq. $R = 14, p = 0.5$	50.40	0.314	313.167	0.421	0.241	242.153	0.418
sim. $R = 14, p = 0.5$	50.40	0.340	183.844	0.421	0.274	147.588	0.421
seq. $R = 14, p = 0.2$	20.16	0.314	779.882	0.421	0.244	608.381	0.423
sim. $R = 14, p = 0.2$	20.16	0.386	261.163	0.423	0.338	222.834	0.426

3 Inventory Allocation

3.1 Introduction

An RMFS is flexible in operations. Pods do not need to have a fixed position in the storage area, but can instead be repositioned continually throughout the day, see also Wurman & Enright (2011). Inventory can thus be positioned close to the workstations as needed. In addition, a product can be stored across multiple pods that can be positioned independently from each other. The travel times of the pods decrease if the inventory of a product is spread across multiple pods, because it becomes more likely that a suitable pod is located close to a workstation, and because different pods can be located close to different workstations. If the inventory on a pod drops below a threshold level, that pod is transported to a replenishment station to be fully replenished. In an e-commerce warehouse, one of the main performance metrics is the order throughput time, which improves if the travel times decrease. An interesting, unresolved aspect in the design of an RMFS is the optimal number of pods over which a product's inventory should be spread to minimize order throughput time. Availability of products is another benefit of spreading inventory across multiple pods. If all inventory of a product is allocated to one pod, there is a risk of temporary unavailability of that product for picking when that pod is waiting for replenishment, or when it is in use for another order. Since in e-commerce environments most orders are single-line, temporary unavailability of that pod directly delays the orders for that SKU. If inventory is spread across multiple pods, it becomes less likely that an order for many units can be fulfilled with inventory from a single pod.

In addition, if inventory is allocated across multiple pods, replenishment happens more frequently because the inventory per product on a pod will drop below the replenishment level sooner. This is inefficient as more trips are needed to replenish the same number of units. In both cases, orders for that product are delayed and order throughput time increases. In practice we observe that while some companies divide the inventory of a product over many locations (Amazon for example), others choose to keep products closer together (Timberland for example).

The impact of the trade-off also depends on the replenishment level. A higher replenishment level per product on a pod means that replenishment happens more frequently and may therefore cause additional robot travel time and additional queueing at the workstations. However, it also means that the average inventory on a pod is higher, causing the larger sized orders to wait less. The queueing at the workstations is also influenced by the ratio of the number of pick stations to replenishment stations. A higher replenishment level does not necessarily lead to more queueing if the number of replenishment stations is also higher. If the number of pods per SKU and the replenishment level are not jointly optimized, long and unnecessary delays may occur that can have a large impact on order throughput time. If the ratio of the number of pick stations to replenishment stations is not optimized, pick stations may have unacceptably low utilization while too much queueing occurs at the replenishment stations, or vice versa.

In view of these trade-offs, this chapter studies how to minimize the order throughput time by optimizing three decision variables: (1) the number of pods per product, (2) the ratio of the number of pick stations to the number of replenishment stations, and (3) the replenishment level per pod. We develop a novel queueing approach to jointly analyze the effect of these three variables. The inventory control policies are embedded in the queueing model's state transitions. The analytical model is developed to gain more insights into the system and to solve small instances, but unfortunately it cannot solve large, realistic instances, which are analyzed using simulation instead. Section 3.2 discusses the literature and motivates why a queueing model is suitable

for analyzing these decision variables. To analyze the decisions, Section 3.3 develops a new type of queueing network, the cross-class matching multi-class Semi-Open Queueing Network (SOQN) and shows several counter-intuitive necessary stability conditions. Section 3.4 provides the results and insights. Section 3.5 concludes and presents a future outlook.

3.2 Literature

Some key features of the system are that robot travel times are stochastic, that queueing at the workstations comprises a substantial part of the order throughput time and that suitable pods have to be retrieved that can fulfill the orders. This implies that traditional inventory models such as (r, Q) or (s, S) policies cannot be used, as these cannot model some or all of these characteristics and their influence on the order throughput time. On the other hand, queueing networks have been used extensively for analyzing the performance of autonomous vehicle storage and retrieval systems (AVS/RS) and automated storage and retrieval systems (AS/RS). These networks can optimize key decision variables, because the low computation time allows evaluation of a large set of parameters. For example, Kuo et al. (2007) use queueing models to predict vehicle utilization and service, waiting and cycle times while varying the number of aisles, storage columns per aisle, storage tiers in the system, vehicles in the system, and the number of lifts in AVS/RS. Fukunari & Malmborg (2009) estimate the expected utilization of resources in an AVS/RS machine using a queueing model that incorporates both single and dual command cycles. Queueing networks can also incorporate the stochasticity of vehicle traveling and worker speed and can capture the resulting congestion effects (see Marchet et al. (2013), Roy et al. (2013), Roy et al. (2015a), Roy et al. (2015b), Roy et al. (2016) and Tappia et al. (2016)). Networks where orders arrive and depart from the system can be divided into two broad categories: Open Queueing Networks (OQN) (Heragu et al. (2011)) and Semi-Open Queueing Networks (SOQN). SOQNs can

capture the matching of different kinds of resources and can therefore include the time an order has to wait before being matched with a vehicle. This could be used to model the matching of orders and pods. For example, Roy et al. (2012) use a multi-class semi-open queueing network to analyze the performance impact of system parameters such as the number of vehicles and lifts, the depth-to-width ratio, and the number of zones. They also study the impact of operational decisions such as vehicle assignment rules on vehicle utilization and order cycle time. A disadvantage of SOQNs is the absence of product form solutions and exact solutions, only approximations exist. Ekren et al. (2014) apply the matrix-geometric method to analyze a SOQN for an AVS/RS, which results in quite accurate approximations for the metrics. Roy et al. (2012) develop a new decomposition technique to evaluate the system.

Lamballais et al. (2017) and Nigam et al. (2014) develop SOQN and CQN queueing networks to estimate the performance of picking operations in an RMFS. Lamballais et al. (2017) optimize the layout of an RMFS warehouse by estimating the expected order cycle time, workstation utilization, and robot utilization for a given layout and by determining the optimal dimensioning of the storage area and the optimal placement of the workstations. They do not consider replenishment but only look at pick operations.

However, a disadvantage of these SOQN models is that only orders and pods of the same type can be matched. In our system, if pods and orders are matched there may be some asymmetry: an order that requires a certain number of units can be fulfilled with a pod that contains that number of units *or more*. In addition, the optimal spreading of the inventory of a product across storage shelves has not been researched yet.

This chapter studies how inventory should be spread across pods, how the ratio of pick stations to replenishment stations should be set, and what the replenishment level should be, such that the order throughput time is minimized.

3.3 Model

We wish to determine the optimal number of pods over which the inventory of a product should be spread, the optimal ratio of the number of pick stations to replenishment stations, and the optimal replenishment level of a pod. We construct a queueing network using two main ideas. The first idea is to use the various movements of a pod as queueing stations within the network. The second idea is to capture the number of inventory units of a product per pod as classes within the queueing network. The class of a pod signifies how many units of the product are left on the pod, and the class of an order indicates how many units the order requires. These two ideas are explained in detail in the sections below. In modeling the queueing network, we make the following assumptions to keep the model tractable. (1) Orders are assumed to be single-line orders. This is not too strong a simplification as RMFSs were built especially for e-commerce order fulfillment. An order can still require multiple units of a SKU. In our context, a small order thus requires few units of a SKU and a large order requires many units of a SKU. (2) Orders arrive according to a Poisson process. (3) Pods are dedicated to a single SKU. i.e., a pod can carry only one SKU at a time and is always replenished with the same SKU. The inventory of a SKU can be spread over multiple pods. (4) The maximum number of units in inventory per product is fixed, as is the number of workstations. (5) The pick or replenishment station at which a pod will be handled is chosen randomly. (6) Multiple pick stations can be modeled as one queueing station with multiple servers and similarly for multiple replenishment stations. (7) Pods are stored at any of the storage locations with equal probabilities; no distinction is made between fast and slow moving SKUs. (8) If multiple, equally suitable pods are available for order picking, the one that is nearest to the workstation will be fetched. Note that, if the inventory of a product is spread across more pods, travel times will become shorter on average, because it becomes more likely that a suitable pod is close to a pick station. This is elaborated in section 3.3.2. (9) The number of robots is not a constraint and does not

delay any of the processes. (10) An order has to be matched with exactly one pod and "order splitting" is not allowed. This implies that if each of the pods carrying a certain SKU have some, but insufficient, units for an order, the order cannot be matched and needs to wait until one of the pods is replenished. (11) A pod can only be matched with one order at a time and "order merging" is not allowed. This implies that if there are two orders that each require one unit and there is a single pod with two units, the pod will be matched with the orders sequentially.

3.3.1 Pod Movement

From the perspective of a pod, the following eight processes occur, see Figures 3.3.1 and 3.3.2. The pod (1) waits to be matched with an order, (2) waits for a robot to come to its storage location, (3) moves to the pick station, (4) queues for its turn at the pick stations and then has items picked from it. If its inventory is not below the replenishment level, (5) the pod returns to the storage area, otherwise it (6) moves to a replenishment station, (7) queues for its turn and is replenished and (8) returns to the storage area.
Each of these processes can be modeled as a queue, where the distribution of the travel times in a situation becomes the distribution of the service time of the corresponding queue. The pick stations are modeled as the servers of the queueing station corresponding to process (4) and the replenishment stations are modeled as the servers of the queueing station corresponding to process (7). The queueing network, which is shown in Figure 3.3.2, is a Semi-Open Queueing Network that captures the matching of an order to a pod. The numbers in Figure 3.3.1 and Figure 3.3.2 show which process corresponds with which queue and Table 3.3.1 shows the modeling approach for each of the decision variables. Here S is the number of SKUs, M^s is the number of pods of SKU s, r is the ratio of the number of pick stations to replenishment stations and ξ is the replenishment level, which is expressed as a percentage of the capacity of a pod, U.

Figure 3.3.1: Illustration of pod movement

Table 3.3.1: Modeling approach for each of the decision variables

Symbol	Decision variable	Modeling approach
$\sum_{s=1}^{S} M^s$	Total number of pods	Number of "tokens" circulating in the SOQN (see Buitenhek et al. (2000))
r	Ratio of the number of pick stations to replenishment stations	Ratio of the number of servers at the pick queue to the number of servers at the replenishment queue
ξ	Replenishment level	Pod classes in the SOQN

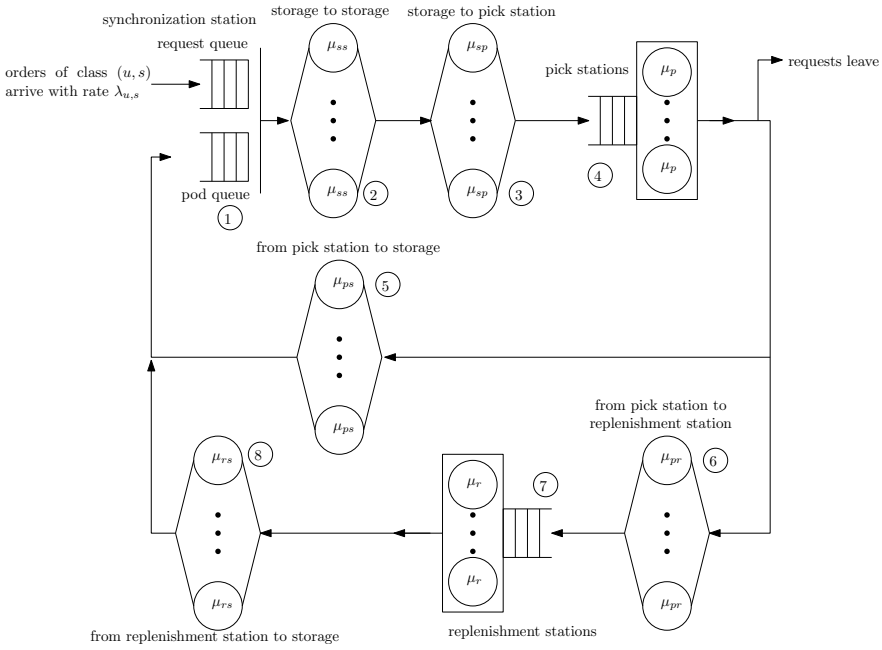


Figure 3.3.2: Queueing model of pod movements in the RMFS

3.3.2 Travel Times and Service Times

For steady state performance analysis, the process in Figure 3.3.1 repeats indefinitely and each of these eight processes can be translated one-to-one to a queue, as shown in Figure 3.3.2. The handling times at the pick and replenishment stations are assumed to follow an exponential distribution. The number of servers at queue (4) equals the number of pick stations in the RMFS and the number of servers at queue (7) equals the number of replenishment stations. The travel times are modeled as Infinite Server (IS) stations, since pods can start traveling immediately and do not have to wait for other pods to finish. At these IS stations, the average service time corresponds to the average time needed for traveling. The aisles have a single travel direction to prevent deadlock and reduce aisle congestion. Unloaded robots can move underneath the pods and do not need to use the aisles. Movement through an aisle with a pod in the area between the workstations and storage area is also single directional.

This means that the travel distance between any two locations can be calculated exactly (see also Lamballais et al. (2017)). We make the following assumptions when calculating the travel times: (1) Robot velocity is constant. This can be an average velocity that takes acceleration and deceleration into account (see also Lamballais et al. (2017)). (2) Delays due to robot congestion or blocking in aisles, battery recharges, or robot downtime do not occur.

As mentioned earlier, we assume that storage and retrieval occur at a random location. For a single pod, the distribution of the travel time can be created by calculating the travel time from the current location to every possible destination and giving these equal probabilities of occuring. These probabilities can be adjusted to suit the application, for example, to include storage zones for products with different pick frequencies (see Lamballais et al. (2017)).

For multiple pods, the travel time distribution can be created as follows. Suppose that the cumulative distribution function (cdf) of the travel times

of a single pod is given by $F(x)$. The travel times decrease if the inventory of a product is spread across multiple pods, because the pod nearest to a workstation can be chosen. The cdf of the travel times of the nearest pod of SKU s is given by $\tilde{F}(x) = 1 - [1 - F(x)]^{M^s}$. For any IS station modeling travel times, the service time distribution has the cdf $\tilde{F}(x)$ corresponding to those travel times.

3.3.3 Inventory Levels

If every order line requires only one unit of a product, the queueing model can be solved using the methods in Buitenhek et al. (2000) and Bolch et al. (2006). If an order line requires more than one unit, the behavior of the queueing network becomes more complicated, because a pod with u remaining units cannot fulfill an order line that needs j units if $j > u$. It is therefore necessary to keep track of the number of units per product on each pod and the number of units required by each order. The queueing network achieves this by modeling the number of units remaining on a pod as the class of that pod and the number of units an order requires as the class of that order. In other words, a pod dedicated to SKU s with u units remaining has class (u, s). Since pods are dedicated to SKUs, the SKU class s of a pod cannot change. Similarly, orders for a product s that require u units have class (u, s) and arrive with a rate $\lambda_{u,s}$. Since orders are single-line orders, s simply denotes the product of the order. For a class (u, s), index u refers to the "inventory class", and index s refers to the "SKU class".
The replenishment level, denoted by ξ, is a percentage between 0% and 100%, the maximum number of units on a pod is denoted by U, so $0 \leq u \leq U$, and the replenishment point is ξU. Consider a pod that leaves a pick station. If the pod has inventory class $j, j > \xi U$, it returns to the storage area, but if it has inventory class $j \leq \xi U$, it moves to a replenishment station. In other words, the routing depends on the inventory class of the pod, and the routing probability to go from process (4) to process (6) in Figure 3.3.2 is either 0 or 1.

At the synchronization station (see process (1) in Figure 3.3.2), a pod of class (j, s) can be matched with an order of class (i, s) if $i \leq j$. We call this kind of matching "cross-class" matching. Due to this cross-class matching, both orders and pods with the same SKU class may be waiting at the synchronization station simultaneously. This happens if $j < i$ for all pods and all orders. If multiple orders are available, the pod is matched with the order of the highest suitable class available. In other words, the pod is matched with the order that requires the most units, but no more units than the current pod inventory. At the pick station, the pod's inventory class is lowered after picking. If the pod belonged to class (j, s) and was matched with an order of class (i, s), then the pod's class after picking changes to class $(j - i, s)$, because $j - i$ is the number of units left after picking. Figure 3.3.3 illustrates these class switches for a situation where the maximum inventory level is six units and the replenishment level is zero.

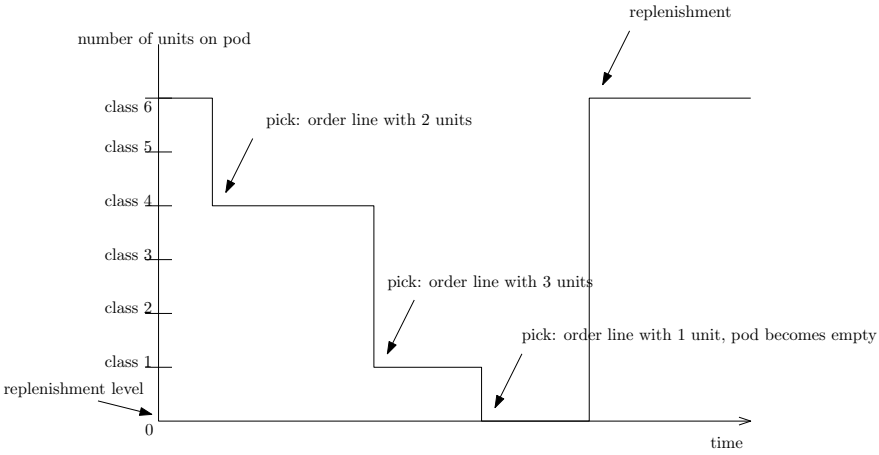


Figure 3.3.3: Illustration of a pod's class switching

Since pods are dedicated to a single SKU, a switch from a class (u, s) to a class (u', s') is not allowed for any $s \neq s'$.

3.3.4 Creating the Compact Queueing Model

SOQN models do not have a product form, which means that exact solutions do not exist. Instead, SOQN models are solved using good approximation methods (Buitenhek et al. (2000), Jia (2005)). However, existing methods for analyzing multi-class SOQN models cannot be applied to the model in Figure 3.3.2, because these methods cannot analyze a network that uses cross-class matching. Therefore, we construct a novel Continuous Time Markov Chain (CTMC) to analyze the multi-class SOQN with cross-class matching. Since the size of the state space of the Markov Chain corresponding to the queueing network in Figure 3.3.2 grows rapidly in the number of pods and classes, a (nearly) equivalent, compact queueing network is created with its state space growing less rapidly. The Markov Chain of this compact queueing network can then be used to analyze the system. To transform the queueing network in Figure 3.3.2 to the compact queueing network requires an additional, intermediate queueing network, which is shown in Figure 3.3.4.

The transformation works as follows. In the queueing network in Figure 3.3.2, no queueing occurs at the IS stations that model travel. Interaction of the robots via queueing only occurs at the pick and replenishment stations, and at the synchronization queue. The network can therefore be reduced to three queues, a synchronization queue, a pick queue, and a replenishment queue. This reduction is done by grouping the IS stations and the pick and replenishment queues into two groups as shown in Figure 3.3.4. Each group can then be transformed into an equivalent load-dependent queue using Norton's theorem (Bolch et al. (2006)). This results in the compact queuing network shown in Figure 3.3.5. To create the two groups, the IS station in Figure 3.3.2, process 5, needs to be changed because it cannot readily be incorporated into a group with the pick stations or into a group with the replenishment stations. However, in a layout where the pick stations and replenishment stations are next to each other, at the same side of the storage area, processes 5 and 8 modeling travel from pick stations to storage and travel from replenishment stations to storage will have about identical service

time distributions. Process 5 can be directly after the pick stations, so in other words, even if the pod goes for replenishment, it first has to visit this IS station. However, by omitting process 8 in the other group of queues, the network consisting of two groups of queues still contains the same underlying processes. This results in two groups of queues as shown in Figure 3.3.4.
In the compact queueing model in Figure 3.3.5, the service rate at the picking queue with i pods in the queue is $\tilde{\mu}_p(i)$ and the service rate at the replenishment queue with i pods is $\tilde{\mu}_r(i)$. As the number of pick stations is equal to or larger than one, $\tilde{\mu}_p(1) \leq \tilde{\mu}_p(i)$ for $i > 1$. A similar statement holds for $\tilde{\mu}_r(i)$. To further reduce the size of the state space, we change the class switching mechanism. Class switching of the pods happens before rather than after the picking queue. This means that the information about which class of orders was matched with each pod does not have to be included in the state space of the Markov Chain. This reduces the state space without affecting the results that can be obtained from the queueing network.

3.3.5 Model States and Transitions

This section explains the states and transitions of the compact queueing model in Figure 3.3.5. The total number of inventory classes per SKU is $U + 1$, since a pod can also be empty (inventory class zero). The state space of the Markov Chain for the compact queueing network in Figure 3.3.5 consists of four parts: (1) A "waiting orders part", denoted by $\boldsymbol{O}$, which is a vector that contains the number of orders of each class that are waiting at the synchronization station. Since orders for zero units cannot arrive, the dimension of $\boldsymbol{O}$ equals $U \times S$. (2) A "waiting pods part", denoted by $\boldsymbol{Q}$, which is a vector that contains the number of pods of each class that are waiting at the synchronization station. Since pods with equal or fewer units than replenishment level $\lfloor \xi U \rfloor$ go for replenishment and not to the synchronization station, the dimension of $\boldsymbol{Q}$ equals $(U - \lfloor \xi U \rfloor) \times S$. (3) A "picking part", denoted by $\boldsymbol{P}$, that contains the number of pods of each class that are waiting to be picked. As discussed earlier, to reduce the size of the

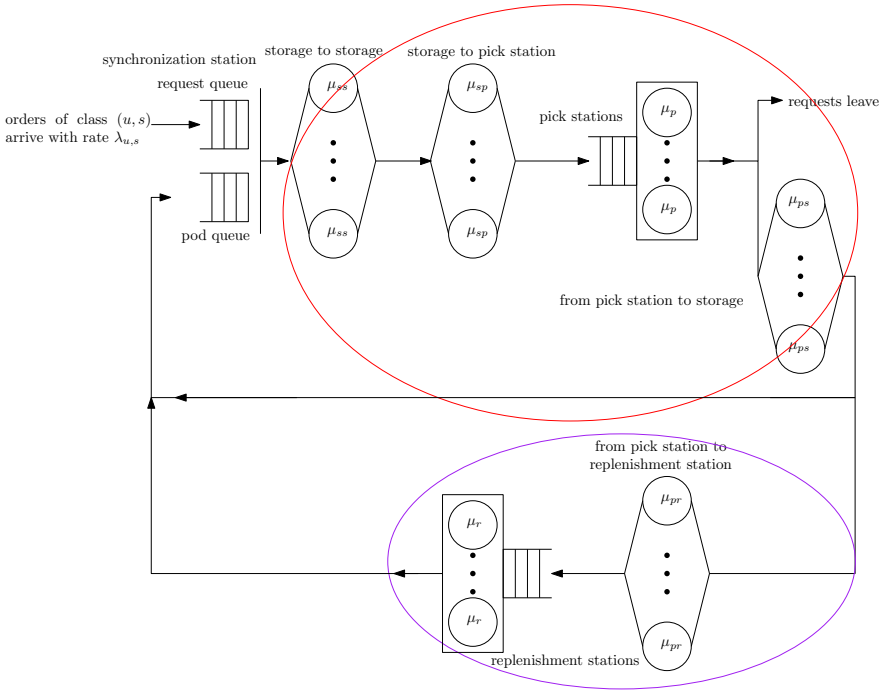


Figure 3.3.4: Intermediate queueing model

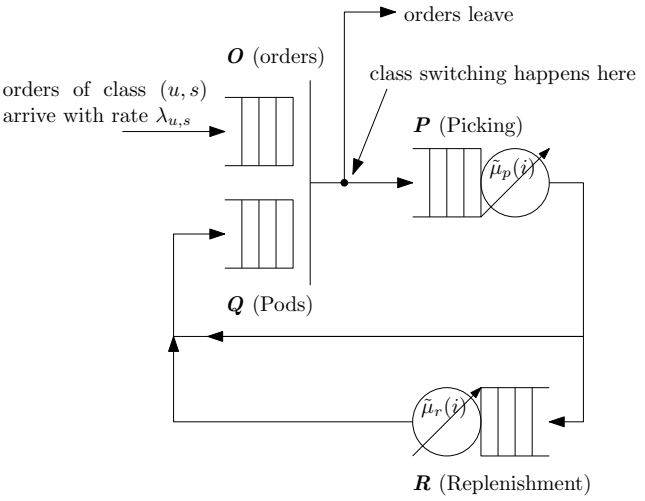


Figure 3.3.5: The compact model

state space, class switching happens before this queue and hence the order leaves the system before this queue. This reduction trick does not change the average order throughput time, since the average service time for the pods at the picking queue is known and is the same as it would be for the orders. This also means that the dimension of $\boldsymbol{P}$ equals $U \times S$ rather than $(U+1) \times S$, because the number of units of the order have already been subtracted from the class of the pod. Hence a pod at the pick queue cannot be full. Switching class before the pick queue avoids including information about the orders and order assignment to pods in the Markov Chain, which strongly reduces the size of the Markov Chain. (4) A "replenishment part", denoted by vector $\boldsymbol{R}$, which contains the number of pods of each class that are waiting to be replenished at the replenishment queue. Since only pods with $\lfloor \xi U \rfloor$ units or less go for replenishment, including pods that are empty, the dimension of $\boldsymbol{R}$ equals $(\lfloor \xi U \rfloor + 1) \times S$. A state ψ can hence be described as $\psi = (\boldsymbol{O}, \boldsymbol{Q}, \boldsymbol{P}, \boldsymbol{R})$.

Let $\boldsymbol{O}^{\psi}_{u,s}$ denote the number of orders of class (u,s) waiting at the synchronization station in state ψ, $\boldsymbol{Q}^{\psi}_{u,s}$ the number of pods of class (u,s) waiting at

the synchronization station in state ψ, $\boldsymbol{P}^{\psi}_{u,s}$ the number of pods of class (u,s) at the pick queue in state ψ, and $\boldsymbol{R}^{\psi}_{u,s}$ the number of pods of class (u,s) at the replenishment queue in state ψ. To limit the number of states, the total number of orders allowed to wait at the synchronization station is limited to K, in the sense that for any state ψ it must hold that $\sum_{u,s} \boldsymbol{O}^{\psi}_{u,s} \leq K$. In a stable system, the number of orders waiting at the synchronization station should not grow too large. Therefore, as long as K is not too small, its effect on the performance metrics is negligible. The size of the state space is also limited by the number of pods per SKU, denoted by M^s, in the sense that $\sum_u \left(\boldsymbol{Q}^{\psi}_{u,s} + \boldsymbol{P}^{\psi}_{u,s} + \boldsymbol{R}^{\psi}_{u,s} \right) = M^s, \forall s, \psi$. Appendix 3.A shows how to calculate the size of the state space for various parameters.

Three categories of transitions exist: (1) The arrival of an order, which either (1a) matches with a pod, (1b) queues at the synchronization station, or (1c) is rejected from the system. (2) The completion of picking a pod, after which either (2a) the pod matches with an order, (2b) the pod queues at the synchronization station, or (2c) the pod goes for replenishment. (3) The completion of replenishing a pod, after which either (3a) the pod matches with an order, or (3b) the pod queues at the synchronization station. In other words, for each category of transition, the entity involved (an order or a pod) either matches or does not match with its complement (pod or order). In addition to this, the transitions also include order rejection (1c) and replenishment (2c). Order rejection occurs when the arriving order cannot be matched with a pod and K orders are already queueing at the synchronization station.

If the inventory class of a pod falls at or below $\lfloor \xi U \rfloor$ after picking, the pod goes to the replenishment station to be replenished. After replenishment, the pod has class (U,s) so it can match with any waiting order for SKU s. Table 3.3.2 shows for each transition: the transition rate, the requirements that the state must meet for the transition to be possible, and how the state changes due to the transition. No information about the sequence of arrivals is stored in the state, so at the synchronization, pick and replenishment queue the queueing discipline is Random rather than FCFS. The number of states

is finite and all the transitions are known, a generator matrix $\boldsymbol{\Gamma}$ and the state probabilities $\boldsymbol{\pi}$ can be calculated via $\boldsymbol{\pi}\boldsymbol{\Gamma} = 0$ under the condition that $\boldsymbol{\pi}\mathbf{1} = 1$ (Bolch et al. (2006), page 69). Once the state probabilities have been calculated, they can be used to derive any metric of interest for the system.

Table 3.3.2: Transitions from a state ψ to a state σ

No	Transition rate	Requirements for transition of current state ψ to next state σ	Change of current state ψ to next state σ after transition
1a	$\sum_{u,s} \lambda_{u,s}$	$\exists j \vert j \geq u,\, \boldsymbol{Q}^{\psi}_{j,s} > 0$	$h = \underset{j}{\operatorname{argmin}}\{j \geq u,\, \boldsymbol{Q}^{\psi}_{j,s} > 0\}$ $\boldsymbol{Q}^{\sigma}_{h,s} \coloneqq \boldsymbol{Q}^{\psi}_{h,s} - 1,\, \boldsymbol{P}^{\sigma}_{h-u,s} \coloneqq \boldsymbol{P}^{\psi}_{h-u,s} + 1$
1b	$\sum_{u,s} \lambda_{u,s}$	$\nexists j \vert j \geq u,\, \boldsymbol{Q}^{\psi}_{j,s} > 0$ $\sum_{u,s} \boldsymbol{O}^{\psi}_{u,s} < K$	$\boldsymbol{O}^{\sigma}_{u,s} \coloneqq \boldsymbol{O}^{\psi}_{u,s} + 1$
1c	$\sum_{u,s} \lambda_{u,s}$	$\nexists j \vert j \geq u,\, \boldsymbol{Q}^{\psi}_{j,s} > 0$ $\sum_{u} \boldsymbol{O}^{\psi}_{u,s} = K$	State remains the same, $\sigma = \psi$
2a	$\tilde{\mu}_p(\sum_{u,s} \boldsymbol{P}^{\psi}_{u,s})$	$\boldsymbol{P}^{\psi}_{u,s} > 0,\, u > \xi U$ $\exists j \vert u \leq j,\, \boldsymbol{O}^{\psi}_{j,s} > 0$	$h = \underset{j}{\operatorname{argmax}}\{j \leq u,\, \boldsymbol{O}^{\psi}_{j,s} > 0\}$ $\boldsymbol{O}^{\sigma}_{h,s} \coloneqq \boldsymbol{O}^{\psi}_{h,s} - 1,\, \boldsymbol{P}^{\sigma}_{u,s} \coloneqq \boldsymbol{P}^{\psi}_{u,s} - 1$ $\boldsymbol{P}^{\sigma}_{u-h,s} \coloneqq \boldsymbol{P}^{\psi}_{u-h,s} + 1$
2b	$\tilde{\mu}_p(\sum_{u,s} \boldsymbol{P}^{\psi}_{u,s})$	$\boldsymbol{P}^{\psi}_{u,s} > 0,\, u > \xi U$ $\nexists j \vert u \leq j,\, \boldsymbol{O}^{\psi}_{j,s} > 0$	$\boldsymbol{P}^{\sigma}_{u,s} \coloneqq \boldsymbol{P}^{\psi}_{u,s} - 1$ $\boldsymbol{Q}^{\sigma}_{u,s} \coloneqq \boldsymbol{Q}^{\psi}_{u,s} + 1$
2c	$\tilde{\mu}_p(\sum_{u,s} \boldsymbol{P}^{\psi}_{u,s})$	$\boldsymbol{P}^{\psi}_{u,s} > 0,\, u \leq \xi U$	$\boldsymbol{P}^{\sigma}_{u,s} \coloneqq \boldsymbol{P}^{\psi}_{u,s} - 1,\, \boldsymbol{R}^{\sigma}_{u,s} \coloneqq \boldsymbol{R}^{\psi}_{u,s} + 1$
3a	$\tilde{\mu}_r(\sum_{u,s} \boldsymbol{R}^{\psi}_{u,s})$	$\boldsymbol{R}^{\psi}_{u,s} > 0$ $\exists j \vert \boldsymbol{O}^{\psi}_{j,s} > 0$	$h = \underset{j}{\operatorname{argmax}}\{\boldsymbol{O}^{\psi}_{j,s} > 0\}$ $\boldsymbol{O}^{\sigma}_{h,s} \coloneqq \boldsymbol{O}^{\psi}_{h,s} - 1,\, \boldsymbol{R}^{\sigma}_{u,s} \coloneqq \boldsymbol{R}^{\psi}_{u,s} - 1$ $\boldsymbol{P}^{\sigma}_{U-h,s} \coloneqq \boldsymbol{P}^{\psi}_{U-h,s} + 1$
3b	$\tilde{\mu}_r(\sum_{u,s} \boldsymbol{R}^{\psi}_{u,s})$	$\boldsymbol{R}^{\psi}_{u,s} > 0$ $\nexists j \vert \boldsymbol{O}^{\psi}_{j,s} > 0$	$\boldsymbol{R}^{\sigma}_{u,s} \coloneqq \boldsymbol{R}^{\psi}_{u,s} - 1,\, \boldsymbol{Q}^{\sigma}_{U,s} \coloneqq \boldsymbol{Q}^{\psi}_{U,s} + 1$

As an example, suppose that there is just one SKU, so $s = 1$, and that $U = 2$ and $\xi = 0$, then $\boldsymbol{O} = (\boldsymbol{O}_{1,1}, \boldsymbol{O}_{2,1})$, $\boldsymbol{Q} = (\boldsymbol{Q}_{1,1}, \boldsymbol{Q}_{2,1})$, $\boldsymbol{P} = (\boldsymbol{P}_{0,1}, \boldsymbol{P}_{1,1})$ and $\boldsymbol{R} = (\boldsymbol{R}_{0,1})$. Suppose furthermore that there are two pods in the system. One is waiting at the pick queue with one unit left, the other is waiting at the replenishment queue and an order is waiting for one unit at the synchronization station. The state would then be $\psi = ((1,0), (0,0), (0,1), (1))$ and Figure 3.3.6 shows the transitions into and out of this state. In Figure 3.3.6, $\lambda_{1,1}$ is the arrival rate of orders requiring just one unit, $\lambda_{2,1}$ is the arrival rate of orders requiring two units, $\tilde{\mu}_p(1)$ is the service rate at the pick station

with one pod, $\tilde{\mu}_p(2)$ is the service rate with two pods, $\tilde{\mu}_r(1)$ is the service rate at the replenishment queue with one pod, and $\tilde{\mu}_r(2)$ is the service rate with two pods.

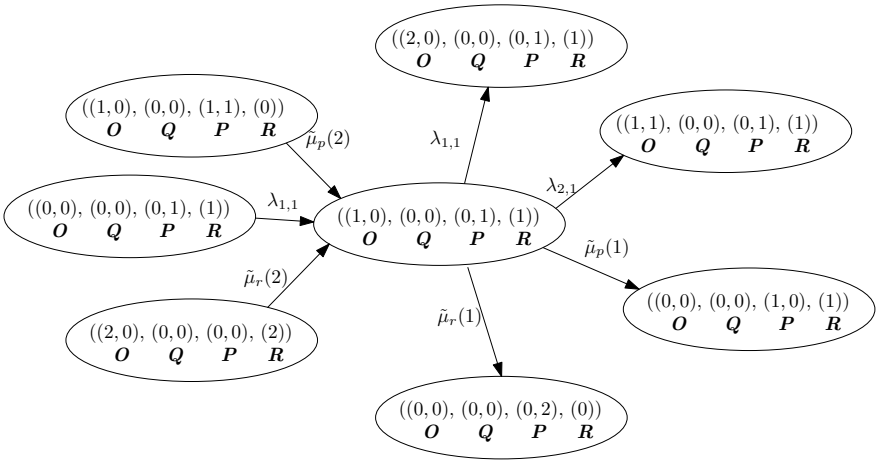


Figure 3.3.6: Example of transitions for $U = 2$ and $\xi = 0$

3.3.6 Aggregate Classes

A disadvantage of using classes to indicate the number of units is that the number of classes can grow large quickly. To mitigate this disadvantage, "aggregate classes" can be constructed for each SKU s. An aggregate class contains a range of m inventory classes of SKU s. An example would be to create three aggregate classes for each SKU, where aggregate class number 1 is labeled "low", number 2 is labeled "medium" and number 3 is labeled "high". If $U = 29$, and $m = 10$, then aggregate class low would contain inventory classes 0 to 9, aggregate class medium would contain inventory classes 10 to 19, and aggregate class high would contain inventory classes 20 to 29. So a pod with 14 units would belong to aggregate class medium. The main difference that comes with using aggregate classes is that the way in which a pod changes class is no longer deterministic, but stochastic. As

described earlier, when using inventory classes, the inventory class of the pod after picking equals the inventory class of the pod before picking minus the inventory class of the order that the pod is matched with, so in other words the class changes of the pod are completely deterministic. This is not the case when using aggregate classes. For example, if a pod with 14 units is matched with an order for 2 units, then the pod has 12 units left after picking. When using inventory classes this means a class change from class 14 to class 12, but when using aggregate classes the pod retains its aggregate class medium. In other words, a pod of aggregate class medium matched with an order of aggregate class low and after picking the pod had aggregate class medium. However, had the order required 6 units, the pod would have been left with 8 units after picking and would have changed from aggregate class medium to aggregate class low. In other words, a pod of aggregate class medium matched with an order of aggregate class low and after picking the pod had aggregate class low. As this example shows, when using aggregate classes the way in which a pod changes class is probabilistic rather than fixed.

The main assumptions for adopting aggregate classes are the following. (1) Both pods and orders have aggregate classes, i.e., it is not possible that pods are designated with aggregate classes, whereas orders are designated with inventory classes. (2) When aggregate classes are used, the exact number of units needed by orders and left on pods is not tracked. (3) Every aggregate class contains the same number of m inventory classes. (4) The replenishment level is an inventory class and determines the probability of going for replenishment. In the example, if the replenishment level is 12, a pod of aggregate class low goes for replenishment with probability 1, and a pod of class medium goes for replenishment with probability 0.3. (5) Orders match with a pod of the same or higher aggregate class. If an order matches with a pod of the same aggregate class, we assume that the number of units that the order actually requires is less than or equal to the number of units that are actually left on the pod.

With inventory classes, if a pod of class a matches with an order of class

b, the class of the pod would be $a - b$ after picking. As it turns out, under the assumptions above a similar rule holds for aggregate classes, shown in Theorem 1.

Theorem 1. *If a pod of aggregate class number α, with $\alpha \geq 1$, matches with an order of aggregate class number β, with $1 \leq \beta \leq \alpha$, then with a probability p the pod will have aggregate class number $\alpha - \beta + 1$ after picking and with probability $1 - p$ the pod will have aggregate class number $\alpha - \beta$, where for $\alpha > \beta$ it holds that $p \to 0.5$ as $m \to \infty$ and for $\alpha = \beta$ it holds that $p \to 1$ as $m \to \infty$.*

Proof. Let an aggregate class number δ contain the inventory classes in the range $[\underline{n}_\delta, \overline{n}_\delta]$, $\underline{n}_\delta \equiv (\delta - 1)m$ and $\overline{n}_\delta \equiv \delta m - 1$, so $m = \overline{n}_\delta - \overline{n}_{\delta-1} = \overline{n}_\delta - (\underline{n}_\delta - 1) = \overline{n}_\delta - \underline{n}_\delta + 1$. For an inventory class d, $d \in [\underline{n}_\delta, \overline{n}_\delta]$, define $\hat{d}$ as $\hat{d} \equiv d - \underline{n}_\delta$, which implies that $0 \leq \hat{d} < m$. A pod of aggregate class number α will, in reality, have a units remaining, $a \in [\underline{n}_\alpha, \overline{n}_\alpha]$, and an order of aggregate class number β will, in reality, require b units, $b \in [\underline{n}_\beta, \overline{n}_\beta]$. After picking, the pod will have $a - b$ units left. Subsequently, equations (3.1) to (3.4) hold.

$$a - b = \hat{a} + \underline{n}_\alpha - \hat{b} - \underline{n}_\beta \tag{3.1}$$

$$= \hat{a} + (\alpha - 1)m - \hat{b} - (\beta - 1)m \tag{3.2}$$

$$= \hat{a} - \hat{b} + ((\alpha - \beta + 1) - 1)m \tag{3.3}$$

$$= \hat{a} - \hat{b} + \underline{n}_{(\alpha - \beta + 1)} \tag{3.4}$$

From equation (3.4) it can be seen that if $\hat{a} - \hat{b} < 0$, the pod will be in aggregate class number $\alpha - \beta$ as the number of its units falls below the lower boundary of class $\alpha - \beta + 1$. However, if $\hat{a} - \hat{b} \geq 0$, the number of units on the pod will be at or above the lower boundary of aggregate class number $\alpha - \beta + 1$, so the pod will be in aggregate class number $\alpha - \beta + 1$. If $\alpha > \beta$, then the probability that $\hat{a} > \hat{b}$ equals the probability that $\hat{a} < \hat{b}$ and as $m \to \infty$ the probability that $\hat{a} = \hat{b}$ will go to zero, hence $p \to 0.5$. If $\alpha = \beta$, then as stated earlier, it is assumed that $\hat{a} \geq \hat{b}$ and hence $p \to 1$ as $m \to \infty$ as the probability that $\hat{a} = \hat{b}$ goes to zero. □

3.3.7 Simulation and Validation

As can be seen from Table 3.A.1 in Appendix 3.A, the number of states in the Markov Chain grows rapidly in the design parameters. Therefore we also develop a queueing network simulation, built in Java for analyzing large test cases. The network simulated is the compact queueing network depicted in Figure 3.3.5, where the warehouse layout in Figure 3.4.1 was used to create the travel times. To validate the results from the simulation, Table 3.3.3 shows the results for three instances. Each instance considers one SKU, one pick station, one replenishment station, and a replenishment level of zero for both the Markov Chain (MC) and the simulation (Sim), with ten runs each of ten weeks of working hours simulated, of which one third serves as the warm-up period. The order arrival rate equals 18 orders per hour per inventory class. The resulting confidence intervals were less than 1 percent of the averages and have therefore been omitted. For the first instance $U = 2$, $M^s = 2$, $K = 2$ resulting in 53 states in the MC, for the second instance $U = 3$, $M^s = 6$, $K = 2$ resulting in 2814 states in the MC, and for the third instance $U = 3$, $M^s = 6$, $K = 5$ resulting in 9324 states in the MC, see also Table 3.A.1. The pod utilization, denoted by

ρ_{pod}, is the percentage of time that a pod on average is being carried by a robot. ρ_{pick} denotes the utilization of the pick stations, ρ_{repl} the utilization of the replenishment stations, t_{ot} the average order throughput time, L_s the average number of orders in the system, L_o the average number of orders waiting at the synchronization station, and L_p the average number of pods waiting at the synchronization station. These measures can be calculated exactly from the Markov Chain as shown in Equations (3.5) to (3.11). Here $\boldsymbol{\pi}_\psi$ denotes the steady state probability to be in state ψ and $\mathbb{1}_{[x]}$ denotes the indicator function, which equals 1 if condition x is true and equals 0 is condition x is false. The order throughput time is calculated as the average number of orders in the system (L_s) divided by the total arrival rate. As no more than K orders can be present at the synchronization station, orders arriving at the system may be rejected, which means that the actual arrival rate will be lower than $\sum_{u,s} \lambda_{u,s}$. This is taken into account via the indicator function in Equation (3.11).

$$\rho_{\text{pod}} = \sum_{\psi} \left(\boldsymbol{\pi}_\psi \times \frac{\sum_{u,s} \boldsymbol{P}^{\psi}_{u,s} + \boldsymbol{R}^{\psi}_{u,s}}{\sum_s M^s} \right) \tag{3.5}$$

$$\rho_{\text{pick}} = \sum_{\psi} \boldsymbol{\pi}_\psi \mathbb{1}_{\left[\sum_{u,s} \boldsymbol{P}^{\psi}_{u,s} > 0\right]} \tag{3.6}$$

$$\rho_{\text{repl}} = \sum_{\psi} \boldsymbol{\pi}_\psi \mathbb{1}_{\left[\sum_{u,s} \boldsymbol{R}^{\psi}_{u,s} > 0\right]} \tag{3.7}$$

$$L_s = \sum_{\psi} \boldsymbol{\pi}_\psi \sum_{u,s} (\boldsymbol{O}^{\psi}_{u,s} + \boldsymbol{P}^{\psi}_{u,s}) \tag{3.8}$$

$$L_o = \sum_{\psi} \boldsymbol{\pi}_\psi \sum_{u,s} \boldsymbol{O}^{\psi}_{u,s} \tag{3.9}$$

$$L_p = \sum_{\psi} \boldsymbol{\pi}_\psi \sum_{u,s} \boldsymbol{Q}^{\psi}_{u,s} \tag{3.10}$$

$$t_{ot} = \frac{L_s}{\sum_{\psi} \boldsymbol{\pi}_\psi \mathbb{1}_{\left[\sum_{u,s} \boldsymbol{O}^{\psi}_{u,s} < K\right]} \sum_{u,s} \lambda_{u,s}} \tag{3.11}$$

Table 3.3.3 shows that the outcomes of the simulations agree with the results from the exact calculations based on the Markov Chain. Table 3.3.4 shows the validation for using the aggregate classes. The results all come from simulation and show two experiments. In the first experiment, each SKU is spread over two pods, whereas in the second experiment each SKU is spread over six pods. In both experiments, the simulations with aggregate classes, indicated with "Agg.", had four aggregate classes. For the simulation with normal inventory classes, case "Unagg." (a) has 20 inventory classes, case "Unagg." (b) 40 inventory classes and case "Unagg." (c) 80 inventory classes. As can be seen, the results for the cases with aggregate classes are close to the results for the cases with normal inventory classes. In other words, using the aggregate classes does not reduce the accuracy of the performance estimates while simultaneously complexity is greatly reduced since far fewer classes are needed to model the system.

Table 3.3.3: Validation of the simulation with the Markov Chain

	$\rho_{\text{pod}}(\%)$	$\rho_{\text{pick}}(\%)$	$\rho_{\text{repl}}(\%)$	$t_{ot}(s)$	L_s(# orders)	L_o(# orders)	L_p(# pods)
MC 1	74.1	70.3	46.1	225.6	1.736	0.795	0.517
Sim 1	74.1	70.0	45.9	226.7	1.744	0.805	0.519
MC 2	41.0	81.8	56.7	158.7	2.116	0.478	3.542
Sim 2	41.2	81.2	56.6	157.0	2.106	0.460	3.527
MC 3	42.6	82.9	58.2	238.0	3.291	1.592	3.443
Sim 3	42.8	82.2	58.0	233.4	3.237	1.535	3.434

Table 3.3.4: Validation of the use of aggregate class via simulation

	$\rho_{\text{pod}}(\%)$	$\rho_{\text{pick}}(\%)$	$\rho_{\text{repl}}(\%)$	$t_{ot}(s)$	L_s(# orders)	L_o(# orders)	L_p(# pods)
Agg.1	74.5	70.3	43.5	358.7	2.867	1.893	0.510
Unagg. 1a	75.2	70.8	44.1	379.6	3.037	2.058	0.496
Unagg. 1b	74.4	70.6	43.1	376.7	3.012	2.033	0.512
Unagg. 1c	74.1	70.6	42.8	370.9	2.961	1.983	0.518
Agg.2	24.9	62.0	40.1	122.6	0.982	0.002	4.506
Unagg. 2a	24.9	62.0	40.3	122.8	0.981	0.002	4.505
Unagg. 2b	24.7	62.1	39.8	122.3	0.981	0.002	4.516
Unagg. 2c	24.7	62.1	39.5	122.7	0.984	0.002	4.517

3.3.8 Necessary Stability Conditions

Finding order arrival rates for which the queueing network is stable is much more interesting for cross-class matching multi-class SOQNs than for other types of SOQNs, and leads to some surprising results. For example, suppose that a pod can carry a maximum of 6 units of a SKU, the replenishment level is zero, and orders only arrive for either 1 or 5 units. Then the system is unstable if the arrival rate for orders of 1 unit is lower than the arrival rate for orders of 5 units, because each time an order of 5 units is matched to a pod, an order of 1 unit must also be matched to that pod before it goes to replenishment. In other words, the system can become unstable if the arrival rates of certain inventory classes become *too low*, which is an uncommon condition. There are three necessary stability conditions that the system must satisfy for each SKU separately, otherwise it will become unstable. These are the *replenishment level reachability* condition, the *combinatorial matching* condition, and the *maximum capacity* condition, which are explained below. Fulfilling these conditions does not mean that the system will be stable, but fulfilling the sufficient condition mentioned at the end of this section does.

3.3.8.1 Condition 1: Replenishment Level Reachability

As an example, suppose that all inventory for a certain SKU is put on only one pod, $U = 6$ and $\lambda_{u,s} = 0, \forall u, s$ with the exception that $\lambda_{3,s} > 0$ and $\lambda_{4,s} > 0$ and that $\xi = 0\%$ so that a pod only goes for replenishment if it has zero units remaining. The system is unstable, because after a full pod synchronizes with an order of inventory class 4, the pod will change to an inventory class 2 and can no longer match with an order. It will then never reach the replenishment level and never replenish to inventory class 6.
The replenishment level reachability condition checks whether a pod can be trapped at an inventory level where it can no longer match with an order and at the same time cannot go to replenishment. This also means that for this condition the number of pods per SKU does not matter. Let H be the

set of u for which $\lambda_{u,s} > 0$, then the feasible arrival rates for a SKU s must be in the set Λ_s^1:

$$\Lambda_s^1 = \left\{ \lambda_{u,s} \mid \exists n \in \mathbb{N}\,, \quad 0 \leq U - \sum_{u \in H} nu \leq \lfloor \xi U \rfloor \right\} \tag{3.12}$$

3.3.8.2 Condition 2: Combinatorial Matching

Suppose that the necessary stability condition 1 of Equation (3.12) is met. A counterintuitive feature of the necessary stability conditions for a cross-class matching multi-class SOQN is that there can also be lower bounds to the arrival rate of certain classes. An example would be if $\xi = 0$ and $\lambda_{u,s} = 0$ except $\lambda_{1,s} > 0$ and $\lambda_{U-1,s} > 0$ and $\lambda_{1,s} < \lambda_{U-1,s}$. Each time an order of class $U-1$ is matched with the pod, the order before or after must belong to inventory class 1 so that the pod can go for replenishment. This means that if fewer class 1 orders than class $U-1$ orders arrive, class $U-1$ orders cannot all be matched and will build up to infinity. Hence in this case $\lambda_{U-1,s}$ is a strict lower bound for $\lambda_{1,s}$.
More generally, the combinatorial matching condition essentially looks at the order arrival rates relative of each other. Matching orders of certain sizes to a pod may put constraints on which orders can be matched to that same pod in between replenishments. In the example, matching a pod with an order of size $U-1$ means that the pod must be matched with an order of size 1 before or after, otherwise it cannot go for replenishment. Certain order sizes can only be matched to a pod in combination(s) with orders of other sizes, which constrains the arrival rates. This means that for this condition, the number of pods per SKU does not matter.
If $\frac{U}{u}$ is an integer, then orders of inventory class u alone can empty the pod and can therefore always bring the pod below the replenishment level ξ. If $\frac{U}{u}$ is not an integer, an additional condition is needed to ascertain that the

pod can be brought below ξ. Define the set G as all u for which $\frac{U}{u} \notin \mathbb{N}$, then the feasible arrival rates must be in the set Λ_s^2.

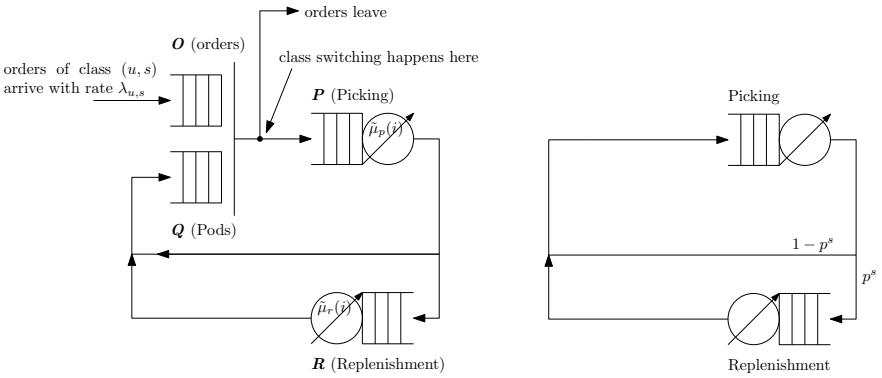
$$\Lambda_s^2 = \Bigg\{ \lambda_{u,s} | \exists w_{g,u} \in \mathbb{N}, U - \lfloor \xi U \rfloor \leq g + \sum_{u=1}^{U} (u \times w_{g,u}) \leq U \; \forall g \in G, \sum_{g \in G} w_{g,u} \lambda_{g,s} \leq \lambda_{u,s} \; \forall u \notin G \Bigg\} \tag{3.13}$$

Here $U - \lfloor \xi U \rfloor \leq g + \sum_{u=1}^{U} (u \times w_{g,u}) \leq U$ means that it must be possible to match an order of inventory class g to a pod together with an integer number $(w_{g,u})$ of orders of other inventory classes u, such that the amount $g + \sum_{u=1}^{U} (u \times w_{g,u})$ is equal to or larger than the number $U - \lfloor \xi U \rfloor$ required to bring the pod below the replenishment level, but equal to or less than the maximum number U on a pod. As an example, assume again that $\xi = 0$, $\lambda_{u,s} = 0$ except $\lambda_{1,s} > 0$ and $\lambda_{U-1,s} > 0$, and $\lambda_{1,s} < \lambda_{U-1,s}$. Then $G = \{U - 1\}$ and $w_{U-1,u} = 1$, so that the equation (3.13) becomes $U \leq U - 1 + 1 \times 1 \leq U$, $1 \times \lambda_{U-1,s} \leq \lambda_{1,s}$

3.3.8.3 Condition 3: Maximum Capacity

The maximum capacity necessary condition examines the number of trips to the pick stations and replenishment stations that are made given the order arrival rates. If the amount of time needed to do picking and replenishment per order times the arrival rate exceeds one, then the system cannot be stable. Here we disregard the time a pod is queueing at the stations, which means that this condition provides an upper bound on the order arrival rates. The system operates at maximum capacity when the pods do not have to wait for an order, but instead immediately synchronize with an order as soon as they reach the synchronization station. For each SKU s, this corresponds to a CQN where the pods go to replenishment with a probability p^s. Figure

3.3.7a shows the compact queueing network, a SOQN model, and Figure 3.3.7b the CQN model that corresponds with the compact model working at full capacity, i.e., the orders are always waiting.



(a) SOQN model, multiple inventory classes

(b) CQN model, single class

Figure 3.3.7: The compact queueing model and the corresponding CQN model

Suppose that ϕ^s is the average number of times per time unit that the pod goes for replenishment for SKU s. For each SKU s, $\lambda_{u,s}$ orders of inventory class u arrive per time unit that each needs u units. In a stable system, all these orders across all inventory classes are fulfilled and the units needed to fill these orders come from replenishment. This implies that the total number of units needed per time unit, divided by the number of units put on the pod during replenishment, equals ϕ^s, see also equation (3.14). Suppose furthermore that θ^s is the average number of times per SKU s and per time unit that the pod goes through the CQN. A pod carries one order at any point in time. In a stable system, it needs to synchronize with every order and for every order the pod carries it goes through the network once. θ^s can then be calculated as shown in equation (3.15). Assuming that the replenishment level is zero, probability p^s can be calculated as shown in equation (3.16). If $p^s \notin [0, 1]$, the network is not stable.

$$\phi^s = \frac{\sum_{u=1}^{U} u\lambda_{u,s}}{M^s U} \qquad \forall s \tag{3.14}$$

$$\theta^s = \sum_{u=1}^{U} \frac{\lambda_{u,s}}{M^s} \qquad \forall s \tag{3.15}$$

$$p^s = \frac{\phi^s}{\theta^s} \qquad \forall s \tag{3.16}$$

The pick and replenishment queues are load-dependent queues and for both queues it holds that the largest service time happens when one pod is at the queue, since then only one workstation is operating at a time. The service time is denoted by $\tilde{\mu}_p^{-1}(i)$ for the pick queue and by $\tilde{\mu}_r^{-1}(i)$ for the replenishment queue. As described earlier in section 3.3.4, $\tilde{\mu}_p(1) \leq \tilde{\mu}_p(i)$ for $i > 1$, therefore using $\tilde{\mu}_p^{-1}(1)$ as the time picking takes provides an upper bound on the picking time needed. $\tilde{\mu}_r^{-1}(1)$ gives an upper bound for replenishment. Per time unit and per SKU s, the pod goes to the pick queue θ^s times, where each time it will spend on average at most $\tilde{\mu}_p^{-1}(1)$ time units. This means that per time unit, a pod is, on average, busy with picking at most $\theta^s \tilde{\mu}_p^{-1}(1)$ time. Furthermore, per time unit the pod goes to the replenishment queue ϕ^s number of times, where each time it spends, on average, at most $\tilde{\mu}_r^{-1}(1)$ time units. This means that per time unit, a pod will on average be busy with replenishment at most $\phi^s \tilde{\mu}_r^{-1}(1)$ time. Let t_{busy} denote the percentage of a time unit that a pod is used. In a stable system, this all needs to fit within one time unit, therefore the order arrival rates needs to be in the set Λ_s^3 as depicted in equation (3.18).

$$t_{\text{busy}} = \theta^s \tilde{\mu}_p^{-1}(1) + \phi^s \tilde{\mu}_r^{-1}(1)) \tag{3.17}$$

$$\Lambda_s^3 = \{\lambda_{u,s} | p^s \in [0,1], t_{\text{busy}} \in [0,1]\} \tag{3.18}$$

3.3.8.4 Sufficiency Condition

For the necessary stability conditions it holds that $\Lambda_s^2 \subset \Lambda_s^1$ but neither Λ_s^1 and Λ_s^3, nor Λ_s^2 and Λ_s^3 are subsets of each other. Figure 3.3.8 shows an example of these sets for a situation with one pod (so $M^s = 1$), $U = 6$, $\xi = 0\%$ and $\lambda_{u,s} = 0$ except $\lambda_{1,s} > 0$ and $\lambda_{5,s} > 0$. Here Λ_s^2 contains all $\lambda_{1,s}$ and $\lambda_{5,s}$ for which it holds that $\lambda_{5,s} \leq \lambda_{1,s}$. For $\lambda_{5,s} > \lambda_{1,s}$ orders for 5 units would not be fulfilled sufficiently often. In this example, $\phi^s = \frac{1}{6}\lambda_{1,s} + \frac{5}{6}\lambda_{5,s}$ and $\theta^s = \lambda_{1,s} + \lambda_{5,s}$, so $t_{\text{busy}} = \lambda_{1,s}\left(\tilde{\mu}_p^{-1}(1) + \frac{1}{6}\tilde{\mu}_r^{-1}(1)\right) + \lambda_{5,s}\left(\tilde{\mu}_p^{-1}(1) + \frac{5}{6}\tilde{\mu}_r^{-1}(1)\right)$. At maximum t_{busy} can be 1. Solving for $t_{\text{busy}} = 1$, we find that $\lambda_{5,s}$ can at most be $\frac{\lambda_{1,s}\left(\tilde{\mu}_p^{-1}(1)+\frac{1}{6}\tilde{\mu}_r^{-1}(1)\right)}{\tilde{\mu}_p^{-1}(1)+\frac{5}{6}\tilde{\mu}_r^{-1}(1)}$.

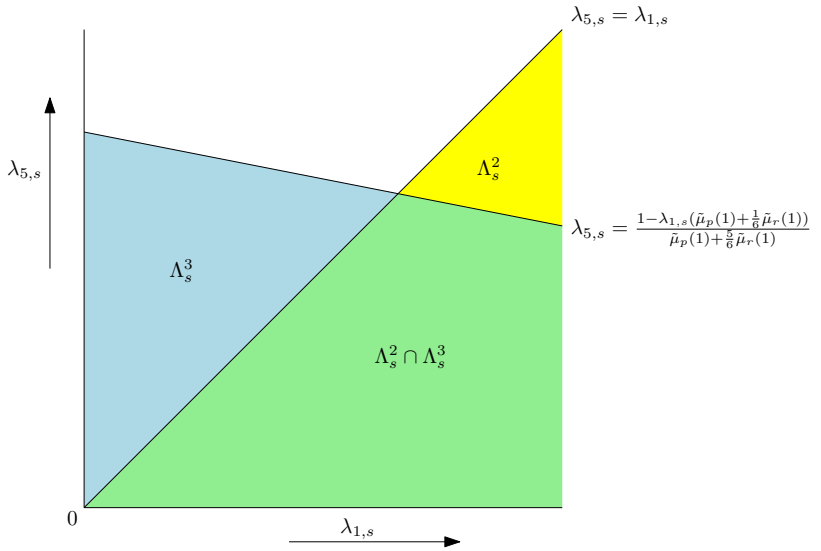


Figure 3.3.8: Example of arrival rates that meet the combinatorial matching and the maximum capacity necessary stability conditions

The three necessary stability conditions do not guarantee that the system will be stable. To see this, consider a system with one SKU and one pod, $U = 3, \xi = 0\%$ and $\lambda_{1,s} = \lambda_{2,s} > 0, \lambda_{3,s} = 0$, where the arrival rates are

such that the maximum capacity necessary stability condition is met, i.e., they are in Λ_s^3. These arrival rates would be in Λ_s^1, as the replenishment level can clearly be reached, and would be in Λ_s^2, for example, by setting $w_{g,u} = 1 \; \forall g, u$. This system would be stable if every time the pod comes back from replenishment, it would first match with an order of inventory class 1 and later with an order of inventory class 2 or vice versa. However, after replenishment the pod could match twice with an order of inventory class 1, after which it will need to match again with an order of inventory class 1, else it cannot go to replenishment. Each time this happens, three orders of inventory class 2 cannot be matched. This means that the stock of inventory class 2 orders builds up and thus that the system is unstable, despite the fact that the necessary stability conditions are met.
The problem is that the pod will not reach the replenishment level *sufficiently* often to prevent the build-up of orders in the queue. A sufficient condition to prevent this situation would be that $\lambda_{1,s}$ orders for each SKU s is very large relative to the arrival rates of the other inventory classes. If there are enough orders for 1 unit, the pod will reach the replenishment level sufficiently often. Theorem 2 shows a sufficient condition.

Theorem 2. *Assuming that* $\lambda_{u,s} \in \Lambda_s^2 \cap \Lambda_s^3$ *for* $1 \leq u \leq U$, $1 \leq s \leq S$, *it holds that* $\lambda_{1,s} \geq \sum_{u=2}^{U} \left((U - \lfloor \xi U \rfloor) - u \right) \lambda_{u,s}$ *is a sufficient condition for stability of the network*

Proof. Let C be a collection of orders that such $\sum_{o_{u,s} \in C} u = U - (\lfloor \xi U \rfloor + 1)$, with $o_{u,s}$ denoting an order of class (u, s). Here C can have multiple orders of the same class (u, s), but they all have to be for the same SKU s. If a pod of class (U, s) would be sequentially matched with all orders in C and only with the orders in C, the pod would attain class $(\lfloor \xi U \rfloor + 1, s)$. Let $\mathfrak{C}$ be the set of all such collections C that meet this condition. Let η_C be the rate at which a pod of class (U, s) is matched with a sequence of orders that contains

orders of the same classes in the same number as C. $\lambda_{1,s} \geq \sum_{C \in \mathfrak{C}} \eta_C$ is not a sufficient condition, because inventory class $(1, s)$ may also appear one or multiple times in a collection C. For an inventory class u, let C_u be the set of all collections C that contain at least one order of inventory class u. Define $L_u = \sum_{C \in C_u} \eta_C$, then L_u is the rate with which a pod attains class $(\lfloor \xi U \rfloor + 1, s)$ due to a collection C that contained an order of inventory class u. Besides an order of inventory class u, it could happen that this collection C consists only of orders of inventory class $(1, s)$. If a pod is matched with an order of inventory class u, it is certain that the pod will reach the replenishment level if it is also matched with $(U - \lfloor \xi U \rfloor) - u$ number of orders of inventory class $(1, s)$. Therefore, the pod will reach the replenishment level *sufficiently* often if $\lambda_{1,s} \geq \sum_u ((U - \lfloor \xi U \rfloor) - u) L_u$. As $\lambda_{u,s} \geq L_u$, $\lambda_{1,s} \geq \sum_{u=2}^{U} ((U - \lfloor \xi U \rfloor) - u) \lambda_{u,s}$ is a sufficient condition. □

3.4 Results

Figure 3.4.1 shows the layout and Table 3.4.1 shows the parameters used to generate the results in this section. In Figure 3.4.1 on the left side, there are six workstations, separated by a traffic corridor from the storage area. In the storage area, stored pods are shown as dark grey squares and empty storage locations as light grey squares. The time needed for pod lifting and storing, the robot speed, and the distribution of the pick and replenishment time are based on Lamballais et al. (2017). However, the Markov Chain would become far too large when using these parameters. Therefore we simulated the queueing network describing the pod movements as depicted in the Figure 3.3.2, without aggregate classes, to generate the results in this section. The simulation time was 604800 seconds per run, with 10 runs and a

warm-up time of 201600 seconds. The width of the 95% confidence intervals were typically about 1% of the average values of the metrics.

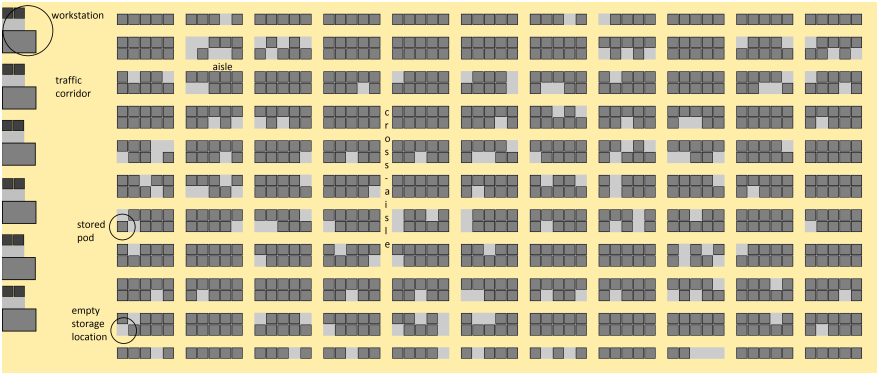


Figure 3.4.1: Top view of the warehouse layout used for the experiments

Table 3.4.1: Parameters used in all the experiments

Parameter	Value	Parameter	Value
Number of aisles	10	Number of cross-aisles	10
Number of storage locations	1100	Number of workstations	6
Time for pod lifting and storing	1 (s)	Robot speed	1.3 (m/s)
Pick time & replenishment time (Identically distributed)	Exponentially distributed, mean is 8 (s)	$\sum_u \lambda_{u,s}$	2 orders per hour per SKU
K	1200	Number of SKUs S	100
Layout width	98.8 meters	Layout length	41.6 meters

To investigate the optimal number of pods to use for inventory, we keep the maximum inventory in the experiments at 36 units per SKU and vary the number of pods per SKU, M^s, which can be 1, 2, 3, 4 or 6, while keeping the layout fixed. The number of SKUs S is 100 and the SKUs are identical. Taken together, this means that the total number of pods that are actively used in the system varies from 100 to 600. The total number of pods in the

system is always 935, which equals 85% of the number of storage locations. The average travel time to bring a pod to a pick station strongly decreases in the number of pods. For the northernmost workstation, the average travel time is 43.4, 34.1, 29.5, 26.6, and 23.0 seconds for 1, 2, 3, 4, and 6 pods respectively. In the last case with 6 pods, $U = 6$, which means that the largest order that can be matched with a pod has 6 units. Therefore in order to keep the experiments comparable, $\lambda_{u,s} > 0$ for $1 \leq u \leq 6$ and $\lambda_{u,s} = 0$ for $u > 6$ across all experiments. To meet the sufficient condition for stability, the arrival rates are $\lambda_{2,s} = \lambda_{3,s} = \lambda_{4,s} = \lambda_{5,s} = \lambda_{6,s}$ and $\lambda_{1,s} = 10 \times \lambda_{2,s}$. Having far more orders arrive for 1 unit than for multiple units seems reasonable, as the system is typically used in an e-commerce environment.
To investigate the optimal ratio of the number of pick stations to replenishment stations, we analyze 5 cases. Let the ratio of the number of pick stations to replenishment stations be denoted by r, where $r = (i, j)$ means that in Figure 3.4.1 the upper i workstations are pick stations and the j lower workstations are replenishment stations. The 5 cases we consider are $r = (1, 5)$, $r = (2, 4)$, $r = (3, 3)$, $r = (4, 2)$ and $r = (5, 1)$.
Finally, to investigate the optimal replenishment level per pod, we study the following cases: (1) a pod is not replenished unless it is empty, $\xi = 0\%$, (2) a pod is not replenished unless it is at least half empty, $\xi = 50\%$, and (3) a pod is replenished every time after it has visited a pick station, $\xi = 100\%$.

From Table 3.4.2 we can conclude that spreading inventory across as many pods as possible (6 in the experiments) appears to consistently result in the lowest order throughput times. If the inventory of a SKU is contained on a single pod, the replenishment level has a much larger influence on the order throughput time than if the inventory is spread across multiple pods. The order throughput time is about one and a half to two times longer when a single pod is used. This is due to a longer average travel time for a single pod from storage to a pick station and due to a higher probability that no pod is available due to replenishment. We can also conclude that in most cases the optimal ratio is $r = (4, 2)$. In addition, it is clear that

Table 3.4.2: Experiment results, t_{ot} in seconds and the utilizations in percentages

		$\xi = 0\%$				$\xi = 50\%$				$\xi = 100\%$			
M^s , U	r	t_{ot}	ρ_{pod}	ρ_{pick}	ρ_{repl}	t_{ot}	ρ_{pod}	ρ_{pick}	ρ_{repl}	t_{ot}	ρ_{pod}	ρ_{pick}	ρ_{repl}
1 , 36	(1, 5)	414.4	11.1	80.7	0.9	168.3	11.2	80.5	1.7	171.0	13.2	80.4	16.1
1 , 36	(2, 4)	330.2	7.4	40.5	1.1	92.7	7.5	40.3	2.1	97.2	9.7	40.5	20.1
1 , 36	(3, 3)	332.7	7.2	26.9	1.5	89.3	7.3	26.9	2.8	93.8	9.6	27.0	27.0
1 , 36	(4, 2)	328.5	7.1	20.1	2.2	88.6	7.3	20.3	4.2	93.2	9.7	20.1	40.2
1 , 36	(5, 1)	323.7	7.1	16.1	4.5	88.9	7.3	16.2	8.4	103.9	13.1	16.1	80.0
2 , 18	(1, 5)	172.0	5.7	80.9	1.8	148.3	5.6	80.1	3.2	146.3	6.5	80.4	16.1
2 , 18	(2, 4)	89.7	3.5	40.3	2.2	77.0	3.6	40.4	3.9	77.2	4.6	40.4	20.1
2 , 18	(3, 3)	85.6	3.4	26.8	3.0	73.7	3.5	26.7	5.2	74.0	4.5	26.9	27.0
2 , 18	(4, 2)	85.4	3.4	20.1	4.4	73.0	3.5	20.1	7.8	73.3	4.6	20.2	40.4
2 , 18	(5, 1)	86.3	3.4	16.0	8.9	73.3	3.5	16.1	15.9	74.0	6.5	16.0	80.4
3 , 12	(1, 5)	147.3	3.6	80.5	2.7	141.4	3.7	80.4	4.5	147.2	4.4	81.2	16.2
3 , 12	(2, 4)	74.5	2.3	40.1	3.4	72.2	2.4	40.3	5.6	72.2	3.0	40.4	20.2
3 , 12	(3, 3)	72.2	2.2	26.9	4.5	69.1	2.3	26.7	7.6	69.2	3.0	26.9	27.0
3 , 12	(4, 2)	71.5	2.2	20.2	6.7	68.6	2.3	20.3	11.4	68.6	3.0	20.1	40.1
3 , 12	(5, 1)	70.7	2.2	16.1	13.4	68.7	2.4	16.1	22.8	68.8	4.4	16.2	81.0
4 , 9	(1, 5)	158.2	2.8	80.7	3.6	140.0	2.8	80.4	5.4	140.3	3.2	80.7	16.2
4 , 9	(2, 4)	87.5	1.7	40.2	4.5	69.5	1.8	40.4	6.7	69.3	2.2	40.3	20.2
4 , 9	(3, 3)	81.1	1.7	26.8	5.9	66.4	1.7	26.9	8.9	66.3	2.2	26.8	26.7
4 , 9	(4, 2)	83.4	1.7	20.1	9.0	65.7	1.7	20.2	13.4	65.7	2.2	20.3	40.4
4 , 9	(5, 1)	84.8	1.7	16.1	17.9	66.0	1.8	16.1	26.5	65.9	3.2	16.0	80.2
6 , 6	(1, 5)	139.9	1.8	80.5	5.4	137.0	1.9	80.4	7.8	144.5	2.2	80.8	16.2
6 , 6	(2, 4)	66.0	1.2	40.0	6.7	65.9	1.2	40.3	9.7	65.9	1.4	40.3	20.2
6 , 6	(3, 3)	63.3	1.1	26.9	8.9	63.1	1.2	27.0	13.0	62.9	1.4	26.8	26.8
6 , 6	(4, 2)	62.6	1.1	20.1	13.4	62.4	1.2	20.1	19.4	62.5	1.4	20.1	40.1
6 , 6	(5, 1)	63.1	1.1	16.1	26.7	62.6	1.2	16.1	39.0	62.6	2.1	16.1	80.4

$r = (1, 5)$ performs much worse than the other ratios, and the optimal one in terms of lowest order throughput time depends both on M^s and ξ. Lastly, we can conclude that the optimal replenishment level is $\xi = 50\%$ in the majority of experiments. Setting the replenishment level as $\xi = 0\%$ leads to the highest order throughput times in all experiments except in one case, when $M^s = 6$, $U = 6$ and $r = (1, 5)$. The utilization of the replenishment stations is typically very low, unless $\xi = 100\%$, which means the pod goes to replenishment after every pick. Appendix 3.B shows the same results but for equal order arrival rates across inventory classes. From Table 3.B.1 we can conclude that the system is not stable if $\xi = 0\%$, even though the necessary stability conditions are met. This happens because insufficient orders of inventory class 1 arrive, which means that the pod stays in inventory class 1 far too long and does not go for replenishment sufficiently often.

3.5 Conclusions and Future Work

The results show that each of the decision variables can be optimized. The order throughput time appears lowest when inventory is spread across as many pods as possible, when the ratio of the number of pick stations to replenishment stations is 4 to 2, and when the replenishment level is 50%. The relationship between order throughput time and the ratio of the number of pick stations to replenishment stations has a skewed U shape. Moreover, the replenishment level has a strong influence on the utilization of the replenishment stations, which can become very low. Tables 3.5.1 and 3.5.2 summarize some of the additional insights.

Table 3.5.1: Managerial insights regarding stability

Stability of pick operations	Small arrival rate of orders of 1 unit	Large arrival rate of orders of 1 unit
Replenish when pod is empty ($\xi = 0$)	Not stable	Stable
Replenish when still some units left on pod ($\xi > 0$)	Stable	Stable

Table 3.5.2: Managerial insights regarding decision variables

Interaction decision variables	Inventory of a product on a single pod	Inventory of a product on multiple pods
Ratio of the number of pick stations to replenishment stations	Choosing the wrong ratio can increase order throughput time by about 25%	Choosing the wrong ratio can increase order throughput time by about 100%
Replenishment level	Large impact on order throughput time, best to replenish a pod before it becomes empty	Small impact on order throughput time, best to replenish a pod before it becomes empty

This chapter focused on several important tactical decisions, but there are many promising directions for future research focusing on operational decisions. For example, an RMFS is flexible in capacity as robots can be added quickly and workstations can be opened and closed as needed. The system can thus dynamically increase and decrease the amount of resources used and how much of the resources to dedicate to different types of activities, something that has not yet been researched. Another interesting feature is that the system's decisions can be decentralized to a high degree. Kiva Systems decentralized robot movement and collision detection already in the earliest implementation, but other elements such as route planning, task scheduling, and resource allocation can also be decentralized (see Wurman et al. (2008)). No research has yet been conducted on the interplay between the algorithms and policies for each of these elements.

Appendix

3.A Size of the State Space

This appendix explains how to calculate the total number of states possible in the Markov Chain. The explanation here assumes there is only one SKU, so $S = 1$, but for $S > 1$, the number of states is just the number of states when $S = 1$ multiplied by S. The total number of states is constrained by the maximum number of orders that can wait at the synchronization station and by the number of pods. There can be at most K orders at the synchronization station. Let the total number of possible ways to distribute k orders across the order classes and for all k be denoted by N_O^C. These orders can be of any of U classes (the dimension of $\boldsymbol{O}$ equals U as explained above), so the possible number of combinations is $\binom{U+k-1}{k}$ as this is similar to choosing with repetition k places out of a possible number of U places. Then N_O^C is given by Equation (3.19). Let N_P^C denote the number of possible ways that M^s pods can be distributed across $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$. This is again similar to choosing with repetition, and Equation (3.20) shows how to calculate N_P^C.

$$N_O^C = \sum_{k=0}^{K} \binom{U+k-1}{k} \tag{3.19}$$

$$N_P^C = \binom{2U+M^s}{M^s} \tag{3.20}$$

It is not possible to combine every instance of an order part with every instance of the other three parts, because whenever a pod of inventory class u is at the synchronization station, it is not possible to also have orders of inventory class u or lower there. In other words, $\boldsymbol{O}$ constrains the number of inventory classes u in $\boldsymbol{P}$ for which $\boldsymbol{P}_{u,s}$ can be bigger than zero. Let $N^Y_{j,s}$ be the number of combinations in $\boldsymbol{O}$ for which $\boldsymbol{O}_{u,s} = 0, u \leq j$ and $\boldsymbol{O}_{j+1,s} > 0$, so $N^Y_{j,s} = \{\boldsymbol{O} | \boldsymbol{O}_{u,s} = 0, 0 \leq u \leq j, \boldsymbol{O}_{j+1,s} > 0\}, j = 0, \ldots, U-1$. Suppose that there are k orders at the synchronization station, with $k = 1, \ldots, K$. For $N^Y_{j,s}$, this means there can be h orders of class $j+1$ and $k-h$ at classes $j+2$ to $U-1$. Let $N^Y_{j,s,k,h}$ denote all possible situations where k orders are waiting at the synchronization station, with no orders of class j or lower and h orders of class $j+1$, then:

$$N^Y_{j,s,k,h} = \begin{cases} \binom{U+k-h-j-2}{k-h} & \text{if } U+k-h-j-2 \geq 0 \\ 1 & \text{(no classes } > j \text{ left to distribute orders to)} \end{cases} \tag{3.21}$$

$$N^Y_{j,s} = \sum_{k=1}^{K} \sum_{h=1}^{k} N^Y_{j,s,k,h} \qquad j = 0, \ldots, U-1 \tag{3.22}$$

In the above, $j \leq U-1$, because $N^Y_{U,s} = 0$ and therefore does not follow the structure and formula above.
Let $N^F_{j,s}$ be the number of combinations in $(\boldsymbol{Q}, \boldsymbol{P}, \boldsymbol{R})$ for which $\boldsymbol{Q}_{u,s} > 0, u \leq j$ and let N^Z be the total number of states in the Markov Chain. Assume that the replenishment level is zero, so pods are only replenished when they are empty.

$$N_{j,s}^{F} = \binom{2U + M^s - j}{M^s}, \qquad j = 0, \ldots, U \tag{3.23}$$

$$N^{Z} = \sum_{s=0}^{S} \sum_{j=0}^{U} N_{j,s}^{Y} N_{j,s}^{F} \tag{3.24}$$

Table 3.A.1 shows the number of states for various combinations of K, U and M^s. It seems that if K increases by 10, N^Z roughly doubles. Another pattern that becomes apparent is that the higher U is, the more rapid N^Z grows in M^s.

Table 3.A.1: Number of states N^Z, $S = 1$

N^Z	$K = 2$	$K = 5$	$K = 10$	$K = 20$	$K = 30$	$K = 40$
$U = 2$, $M^s = 2$	53	155	445	1475	3105	5335
$U = 2$, $M^s = 6$	462	1050	2590	7770	15750	26530
$U = 3$, $M^s = 2$	155	708	3263	18998	57233	127968
$U = 3$, $M^s = 6$	2814	9324	35574	183624	529074	1155924
$U = 4$, $M^s = 2$	360	2430	17290	171810	732105	2116675
$U = 4$, $M^s = 6$	12105	56313	322773	2802393	11368863	32049183
$U = 5$, $M^s = 2$	721	6882	73073	1206580	7190337	26714094
$U = 5$, $M^s = 6$	41283	261030	2186327	31200246	176430265	638136884

3.B Results with Equal Order Arrival Rates

Table 3.B.1 shows the results for equal order arrival rates. For ξ =50% and ξ =100%, the results are nearly identical to the results in section 3.4, but for ξ =0%, the results are unstable. The order arrival rates meet all the necessary stability conditions, but these are not sufficient and it turns out that the order throughput time simply becomes longer if the simulation time is larger. Let T_K be the percentage of time the system spends in a state

where the number of orders at the synchronization stations equals K. If T_K exceeds 1%, we consider the system unstable and do not show results. The reason the system is unstable is that the pods in the system remain in state $(1, s)$ too often. This happens, because of the following. As $U = 6$, if an order of class $(5, s)$ arrives and matches with a full pod, the pod goes to state $(1, s)$ and can only be matched with an order of class $(1, s)$. This also happens if the pod is matched with an order of class $(2, s)$ and an order of class $(3, s)$ or alternatively an order of class $(4, s)$ and an order of class $(1, s)$. As the arrival rate of orders of class $(1, s)$ equals the arrival rate of orders of class $(5, s)$, orders of class $(1, s)$ do not arrive sufficiently often to bring a pod of class $(1, s)$ to the replenishment level.

Table 3.B.1: Results for experiments with equal order arrival rates, t_{ot} in seconds and the utilizations in percentages

		$\xi = 0\%$				$\xi = 50\%$				$\xi = 100\%$			
M^s , U	r	t_{ot}	ρ_{pod}	ρ_{pick}	ρ_{repl}	t_{ot}	ρ_{pod}	ρ_{pick}	ρ_{repl}	t_{ot}	ρ_{pod}	ρ_{pick}	ρ_{repl}
1 , 36	(1, 5)	1192.0	37.8	20.1	0.4	142.8	38.8	20.1	0.7	182.5	48.3	20.1	4.0
1 , 36	(2, 4)	1237.8	36.7	10.0	0.5	135.9	37.9	10.2	0.9	177.0	48.2	10.1	5.0
1 , 36	(3, 3)	1144.6	36.4	6.8	0.7	133.9	37.5	6.7	1.2	170.1	47.2	6.7	6.6
1 , 36	(4, 2)	1166.7	36.0	5.0	1.0	133.0	37.1	5.0	1.8	175.4	48.1	5.0	10.1
1 , 36	(5, 1)	1222.4	36.1	4.0	2.0	132.7	37.1	4.0	3.6	180.8	49.0	4.1	20.2
2 , 18	(1, 5)	973.6	18.5	20.4	0.8	81.8	19.3	20.1	1.3	84.2	23.1	20.0	4.0
2 , 18	(2, 4)	949.6	17.6	10.0	1.0	76.8	18.5	10.1	1.7	79.2	22.6	10.0	5.1
2 , 18	(3, 3)	872.0	17.5	6.7	1.3	76.3	18.3	6.7	2.2	78.8	22.7	6.7	6.7
2 , 18	(4, 2)	907.4	17.4	5.0	2.0	76.2	18.4	5.1	3.3	78.8	22.9	5.1	10.2
2 , 18	(5, 1)	935.1	17.4	4.0	3.9	76.1	18.4	4.1	6.7	79.6	23.4	4.1	20.2
3 , 12	(1, 5)	1048.7	12.2	20.1	1.2	74.1	12.9	20.1	1.9	74.2	15.1	20.1	4.0
3 , 12	(2, 4)	1166.6	11.7	9.8	1.5	69.4	12.5	10.0	2.3	69.9	14.7	10.1	5.0
3 , 12	(3, 3)	1143.8	11.7	6.8	2.0	68.9	12.3	6.7	3.1	69.0	14.5	6.7	6.7
3 , 12	(4, 2)	1088.3	11.6	5.0	2.9	68.7	12.4	5.0	4.6	68.7	14.8	5.1	10.2
3 , 12	(5, 1)	1048.9	11.5	4.0	5.8	68.8	12.4	4.0	9.3	69.4	15.1	4.0	20.2
4 , 9	(1, 5)	1963.8	9.3	20.2	1.6	71.0	9.8	20.2	2.2	70.9	11.1	20.2	4.1
4 , 9	(2, 4)	2013.2	8.9	10.1	2.0	66.2	9.5	10.1	2.8	66.1	10.8	10.1	5.1
4 , 9	(3, 3)	1897.2	8.9	6.7	2.6	65.7	9.3	6.6	3.7	65.8	10.7	6.7	6.7
4 , 9	(4, 2)	1979.7	8.8	5.0	3.9	65.6	9.4	5.1	5.5	65.6	10.8	5.0	10.1
4 , 9	(5, 1)	2022.5	8.8	4.1	7.9	65.7	9.5	4.0	11.1	65.8	11.1	4.0	20.2
6 , 6	(1, 5)	–	–	–	–	67.5	6.7	20.1	3.0	67.4	7.3	20.1	4.0
6 , 6	(2, 4)	–	–	–	–	63.0	6.5	10.0	3.8	62.8	7.0	10.0	5.0
6 , 6	(3, 3)	–	–	–	–	62.6	6.6	6.8	5.1	62.5	7.0	6.7	6.7
6 , 6	(4, 2)	–	–	–	–	62.4	6.6	5.1	7.6	62.2	7.1	5.1	10.1
6 , 6	(5, 1)	–	–	–	–	62.5	6.6	4.0	15.3	62.5	7.2	4.0	20.0

4 Resource Reallocation

4.1 Introduction

Handling peak demand for order fulfillment is a challenge for e-commerce warehouses. According to Gue et al. (2014), e-commerce is challenging for three reasons. First, it requires additional services such as packaging following the picking operations. Second, e-commerce warehouses usually store a very large number of stock keeping units (SKUs), or products, that are picked as individual units rather than as cartons or pallets, making operations complex and labor intensive. Third, e-commerce creates a high pressure on reducing the customer waiting time, with tight delivery schedules, such as same-day delivery, becoming increasingly common. This overlaps with findings of MHI & Deloitte (2015) and MHI & Deloitte (2016), who identify high customer demand for faster response times and rising expectations of customer service as important current trends and note that the material handling industry is increasingly adopting robotics and automation to create competitive advantage (see Azadeh et al. (2017)).
In e-commerce, demand fluctuations tend to be strong and the influx of new products and the outflux of outdated products are large. The Christmas season is an absolute peak demand season that requires hiring temporary workers to expand the capacity of the warehouse from November to December. Other important peak demand periods are the evening and weekend, when consumers have spare time to shop online, see also Agatz et al. (2008) and Patil & Divekar (2014).

This chapter focuses on handling short term peak demand for a warehouse operating a Robotic Mobile Fulfillment System (RMFS), with rates alternating between high and low levels. We look at two short term peak demand situations, (1) the "daily" situation where high demand occurs in the evening and low demand during the rest of the day, and (2) the "week" situation, where high demand occurs during the weekend and low demand during working days.

The two main processes in an RMFS are the picking and the replenishment processes. When an order arrives, it is assigned to a pick station, where it waits until all the items it needs are picked from the pods brought to the pick station. Whereas picking involves handling one or a few products per pod each in a small quantity, replenishment involves multiple units per product. When suppliers deliver goods at the warehouse they are usually first stored in a reserve area elsewhere in the warehouse. Only when the system needs units of a product, a pallet with that product is sent to a replenishment station, where a worker can then put individual units on the pods (Wulfraat, 2012).

One of the main characteristics of an RMFS is the ability to reallocate workers from pick stations to replenishment stations and reallocate robots from working on picking tasks to working on replenishment tasks. This characteristic is known as the dynamic reallocation of resources across picking and replenishment activities, where the workers and robots are the resources. Dynamic reallocation of resources can be achieved quickly and in real time in an RMFS. The workstations in an RMFS are suitable for both picking and replenishment. Pick stations can be converted to replenishment stations in a short amount of time and vice versa. Alternatively, it is possible to have additional pick and replenishment stations that can be opened and closed on demand. For example, if the number of pick stations needs to be decreased by one and the number of replenishment stations increased by one, a worker at a pick station can close that station, move to a closed replenishment station, and open that replenishment station. A robot can be assigned to either a pick task or a replenishment task, after it returns a pod to a storage location.

In other words, robots can in principle be reallocated continually. Therefore, set-up or switching costs due to resource reallocation are relatively low in an RMFS, and can occur frequently to maximize performance under changing order arrival rates.

Resource reallocation does influence other costs, in particular those related to the customer waiting time and inventory availability, which depend on both the picking and the replenishment process. Too few resources in the picking process means that the customer waiting time will become too long, a metric that e-commerce companies are keen on minimizing. Inventory availability, however, increases as the replenishment process has more resources and pods are quickly replenished. With more resources allocated to the picking process, customer waiting time should decrease as products are fulfilled more quickly. However, if too few resources are allocated to the replenishment process, the inventory level in the storage area becomes low. Inventory availability in an RMFS is tied to the inventory held on the pods; RMFSs typically have only a few days worth of inventory and a stock-out in the storage area may mean unavailability of a product for at least a day (Wurman et al. (2008) and Wulfraat (2012)). In e-commerce, low inventory availability may lead to lost sales as customers will place their order with a competitor that does have stock immediately available. The resources are the robots and the workers, the reallocation happens between the picking and replenishment process, the costs are those related to customer waiting time and lost sales, and strong demand fluctuation means alternating between a high and a low order arrival rate. This chapter focuses on using resource reallocation to minimize costs under strong demand fluctuation, modeled as an Markov Modulated Poisson Process (MMPP). The number of workers is kept fixed, so that wages do not factor into the costs. Furthermore, resource reallocation can happen frequently as set-up or switching costs are considered negligible. The goal is to find an optimal policy for allocating robots and workstations to picking and replenishment activities given the number of unfulfilled orders and the number of pods that need replenishment. The optimal policy must balance the cost of customer waiting time with the cost of unavailability of inventory.

The optimal policy is compared to two benchmark policies used commonly in practice. The contributions of this chapter are threefold. (1) We show how to extend the queueing networks used for modeling an RMFS in the literature, to include both robots and inventory pods rather than either one of these two. We also show how to integrate the extended queueing networks into an MDP model. (2) We consider two resources to be used in two processes, whereas previous work studies how to optimize the use of only one resource in only one process. This generalization allows us to study the trade-off of reallocating resources between the picking process and the replenishment process, which influences customer waiting time and inventory availability. This trade-off is particularly relevant for e-commerce warehouses, as they need to deliver quickly while also maintaining sufficient inventory to prevent lost sales. (3) Our approach shows the benefits of being able to reallocate resources quickly and frequently. The cost reductions achieved in an RMFS, where resources can be reallocated quickly and frequently, are compared with cost reductions under benchmark policies where resources are reallocated less often. (4) We obtain optimal policies while modeling time-varying demand using a two-phase MMPP.

This chapter is structured as follows: Section 4.2 discusses the literature, Section 4.3 develops a queueing network to model the performance of the system given a fixed sets of resources, Section 4.4 develops a Markov Decision Process (MDP) model for finding the optimal policy of allocating resources, Section 4.5 gives the stability conditions for this MDP and the structure for the optimal policy, Section 4.6 validates the performance of the queueing network by comparing it with a detailed, realistic simulation of an RMFS, and shows the results of applying the MDP model to several case studies, and Section 4.7 draws conclusions and explores promising avenues for further research.

4.2 Literature

This chapter studies the RMFS under strong demand fluctuations and resource reallocation. We therefore first examine the RMFS literature, especially existing queueing networks for the RMFS, second we discuss how demand fluctuations are typically modeled in stochastic models as a Markov Modulated Poisson Process (MMPP), thirdly we look at MDPs have been used in warehouse situations, and finally we review how MDPs have been used for resource allocation in manufacturing situations.

The RMFS is described generally and compared with other robotized warehousing systems in Azadeh et al. (2017), and described in more detail by Wurman & Enright (2011) and Wurman et al. (2008), the developers of the first RMFS. They identify numerous, challenging operational decision problems, such as path planning, task allocation, order assignment to pick stations, pod storage allocation, and inventory pod selection. Path planning in an RMFS has been studied by Merschformann et al. (2017a), while Merschformann et al. (2018) address the above-mentioned decision problems together, except for path planning. Boysen et al. (2017) provide methods for optimally batching and sequencing picking orders, and for sequencing the pods transported to a pick station. In contrast, both Lamballais et al. (2017), Lamballais et al. (2018b), Zou et al. (2017), Zou et al. (2018) and Nigam et al. (2014) focus on the warehouse design aspects of an RMFS. They model the operations in an RMFS using queueing networks. Nigam et al. (2014) build queueing network models to analyze the throughput time of single-line orders in an RMFS. Lamballais et al. (2017) design queueing models that incorporate realistic movement of the robots, storage zones and multi-line orders in an RMFS. These models provide accurate estimates for workstation and robot utilization and order throughput time. They find how the dimensions for the storage area and placement of workstations around the storage area impact the throughput times of single- and multi-line orders. Zou et al. (2017) examine how to determine the routing probabilities, i.e. the probability with which a robot chooses a certain destination worksta-

tion for transporting the pod. They analytically derive the best size of a storage block. Moreover, they find that an assignment rule incorporating the different handling speeds of the workers leads to significantly better routing probabilities, and they provide a neighborhood search algorithm that performs close to optimal.

Zou et al. (2018) study battery management in RMFSs, and compare three policies: battery swapping, automated plug-in charging, and inductive charging. They build a semi-open queueing network to evaluate these policies and conclude that, if robot prices are low and required retrieval transaction throughput time are small, inductive charging outperforms the other two policies in annual costs. They also find that ignoring battery recovery underestimates the system costs and the number of robots required.

Lamballais et al. (2018b) study three key variables, namely the number of pods per product, the ratio of the number of pick stations to replenishment stations, and the replenishment level. They find that these variables interact and together affect the order throughput time and the stability of the system in the long run. Their model expands the work by Lamballais et al. (2017) by including replenishment in the queueing network. They build a new type of Semi-Open Queueing Network (SOQN), the cross-class matching SOQN, to analyze the pick and replenishment operations simultaneously.

In e-commerce fulfillment applications, the RMFS has to deal with strong demand fluctuations and periods of peak demand. Tunc et al. (2011) show that assuming that demand is stationary when it is not, can lead to high total expected costs, even when the optimal stationary inventory policy is used. Peak demand or seasonal demand can be incorporated in stochastic models by modeling the order arrival process as a Markov Modulated Poisson Process (MMPP) (Fischer & Meier-Hellstern, 1992). Prabhu & Zhu (1989) and van Hoorn & Seelen (1983) explore the effect of using an MMPP arrival process for queueing systems and derive the properties of the waiting time, idle time and busy period. Ching et al. (1997) construct fast algorithms for solving queueing systems with MMPP inputs and Dhingra et al. (2017) solve a Semi-Open Queueing Network (SOQN) with MMPP input using the

Matrix Geometric Method.
In other words, demand fluctuates and orders arrive at the warehouse stochastically. It is therefore often modeled as a Markov Decision Process (MDP), where the problem is to optimize the flow of goods. Seidscher & Minner (2013) and Archibald (2007) use an MDP to optimally control transshipments between warehouses. The paper of Li (2013) focuses more on optimal replenishment. The author studies an inventory system with two arrival processes: the standard order arrival process and the return of goods. The author uses an MDP to model the dynamics of the returns of goods and to derive optimal replenishment policies.
The resource allocation problem has been studied in the context of manufacturing. The resources are a number of identical machines, which can be allocated production, and the problem is to find the optimal production rate while keeping inventory (costs) under control. Crabil (1974) shows how to optimally control a production system with variable service rates. Bhat & Krishnamurthy (2012) derive the structure of the optimal production and inventory policies for a manufacturing system subject to MMPP arrivals. Bhat & Krishnamurthy (2013) model the problem of finding optimal production rates for a manufacturer with one product serving multiple classes of customers under seasonal demand as an MDP. They demonstrate how to calculate the optimal production rates and show that the optimal rates are season-dependent. Bhat & Krishnamurthy (2015) show the value of having flexibility in both the production rates and the inventory levels in a manufacturing system subject to seasonal demand. Bhat & Krishnamurthy (2016a) look at a similar environment but focus on the characteristics of the seasonal demand. They examine how the differences in average demand, in average season duration and duration variability, in the randomness of the sequence of seasons and in skewed seasonality have an effect on a manufacturing system where production rates and inventory levels can be flexibly controlled. Bhat & Krishnamurthy (2016b) build on the previous work by adding service level constraints to each season. In all of these studies, the model includes one type of resource and one type of process to allocate the

resource to.
The queueing networks in Lamballais et al. (2017) and Lamballais et al. (2018b) model either robots or pods, but the research in this chapter needs to model both resource types. The robots need to be included in the model as they are reallocated between the picking and replenishment process, and the pods are needed to model inventory availability. In addition, existing resource reallocation MDP models, such as the models of Bhat & Krishnamurthy (2015), Bhat & Krishnamurthy (2016a) and Bhat & Krishnamurthy (2016b) in a manufacturing settings, only model one process and one resource, whereas we examine a problem with two processes, picking and replenishment, and two resource types, workers and robots. Section 4.3 describes the queueing network used in this chapter, and section 4.4 describes the MDP for two processes and two resources and how to integrate the queueing network in that MDP.

4.3 Queueing Models

This section develops a queueing network model to estimate the system performance when the resources allocated to pick and replenishment activities are fixed. This chapter builds on and extends the queueing networks in Chapter 2 and Chapter 3.

4.3.1 Assumptions

The model makes the following assumptions. (1) The orders arrive according to a Markov Modulated Poisson Process (MMPP). We capture the time-varying nature of demand with different demand seasons. Within each demand season, the orders arrive according to a Poisson process. As van Nieuwenhuyse & De Koster (2009) point out, a Poisson process is a reasonable assumption if a large customer base makes ordering decisions independently

and under similar conditions. (2) During a period of high demand, the system can be temporarily unstable, with queues building up, as long as together with periods of low demand the system is stable in the long run. (3) We aggregate the SKUs to one SKU. We are only interested in the total workloads for the picking and replenishment process, so this simplifying assumption allows us to focus on the behavior of the system as a whole and not on individual products. We elaborate more on this assumption at the end of this subsection. (4) At the replenishment stations there are always pallets waiting with goods that need to be placed on the inventory pods. (5) Robot congestion or gridlock does not occur. Unloaded robots can move underneath parked pods, which gives them more possible routes and options to adjust their chosen route than robots carrying pods, which have to travel in the aisles. We assume single-directional travel in the aisles, which strongly decreases the congestion effects and the likelihood of gridlock. (6) The robot dwelling policy is the Point Of Service Completion (POSC) policy.

Two assumptions need further elaboration, namely assumption (1) about MMPP arrivals, and assumption (3) requires more discussion on the probability that a pod needs to go to a replenishment station after visiting a pick station. This probability is denoted by q. In an MMPP arrival process, the external demand is modeled as an irreducible continuous time Markov Chain, where the order arrival rate depends on the state of the external demand. As long as external demand is in a state s, orders arrive to the system according to a Poisson process with rate λ_s. When external demand moves from state s to another state s', orders start to arrive according to a Poisson process with rate $\lambda_{s'}$ instead of rate λ_s. External demand moves from state s to another state according to an exponential process with average time ν_s^{-1} (Fischer & Meier-Hellstern, 1992). In e-commerce applications, demands varies throughout the year, with many peaks and troughs. Wulfraat (2012) shows a graph illustrating such a demand pattern. A MMPP distribution with two phases seems to fit the data well, namely low demand with arrival rate λ_l and average duration ν_l^{-1}, and high demand with arrival rate λ_h and average duration ν_h^{-1}. The latter phase of high demand models periods of

peak demand.
Assumption (3) means that we treat all SKUs as if they were the same, so in our model we effectively have only SKU. The advantage of aggregating all SKUs into one SKU is that fluctuations of individual SKUs are canceled out. One SKU may experience a high demand while another is experiencing a low demand, and by aggregating all SKUs such individual differences are averaged out. One, aggregated SKU is more realistic in conjunction with our MMPP distribution, which models periods of high and low demand for the warehouse as a whole. A consequence of aggregating the SKUs to one SKU in the model, is that an order can be immediately matched with a pod when it arrives at the system. This is realistic as in practical implementations of an RMFS, it is unlikely that an order would be released for which no stock is available in the storage area. The main disadvantage of modeling only one SKU rather than all products separately, is that we cannot keep track of the number of units per product on each pod, which would have allowed us to accurately model when a pod needs to go to replenishment. However, as long as we can accurately estimate the probability q that a pod needs to go for replenishment after visiting a picking station, the average customer waiting time and the availability of inventory are not affected. The probability q is exogenous to our model and can be observed in real RMFS applications. It is not the probability that a pod directly goes to a replenishment station after picking, since the pod may spend some time in storage before a robot becomes available to collect it from storage and to transport it. Rather, q is the probability that the pod will be needed for replenishment, either because it has run low on inventory itself, or because a product is running low on inventory and the pod is needed for storing the units of that product.

4.3.2 Description of the Complete Queueing Network

The complete queueing network is shown in Figure 4.3.1. Our network is a special class of queueing networks known as semi-open queueing networks

(SOQN)(see Roy (2016)). SOQNs are particularly useful to capture external transaction waiting times and capture the synchronization between the orders and the resources. It is open with respect to the orders and closed with respect to the resources. Each queue in Figure 4.3.1 is labeled and in the following, queue $[x]$ will refer to queue number x in Figure 4.3.1. Three types of entities move through this network: orders, robots, and pods. The robots are dedicated to either picking or replenishment tasks and cannot switch to a different task.

The picking process works as follows, see Figure 4.3.1. Orders arrive at a synchronization station, queue [1], at either a low arrival rate λ_l or a high arrival rate λ_h. The time that a period of low demand or of high demand lasts is exponentially distributed with mean ν_l^{-1} and mean ν_h^{-1}, respectively. An order is matched at queue [1] with a suitable pod, after which the pod with assigned order is matched with an idle picking robot at the next synchronization station, queue [2]. The picking robot with assigned order and pod then moves through two Infinite Server (IS) queueing stations, queues [3] and [4], that model ([3]) the travel from the dwell location of the picking robot to the storage location of the pod and ([4]) the travel from the storage location of the pod to a pick station. The service times of the IS queueing stations correspond to the respective traveling times. The distributions can be calculated by assigning probabilities to every storage location, that indicate the likelihood of pod retrieval from that storage location for transport. As the travel times between any two points in the warehouse can be calculated using closed form expressions (Lamballais et al., 2017), these probabilities and travel times together completely describe the distribution of the travel times, without the need for any further assumptions. In other words, the travel times follow an empirical distribution.

The RMFS has W_p pick stations, which are modeled as queue [5], a queueing station with W_p servers. Each server operates with a service rate μ_p. The picking robot with assigned order and pod arrives at queue [5], which corresponds to the robot arriving at the buffer of a pick station. After the picker has retrieved a unit from the pod, the order assigned to that pod

leaves the system. The picking robot and its assigned pod proceeds through an IS queueing station, queue [6], which models travel from the pick station to the storage location where the pod must be returned. After storing the pod, the picking robot becomes idle and joins a synchronization station, queue [2] where it can be matched with a new pod and order coming from queue [1]. The picking robot will continue this cycle indefinitely, but the pod only continues this cycle until it has to visit a replenishment station. After every cycle, the pod has to visit a replenishment station with a probability q. In that case it joins a synchronization queue, namely queue [7], which is associated with the replenishment process. Otherwise, it goes to the synchronization station where it can be matched with orders, queue [1].
The replenishment process works as follows. At queue [7], a pod that needs replenishment is matched with an idle replenishment robot. The replenishment robot with its assigned pod then moves through four queueing stations, queues [8], [9], [10] and [11], that are similar to queues [3], [4], [5] and [6] in the picking process. The pod is replenished at queue [10]. The replenishment robot continues this cycle indefinitely, but the pod moves to a synchronization station, queue [1], where it can be matched with an assigned order.
In Figure 4.3.1, μ_{ss}^{-1} is the average robot travel time from a random storage location to another random storage location, μ_{sp}^{-1} is the average robot travel time from a random storage location to a random pick station, μ_{p}^{-1} is the average time that a picking operation takes, μ_{ps}^{-1} is the average robot travel time from a random pick station to a random storage location, μ_{sr}^{-1} is the average robot travel time from a random storage location to a random replenishment station, μ_{r}^{-1} is the average time that a replenishment operation takes and μ_{rs}^{-1} is the average robot travel time from a random replenishment station to a random storage location. As mentioned earlier, the travel times follow an empirical distribution. The pick times at the pick stations follow an exponential distribution with mean μ_{p}^{-1}, and replenishment times at replenishment stations an exponential distribution with mean μ_{r}^{-1}. However, these distributions can also be empirical distributions if there is empirical data available on pick and replenishment times.

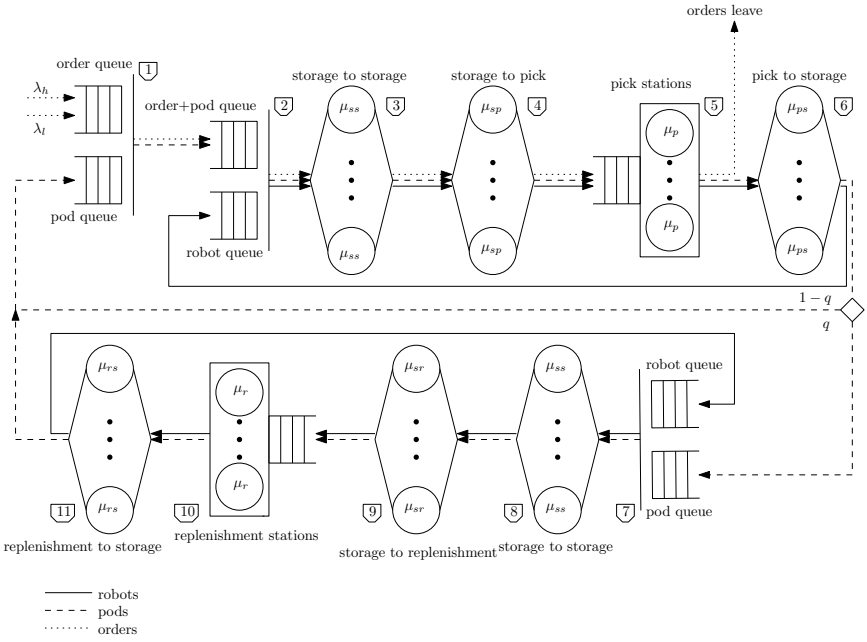


Figure 4.3.1: The Complete Semi-Open Queueing Network

4.3.3 The Compact Network

Section 4.4 develops an MDP model for analyzing dynamic resource reallocation. This MDP model needs to incorporate the service rates of the picking process and the service rates of the replenishment process. Incorporating the complete queueing network in Figure 4.3.1 would lead to a state space that even for small instances becomes too large for analysis, It would have to keep track of the MMPP phase and all the orders, pods and robots at the 11 queueing stations, leading to a high dimensional state space. To keep the size of the state space sufficiently small, this section transforms the complete queueing network in Figure 4.3.1 to a more compact queueing network that makes it amenable to analysis. Note that the network dynamics are still captured in the compact network model.

The transformation from the complete to the compact network is done in two steps and works as follows. The first step is transforming the complete network to an intermediate network. From the complete queueing network, the queues related to the replenishment process, namely queues [8], [9], [10] and [11], are used to form a Closed Queueing Network (CQN), as shown in Figure 4.3.2. In this CQN, let the throughput rate with i robots in the CQN be equal to $\hat{\mu}_r(i)$. Using Norton's theorem, we can transform this CQN into a single equivalent load-dependent queue with service rates $\hat{\mu}_r(i)$ (see Viswanadham & Narahari (1992)). The service rates $\hat{\mu}_r(i)$ for $1 \leq i \leq R$ can be found by varying the number of robots i in the CQN from 1 to R and calculating the resulting throughput rates. By convention, the service rate of an empty load-dependent queue is zero, so $\hat{\mu}_r(0) = 0$.

The first step, where the complete network in Figure 4.3.1 is transformed to the intermediate network in Figure 4.3.4, is an approximation, because the travel times follow an general distribution (estimated empirically), hence reduction of the network into a compact network using Norton's theorem does not result in exact results (see also Viswanadham & Narahari (1992) and Lazowska et al. (1984)). We use the AMVA algorithm of Buitenhek et al. (2000) to calculate the throughput times of the CQNs, $\hat{\mu}_r(i)$, see also

Appendix 2.A.
The resulting load-dependent queue representing the replenishment process is shown in Figure 4.3.3, where λ_p is the arrival rate of robots with pods to the replenishment process. In the same way, we can create a load-dependent queue with service rate $\hat{\mu}_p(i)$ representing the pick process.

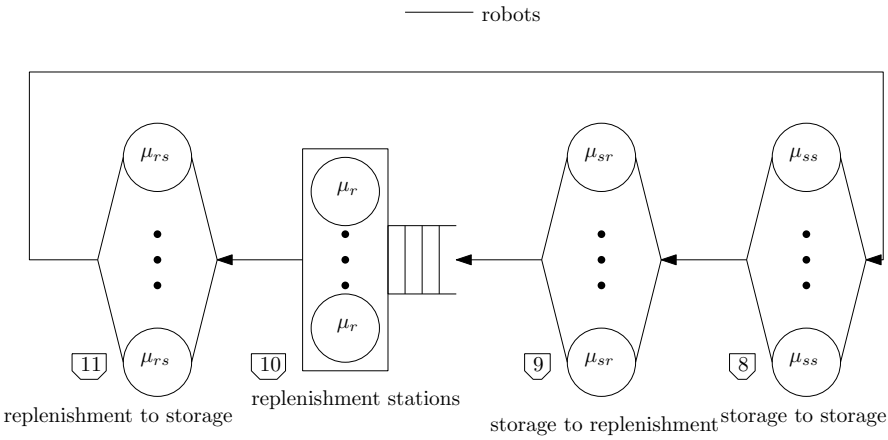


Figure 4.3.2: Part of the complete queueing network associated with the replenishment process

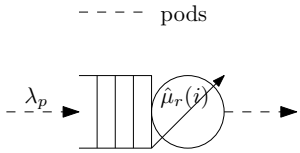


Figure 4.3.3: A load-dependent queue representing the replenishment process in Figure 4.3.2

We then create an intermediate queueing network by replacing queues [8], [9], [10] and [11] in the complete queueing network by the load-dependent queue representing the replenishment process, and by replacing queues [3], [4], [5] and [6] with the load-dependent queue representing the pick process.

The intermediate queueing network is shown in Figure 4.3.4.
In Figure 4.3.4, $\hat{\mu}_p(i)$ is the service rate of the picking process when i pick robots are busy with activities related to picking, and $\hat{\mu}_r(i)$ is the service rate of the replenishment process when i replenishment robots are busy with activities related to replenishment.

The second step is to simplify the intermediate queueing network further in order to arrive at the compact network. A complication is that the arrival rate of robots with pods, λ_p, is unknown for both the pick and replenishment process. In other words, in Figure 4.3.4, we do not know the arrival rate or robots carrying pods for queues [3] and [5], representing the picking and replenishment process respectively. Therefore, we use an approximation to simplify the intermediate network in Figure 4.3.4. Appendix 4.A shows how to transform a queueing network consisting of a synchronization station connected to a load-dependent queue into a queueing network consisting only of a load-dependent queue. It makes it possible to eliminate two of the three synchronization stations in the intermediate queueing network. Queues [2] and [3] in the intermediate queueing network are replaced by a load-dependent queue with the same service rates $\hat{\mu}_p(i)$ as queue [3], while queues [4] and [5] in the intermediate queueing network are replaced by a load-dependent queue with the same service rates $\hat{\mu}_r(i)$ as queue [5]. The resulting queueing network is the compact queueing network shown in Figure 4.3.5. We do not need to actually solve this queueing network, since we are only interesting in calculating $\hat{\mu}_p(i)$ and $\hat{\mu}_r(i)$.

The validation of the compact queueing network with a realistic and detailed simulation of an RMFS is shown in Section 4.6.
The next section introduces the MDP model, which entails adopting new notation. For the sake of convenience, Table 4.3.1 contains an overview of the notation used throughout the main text.

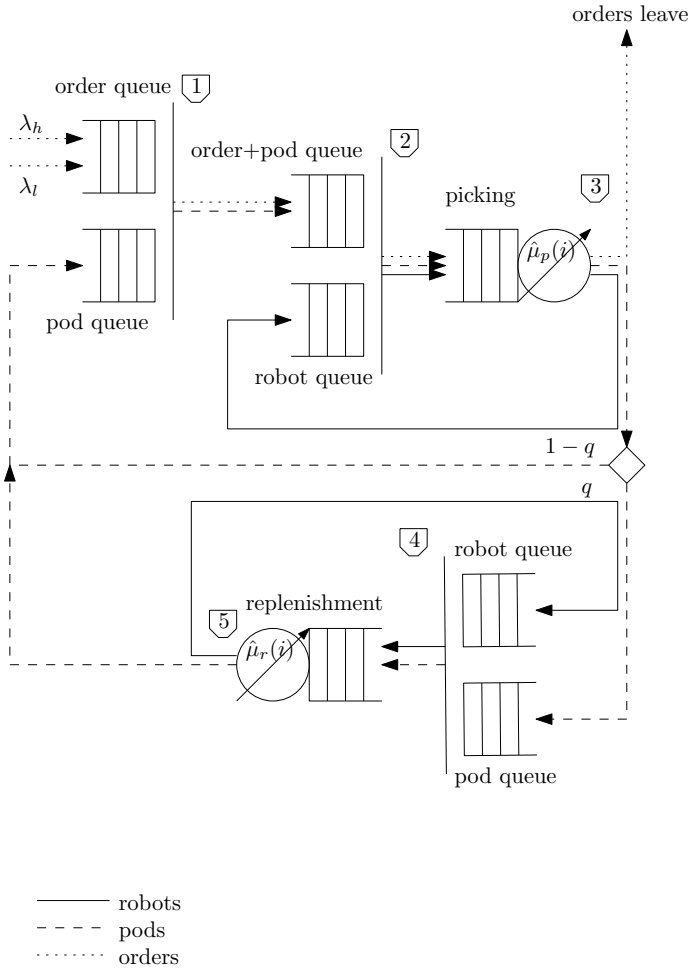


Figure 4.3.4: The Intermediate Queueing Network

Table 4.3.1: Notation overview

Symbol(s)	Meaning
a, A	a is an allocation of workstations and robots to the pick process, and A is the set of all allocations, so $a \in A$
C_{cwt}, C_{ls}	the cost of customer waiting time, and of lost sales, respectively
C_p, C_r	the cost associated with picking and replenishment, respectively
f^*, f^F, f^{DD}, f^{SD}, f^{CD}	the optimal, fixed, demand dependent, state dependent, and cost dependent policies, respectively
M, M_p, M_r	the number of pods in total, in the pick process, and in the replenishment process, respectively
O_σ, O_t	the number of orders at the synchronization station, and the total number of orders in the system, respectively
q	the probability for a pod to go to replenishment after picking
R, R_p, R_r	the number of robots in total, in the pick process, and in the replenishment process, respectively
s, $\mathcal{S}$	s is a state in the MDP and $\mathcal{S}$ is the set of all states, so $s \in \mathcal{S}$
W, W_p, W_r	the number of stations in total, allocated to the pick process, and to the replenishment process, respectively
γ_s	the fraction of orders that are rejected from the system
λ_Δ, λ_h, λ_l	the arrival rate in phase Δ, during a period of high demand, and during a period of low demand, respectively
$\bar{\lambda}$, λ_e	the long term average order arrival rate to the system, and the effective arrival rate (excludes rejected orders), respectively
$\bar{\rho}$	the (long term) average utilization of the systems
μ_p, μ_r	the avg. pick rate and the avg. repl. rate, respectively
μ_{ss}^{-1}, μ_{sp}^{-1}, μ_{ps}^{-1}, μ_{sr}^{-1}, μ_{rs}^{-1}	the avg. robot travel time from storage to storage, storage to pick station, pick station to storage, storage to replenishment station,and replenishment station to storage, respectively
$\hat{\mu}_p^a(i)$, $\hat{\mu}_r^a(i)$	the rates at which a pod leaves the picking process, and the replenishment process, respectively, given that there are i pods and given an allocation a
ν_Δ^{-1}, ν_h^{-1}, ν_l^{-1}	the average duration time of a phase Δ, of a period of high demand, and of a period of low demand, respectively
ψ_s	the stock-out probability for a SKU
ω	the order cap, $O_\sigma \leq \omega$
ζ	parameter used for approximating the stock-out probability

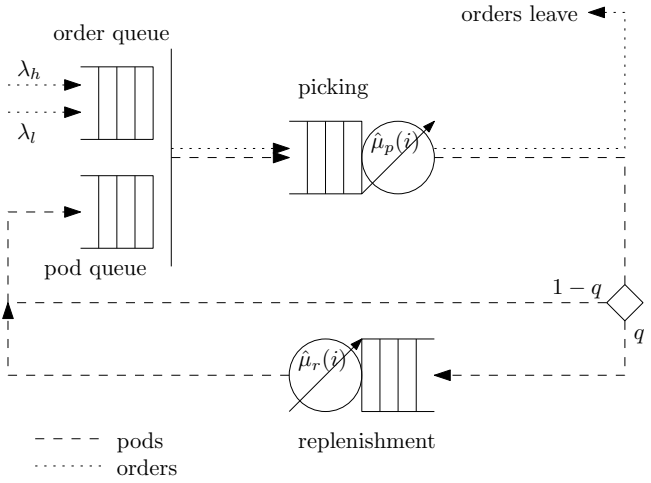


Figure 4.3.5: The Compact Queueing Network

4.4 The Markov Decision Process

This section explains the MDP model used to find the optimal policies for resource reallocation of the workers and robots. It is a discretized MDP with a discrete action space embedded in continuous time with an infinite time horizon. The two resources that can be reallocated are workstations and robots, which can be allocated to either picking or replenishment activities. We will assume that the number of workstations, W, and the number of robots, R, are fixed, so that determining the number of pick stations implicitly determines the number of replenishment stations, and determining the number of pick robots implicitly determines the number of replenishment robots. The number of pick stations is denoted by W_p, the number of replenishment stations by W_r, the number of pick robots by R_p, and the number of replenishment robots by R_r. A resource allocation a describes the number of pick stations, W_p, and the number of pick robots, R_p, that will be used and thus implicitly the number of replenishment stations, W_r, and the number of replenishment robots, R_r. In other words, an allocation a is

given by $a = (W_p, R_p)$, and implicitly sets W_r and R_r as $W_r = W - W_p$ and $R_r = R - R_p$.

4.4.1 A State in the MDP

A state s in the MDP is constructed as follows. The first element that describes a state s is the demand phase Δ, where $\Delta = l$ in the low demand phase of the MMPP and $\Delta = h$ in the high demand phase of the MMPP. The order arrival rate is denoted by λ_Δ, and becomes λ_l and λ_h in the low and high demand phase, respectively. The second element of a state s is O_σ, the number of orders in the order queue that have not been finished yet. The total number of pods in the system is denoted by M, the number of pods in the pick process by M_p, and the number of pods in the replenishment process by M_r. The third element of a state s is M_p and the fourth element is M_r. M_p and M_r indicate the workload of the picking and replenishment process respectively and therefore contain vital information for resource reallocation decisions. The number of pods stored in the storage area not involved in any activities is implicitly given by $M - M_p - M_r$. Taken together, a state s in the MDP is given by $s = (\Delta, O_\sigma, M_p, M_r)$.

In a state s, five different types of transitions are possible with reference to the compact queueing network in Figure 4.3.5. These transitions are denoted by letters A to E: (A) an order arrives to the order queue, (B) the picking of a pod finishes, the order assigned to that pod leaves the system, and the handled pod moves from the picking queue to the pod queue, (C) the picking of a pod finishes, the order assigned to that pod leaves the system, and the handled pod moves from the picking queue to the replenishment queue, (D) the replenishment of a pod finishes and the handled pod moves from the replenishment queue to the pod queue, and (E) the demand phase rate Δ changes. After a transition, if Δ was h, it is l, and if Δ was l, it is h.

In an MDP, at each transition from a state s, an action has to be chosen from the action space $A = \{a_1, \ldots, a_i, \ldots, a_n\}$. Here n is the number of

possible resource allocations. The set A of possible resource reallocations is the same for all states.

Theoretically, the total number of possible resource allocations is $n = (W + 1) \times (R + 1)$, as between 0 and W workstations can be allocated to picking and between 0 and R robots can be allocated to picking. However, not every possible allocation need to be included in A. It is not necessary, for example, to include an allocation with one pick robot and multiple pick stations, as that is quite clearly inefficient. In addition, the action space A should be limited to only a few possible actions for the sake of solving the MDP in numerical experiments.

The compact queueing network in Figure 4.3.5 assumes that the number of pick stations, replenishment stations, pick robots, and replenishment robots is fixed. Therefore, for every action $a \in A$, we need to solve a different queueing network to find the rates at which pods complete the picking and replenishment processes. Given a resource allocation a, the service rates for the picking process are denoted by $\hat{\mu}_p^a(M_p)$ and the service rates for the replenishment process by $\hat{\mu}_r^a(M_r)$.

The transition probabilities can be constructed as follows. For a transition of type (A), the system moves out of a state s with rate λ_Δ, the rate at which orders arrive at the system. For transitions (B) and (C), given a resource allocation a, the rate is $\hat{\mu}_p^a(M_p)$, the rate at which the picking of a pod finishes given that M_p pods are involved in the picking process. Similarly for transition (D), the rate is $\hat{\mu}_r^a(M_r)$, the rate at which replenishment of a pods finishes, given that M_r pods are involved in the replenishment process. Transition (E) occurs with rate ν_Δ, the rate at which the order arrival rate changes in state s. For any state s, the total rate at which the system moves out of s is thus $\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta$ The transition probability $P^a(s'|s)$ denotes the probability to move from a state s to a state s' when choosing resource allocation a.

To ensure that the queueing network and the system are both stable, we will assume that there is a cap ω on the number of orders at the synchronization station. The number of orders in the system, O_t, consists of two components:

(1) the number of orders waiting at the synchronization station in the compact queueing network, O_σ, and (2) the number of orders being processed, which equals the number of pods involved in picking activities, M_p, i.e., pods being transported to or from the pick stations or being handled at the pick stations. In other words, in any state s, $O_t = O_\sigma + M_p$ and $O_\sigma \leq \omega$. If $O_\sigma = \omega$, no orders can enter the system and thus transition (A) cannot occur. If $M_p = 0$, transitions (B) and (C) cannot occur and their transition probability is zero. Similarly, if $M_r = 0$ transition (D) cannot occur and the corresponding transition probability is zero. The transitions and their probabilities are shown in Table 4.4.1. In Table 4.4.1, $\tilde{\Delta}$ is as follows: if $\Delta = h$, then $\tilde{\Delta} = l$, and if $\Delta = l$, then $\tilde{\Delta} = h$. The MMPP arrival process has thus been explicitly woven into the transition probabilities of the MDP.

Table 4.4.1: Possible transitions from a state $s = (\Delta, O_\sigma, M_p, M_r)$, with $\Delta \in \{l, h\}$, and $a \in A$

Type	Condition(s)	Transition probability		
(A)	$O_\sigma = \omega, M_p + M_r = M$	$P^a(\Delta, O_\sigma, M_p, M_r \mid s)$	$=$	$\frac{\lambda_\Delta}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(A)	$O_\sigma < \omega, M_p + M_r = M$	$P^a(\Delta, O_\sigma + 1, M_p, M_r \mid s)$	$=$	$\frac{\lambda_\Delta}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(A)	$O_\sigma < \omega, M_p + M_r < M$	$P^a(\Delta, O_\sigma, M_p + 1, M_r \mid s)$	$=$	$\frac{\lambda_\Delta}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(B)	$O_\sigma = 0, M_p > 0$	$P^a(\Delta, O_\sigma, M_p - 1, M_r \mid s)$	$=$	$\frac{(1-q)\hat{\mu}_p^a(M_p)}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(B)	$O_\sigma > 0, M_p > 0$	$P^a(\Delta, O_\sigma - 1, M_p, M_r \mid s)$	$=$	$\frac{(1-q)\hat{\mu}_p^a(M_p)}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(C)	$M_p > 0$	$P^a(\Delta, O_\sigma, M_p - 1, M_r + 1 \mid s)$	$=$	$\frac{q\hat{\mu}_p^a(M_p)}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(D)	$O_\sigma = 0, M_r > 0$	$P^a(\Delta, O_\sigma, M_p, M_r - 1 \mid s)$	$=$	$\frac{\hat{\mu}_r^a(M_r)}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(D)	$O_\sigma > 0, M_r > 0$	$P^a(\Delta, O_\sigma - 1, M_p + 1, M_r - 1 \mid s)$	$=$	$\frac{\hat{\mu}_r^a(M_r)}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$
(E)	-	$P^a(\tilde{\Delta}, O_\sigma, M_p, M_r \mid s)$	$=$	$\frac{\nu_\Delta}{\lambda_\Delta + \hat{\mu}_p^a(M_p) + \hat{\mu}_r^a(M_r) + \nu_\Delta}$

4.4.2 Solving the MDP Model

The goal is to allocate resources in such a way that the cost associated with customer waiting time and lost sales are minimized. Each workstation has exactly one worker, and since the number of workstations is fixed, wages can be excluded from the analysis. An important cost factor is the customer waiting time, which in an e-commerce setting should be kept as low as possible. Another import cost factor is the availability of inventory, as not having products in stock results in a long delay in the fulfillment of an order, or possibly lost sales. Puterman (1994) argues that the average cost criterion is suitable when decisions are made frequently, because then the discount factor of future costs is close to one. For the MDP in this chapter that is arguably the case, as state transitions typically happen within seconds of each other. Therefore, the optimality criterion used in this chapter is the average costs per time unit. Minimizing the average customer waiting time can be done by minimizing the average number of orders in the system, as the arrival process is given. The order cap ω leads to lost sales when $O_\sigma = \omega$, which creates an additional cost C_{ls} per lost order (lost sales). When $O_\sigma = \omega$, lost sales occur at the order arrival rate λ_Δ. During the high demand phase, the order arrival rate exceeds the system capacity, meaning that some of orders need to be processed during the low demand phase. Another interpretation for ω is that it is the maximum number of orders that can be transferred from the high demand phase to the low demand phase (or vice versa, although that situation should be rather unlikely).

Let $\mathbb{1}^s_{[O_\sigma=\omega]}$ be a function that equals 1 if $O_\sigma = \omega$ in state s of the MDP and 0 otherwise, and let C_{cwt} be the customer waiting time cost per time unit per order. The unavailability of inventory is approximated by using $\frac{M_r}{M}$ since this is the fraction of empty pods relative to the total number of pods. The cost of unavailability is thought of as lost sales: a fraction ψ_s of incoming orders finds that the storage area does not contain the desired product and the customers buy the product elsewhere. In a typical RMFS, each SKU tends to be spread across multiple pods, therefore ψ_s increases nonlinearly

in the number of empty pods. Appendix 4.C discusses in more detail how ψ_s can be derived based on realistic data with multiple SKUs. Appendix 4.C shows that $\left(\frac{M_r}{M}\right)^{\zeta}$ offers a reasonable approximation of the stock-out probability, with ζ a parameter. The formula is shown in Equation (4.1). The order cap together with inventory unavailability mean that the fraction of incoming orders that are rejected from the system, denoted by γ_s, is as given in Equation (4.2). The total cost $C_T(s)$ per time unit to be in state s is given by Equation (4.3). The total cost C_T can be partitioned in a pick component and a replenishment component. Let the costs associated with picking be denoted with C_p and the costs associated with replenishment as C_r. Delays in the picking process causes customer waiting costs, and it may cause orders to be rejected if $\mathbb{1}^s_{[O_\sigma=\omega]} = 1$. Delays in the replenishment process causes lost sales. Hence C_p and C_r are as given in Equations (4.4) and (4.5).

$$\psi_s = \left(\frac{M_r}{M}\right)^{\zeta} \tag{4.1}$$

$$\gamma_s = \max\left(\mathbb{1}^s_{[O_\sigma=\omega]}, \psi_s\right) \tag{4.2}$$

$$C_T(s) = O_t C_{cwt} + \gamma_s \lambda_\Delta C_{ls} \tag{4.3}$$

$$C_p = O_t C_{cwt} + \mathbb{1}^s_{[O_\sigma=\omega]} \lambda_\Delta C_{ls} \tag{4.4}$$

$$C_r = \psi_s \lambda_\Delta C_{ls} \tag{4.5}$$

A policy f describes for every state s which allocation a should be chosen when a transition occurs. For a state s, let π^f_s be the stationary probability to be in state s under a policy f. We want to solve the MDP for an infinite horizon, as we want to analyze the performance of the RMFS in the long run, and hence choose the expected cost minimization criterion. It therefore makes sense to try to minimize the expected costs. If unique stationary state

probabilities π_s^f exist, the expected cost C_A^f per time unit under policy f can be expressed as:

$$C_A^f = \sum_{s \in \mathcal{S}} \pi_s^f C_T(s) \tag{4.6}$$

In Equation (4.6), $\mathcal{S}$ is the set of all states, and $C_T(s)$ is the total cost as described in Equation (4.3). We can now show that the average cost C_A^f per time unit under any policy f exists and is a finite quantity as follows. We have that $O_\sigma \leq \omega$, $M_r \leq M$, $O_t \leq \omega + M$, and $\lambda_\Delta \leq \lambda_h$. Furthermore, the number of states $s = (\Delta, O_\sigma, M_p, M_r)$ is finite as Δ can take two values, $O_\sigma \leq \omega$, and both M_p and M_r can take integer values from 0 to M. Therefore, the total cost $C_T(s)$ and the number of states are bounded as follows:

$$C_T(s) \leq (\omega + M)C_{cwt} + \lambda_h C_{ls} < \infty \tag{4.7}$$

$$|\mathcal{S}| \leq 2 \times \omega \times (M+1) \times (M+1) < \infty \tag{4.8}$$

Equation (4.8) shows that the number of states is finite, which means that the sum in Equation (4.6) is over a finite number of terms. As Equation (4.7) shows that $C_T(s) < \infty$, and since π_s^f is a probability, each of the summation terms in Equation (4.6) is finite. As a finite sum of finite terms is finite, it must hold that $C_A^f < \infty$. In other words, for each policy f the expected cost per time unit is finite.

An optimal policy f^* is a policy that minimizes the average cost per time unit. In other words, the optimality criterion is:

$$f^* \in \underset{f}{\operatorname{argmin}}\ C_A^f \tag{4.9}$$

Unique stationary state probabilities π_s^f exist under policy f, if two conditions are met (Bolch et al., 2006). First of all, the Markov Chain describing the system must be aperiodic. Secondly, given a reference state s_0, the expected time $\tau(s, s_0)$ needed to go from any state s to the reference state s_0, must be bounded by a number $T < \infty$. In other words, $\tau(s, s_0) \leq T < \infty \quad \forall s \in \mathcal{S}$. In addition, the reference state s_0 must be positive recurrent.
For the MDP described above, we can define a reference state $s_0 = (\lambda_l, 0, 0, 0)$. In s_0 the MMPP phase is low, the system contains no orders, and no pods are involved in picking or replenishment. Given this definition of the reference state s_0, the conditions for the existence of the unique stationary state probabilities π_s^f coincide with the assumption that the system is stable under policy f. Therefore, an optimal policy f^* is unique and therefore we can refer to *the* optimal policy f^*. Let X be the Continuous Time Markov Chain describing the system and let X_n be the state of the Markov Chain at time n. The stationary state probabilities π_s^f, independent of the initial state i, are given by:

$$\pi_s^f = \lim_{n \to \infty} P^f \{X_n = s | X_0 = i\} \quad \forall s \in \mathcal{S} \tag{4.10}$$

The stationary state probabilities $\pi_s^{f^*}$ of the optimal policy f^* can be calculated by solving the following Linear Program:

$$\min_{x_{s,a}} \sum_{s\in\mathcal{S}} \sum_{a\in A} x_{s,a} C_T(s) \tag{4.11}$$

subject to

$$\sum_{a\in A} x_{s,a} = \sum_{i\in\mathcal{S}} \sum_{a\in A} x_{i,a} P^a(s|i) \qquad \forall s \in \mathcal{S} \tag{4.12}$$

$$\sum_{i\in\mathcal{S}} \sum_{a\in A} x_{i,a} = 1 \tag{4.13}$$

$$x_{s,a} \geq 0 \qquad \forall s \in \mathcal{S}, \forall a \in A \tag{4.14}$$

We have that $\pi_s^f = \sum_{a\in A_s} x_{s,a}$. In other words $x_{s,a}$ is the steady state fraction of time the system spends in state s and allocation a is chosen. Equation 4.11 is the objective function and minimizes the total costs of the system, which can also be written as $\sum_{s\in\mathcal{S}} \pi_s^f C_T(s)$. Constraint 4.12 incorporates the transitions between states. Constraint 4.13 ensures that in every state an allocation is chosen and constraint 4.14 ensures that allocations cannot be chosen a negative number of times. By solving the linear program, we obtain $x_{s,a}^*$, the values for $x_{s,a}$ that minimize the costs. From $x_{s,a}^*$ we can calculate $\pi_s^{f^*} = \sum_{a\in A_s} x_{s,a}^*$, and we can also construct the optimal policy f^*. The optimal policy f^* is to choose allocation a in state s with a probability equal to $x_{s,a}^*$.

Instead of using Linear Programming, it is also possible to use a Dynamic Programming approach. The Bellman equation corresponding to the MDP in this chapter is given in Equation (4.15):

$$v_t^*(s) = \min_{a\in A} \left\{ C_T(s) + \sum_{s'\in\mathcal{S}} P^a(s|s') v_{t-1}^*(s') \right\} \tag{4.15}$$

In Equation (4.15), t indicates the time period and $v_t^*(s)$ is the expected total cost of the system, if the system is in state s between at time point t. The Dynamic Program can be solved using well-known methods such as policy iteration or value iteration.

4.4.3 Benchmark Policies

To understand the benefits of using the optimal policy f^*, we will compare the minimal costs under policy f^* with four benchmark policies. Two of these benchmark policies are derived from practice, and two are customized for the MDP in this chapter. The four benchmark policies are: (1) a fixed policy f^F that always uses the same allocation a^F regardless of the state s, (2) a demand-dependent policy f^{DD} that always uses an allocation a_h^{DD} for states with a high arrival rate and an allocation a_l^{DD} for states with a low arrival rate, (3) a cost-dependent policy f^{CD} that always uses an allocation a_p^{CD} if $C_p \geq C_r$, and a_r^{CD} if $C_p < C_r$, and (4) a state-dependent policy f^{SD}, where for each allocation we evaluate expected total cost after a transition. More specifically, given the current state s, let the set $\mathcal{S}'$ be the set of all possible states after a transition, and let $P^a(s'|s)$ be the probability to transit from the current state s to another state s'. The expected total cost $\tilde{C}(a)$ of an allocation a is then as given in Equation (4.16). The state-dependent policy f^{SD} chooses the allocation a^{SD} that results in the lowest expected total costs after the next transition, as shown in Equation (4.17).

$$\tilde{C}(a) = \sum_{s' \in \mathcal{S}'} C_T(s') P^a(s'|s) \tag{4.16}$$

$$a^{SD} = \arg\min_{a \in A} \tilde{C}(a) \tag{4.17}$$

The fixed policy f^F is similar to an RMFS where workstations and robots are dedicated to either picking or replenishment, whereas the demand-dependent policy f^{DD} is similar to an RMFS where workstations and robots are only dedicated to picking or replenishment during work shifts, but can be reallocated between shifts. The state-dependent policy f^{SD} chooses an allocation where the ratio between pick rates and replenishment rates matches the ratio between the costs associated with picking and the costs associated with replenishment as closely as possible. Some allocations a will reduce the number of waiting orders O_t by allocating the majority of workstations and robots to the picking workload, whereas other allocations a will primarily reduce the number of pods waiting for replenishment, M_r. Naturally, some allocations a will strike a balance between the picking and replenishment workloads. The fixed policy f^F always chooses a balanced allocation, whereas the demand-dependent policy chooses an allocation that mainly reduces O_t when the MMPP phase is high and an allocation that mainly reduces M_r when the MMPP phase is low. The main benefit of policy f^* may lie in the fact that it can choose to spend most workers and robots on picking when O_t is high even when the MMPP phase is low.

4.5 Average Arrival Rate and Stability Conditions

This section establishes necessary stability conditions for the system studied in this chapter. The MMPP contains two phases, (1) low and (2) high, with respective stationary probabilities $\hat{\pi} = (\hat{\pi}_l, \hat{\pi}_h)$. The duration of these phases are exponentially distributed, with ν_l^{-1}, the average time for the low phase, and ν_h^{-1}, the average time for the high phase. The corresponding generator matrix is:

$$\mathbf{\Gamma} = \begin{bmatrix} -\nu_l & \nu_l \\ \nu_h & -\nu_h \end{bmatrix} \tag{4.18}$$

It must hold that $\hat{\pi}\mathbf{\Gamma} = 0$ and $\hat{\pi}\mathbf{1} = 1$. Solving the corresponding system of equations leads to Equation (4.19).

$$\hat{\pi}_h = \frac{\nu_h^{-1}}{\nu_h^{-1} + \nu_l^{-1}}, \qquad \hat{\pi}_l = \frac{\nu_l^{-1}}{\nu_h^{-1} + \nu_l^{-1}} \tag{4.19}$$

The average arrival rate $\bar{\lambda}$ is the average order arrival rate in the long run across all demand periods. The effective arrival rate λ_e is the long term average rate at which the orders are actually admitted to the system. The effective arrival rate λ_e takes into account that orders can become lost sales and do not enter the system, and it therefore depends on the chosen policy. The average arrival rate $\bar{\lambda}$ and the effective arrival rate λ_e are defined in Equation (4.20) and (4.21), respectively. It naturally holds that $\bar{\lambda} \geq \lambda_e$.

$$\bar{\lambda} = \hat{\pi}_h \lambda_h + \hat{\pi}_l \lambda_l \tag{4.20}$$

$$\lambda_e = \sum_{s \in \mathcal{S}} \pi_s^f (1 - \gamma_s) \lambda_\Delta \tag{4.21}$$

For assessing upfront whether the system can be stable, λ_e cannot be used as it depends on the operational policy. Instead, we use $\bar{\lambda}$ to assess whether the system can be stable. Naturally, if $\bar{\lambda}$ is smaller than the order throughput capacity, TH of the system, $\bar{\lambda} \leq TH$, then the stability of the system is assured. However, the throughput of the system depends on the allocation a of both types of resources.

Let TH^a be the system throughput if the allocation of resources is fixed to allocation a and the allocation does not change. In order to find TH, we eliminate the synchronization station from the compact queueing network in Figure 4.3.5 to arrive at the Closed Queueing Network in Figure 4.5.1. In the latter queueing network, pods do not have to wait for orders and hence

the throughout is higher than in the former queueing network. Given an allocation a and M, let $TH^a(M)$ denote the throughout rate of the Closed Queueing Network in Figure 4.5.1. The network in Figure 4.5.1 can be analyzed with a single-class exact Mean Value Analysis (MVA) for Load-dependent queues as can be found in Jia (2005) and Buitenhek et al. (2000), and this yields $TH^a(M)$.

pods

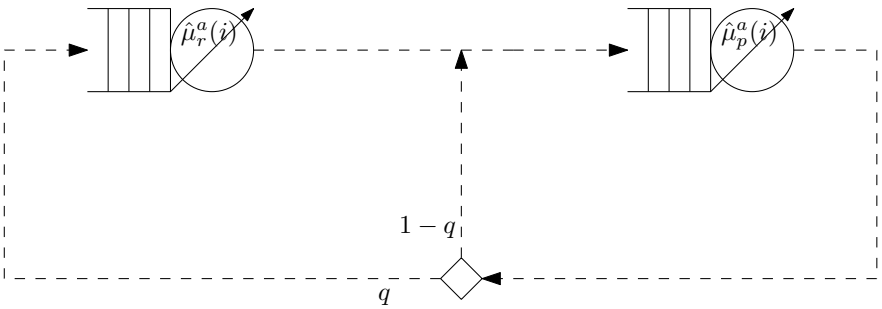


Figure 4.5.1: Closed queueing network based on the compact queueing network in Figure 4.3.5

TH_{max}, as given in Equation (4.22), is the maximum throughput the system can achieve if a fixed resource allocation is chosen, and TH_{min} is the minimum.

$$TH_{max} = \max_{a \in A} TH^a(M) \tag{4.22}$$

$$TH_{min} = \min_{a \in A} TH^a(M) \tag{4.23}$$

The system is stable under any resource allocation if $\bar{\lambda} \leq TH_{min}$. If it is possible to choose the resource allocation upfront, then a resource allocation

can be chosen such that the system is stable if $\bar{\lambda} \leq TH_{max}$. It is therefore possible to calculate for any fixed policy f^F, whether the system will be stable. Moreover, we can draw further conclusions with regards to existence of a demand-dependent policy that results in a stable system, as shown in Theorem 3.

Theorem 3. *If a fixed policy f^F exists, such that $\bar{\lambda} \leq TH_{max}$, meaning that the system is stable, then there also exists a demand-dependent policy f^{DD} for which the system is stable.*

Proof. For a fixed policy f^F, a^F is the allocation used and $TH^{a^F}(M)$ is the order throughput capacity of the system under f^F. Let f^F be such that $TH^{a^F}(M) = TH_{max}$ and let the system be stable under f^F, i.e. $\bar{\lambda} \leq TH_{max}$. For a demand-dependent policy f^{DD}, let the order throughput capacity be denoted with $TH^{(f^{DD})}$. If we choose the demand-dependent policy f^{DD} to be such that $a_h^{DD} = a^F$ and $a_l^{DD} = a^F$, then $TH^{(f^{DD})} = TH^{a^F}(M)$ as the same action would be chosen in each state for both f^{DD} and f^F. We can therefore choose a demand-dependent policy f^{DD} such that $TH^{(f^{DD})} = TH_{max}$, and since $\bar{\lambda} \leq TH_{max}$ the system would be stable under f^{DD}. □

4.6 Numerical Experiments

This section describes the numerical experiments. Subsection 4.6.1 describes the first numerical experiment, called the "small instance experiment", where the optimal policy is compared with the benchmark policies on a set of small instances. An explanation of the policies can be found in Section 4.4.3. The aim of this experiment is to be able to compare the benchmark policies with an optimal policy to obtain an impression of their performance. The optimal

policy can only be determined if the number of states is small, which limits on the number of pods M and the order cap ω. Both M and ω are small as a result.

Subsection 4.6.2 describes the second numerical experiment, a comparison of the benchmark policies on a set of large instances. This experiment is called the "large instance experiment", because the size of the systems in terms of number of pods, number of workstations, and size of the layouts, are sufficiently large to represent real systems. For these large instances, the optimal policy could not be determined, but nevertheless the experiment leads to some new insights due to the benchmark policies.

Throughout all experiments, the same layouts, parameters, and resource allocations are used. Three layouts are used in total, a small, a medium, and a large layout. The layouts are described in Table 4.6.1 and depicted in Figures 4.6.1 and 4.6.2. In Figure 4.6.1, the numbers next to the workstations indicate the order of activation. For example, if there are two replenishment stations in our MDP model, that means that on the left side of Figure 4.6.1 the workstations with a 1 and 2 next to them will be used as replenishment stations, while with three replenishment stations, the replenishment station with a 3 next to it would also be used as a replenishment station.

In Table 4.6.1, μ_{ss}^{-1}, μ_{sp}^{-1}, μ_{ps}^{-1}, μ_{sr}^{-1}, and μ_{rs}^{-1} are the average travel times, as described in Section 4.3.2. The corresponding travel time variances are denoted by $\hat{V}_{ss}$, $\hat{V}_{sp}$, $\hat{V}_{ps}$, $\hat{V}_{sr}$, and $\hat{V}_{rs}$. We have that $\mu_{sp}^{-1} = \mu_{ps}^{-1} = \mu_{sr}^{-1} = \mu_{rs}^{-1}$ and that $\hat{V}_{sp} = \hat{V}_{ps} = \hat{V}_{sr} = \hat{V}_{rs}$, due to the symmetry of the layouts, e.g. the placement of the pick stations around the storage area mirrors the placements of the replenishment stations around the storage area in all layouts. The order arrival rates differ across layouts and across experiments and are shown in Table 4.6.2.

Table 4.6.2 shows the parameters that are used throughout the numerical experiments. It includes the parameter needed for calculating the empirical travel time distributions, i.e. to calculate μ_{ss}^{-1}, μ_{sp}^{-1}, μ_{ps}^{-1}, μ_{sr}^{-1}, μ_{rs}^{-1}, $\hat{V}_{ss}$, $\hat{V}_{sp}$,

Table 4.6.1: Layouts variables (with block size: 2x4)

Name	M	W	Aisles	$\times$	Cross-aisles	μ_{ss}^{-1}	$\hat{V}_{ss}$	$\mu_{sp}^{-1} = \mu_{ps}^{-1} = \mu_{sr}^{-1} = \mu_{rs}^{-1}$	$\hat{V}_{sp} = \hat{V}_{ps} = \hat{V}_{sr} = \hat{V}_{rs}$
Small	550	4	8	$\times$	8	18.4 s	90.0 s^2	34.5 s	118.8 s^2
Medium	1149	6	12	$\times$	12	26.6 s	188.3 s^2	45.3 s	250.0 s^2
Large	1965	8	16	$\times$	16	34.8 s	322.5 s^2	56.1 s	429.2 s^2

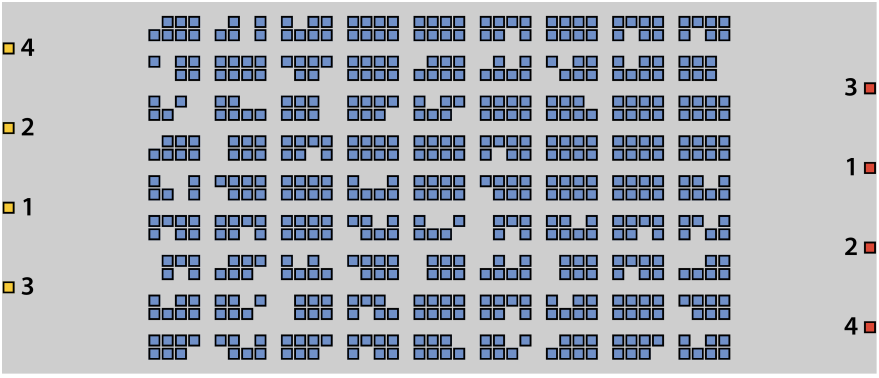
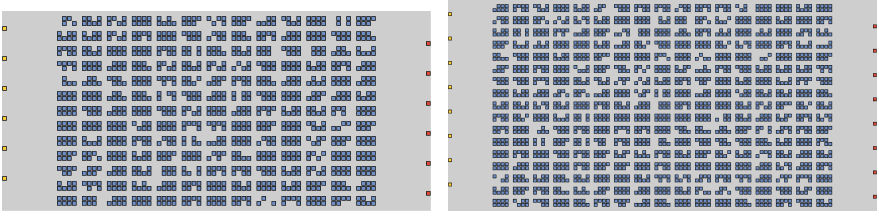


Figure 4.6.1: Top view of the small layout, including station activation sequence numbering, with the storage area in the middle, replenishment stations to the left, and pick stations to the right



(a) Medium layout **(b)** Large layout

Figure 4.6.2: Top view of the medium and large layout, with the storage area in the middle, replenishment stations to the left, and pick stations to the right

$\hat{V}_{ps}$, $\hat{V}_{sr}$, and $\hat{V}_{rs}$. These parameters are the robot acceleration, robot deceleration, the average robot speed, the maximum robot speed, the time needed to lift and to store a pod, and the time a robot needs to make a 90 degree turn. We estimated these parameters by observing real implementations of RMFSs.

In addition, Table 4.6.2 also includes the other parameters needed for the queueing networks, namely the probability q that a pod needs to go for replenishment after visiting a pick station, the average pick time (μ_p^{-1}) and the average replenishment time (μ_r^{-1}) needed for the complete queueing network. The q is exogenous to the model we present, and can be observed in existing implementations. It is, however, not the probability to directly go to a replenishment station after picking. A pod that has entered the replenishment process, shown in Figure 4.3.2, may spend a considerable amount of time in store before it is transported to a replenishment station by a robot. In other words, q is the probability that a pod will need replenishment, or can be needed to replenish a product running low on inventory. We estimated these parameters from observations at an RMFS at a Dutch retailer and found that $q = 20\%$.

The parameters in Table 4.6.2 that are related to the costs are C_{cwt}, C_{ls}, and ζ. The stock-out probability ζ is derived in Appendix 4.C. For the cost of a lost sale, we estimate that in an E-commerce environment the cost is on average 20 € per order. The customer waiting in an E-commerce environment stems from the idea that if a customer has to wait too long, the customer may cancel the order and buy the product(s) elsewhere, either online or in a retail store, or may be less inclined to place a future order with the company. Assuming that a waiting time of 24 hours is equivalent to a lost sale, the cost of customer waiting time is roughly 0.00023 € per order per second.

The results for both the large instance experiment and the small instance experiment are generated by running a discrete event simulation of the corresponding Markov Decision Process. The state in this simulation is s and the transition to another state occurs as detailed in Table 4.4.1. Table 4.6.2 show the simulation time per simulation run and the number of simulation

runs. The simulation time is 31536000 seconds, which equals one year and is sufficiently long for the system to have reached a long-run equilibrium.
Table 4.6.2 also contains parameters that differ between the large instance experiment and the small instance experiment, namely the order cap ω and the average arrival rate $\bar{\lambda}$. As mentioned earlier, in the small instance experiment, the order cap ω needs to be kept small. However, in real systems, an order is not rejected due to some order cap, but is instead temporarily put in a backlog until the system can process it. Therefore, in the large instance experiment the order cap should have a minimal influence, and consequently is set to a large value.
The order arrival rates differ across the layouts, because the travel times differ strongly across the layouts and this affects the order throughput capacity of the system. The order arrival rates also differ between the two experiments, because in the small instance experiment the number of pods M is small and this reduces the order throughput capacity. The order arrival rates were chosen such that the number of picking completion events (transitions (B) and (C)) where (nearly) equal to the number of replenishment completion events (transition (D)) under the state-dependent policy.
Lastly, Table 4.6.3 shows the resource allocations, labeled a_1 to a_{11}, and how these resource allocations translate to number of pick stations and pick robots for each layout. The ratios and fractions used in these resource allocations are based on our observations at an RMFS implementation at a Dutch retailer.

4.6.1 Small Instance Experiment Results

Table 4.6.5 shows a comparison of the optimal policy with the four benchmark policies for small instances, i.e. instances with low M and small ω. The M and ω are low to keep the number of states in the MDP sufficiently small that it can be solved to optimality. The optimal policy was determined with the open source MDP toolbox from INRA, which is described in Chadès et al. (2014).

Table 4.6.2: Parameters used throughout the experiments

Parameter	Value	Parameter	Value
Pod storage time	3.0 s	Pod lift time	3.0 s
Avg. robot speed	1.3 $\frac{m}{s}$	Max. robot speed	1.5 $\frac{m}{s}$
Robot acceleration	0.5 $\frac{m^2}{s}$	Robot deceleration	0.5 $\frac{m^2}{s}$
Robot turn time	2.5 s	Prob. repl. after picking (q)	20.0%
Avg. pick time (μ_p^{-1})	10.0 s	Avg. repl. time (μ_r^{-1})	30.0 s
Number of simulation runs	10 runs	Simulation time	31536000 s
Cost of waiting time (C_{cwt})	0.00023 €/order/s	Cost per lost sale (C_{ls})	20.0 €/order
Stock-out probability parameter ζ	7		
Small instance experiment		Large instance experiment	
ω	20 orders	ω	2000000 orders
$\bar{\lambda}$, small layout	360.0 orders/h	$\bar{\lambda}$, small layout	468.0 orders/h
$\bar{\lambda}$, medium layout	432.0 orders/h	$\bar{\lambda}$, medium layout	540.0 orders/h
$\bar{\lambda}$, large layout	504.0 orders/h	$\bar{\lambda}$, large layout	612.0 orders/h

Table 4.6.3: Resource allocations a per layout, $a = (W_p, R_p)$

	Small Layout ($W = 4$)			Medium Layout ($W = 6$)			Large Layout ($W = 8$)		
	$R = 16$	$R = 24$	$R = 32$	$R = 24$	$R = 36$	$R = 48$	$R = 32$	$R = 48$	$R = 64$
$a_1 = (W_p = 0, R_p = 0)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
$a_2 = (W_p = \lfloor 0.45W \rfloor, R_p = \lfloor 0.45R \rfloor)$	(1, 7)	(1, 10)	(1, 14)	(2, 10)	(2, 16)	(2, 21)	(3, 14)	(3, 21)	(3, 28)
$a_3 = (W_p = \lfloor 0.45W \rfloor, R_p = \lfloor 0.60R \rfloor)$	(1, 9)	(1, 14)	(1, 19)	(2, 14)	(2, 21)	(2, 28)	(3, 19)	(3, 28)	(3, 38)
$a_4 = (W_p = \lfloor 0.45W \rfloor, R_p = \lfloor 0.75R \rfloor)$	(1, 12)	(1, 18)	(1, 24)	(2, 18)	(2, 27)	(2, 36)	(3, 24)	(3, 36)	(3, 48)
$a_5 = (W_p = \lfloor 0.60W \rfloor, R_p = \lfloor 0.45R \rfloor)$	(2, 7)	(2, 10)	(2, 14)	(3, 10)	(3, 16)	(3, 21)	(4, 14)	(4, 21)	(4, 28)
$a_6 = (W_p = \lfloor 0.60W \rfloor, R_p = \lfloor 0.60R \rfloor)$	(2, 9)	(2, 14)	(2, 19)	(3, 14)	(3, 21)	(3, 28)	(4, 19)	(4, 28)	(4, 38)
$a_7 = (W_p = \lfloor 0.60W \rfloor, R_p = \lfloor 0.75R \rfloor)$	(2, 12)	(2, 18)	(2, 24)	(3, 18)	(3, 27)	(3, 36)	(4, 24)	(4, 36)	(4, 48)
$a_8 = (W_p = \lfloor 0.75W \rfloor, R_p = \lfloor 0.45R \rfloor)$	(3, 7)	(3, 10)	(3, 14)	(4, 10)	(4, 16)	(4, 21)	(6, 14)	(6, 21)	(6, 28)
$a_9 = (W_p = \lfloor 0.75W \rfloor, R_p = \lfloor 0.60R \rfloor)$	(3, 9)	(3, 14)	(3, 19)	(4, 14)	(4, 21)	(4, 28)	(6, 19)	(6, 28)	(6, 38)
$a_{10} = (W_p = \lfloor 0.75W \rfloor, R_p = \lfloor 0.75R \rfloor)$	(3, 12)	(3, 18)	(3, 24)	(4, 18)	(4, 27)	(4, 36)	(6, 24)	(6, 36)	(6, 48)
$a_{11} = (W_p = W, R_p = R)$	(4, 16)	(4, 24)	(4, 32)	(6, 24)	(6, 36)	(6, 48)	(8, 32)	(8, 48)	(8, 64)

For each policy f in Table 4.6.4, results were generated via a simulation of the Markov Decision Process, where allocations were chosen according to the policy f. For each policy f, 10 simulation runs were performed and the average costs C_A^f are an average across these 10 simulation runs. For each of the five policy categories, namely F, DD, CD, SD, and O (Optimal), we examine the average costs of the policies in that category and report the lowest cost among these policies. For example, Table 4.6.4 shows five fixed policies, namely F_1 to F_5, where the expected costs per second are $C_A^{F_1}, \ldots, C_A^{F_5}$ respectively. We denote the minimal cost in a policy category X as $C_A^{X^*}$. For the F, DD, and CD policy categories, the minimal cost is given in Equations (4.24), (4.25), and (4.26), respectively. The SD and O categories only have one policy each, so the minimal cost in that category equals the average cost per second C_A of the one policy in that category.

$$F^* = \arg\min \left(C_A^{F_1}, \ldots, C_A^{F_5} \right) \tag{4.24}$$

$$DD^* = \arg\min \left(C_A^{DD_1}, \ldots, C_A^{DD_9} \right) \tag{4.25}$$

$$CD^* = \arg\min \left(C_A^{CD_1}, \ldots, C_A^{CD_9} \right) \tag{4.26}$$

The different settings in Table 4.6.5 include the three layouts (with corresponding number of robots) and three MMPP scenarios, creating nine settings in total. In the first MMPP scenario, the low demand phase lasts on average 4 times as long as the high demand phase, while during the high demand phase 4 times as many orders arrive compared to the low demand phase. It is a MMPP scenario where the differences between the high and the low phase are moderate. In the second MMPP scenario, the ratio is further skewed to 1:7. This MMPP scenario aims to represent a normal workday, where customers have three hours of spare time in the evening during which they shop online. In contrast, the rest of the day is the low demand phase. The third MMPP scenario does not aim to represent one day, but rather

one week. The high demand phase represents the weekend, when customers have more time to pursue online shopping, whereas the low demand phase represents the work week.

The results in Table 4.6.5 show the 95% confidence intervals across the 10 simulation runs of the best policies per category. The confidence intervals are the smallest for MMPP scenario 1, and the largest for MMPP scenario 3. The best cost-dependent policy leads to higher costs on average than the best policies from the other categories. The higher costs are caused by the small value for order cap ω; the ω should be set to a high number as discussed earlier, and as is done for the large instance experiment, but is set to a small value here to limit the number of states. The small value for ω has a disproportionate effect on the picking costs that the cost-dependent policy uses, which means it will favor allocating resources to the picking process over replenishment process. However, with just $M = 30$ pods, replenishment costs due to stock-out will consequently become larger. In other words, the small value for ω distorts the cost-dependent policies.

Also, for the small layout, the optimal policy does seem to clearly outperform the best policies from the other categories. However, for the other layouts, the confidence intervals of the optimal policy and the best F, DD, and SD policies mostly overlap. Figure 4.6.3 provides an overview of the average cost across all policies. There is a group of policies that lead to low costs and another group that leads to high costs, with a large gap in between the two groups. However, across MMPP scenarios the performance does not vary much, indicating that the policies are robust against time-varying demand.

The main observation from the results in Table 4.6.5 is that confidence intervals of the benchmark policies mostly overlap with the confidence intervals from the optimal policies. In other words, the benchmark policies perform similarly to the optimal policy. We can therefore focus on the benchmark policies for real-life larger instances.

Table 4.6.4: Overview of policies used in the small instance experiment. The SD policy is given in Equation 4.17, and allocations $a_1, \ldots, a_{11}$ can be found in Table 4.6.3.

Policy	Allocation	Policy	Allocation	Policy	Allocation
F_1	$a^F = a_1$	DD_1	$a_l^{DD} = a_1$, $a_h^{DD} = a_6$	CD_1	$a_p^{CD} = a_1$, $a_r^{CD} = a_6$
F_2	$a^F = a_2$	DD_2	$a_l^{DD} = a_1$, $a_h^{DD} = a_{10}$	CD_2	$a_p^{CD} = a_1$, $a_r^{CD} = a_{10}$
F_3	$a^F = a_6$	DD_3	$a_l^{DD} = a_1$, $a_h^{DD} = a_{11}$	CD_3	$a_p^{CD} = a_1$, $a_r^{CD} = a_{11}$
F_4	$a^F = a_{10}$	DD_4	$a_l^{DD} = a_2$, $a_h^{DD} = a_6$	CD_4	$a_p^{CD} = a_2$, $a_r^{CD} = a_6$
F_5	$a^F = a_{11}$	DD_5	$a_l^{DD} = a_2$, $a_h^{DD} = a_{10}$	CD_5	$a_p^{CD} = a_2$, $a_r^{CD} = a_{10}$
		DD_6	$a_l^{DD} = a_2$, $a_h^{DD} = a_{11}$	CD_6	$a_p^{CD} = a_2$, $a_r^{CD} = a_{11}$
		DD_7	$a_l^{DD} = a_6$, $a_h^{DD} = a_6$	CD_7	$a_p^{CD} = a_6$, $a_r^{CD} = a_6$
		DD_8	$a_l^{DD} = a_6$, $a_h^{DD} = a_{10}$	CD_8	$a_p^{CD} = a_6$, $a_r^{CD} = a_{10}$
		DD_9	$a_l^{DD} = a_6$, $a_h^{DD} = a_{11}$	CD_9	$a_p^{CD} = a_6$, $a_r^{CD} = a_{11}$

Table 4.6.5: Results for the small instance experiment: The 95% confidence intervals of the costs for the benchmark policies and optimal policy (costs in € per second), with $M = 30$, and $\omega = 20$, # states = 2232

Layout	R	MMPP	ν_h^{-1}	ν_l^{-1}	λ_h:λ_l	$C_A^{F^*}$	$C_A^{DD^*}$	$C_A^{SD^*}$	$C_A^{CD^*}$	$C_A^{O^*}$
Small	16	1	3 h	12 h	1:4	[0.50, 0.52]	[0.50, 0.52]	[0.50, 0.52]	[0.61, 0.64]	[0.45, 0.47]
Medium	24	1	3 h	12 h	1:4	[0.61, 0.64]	[0.61, 0.65]	[0.58, 0.60]	[0.72, 0.76]	[0.56, 0.60]
Large	32	1	3 h	12 h	1:4	[0.78, 0.82]	[0.76, 0.79]	[0.79, 0.81]	[0.88, 0.91]	[0.77, 0.79]
Small	16	2	3 h	21 h	1:7	[0.67, 0.73]	[0.64, 0.69]	[0.67, 0.72]	[0.70, 0.74]	[0.63, 0.66]
Medium	24	2	3 h	21 h	1:7	[0.80, 0.87]	[0.80, 0.87]	[0.79, 0.85]	[0.90, 0.94]	[0.79, 0.86]
Large	32	2	3 h	21 h	1:7	[1.02, 1.09]	[1.02, 1.08]	[0.98, 1.05]	[1.08, 1.18]	[1.00, 1.09]
Small	16	3	36 h	132 h	3:11	[0.47, 0.54]	[0.49, 0.55]	[0.44, 0.50]	[0.57, 0.72]	[0.41, 0.47]
Medium	24	3	36 h	132 h	3:11	[0.54, 0.61]	[0.53, 0.65]	[0.53, 0.58]	[0.70, 0.80]	[0.51, 0.56]
Large	32	3	36 h	132 h	3:11	[0.69, 0.84]	[0.69, 0.81]	[0.72, 0.87]	[0.81, 0.96]	[0.71, 0.86]

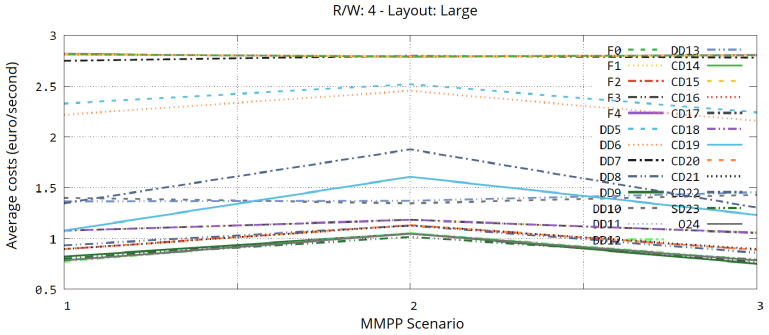


Figure 4.6.3: Average costs for policies in the small instance experiment across MMPP Scenarios

4.6.2 Large Instance Experiment Results

Table 4.6.6 shows the results for the four benchmark policies on large instances. For the sake of brevity, no confidence intervals are shown. The number of states for these instances is so large that solving the MDP optimally is not computationally feasible. The policies used are described in Table 4.6.7. The policies vary widely in their costs and can become as large as 350 € per second, which indicates an unsustainable, large order backlog. Under such policies, the system is clearly unstable.

The best policies in each policy category, namely F^*, DD^*, CD^*, and SD^* and the associated minimal costs for each policy category, namely $C_A^{F^*}$, $C_A^{DD^*}$, $C_A^{CD^*}$, and $C_A^{SD^*}$, are calculated as described in Section 4.6.1 for the small instance experiment.

Besides a larger ω and a higher number of pods M in each layout, namely the values as shown in Table 4.6.1, the large instance experiment contains more MMPP scenarios than the small instance experiment. Also, the number of robots per workstation, denoted by R/W, is varied. MMPP scenarios 8, 9, and 10 are meant to represent a typical week, with the weekend represented

by the high demand phase as customers shop more in the weekend, and the work week represented by the low demand phase.
First of all, we can see that, for the same layout and policy category, the average costs differ widely across the MMPP scenarios. This cost difference shows that not only the average order arrival rate matters, but also the length of a period of peak demand and the height of the peak affect costs. For different MMPP scenarios, different policies and types of policies lead to the lowest cost. Secondly, we can see that with four robots per workstation the costs are much higher, indicating that when only four robots are present, orders have to wait much longer to be fulfilled. However, for the best CD policy, the costs are typically only a fraction of the costs under the F and DD policies, showing that resource reallocation can reduce costs sharply when the number of robots is limited. In other words, even when the number of robots is sufficient to process all orders on average, the number of robots and the policy employed can have a strong, non-linear impact on the customer waiting time. Lastly, in all cases either the state-dependent policy or a cost-dependent policy deliver the lowest average costs. The state-dependent policy and the cost-dependent policies exploit the fact that they can dynamically reallocate resources, whereas the fixed policies and the demand-dependent policies do not. Therefore, dynamic resource reallocation can contribute to lowering costs in an RMFS. Moreover, the fixed policy and the demand-dependent policy can be applied in other contexts, but the state-dependent policy and cost-dependent policy are specific for the RMFS. An RMFS may therefore provide value to E-commerce companies by lowering costs when deploying the state-dependent or a cost-dependent policy.

4.7 Conclusions

Warehouses have to balance resources between order picking and inventory replenishment tasks. This is particularly important in online retail environments, which suffer from high demand peaks and where customers require

Table 4.6.6: Results for the large instance experiment (costs in € per second)

					Small Layout				Medium Layout				Large Layout			
MMPP	ν_h^{-1}	ν_l^{-1}	λ_h:λ_l	R/W	$C_A^{F^*}$	$C_A^{DD^*}$	$C_A^{SD^*}$	$C_A^{CD^*}$	$C_A^{F^*}$	$C_A^{DD^*}$	$C_A^{SD^*}$	$C_A^{CD^*}$	$C_A^{F^*}$	$C_A^{DD^*}$	$C_A^{SD^*}$	$C_A^{CD^*}$
1	3 h	3 h	1:1	4	26.35	22.80	0.03	0.05	21.92	0.86	0.01	0.01	47.28	2.18	0.05	0.08
1	3 h	3 h	1:1	6	< 0.01	0.12	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
1	3 h	3 h	1:1	8	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
2	3 h	6 h	1:2	4	25.56	52.62	2.06	0.33	23.17	23.98	0.72	0.29	50.30	54.26	3.51	0.41
2	3 h	6 h	1:2	6	0.02	0.47	< 0.01	0.02	0.02	0.02	< 0.01	0.01	0.03	0.04	< 0.01	0.02
2	3 h	6 h	1:2	8	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
3	3 h	9 h	1:3	4	28.67	68.35	4.09	0.68	29.35	47.64	1.87	0.52	49.54	82.73	5.85	0.68
3	3 h	9 h	1:3	6	0.15	0.59	0.09	0.13	0.13	0.25	0.06	0.11	0.19	0.22	0.07	0.13
3	3 h	9 h	1:3	8	0.09	0.07	0.03	0.04	0.02	0.02	< 0.01	0.02	0.03	0.03	< 0.01	0.02
4	3 h	12 h	1:4	4	31.39	77.75	7.51	5.10	32.94	62.88	3.45	4.81	54.69	98.75	9.35	2.57
4	3 h	12 h	1:4	6	0.30	0.63	0.22	0.27	0.29	0.42	0.16	0.23	0.40	0.34	0.20	0.27
4	3 h	12 h	1:4	8	0.20	0.18	0.10	0.12	0.08	0.13	0.04	0.08	0.11	0.15	0.05	0.10
5	3 h	15 h	1:5	4	33.59	86.01	7.49	6.76	31.49	73.78	6.53	7.15	56.39	> 100	11.64	7.36
5	3 h	15 h	1:5	6	0.42	0.73	0.39	0.39	0.47	0.49	0.31	0.47	0.64	0.48	0.39	0.64
5	3 h	15 h	1:5	8	0.33	0.36	0.17	0.22	0.16	0.29	0.11	0.16	0.20	0.31	0.12	0.18
6	3 h	18 h	1:6	4	29.81	93.24	7.70	8.11	32.13	79.84	6.38	8.69	48.69	> 100	13.89	9.66
6	3 h	18 h	1:6	6	0.56	0.92	0.55	0.52	0.71	0.59	0.49	0.71	0.93	0.72	0.53	0.93
6	3 h	18 h	1:6	8	0.40	0.60	0.28	0.29	0.26	0.53	0.19	0.26	0.31	0.39	0.21	0.31
7	3 h	21 h	1:7	4	34.78	96.49	11.28	9.47	33.26	82.01	9.54	10.50	49.63	> 100	12.89	11.49
7	3 h	21 h	1:7	6	0.61	1.15	0.75	0.60	0.98	0.75	0.61	0.98	1.04	1.05	0.70	1.04
7	3 h	21 h	1:7	8	0.48	0.77	0.40	0.38	0.35	0.57	0.29	0.35	0.44	0.45	0.27	0.44
8	36 h	132 h	3:11	4	36.50	20.10	37.13	5.24	52.89	1.58	21.18	5.89	64.04	3.16	24.04	6.66
8	36 h	132 h	3:11	6	0.75	1.23	2.18	0.88	2.60	1.40	2.06	2.60	3.73	1.46	2.67	3.73
8	36 h	132 h	3:11	8	0.51	0.53	1.07	0.48	0.65	0.96	0.57	0.33	0.88	1.07	0.50	0.35
9	48 h	120 h	2:5	4	34.42	32.89	18.63	3.13	27.00	1.42	14.89	1.06	69.79	2.86	26.84	1.39
9	48 h	120 h	2:5	6	0.38	0.46	0.79	0.38	0.98	1.33	0.57	0.29	1.33	1.34	0.71	0.32
9	48 h	120 h	2:5	8	0.28	0.28	0.06	0.06	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
10	60 h	107 h	5:9	4	28.68	47.88	11.61	0.59	34.47	12.32	10.00	0.66	46.36	18.81	19.71	0.81
10	60 h	107 h	5:9	6	0.11	0.13	< 0.01	0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.09	0.09	< 0.01	0.01
10	60 h	107 h	5:9	8	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

Table 4.6.7: Overview of policies used in the large instance experiment. The SD policy is given in Equation 4.17, and allocations $a_1, \ldots, a_{11}$ can be found in Table 4.6.3.

Policy	Allocation	Policy	Allocation	Policy	Allocation
F_1	$a^F = a_2$	DD_1	$a_l^{DD} = a_1$, $a_h^{DD} = a_7$	CD_1	$a_p^{CD} = a_7$, $a_r^{CD} = a_1$
F_2	$a^F = a_3$	DD_2	$a_l^{DD} = a_1$, $a_h^{DD} = a_9$	CD_2	$a_p^{CD} = a_7$, $a_r^{CD} = a_2$
F_3	$a^F = a_4$	DD_3	$a_l^{DD} = a_1$, $a_h^{DD} = a_{10}$	CD_3	$a_p^{CD} = a_7$, $a_r^{CD} = a_3$
F_4	$a^F = a_5$	DD_4	$a_l^{DD} = a_1$, $a_h^{DD} = a_{11}$	CD_4	$a_p^{CD} = a_7$, $a_r^{CD} = a_5$
F_5	$a^F = a_6$	DD_5	$a_l^{DD} = a_2$, $a_h^{DD} = a_7$	CD_5	$a_p^{CD} = a_9$, $a_r^{CD} = a_1$
F_6	$a^F = a_7$	DD_6	$a_l^{DD} = a_2$, $a_h^{DD} = a_9$	CD_6	$a_p^{CD} = a_9$, $a_r^{CD} = a_2$
F_7	$a^F = a_8$	DD_7	$a_l^{DD} = a_2$, $a_h^{DD} = a_{10}$	CD_7	$a_p^{CD} = a_9$, $a_r^{CD} = a_3$
F_8	$a^F = a_9$	DD_8	$a_l^{DD} = a_2$, $a_h^{DD} = a_{11}$	CD_8	$a_p^{CD} = a_9$, $a_r^{CD} = a_5$
F_9	$a^F = a_{10}$	DD_9	$a_l^{DD} = a_3$, $a_h^{DD} = a_7$	CD_9	$a_p^{CD} = a_{10}$, $a_r^{CD} = a_1$
		DD_{10}	$a_l^{DD} = a_3$, $a_h^{DD} = a_9$	CD_{10}	$a_p^{CD} = a_{10}$, $a_r^{CD} = a_2$
		DD_{11}	$a_l^{DD} = a_3$, $a_h^{DD} = a_{10}$	CD_{11}	$a_p^{CD} = a_{10}$, $a_r^{CD} = a_3$
		DD_{12}	$a_l^{DD} = a_3$, $a_h^{DD} = a_{11}$	CD_{12}	$a_p^{CD} = a_{10}$, $a_r^{CD} = a_5$
		DD_{13}	$a_l^{DD} = a_5$, $a_h^{DD} = a_7$	CD_{13}	$a_p^{CD} = a_{11}$, $a_r^{CD} = a_1$
		DD_{14}	$a_l^{DD} = a_5$, $a_h^{DD} = a_9$	CD_{14}	$a_p^{CD} = a_{11}$, $a_r^{CD} = a_2$
		DD_{15}	$a_l^{DD} = a_5$, $a_h^{DD} = a_{10}$	CD_{15}	$a_p^{CD} = a_{11}$, $a_r^{CD} = a_3$
		DD_{16}	$a_l^{DD} = a_5$, $a_h^{DD} = a_{11}$	CD_{16}	$a_p^{CD} = a_{11}$, $a_r^{CD} = a_5$

very short response times. Too many resources in replenishment may lead to delays in fulfilment. However, too many resources allocated to picking, may deplete inventory and ultimately lead to even larger delays in fulfilment. We model this problem for robotic mobile fulfillment systems, which are popular in online retail warehouses. In such a system two different resources, robots and workers, work together to fill orders and replenish inventory. Resources can rapidly switch between tasks (with no or little setup), based on demand and inventory levels. We build an MDP model embedded in a queuing network model that allows to take optimal decisions, for small instances, minimizing operational, customer wait, and lost sales cost. For larger, real-life size instances we find heuristic allocation policies that are close to optimal for small instances. Four heuristic policies are evaluated. The fixed policy keeps the resource allocation fixed, whereas the demand-dependent policy has two different resource allocations, one for high demand and one for low demand. The cost-dependent policy also has two different resource allocations, depending on the ratio of picking to replenishment costs. Finally, the state-dependent policy uses the transition probabilities of the MDP model to estimate the expected cost of an allocation and chooses the one with lowest cost. The cost-dependent and state-dependent policies outperform the fixed and the demand-dependent policies. Continually reallocating resources based on the state of the system (order demand rate, allocation, and number of waiting orders) appears to bring substantial cost savings. The benefits are even more pronounced when the number of robots is small, because then the cost-dependent and/or state-dependent policies can sharply reduce costs compared to the fixed and demand-dependent policies. We also show that the characteristics of peak demand have a strong effect on the costs. Given a fixed average order arrival rate across all demand phases, we varied the length of the peak demand phase and the height of the peak, and found that this influences both the average costs but also the type of policy that minimizes costs. To minimize costs, a system should deploy different resource allocation policies depending on the duration and height of peak demand. The method developed in this chapter to cope with non-stationary demand and dynamic

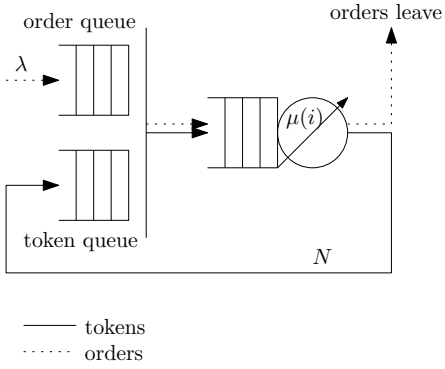
reallocation of resources may be deployed rapidly to other handling systems. Automated systems, such as automated-guided vehicle systems and robotic systems may be the most suitable, as these resources can continually switch between different processes without setup cost, based on software control. For robotic mobile fulfillment systems, an interesting area of future research would be to include the repositioning process in the method. Robots may reposition pods stored within the storage area in order to sort the inventory. Repositioning may reduce the time needed to retrieve pods, and hence reduce the customer waiting time, but repositioning tasks also add to the workload of the robots.

Appendix

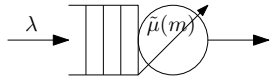
4.A Synchronization with a Load-dependent Queue

This section shows how to transform a queueing network consisting of a synchronization station with a load-dependent queue into a queueing network consisting of only a load-dependent queue. The transformation is a rather accurate approximation, but it is not exact. A queueing network consisting of a synchronization station and a load-dependent queue is shown in Figure 4.A.1a. Figure 4.A.1b shows a queueing network consisting of only a load-dependent queue.

In the queueing network in Figure 4.A.1a, orders arrive with a rate λ at the synchronization station, where the orders are matched with so-called tokens. After being matched with a token the order-token pair goes to a load-dependent queue. If there are i order-token pairs at the load-dependent queue, then the service rate is $\mu(i)$. When service at the load-dependent queue finishes, an order-token pair leaves the load-dependent queue. The order of that order-token pair leaves the system and the token itself returns to the synchronization station. The system contains a total of N tokens. The state space of this system can be described as (j, i), where j represents the number of orders waiting at the synchronization station and i represents the number of order-token pairs at the load-dependent queue. The number of



(a) A queueing network with a synchronization station and a load-dependent queue



(b) A queueing network with only a load-dependent queue

Figure 4.A.1: Queueing Networks

tokens waiting at the synchronization station is denoted by k, with $k = N - i$. If $j > 0$ then it follows that $k = 0$, otherwise an order and token can be matched and go to the load-dependent queue. Similarly, if $k > 0$ then it follows that $j = 0$. Table 4.A.1 shows the transitions from a state (j, i) to another state. Time is considered to be continuous, so it is not possible to increase or decrease j at the exact same moment that i is increased or decreased, and vice versa, nor can the state remain the same after a transition. At each transition either j changes or i changes but not both simultaneously. However, the system is not a Markov chain, because the memoryless property does not apply. If an order arrives at the synchronization station and matches with a token, it moves to the load-dependent queue, where service time is reset / redrawn. However, if an order arrives at the synchronization station and no token awaits, the order stays at the synchronization station and the service time at the load-dependent queue is not reset / redrawn, so the memoryless property does not apply in this situation. Therefore, when $j > 0, i = N$, the transition rate is approximately $\mu(N)$ rather than exactly $\mu(N)$.

Figure 4.A.1b shows a queueing network with only a load-dependent queue, where the service rate is $\tilde{\mu}(m)$ if m orders are waiting at the queue. If we would set the service rates μ as $\mu = \tilde{\mu}$, then the two networks are equivalent when $i \leq N$, but diverge when $i > N$, because the queueing network in Figure 4.A.1a does not have the memoryless property in that case, whereas the queueing network in Figure 4.A.1b does.

Table 4.A.1: Transition rates from state (j, i) to another state x

State (j,i)	$x = (j+1, i)$	$x = (j-1, i)$	$x = (j, i+1)$	$x = (j, i-1)$
$j = 0, i = 0$	0	0	λ	0
$j = 0, 0 < i < N$	0	0	λ	$\mu(i)$
$j = 0, i = N$	λ	0	0	$\mu(N)$
$j > 0, i = N$	λ	(approx.) $\mu(N)$	0	0

4.B Validation of the Queueing Networks

In this section the compact queueing network shown in Figure 4.3.5 is validated with a realistic simulation specifically built for simulating RMFSs. Our simulation framework is called "RAWSim-O" (Merschformann et al. (2017a)) and builds on the work of Hazard et al. (2006a). Merschformann et al. (2018) use RAWSim-O to experiment with different decision rules and policies across multiple decision problems, including pick order assignment, replenishment order assignment, pod storage assignment, pick pod selection, and replenishment pod selection. RAWSim-O therefore includes numerous algorithms for assigning tasks to robots, assigning orders to stations, determining where to store pods, and determining the pods to be transported to the stations. These algorithms are described in Merschformann et al. (2017a) and Merschformann et al. (2018).

RAWSim-O is an agent-based and event-driven simulation, that incorporates realistic robot movement, including physically accurate deceleration and acceleration, blocking by other robots, and advanced path planning algorithms. It

keeps track of all units of all SKUs on all pods, keeps track of all the pick orders and replenishment orders in the system and their due dates, and keeps track of a large variety of KPIs. It incorporates lifting and storing of pods, allows for multiple floors witin the RMFS warehouse, and RMFS specific features like the repositioning pods. Merschformann et al. (2017b) provides a more detailed description of RAWSim-O and the source code of the simulation framework itself is available at `https://github.com/merschformann/RAWSim-O`.
RAWSim-O simulates real-time operations in an RMFS warehouse and may therefore serve to validate the performance of the compact queueing network shown in Figure 4.3.5. Table 4.B.1 shows the results of the validation, where RAWSim-O simulated 24 hours of RMFS operations. We compare the order cycle time, denoted by t_{oc}, which is the time between an order arriving at the pick process and it leaving the pick process. There are some differences between the situation that RAWSim-O simulates and the situation that the compact queueing network models. The queueing network assumes the robots always travel with a constant speed, with no traffic jams, whereas RAWSim-O simulates realistic robot movement, including traffic jams. Moreover, whereas the queueing networks only have pick orders, RAWSim-O also includes replenishment orders, that are fulfilled by bringing empty pods to the replenishment stations. In other words, for RAWSim-O the probability for a pod to go for replenishment after visiting a pick station, denoted by q, is endogenous to the simulations of RAWSim-O, rather than an exogenously observed parameter as it is in the queueing network. In addition, RAWSim-O include multi-line, multi-unit pick orders, and when a pod visits a pick station, a picker may pick multiple units to fulfill lines from multiple pick orders. Lastly, in RAWSim-O workstations have a limited capacity for orders, and the generation of pick orders and replenishment orders is stopped and restarted based on the storage space utilization, see also Merschformann et al. (2018). This is different from the queueing networks, where picking and replenishment never halt and orders arrive continually.
We ran a 1080 simulations in RAWSim-O, namely 108 different settings, i.e. 108 sets of different parameters, and 10 runs per setting. In these simulations,

the average speed per robot varied around just slightly more than 1 m/s and q varied widely but was typically much larger than the 20% we propose in Table 4.6.2. We decided to select a subset of these simulations where $27\% \leq q \leq 31\%$ and where the average robot speed was within $[1.04, 1.06]$ $\frac{m}{s}$. We then analyzed the queueing network with $q = 29\%$ and an average robot speed of 1.05 m/s. The results are shown in Table 4.B.1. The arrival rate, number of robots, and allocation are derived from the simulation output of RAWSim-O and were also used in the queueing network, the number of SKUs is 1000.

We can see that there are two different settings; one case where the medium layout is used and one where the large layout was used. In the former, the order cycle time difference between RAWSim-O and the queueing networks is roughly 10%, whereas in the latter it is nearly 40%. In other words, for one setting, the queueing network is relatively close to RAWSim-O, whereas for the other the difference is too large to be practially useful. However, the main issue is that the q was much higher in most of RAWSim-O's simulations than the 20% we estimated from our observations at a Dutch retailer, which limited the settings we could use for validation.

4.C Stock-out Probability

The stockout probability ψ_s is the probability that a SKU is not in the storage area when an order arrives at the warehouse. In this chapter we aggregate all SKUs into one SKU, so technically a stockout only happens when there is not a single unit left in the storage area, but this is not realistic. Therefore this appendix shows how to calculate the stockout probability ψ_s given a set of SKUs, in a more realistic fashion.

Lamballais et al. (2018b) show that it is beneficial to spread the inventory of a SKU across multiple pods, as this decreases the average travel time of a SKU to a workstation. Let SKU c be distributed across m_c different pods, then that SKU is stocked out if all m_c pods on which the SKU is located

Table 4.B.1: Order cycle times of the compact queueing network and of RAWSim-O

Run	Layout	a	R	λ (orders / h)	t_{oc} RAWSim-O (s)	t_{oc} Queueing Network (s)
1	medium	a_7	12	227.583	51.137	45.148
2	medium	a_7	12	226.375	51.422	45.266
3	medium	a_7	12	227.708	51.275	45.124
4	medium	a_7	12	227.333	51.218	45.264
5	medium	a_7	12	227.583	51.217	45.149
6	medium	a_7	12	228.292	51.125	45.031
7	medium	a_7	12	227.625	51.202	45.196
8	medium	a_7	12	227.000	51.516	45.183
9	medium	a_7	12	227.917	51.301	44.998
10	medium	a_7	12	228.625	51.216	45.091
1	large	a_7	32	485.625	64.665	35.358
2	large	a_7	32	485.042	64.389	35.389
3	large	a_7	32	481.833	64.382	35.469
4	large	a_7	32	485.167	64.420	35.379
5	large	a_7	32	483.917	64.478	35.398
6	large	a_7	32	487.875	64.478	35.334
7	large	a_7	32	483.208	64.376	35.407
8	large	a_7	32	485.125	64.500	35.389
9	large	a_7	32	484.333	64.280	35.391
10	large	a_7	32	484.333	64.192	35.345

are empty. The number of pods is M, and the number of empty pods is M_r. If $m_c > M_r$, then the stockout probability for SKU c is zero, since there is at least one pod left in the inventory, with the SKU. If $m_c \leq M_r$, then the stockout probability for SKU c is calculated as follows. If $m_c = 1$, then a stockout happens with probability $\frac{M_r}{M}$. If $m_c = 2$, then a stockout happens with probability $\frac{M_r}{M} \times \frac{M_r-1}{M-1}$, because the probability that the second pod does not contain SKU c given that the first does not contain SKU c is $\frac{M_r-1}{M-1}$. More generally, a stockout for SKU c occurs with probability:

$$\prod_{i=0}^{m_c-1} \frac{M_r - i}{M - i}, \qquad m_c \leq M_r \tag{4.27}$$

Let $\mathcal{C}$ be the set of all SKUs, let c indicate a SKU, let ϕ_c be the probability that an order needs a unit of SKU c, and let SKU c be distributed across m_c number of pods, and let $\mathbb{1}_{m_c \leq M_r}$ be an indicator function that is one if $m_c \leq M_r$ and zero otherwise. Across all SKUs, the stockout probability ψ_s in state s is then given by Equation (4.28).

$$\psi_s = \sum_{c \in \mathcal{C}} \phi_c \mathbb{1}_{m_c \leq M_r} \left(\prod_{i=0}^{m_c-1} \frac{M_r - i}{M - i} \right), \text{ with } M_r \in s \tag{4.28}$$

Figure 4.C.1 shows five different stock-out probability curves as a function of the fraction of pods to be replenished, $\frac{M_r}{M}$. The first stock-out probability curve, $\psi^{(1)}$, shows what happens if we have two classes of SKUs that are somewhat dissimilar, with $\phi_c = [0.6, 0.4]$ and $m_c = [40, 60]$. The second stock-out probability curve, $\psi^{(2)}$, models the ABC curve, where 20% of the products account for 70% of demand ("A" class products), 30% of products account for 25% of demand ("B" class products), and the remaining 50% of products accounts for 5% of demand ("C" class

products). We model this by setting $\phi_c = [0.7, 0.25, 0.05]$ and $m_c = [20, 30, 50]$. For the third stock-out probability curve, $\psi^{(3)}$, we created 10 classes that are quite different from one another, for the sake of variety. We set $\phi_c = [0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.19]$ and $m_c = [19, 17, 15, 13, 11, 9, 7, 5, 3, 1]$. To see whether we can capture create a similar curve with a simpler function, we also show two curves that are a simply a power of $\frac{M_r}{M}$, namely $\tilde{\psi}^{(4)} = \left(\frac{M_r}{M}\right)^3$ and $\tilde{\psi}^{(5)} = \left(\frac{M_r}{M}\right)^7$. Figure 4.C.1 shows that $\tilde{\psi}^{(5)}$ provides a compromise between $\psi^{(1)}$ and $\psi^{(2)}$ on the one hand and $\psi^{(3)}$ on the other hand. Figure 4.C.1 shows $\psi^{(1)}$, $\psi^{(2)}$, and $\psi^{(3)}$ with continuous lines, because they have been calculated with Equation (4.28), whereas $\tilde{\psi}^{(4)}$ and $\tilde{\psi}^{(5)}$ are shown with dotted lines, because they were calculated with $\left(\frac{M_r}{M}\right)^\zeta$ instead, where ζ is a parameter. We therefore posit that, without any further, specific information about ϕ_c and m_c, $\tilde{\psi}^{(5)}$ offers a reasonable approximation of the stock-out probability as a function of the fraction of pods needing replenishment, i.e. where $\zeta = 7$.

For systems with a large number of SKUs, that are all equally frequently ordered, we propose the following approximation shown Proposition 4. Since in E-commerce environments the order frequency of fast movers is quite different from the order frequency of slow movers, Proposition 4 does not hold for the warehouse environment in the current study.

Proposition 4. *Given that $m_c = m$, $\phi_c = \phi \forall c \in \mathcal{C}$, and that $m \leq M_r$, it holds for single-line orders that $\psi_s = \binom{M_r}{m} / \binom{M}{m}$, where $|\mathcal{C}|$ is the number of SKUs.*

Proof. Equation (4.27) can be rewritten as shown in Equation (4.29), and Equation (4.28) as Equation (4.31). If $m > M_r$ then $\psi_s = 0$. However, if $m \leq M_r$, then Equation (4.31) can be written as Equation (4.32). Since ϕ_c is the probability that an order needs a unit of SKU, and orders are assumed to be single-line orders, we have that $\sum_c \phi_c = 1$. Moreover, since ϕ_c is the

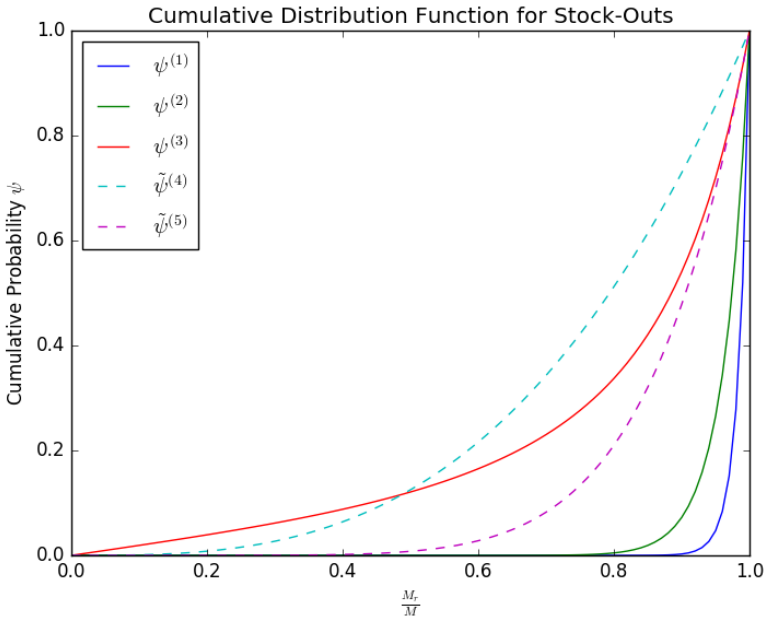


Figure 4.C.1: Cumulative Distribution of the stock-out probability ψ_s, as a function of the fraction of empty pods, $\frac{M_r}{M}$

same probability ϕ, we have that $\sum_c \phi_c = \sum_c \phi = |\mathcal{C}|\phi = 1$. Therefore, Equation (4.32) can be written as Equation (4.33).

$$\prod_{i=0}^{m_c-1} \frac{M_r - i}{M - i} = \frac{M_r!(M - m_c)!}{M!(M_r - m_c)!} = \frac{M_r!(M - m)!m!}{M!(M_r - m)!m!} = \frac{\binom{M_r}{m}}{\binom{M}{m}} \tag{4.29}$$

$$\psi_s = \sum_{c \in \mathcal{C}} \phi_c \mathbb{1}_{m_c \leq M_r} \left(\prod_{i=0}^{m_c-1} \frac{M_r - i}{M - i} \right) \tag{4.30}$$

$$= \sum_{c \in \mathcal{C}} \phi \mathbb{1}_{m \leq M_r} \left(\frac{\binom{M_r}{m}}{\binom{M}{m}} \right) \tag{4.31}$$

$$= \frac{\binom{M_r}{m}|\mathcal{C}|\phi}{\binom{M}{m}} \tag{4.32}$$

$$= \frac{\binom{M_r}{m}}{\binom{M}{m}} \tag{4.33}$$

□

5 Decision Rules

5.1 Introduction

The rise of e-commerce has created the need for new warehousing systems. Traditional, manual picker-to-parts systems work best when orders are large, i.e. consist of many SKUs so that consolidation has to be organized well. However, e-commerce orders are typically small and e-commerce warehouses are often large as they need to contain large assortments of products, which results in long walking distances for the pickers. In contrast to manual picker-to-part systems, automated parts-to-picker systems eliminate the time pickers spend traveling. Thus, they can achieve higher pick rates.
The Robotic Mobile Fulfillment System (RMFS) is an automated parts-to-picker system. Robots transport movable shelves, called "pods", that contain the inventory, back and forth between the storage area and the workstations. As RMFSs eliminate picker walking time, high pick rates can be expected. Implementations suggest that pick rates can improve substantially compared to manual picker-to-parts operations, see also Wulfraat (2012). The systems are mainly used by Amazon, which bought the company that invented the RMFS, Kiva Systems, and has since deployed it in its warehouses (Business Wire (2015)). Recently, competitors such as Swisslog, Interlink, GreyOrange, Mobile Industrial Robots and Scallog have been rolling out their versions of an RMFS.
The RMFS is described in more detail in Wurman & Enright (2011) and Wurman et al. (2008). They mention that numerous operational decisions

problems are yet to be examined in depth, for example the assignment of customer orders to workstations or of pods to a storage locations. Each of these decision problems comes with a trade-off. An order may be assigned to a workstation if it is nearing its due time, but assigning another order that has lines in common with other orders assigned to that workstation may result in more picks per pod and hence a reduction in the number of pod trips. Furthermore, assigning a pod to a storage location that is close to the workstation reduces travel time, but keeping the inventory sorted by assigning pods to favorable storage location if they are likely to be needed in the near future may reduce travel times more.

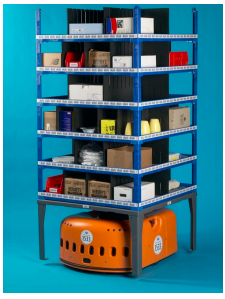
These trade-offs are linked to the number of robots in the system. As an example, with more robots, more trips can be done and hence the order due times can become a more important criterion than the number of picks per pod when selecting a pod to be transported to a workstation. The trade-offs are also linked to the resources and conditions in the warehouse. For example, the more SKUs a warehouse contains, the more difficult it becomes to assign orders to pick stations in such a way that multiple products can be picked from a single pod.

As these examples indicate, a need exists for finding methods to address the decision problems in an RMFS, for research on the performance of RMFSs across performance measures, and for examining performance while varying aspects like the number of robots. This chapter addresses this need. We study the pick order assignment, replenishment order assignment, pick pod selection, replenishment pod selection, and pod storage assignment decision problems and propose several decision rules for each. To see which trade-offs in performance may exist, we use different performance measures. Furthermore, we vary three aspects of the RMFS, namely whether or not return orders need to be processed, the size of the orders, and the number of SKUs in the warehouse. This study focuses on both the pick process and the replenishment process, because a more efficient replenishment process frees up robots for pick tasks. Lastly, the number of pick stations and the number of robots per pick station is varied. Varying these numbers shows

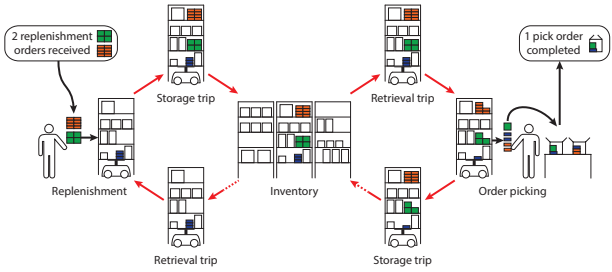
how many pick stations and robots are needed to provide pickers with a near continuous supply of pods.
Section 5.2 describes the RMFS in more detail, Section 5.3 points out related work, Section 5.4 the decision problems, Section 5.5 the decision rules, and Section 5.6 describes the realistic simulation built for evaluating the decision rules, while Section 5.7 explains the evaluation framework, Section 5.8 shows the results of the analysis, and Section 5.9 provides conclusions and directions for future research.

5.2 The Robotic Mobile Fulfillment System

An RMFS consists of shelves on which products are stored (called pods), robots that can move underneath and also carry them (see Figure 5.2.1a), and work stations. After handling a pod at a station it can be returned to a different storage location than where it was retrieved from, hence, inventory can be sorted continuously throughout the day.



(a) Robot carrying a pod (Wurman & Enright, 2011)



(b) The internal storage / retrieval process in RMFSs (red: robot & pod movement

Figure 5.2.1: The essential elements of an RMFS

Figure 5.2.1b shows the storage and retrieval processes, where the robots transport pods between the workstations and the storage area. Starting at

the replenishment station, in the example, two replenishment orders with 4 and 8 units of two SKUs (green & orange) are stored on a pod that was retrieved from the inventory by a robot. The blue SKU also relevant to the process is already available on the pod in focus. After the pod was handled at the station it is stored in inventory again. Next, if the pod is selected for picking at a pick station, it is brought to that station. The operator at the station then picks the units matching the open order lines at the station from the pod and puts them into the bins for the respective pick orders. As soon as a pick order is completed it leaves the pick station and is handled by further warehouse systems. If zoning is in place at the warehouse, the pick order may only be a part of a larger customer order and must be consolidated further with the other partial pick orders in a following sortation process. If the customer order is already completely fulfilled at the pick station, it may be packed into a carton and prepared for shipping immediately with no further handling. The latter may only be possible in e-commerce operations where lines per order are small.

Each pair of storage and retrieval trip is one robot cycle in an RMFS. During one cycle the robot does not set-down or leave the rack until it is returned to a storage location. Note that, the pod may be brought to further replenishment or pick stations between the retrieval and the storage trip, if further replenishment or immediate picking can be done with it. For the sake of clarity we limited the visits per cycle to one station in the example above. While the operation of the robot is cyclic the flow of the inventory units through the system starts at a replenishment station (by storing a replenishment order) and exits at a pick station (by fulfilling a pick order). However, in contrast to other systems there is quite some overhead inventory movement, because all contained units, not only needed ones, are moved when a pod is brought to a station. The same happens during replenishment operations, if non-empty pods are moved to a replenishment station.

Robots navigate their paths through the warehouse using a waypoint system, which is laid out as a grid. A path is a sequence of connected waypoints and all robots have to be guided concurrently along their paths while avoiding

collisions and deadlocks. Robots that are not carrying a pod can move underneath stationary pods and hence take other paths than robots that do carry pods, because the latter cannot use occupied storage locations. The system layout is depicted in Figure 5.2.2 and consists of a storage area where the pods are stored, pick and replenishment stations grouped around the storage area, maneuvering areas between the storage area and the workstations, and per workstation a buffer area. A robot carries a pod from the storage area, via the maneuvering area, to the buffer area of the destination workstation. Only one pod is picked or replenished simultaneously. Workers at the replenishment stations replenish the pods with new inventory. In contrast, workers at the pick stations pick product units to fulfill orders. A picker picks for multiple unfinished/incomplete pick orders at the same time. For both operations the robots need to stop with a pod at a waypoint representing the access point of the respective station. In the buffer area next to each workstation, robots carrying pods can wait for their turn. In the middle of the layout a number of waypoints is used as possible storage locations where pods can be put when they are not used. Every storage location is directly reachable from an aisle and access to a storage location cannot be blocked by stored pods. Travel in the aisles is single-directional to avoid gridlock and reduce congestion.
The system has the ability to adapt to changing demand conditions. E.g., if order arrival rates of some SKUs drop, pods containing those SKUs can be relocated further away from the pick stations. This relocation frees up storage locations near the pick stations for pods containing SKUs with high order arrival rates. Pods can be relocated when returning from a workstation, hence the inventory can be continually sorted in response to changing demand.

5.3 Related Work

To this date no detailed discrete event simulation based research has been done for RMFS. Moreover, most research on RMFSs to date uses queueing

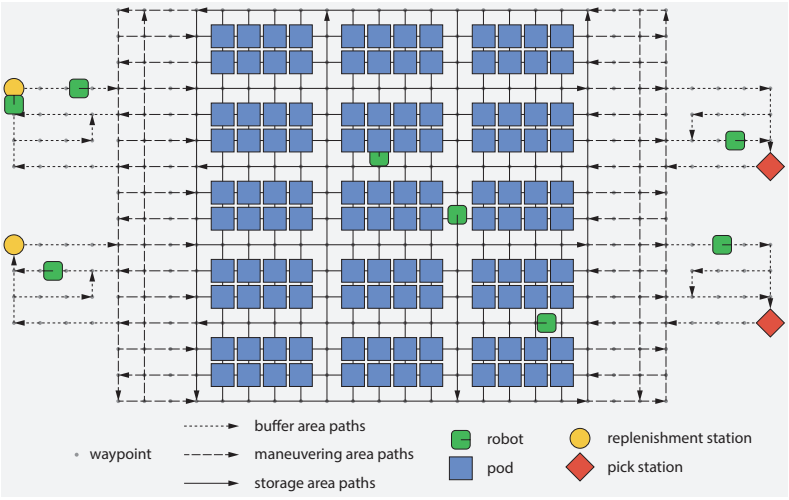


Figure 5.2.2: A top view of an RMFS layout

networks to study design questions on the strategic level. This chapter aims to close the gap by delivering insights about RMFS using a very detailed simulation framework that integrates most dynamic effects an operator faces. Next, we first outline the queuing network based research and close this section with simulation based work.

Nigam et al. (2014) create queueing networks similar to earlier queueing networks used for autonomous vehicle storage and retrieval systems (AVS/RS) and automated storage and retrieval systems (AS/RS) (see Heragu et al. (2011) and Roy et al. (2012)). Their queueing networks capture both pick and replenishment operations but cannot model robot movement realistically. They estimate the order throughput time for single-line orders. Lamballais et al. (2017) create a different queueing network for both single- and multi-line orders, with and without zoning in the storage area, that captures only the pick operations, but that does include realistic robot movement. Their model can accurately estimate the expected order cycle time, workstation utilization and robot utilization. Lamballais et al. (2017) determine how the storage area dimensions and the workstation placement around the storage

area affect the maximum order throughput, by evaluating a large number of possible designs. Lamballais et al. (2018b) develop a queueing network that addresses problems on a tactical level. They show the effect of the number of pods per SKU and of the replenishment level of a pod on order throughput, and they show what the optimal ratio of the number of pick stations to the number of replenishment stations is. They find that it is better to replenish pods before they are entirely empty, even with multiple pods per SKU. Zou et al. (2017) use semi-open queueing networks to analyze the policy for assigning robots to pick stations. The authors find that the random policy is significantly outperformed by the proposed handling-speeds-based assignment rule when facing varying service rates of the pickers. Zou et al. (2018) build a semi-open queueing network for evaluating the effects of battery management in RMFS. The strategies of battery swapping, automated plug-in charging and inductive charging at the pick station are compared. The authors come to the conclusion that battery swapping is generally more expensive than plug-in charging while inductive charging outperforms both in throughput and costs, if robot prices and retrieval times are low.

Wurman & Enright (2011) and Wurman et al. (2008) mention several decision problems on the operational level that they encountered in practice. One of the few studies that address decision problems on the operational level is by Boysen et al. (2017). They provide methods for optimally batching the pick orders and sequencing both the pick orders and the pods transported to the stations. They show that an optimized pick order processing requires only half the number of robots that a pick order process based on simple decision rules would need. Roodbergen et al. (2014) utilize a simulation based approach in order to optimize the warehouse layout of a manual order picking system for an industrial partner. The authors devise an integrated approach taking on to certain design decisions as well as selecting control policies. The simulation is thereby used “as a solution tool and an evaluation system” (see Roodbergen et al. (2014)). Chen et al. (2010) use a simulation based approach for evaluating the performance of policy sets for manual order picking systems. The authors make use of DEA as a tool for obtaining

a comparable performance indicator among the policy sets. Beckschäfer et al. (2017) use a discrete event simulation approach, similar to the approach in this chapter, for assessing storage policies for Automated Grid-based Storage systems. The authors find that even simple strategies improve the system efficiency, which encourages research on more complex strategies. Lamballais et al. (2018a) develop a Markow decision process (MDP) model for addressing the resource reallocation problem, i.e., the problem of deciding how many workers and robots to allocate to the pick process and replenishment process continually throughout time. The assumptions related to replenishment differ strongly across the papers mentioned above, and the number of approaches to replenishment in practical applications is diverse as well.

5.4 Decision Problems

This section introduces the decision problems considered in this chapter and places them within the context of other decision problems in an RMFS. Requests to the system occur via pick orders or replenishment orders. Upon receipt, pallets are broken up into smaller parts consisting of multiple units of one SKU. A replenishment order is a request to place one such part, i.e. a number of units of one specific product, on a pod.

We structure the decisions at the operational level in four steps: (1) Order Assignment (OA), the assignment of pick or replenishment orders to workstations, (2) Task Creation (TC), the creation of tasks for the robots, (3) Task Allocation (TA), the allocation of tasks to robots, and (4) Path Planning (PP), the creation of paths along which the robots will move. There are two kinds of Order Assignment decisions: the assignment of pick orders to pick stations, called the Pick Order Assignment (POA) problem, and the assignment of replenishment orders to replenishment stations, called the Replenishment Order Assignment (ROA) problem. In the second step, a task is defined as transporting a specific pod to a specific workstation and

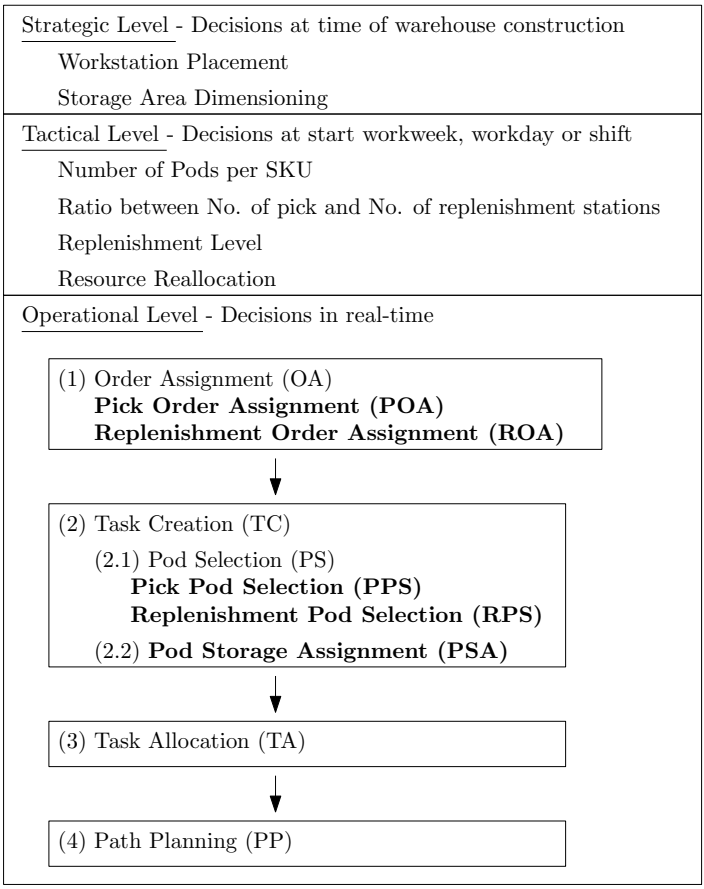


Figure 5.4.1: Hierarchical overview of the decision problems and their relations

back to a specific storage location. Therefore, for each workstation, the Task Creation decision problem includes the two subproblems of (2.1) deciding which pod to select for transportation, the Pod Selection (PS) decision problem, and (2.2) deciding at which storage location to return the pod, the Pod Storage Assignment (PSA) decision problem. The Pod Selection (PS) decision problem differs for the pick and replenishment process, because for

the pick process the due times of the pick orders is important in selecting a pod. Pod selection in the pick process is called Pick Pod Selection (PPS) and pod selection in the replenishment process is called Replenishment Pod Selection (RPS). Task Creation uses the pick order and replenishment order assignments to select suitable pods and subsequently converts the requests for the selected pods into tasks for pod transportation between the workstations and the storage area. Task Allocation creates a trip by building a sequence of tasks for the robots to execute. These sequenced tasks implicitly define trips and serve as input for the Path Planning algorithms, where a path is generated for a robot to follow.

Figure 5.4.1 shows an overview of the decision problems at the strategic, tactical and operational level in an RMFS, with the problems addressed in this chapter in bold. As can be seen in Figure 5.4.1, this chapter focuses on decision problems at the operational level. We use the term "decision rule" to refer to a fairly simple method to solve a decision problem. The aim of this chapter is to evaluate several decision rules per decision problem. Some decision rules may closely resemble common best practices, whereas others may be more specific to RMFS. The Task Allocation decision problem is intertwined with the Path Planning decision problem, which has been addressed by Merschformann et al. (2017a). Therefore we do not consider the Task Allocation and Path Planning decision problems. We do address Pick Order Assignment (POA), Replenishment Order Assignment (ROA), Pick Pod Selection (PPS), Replenishment Pod Selection (RPS), and Pod Storage Assignment (PSA). For Pick Order Assignment, we assume there is a constant backlog, and the pick stations are always filled to full capacity with pick orders. Whenever a pick order is fulfilled and leaves its pick station, a pick order has to be selected from the backlog and assigned to the pick station. For replenishment orders, we assume that the sequence of replenishment orders inbound to the system cannot be altered anymore. This assumption resembles the situation in conventional conveyor-based material handling components that do not allow sequence modification but only load routing. Moreover, we aim to avoid taking decision problems outside of the system's

boundaries into account, e.g., different dispatching rules of preceding systems. The replenishment stations have a finite capacity. If a replenishment order arrives and multiple replenishment stations have capacity left, the ROA decision rule determines to which replenishment station the replenishment order is assigned. If no place is available, replenishment orders are put in a replenishment order backlog. When a replenishment order is fulfilled at one of the replenishment stations, a new replenishment order is chosen from the replenishment order backlog according to the FCFS rule. Table 5.4.1 summarizes the decision problems addressed in this chapter.

At this point we also introduce the concept of "pile-on" (sometimes also called "hit-rate"). Pile-on as a concept refers to the average number of units that are picked from a pod every time a pod is presented to a picker at a pick station. Pile-on as a metric measures the number of units (across all SKUs) picked from a pod when presented to a picker at a pick station, averaged across every visit of a pod to a pick station during the entire time horizon. In other words, pile-on is measured in "units picked per pod visit to a pick station". The higher the pile-on is, the fewer pods need to be transported between the pick stations and the storage area, which may reduce the number of robots needed.

Table 5.4.1: Decision Problems

Abb.	Name	Description	Trigger
POA	Pick Order Assignment	Choosing a pick order from the backlog	When another pick order is fulfilled and leaves the pick station, creating room for the next pick order to be assigned
ROA	Replenishment Order Assignment	Selecting the replenishment station for the next replenishment order	When a replenishment order arrives at the system and one or more replenishment stations have capacity left
PPS	Pick Pod Selection	Selecting a pod to transport to a pick station	When a robot working for a pick station needs a new task
RPS	Replenishment Pod Selection	Select a pod for the next replenishment order	Depends on the ROA decision rule
PSA	Pod Storage Assignment	Choosing a storage location for a pod	When a pod leaves a workstation

5.5 Decision Rules

To solve the operational problems, we define several decision rules per decision problem that are evaluated in a realistic simulation. Several Path Planning algorithms for the RMFS are compared in Merschformann et al. (2017a), therefore this decision problem will not be addressed in this chapter. Thus, we selected WHCA$^*_v$, one of the best performing algorithms from the paper, as the path planning engine for the simulation framework used in this chapter. Additionally, we fix the Task Allocation algorithm to a simple method that first assigns two-thirds of the robots to pick operations and the rest to replenishment operations. Then, it aims to equally distribute the robots across the respective stations. This means a robot will only do tasks related to the station it is assigned to. This section will therefore only describe decision rules for the Pick Order Assignment, Replenishment Order Assignment, Pick Pod Selection, Replenishment Pod Selection and Pod Storage Assignment decision problems.

While replenishment and pick operations are similar in the sense that high throughput should be achieved with few resources, the main asymmetry between both is that for the former the goal is to fill the inventory as quickly as possible and for the latter to empty it as quickly as possible. This means that for replenishment operations we aim to replenish pods fast to have them available for pick operations early while preparing pod content such that it allows for a high pile-on during pick operations. For pick operations we aim to achieve a high pile-on and keeping trips short to fulfill as many orders as possible while also considering due times of the pick orders. Furthermore, we do not allow the sequence of replenishment orders to be modified. In contrast, for pick orders we allow to arbitrarily choose one order from the backlog. Lastly, pick orders have due times. All of this leads to different strategies we focus on per decision problem, instead of fully symmetric rules between pick and replenishment decision problems.

For a more precise description of some of the rules we introduce the notation shown in Table 5.5.1.

Table 5.5.1: Overview of the symbols used in the rule descriptions

Symbol	Explanation
$\mathcal{P}$	Set of all pods
$\mathcal{P}_s^I$	Set of pods heading to station s
$\mathcal{I}$	Set of all SKUs
$\mathcal{O}^B$	Set of pick orders in backlog
$\mathcal{O}_s^S$	Set of pick orders assigned to station s
$C(p,i)$	Number of units of SKU i contained in pod p
$L(o,i)$	Required units necessary to fulfill line i of order o
$D(o,i)$	Remaining units necessary to fulfill line i of order o
t_o^D	Due time of order o
t_o^S	Time of assignment to the station of order o
t	Time of deciding

5.5.1 Pick Order Assignment Rules

A pick station has to be chosen for every pick order submitted to the system and the pick order itself has to be chosen from the order backlog. We consider a pick order backlog of constant size, i.e., as soon as an order is removed from the backlog a new one is generated to replace it. This and the immediate replacement of orders completed at a station lead to only one option available to assign any pick order to: the slot of the just completed order. Hence, the choice of station is not a degree of freedom. The rare occasions of multiple orders to be completed at the same time are handled by assigning the orders to the pick stations randomly. Hence, we only investigate rules for selecting the next pick order from the backlog to fill the only open slot at a station. We devise six rules to solve this problem: “Random”, “FCFS”, “Due-Time”, “Fast-Lane”, “Common-Lines” and “Pod-Match”:

Random The Random rule randomly selects a next pick order from the backlog and is used as a benchmark.

FCFS The FCFS rule assigns the pick order that was first received. The rationale behind this is to keep pick order throughput times short.

Due-Time The Due-Time rule selects the pick order with the earliest due time from the backlog and assigns it to a station. This is a greedy approach aiming to finish the pick orders before their deadline.

Fast-Lane The Fast-Lane rule randomly selects a pick order from the backlog like the Random rule, but keeps one slot at each pick station open for immediately completable pick orders. I.e., only pick orders (o), for whom all lines and all units of inventory are available on the next pod (p_n) will be assigned to this station's "fast-lane" order slot (see Equation 5.1). Thus, orders assigned to the "fast-lane" slot are processed shortly after assignment. The next pod of the station is either a not completely processed pod the picker is currently working on or the next pod in the station's queue, if no such pod is available. In cases where no pod reached the station's queue yet, we consider the pod with the shortest remaining path to estimate the next pod. When facing multiple options we use a random tie-breaker. Note that this rule can be combined with any other proposed POA rule. The reason we combine it with random selection is to better assess the impact of the idea itself.

$$\forall i \in \mathcal{I} : L(o,i) \leq C(p_n,i) \tag{5.1}$$

Common-Lines The Common-Lines rule compares the station's (s) currently assigned pick orders with all orders from the backlog and selects the one with most lines in common for assignment (see Equation 5.2). The rationale behind this is to increase pile-on by exploiting synergies among the pick orders. When facing multiple options we use a random tie-breaker.

$$\underset{o \in \mathcal{O}^B}{\operatorname{argmax}} \sum_{o' \in \mathcal{O}_s^S} \sum_{i \in \mathcal{I}} \left(\begin{cases} 1 & L(o,i) > 0 \wedge L(o',i) > 0 \\ 0 & otherwise \end{cases} \right) \tag{5.2}$$

Pod-Match The Pod-Match rule selects the pick order from the backlog that matches best the pods heading to the station (s) at the moment of assignment best. I.e., the more units of the pick order are already available in the pods the better the match (see Equation 5.3). When facing multiple options we use a random tie-breaker.

$$\underset{o \in \mathcal{O}^B}{\operatorname{argmax}} \sum_{p \in \mathcal{P}_s^I} \sum_{i \in \mathcal{I}} \left(\min\left(C(p,i), D(o,i)\right)\right) \tag{5.3}$$

5.5.2 Replenishment Order Assignment Rules

As a result of the assumptions that replenishment orders arrive in a fixed sequence, we investigate only two different approaches for assigning replenishment orders to the stations, i.e., immediate Random assignment and batching of customer orders that go on the same pod. Hence, we construct two rules for replenishment assignment: "Random" and "Pod-Batch":

Random The Random rule randomly selects a next station with sufficient remaining capacity to allocate incoming replenishment orders to. If no such station is available, the order will wait until one becomes available again.

Pod-Batch The Pod-Batch rule tries to use a pod already selected to go to a replenishment station for assigning the next replenishment order. In other words, the Pod-Batch rule first waits for the Replenishment Pod Selection (Section 5.5.4) rule to decide which orders are assigned to which pod, and then uses the same replenishment station for the orders of one pod. If the replenishment orders do not fit one station, they wait until a station with sufficient capacity becomes available. During this time all consecutive orders are also blocked, because the sequence cannot be altered.

5.5.3 Pick Pod Selection Rules

Every time a robot working for a pick station s requests a next task, a pod suitable for picking at pick station s must be selected. We require for all rules that at least one unit can be picked from the pod. This means that no pod is brought to a station completely in vain and additionally it implies a pile-on of at least 1. The six PPS rules used in this chapter are the "Random", "Nearest", "Pile-on", "Demand", "Lateness", and "Age" rules:

Random The Random rule randomly selects a pod that offers at least one useful unit for picking.

Nearest The Nearest rule selects the pod which has the least estimated path time towards the station according to the path planning algorithm and that offers at least one useful unit for picking.

Pile-on The Pile-on rule selects the pod that offers most units necessary to fulfill the orders at the station (see Equation 5.4). Ties are broken by favoring pods with which more orders can be completed. If ties still persist, they are broken randomly.

$$\underset{p \in \mathcal{P}}{\operatorname{argmax}} \sum_{i \in \mathcal{I}} \sum_{o \in \mathcal{O}_s^S} \left(\min\left(C(p,i), D(o,i)\right)\right) \tag{5.4}$$

Demand The Demand rule selects the pod whose content is most demanded considering the current pick order backlog situation, i.e. the pod with most units demanded in the backlog is chosen (see Equation 5.5). Ties are broken randomly.

$$\underset{p \in \mathcal{P}}{\operatorname{argmax}} \sum_{i \in \mathcal{I}} \sum_{o \in \mathcal{O}^B} \min\left(C(p,i), D(o,i)\right) \tag{5.5}$$

Lateness The Lateness rule aims to finish late pick orders by selecting a pod that offers units needed to fulfill open order lines with most lateness

at the station, i.e., for one order the time the order is late is summed as fractions of the open picks (see Equation 5.6). If no order is late, the resulting ties are broken by using the same metric but replacing $\max\left(t - t_o^D, 0\right)$ with t_o^D, thus, selecting pods for orders whose due times are most imminent.

$$\underset{p \in \mathcal{P}}{\operatorname{argmax}} \sum_{i \in \mathcal{I}} \sum_{o \in \mathcal{O}_s^S} \left(\frac{\min\left(C(p,i), D(o,i)\right)}{\sum_{i' \in \mathcal{I}} D(o,i')} \max\left(t - t_o^D, 0\right) \right) \tag{5.6}$$

Age The Age rule aims to finish the oldest pick orders of a station by selecting a pod that offers units needed to fulfill the oldest open order lines, i.e. for one order the time the order spent assigned to the station is summed as fractions of the open picks (see Equation 5.7)

$$\underset{p \in \mathcal{P}}{\operatorname{argmax}} \sum_{i \in \mathcal{I}} \sum_{o \in \mathcal{O}_s^S} \left(\frac{\min\left(C(p,i), D(o,i)\right)}{\sum_{i' \in \mathcal{I}} D(o,i')} \left(t - t_o^S\right) \right) \tag{5.7}$$

5.5.4 Replenishment Pod Selection Rules

For every replenishment order, a suitable pod with sufficient remaining storage capacity needs to be chosen. The decision is taken right before the replenishment order is assigned to a replenishment station. Depending on the selected ROA and RPS rules both are either invoked simultaneously or, if there is a dependency between the two, one after the other. An example for the latter case is the combination of the PodBatch ROA rule with the Emptiest RPS rule, because the PodBatch rule relies on an already selected pod for the replenishment order. Since Replenishment Pod Selection determines the composition of the pods, it offers many possibilities to create pods with different features, e.g. high frequency pods that combine frequently ordered products, or family-based pods combining products that are often ordered together. If all replenishment orders assigned to the same pod are assigned to the same replenishment station, only one trip is necessary to

place all replenishment orders on the pod, which reduces the number of robot movements.
The five RPS rules used in this chapter are the “Random”, “Emptiest”, “Nearest”, “Least-Demand” and “Class” rules:

Random The Random rule selects a random pod with sufficient remaining capacity.

Emptiest The Emptiest rule assigns replenishment orders to the emptiest pod and reuses the same pod for subsequent replenishment orders until it is full or used at a station.

Nearest The Nearest rule assigns an incoming replenishment order to the nearest pod with sufficient remaining capacity.

Least-Demand With the Least-Demand rule an incoming replenishment order is assigned to the pod currently offering the least demanded inventory, i.e. the pod with the least units offered when compared to the aggregated demand by assigned and backlogged pick orders is selected. Thus, this pod is not useful for pick-operations at the time of selection and by this it is not disadvantageous to block it for replenishment operations.

Class The Class rule assigns incoming replenishment orders to a pod of the same class as the replenishment order, i.e. fast moving SKUs to pods with other fast moving SKUs. The classes are built by a background mechanism for which the cumulative relative amount of pods per class are given. In the experiments for this chapter we use “0.1, 0.3, 1.0”, i.e., three classes where the first class holds 10 % of the pods for the highest frequency SKUs, the second class holds 20 % and the last class holds the remaining ones, which are the ones with the lowest frequency SKUs. To assign a replenishment order of a certain class, the emptiest pod is selected from the pods of that particular class. Similar to the Emptiest rule, a selected pod is used for the subsequent incoming replenishment orders of the same class until no more replenishment orders fit the

pod or until the respective pod completes its visit to a replenishment station.

5.5.5 Pod Storage Assignment Rules

For each pod an unoccupied storage location has to be selected, every time after visiting a pick or replenishment station. PSA is an important aspect of the RMFS, because being able to change the storage location of pods after every visit to a workstation is what makes continuous automatic sorting possible. For PSA, five decision rules are examined, namely the "Random", "Fixed", "Nearest", "Station-Based" and "Class" rules.

Random The Random rule chooses a random free storage location.

Fixed The Fixed rule maintains the initially assigned storage location for all pods.

Nearest The Nearest rule stores pods at the nearest unoccupied storage location in terms of shortest estimated path time. This path time is determined using an A* algorithm that takes the time needed for turning the robot (with or without pod) into account.

Station-Based The Station-based rule is a variant on the Nearest rule, i.e. instead of bringing the pod to a storage location that is nearest to the robot's position the storage location with shortest path time to a pick station is selected. The greatest difference with the Nearest rule is in the storage locations chosen for pods returning from a visit to a replenishment station.

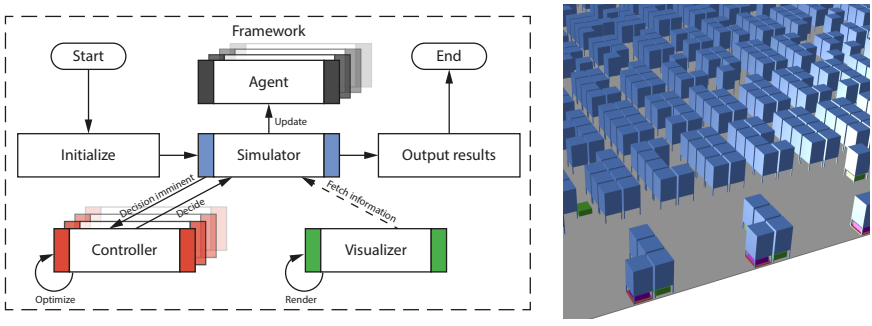
Class The Class rule brings pods back to storage locations of the same class, where classes are constructed in a similar fashion as in the RPS decision problem, but based on the shortest path time to a pick station. Within a class, a storage location for a pod is selected analogously to the Nearest rule.

Table 5.5.2 provides an overview of the decision rules per decision problem and shows how the decision rules are labeled across decision problems. Note that choosing a rule for one decision problem may jeopardize strategies chosen for others. For example, a random Pick Order Assignment may have a negative impact on a Turnover-based approach for assigning replenishment orders to storage locations, because it does not respect the units currently positioned near the pick station while assigning orders to it. Hence, a selection respecting mutual influences has to be done to provide an efficient compilation of rules that is able to adequately overcome the planning problems in such a system.

Table 5.5.2: Overview of the Decision Rules per Decision Problem

Decision Problem	Decision Rules
POA	Random, FCFS, Due-Time, Fast-Lane, Common-Lines, Pod-Match
ROA	Random, Pod-Batch
PPS	Random, Nearest, Pile-on, Demand, Lateness, Age
RPS	Random, Emptiest, Nearest, Least-Demand, Class
PSA	Random, Fixed, Nearest, Station-Based, Class

5.6 Simulation Framework



(a) Overview of the simulation process. **(b)** Visualization screenshot

Figure 5.6.1: RAWSim-O simulation framework

We use a simulation framework, called "RAWSim-O", which is especially written for RMFSs and was inspired by the work of Hazard et al. (2006b). A more detailed description of the framework can be found in Merschformann et al. (2017b), while the source code is available at `https://github.com/merschformann/RAWSim-O`. Similar to Hazard et al. (2006b), we use an agent-based and event-driven simulation focusing at a detailed view of the system. The basic simulation process is managed by the core simulator instance (see Figure 5.6.1a), which is responsible for obtaining the next event and updating the agents. Agents can either represent real entities like robots and stations or virtual entities like process managers, e.g. for emulating order processes. Every decision that has to be made is passed to the corresponding controller. The controller can either immediately decide or can buffer multiple requests in order to optimize and release the decision later on. However, in this chapter we only consider ad-hoc decision rules with the former approach. To allow visual feedback, the ongoing simulation can optionally be rendered in 2D and 3D. The implementation was done in C#.

The level of detail of the simulation is especially high for the simulated movement behavior of the robots. We consider the robot's momentum by emulating acceleration and deceleration behavior, collision avoidance and turning speed (see Table 5.7.2). The emulation employs a continuous time-horizon. The times for activities other than robot movement, e.g. lifting or storing a pod, or picking one unit at a pick station, are constant (see Table 5.7.2). The waypoints allow the emulated robot behavior to match real robot behaviour. Robots that do not carry a pod can traverse underneath stored pods by using the waypoints at which the pods are stored. Furthermore, in the buffers of the workstations, robots can take short-cuts if the buffer is (partially) empty.

Information about the system's state is tracked in a high level of detail, because some decision rules differ with regard to the information they require. For example, all pods and all units on all pods are tracked exactly. Incoming information is divided into a static and a dynamic category. Static information

includes everything describing a system instance and is completely given at start. Static information therefore includes the number and composition of pick stations and replenishment stations, the pods, the robots, and the waypoint system used for robot navigation. All of the decision rules proposed in this chapter differ in their computational complexity and therefore also in the computational time they require to reach decision. They are, however, simple enough to be considered as ad-hoc decisions even for large system sizes.

In contrast to static information, the dynamic information is not completely known beforehand, but becomes available over time. This is the case for incoming pick orders and replenishment orders submitted to the system over time by external processes. While each replenishment order consists of a number of physical units of one SKU, each pick order consists of a set of order lines, each for one SKU, with corresponding units necessary to fulfill the line. We assume for both pick and replenishment orders, that there is a constant order backlog. A constant order backlog means that when an order from the backlog is assigned to a workstation, it is immediately replaced by a newly generated order. By keeping the order backlogs constant, we aim to analyze the system's behavior under constant pressure. However, it also leads to the phenomenon that the system's storage space utilization (utilized space divided by total space available) in the storage area is affected by the performance of the decision rules controlling it, because no further virtual manager steers the process. E.g., if a combination of rules is replenishing quickly, the storage space utilization will increase. In contrast, it will decrease, if the rules are replenishing slowly. Situations in which the storage space utilization is nearing 100%, and only few storage places for new replenishment orders are available, lead to an inefficient replenishment process. To avoid such situations, we pause replenishment order generation, if storage space utilization exceeds 85 % and it is continued after it drops below 65 % again. Analogously, we pause the pick order generation, if storage space utilization drops below 10 % and resume after it exceeds 60 % again. The latter is done to avoid draining the inventory completely. Since in both cases either

the replenishment stations or the pick stations will become inactive due to no further orders to process, the robots will be reassigned to the remaining active stations. This redistribution of robots across the active stations is done at any time a station becomes active or inactive, i.e. at the beginning and end of order generation pauses.
If a new replenishment order is received, first the rules for ROA and RPS are responsible for choosing a replenishment station and a pod (see Figure 5.6.2). The time the decision is taken depends on the active rules. The execution of the assignment can earliest be done as soon as there is sufficient capacity on a pod and a station available. The commit of the assignment technically results in an insertion request (shown as red cylinders), i.e. a request that requires a robot to bring the pod to the workstation. Multiple of these requests are then combined to an insertion task and assigned to a robot by a TA rule. Similarly, after the POA rule selects a pick order from the backlog and the assignment is committed to a pick station, an extraction request (shown as blue cylinders) is generated, i.e. a request that requires bringing a suitable pod to the chosen station. Up to this point, the physical units of SKUs for fulfilling the pick order are not yet chosen. Instead, the decision is postponed and taken right before combining different requests to extraction tasks by PPS and assigning them to robots by TA. This allows the implemented rules to exploit more information when choosing a pod for picking. Hence, we consider PPS as a decision closely interlinked with TA. Furthermore, the system generates store requests (shown as orange cylinders) each time a pod has to be transported to a storage location. The PSA rule only decides the storage location for a pod that is not needed anymore and has to be returned to the storage area. If all requests are already being handled by other robots, the robot will be assigned an idle task, thus, the robot dwells at a dwelling point until needed. Dwelling points can be used to reduce congestion effects if there are only a few active stations compared to the number of robots, e.g. robots waiting at a storage location block others that try to pass by. For this, the robot will park at a free storage location to avoid causing conflicts with other robots. The dwell point

policy uses locations in the middle of the storage area to avoid blocking prominent storage locations next to the stations. Another type of task would be charging, which is necessary when robots run low on battery, however, in this chapter we assume the battery capacity to be infinite, so this type of task is ignored. All of the tasks result in trips (shown as green cylinders), which are planned by a path planning algorithm and executed by the robots. The only exception is when a pod can be used for another task at the same station. The trips are planned by a PP algorithm and the resulting paths are executed by the robots. Figure 5.6.2 shows an abstract overview of these dependencies. The exact times at which the decisions are taken depend on the respective rules, e.g. the Pod-Batch ROA rule assigns a batch of replenishment orders to the first pick station offering sufficient space while the Random ROA rule immediately assigns single replenishment orders to the first station with sufficient capacity available. However, all of the rules have in common that they make assignments greedily while adhering to certain capacity constraints (station capacity, pod capacity, etc.).

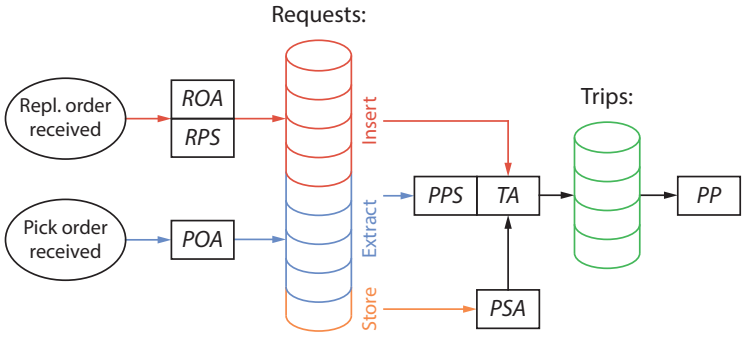


Figure 5.6.2: Order of decisions to be done induced by receiving pick and replenishment orders

5.7 Evaluation Framework

This section describes the evaluation framework used to carry out the research in this chapter. Two central concepts to the evaluation framework are the Rule Configuration (RC) and the Warehouse Scenario (WS). The RC specifies for each decision problem, which decision rule is used. The WS specifies the warehouse layout, number of robots, number of workstations, number of SKUs, whether or not return orders are part of the operations of the warehouse, and pick order size. During one simulation run the RC and WS do not change, so they can be seen as an input to a simulation run.
The evaluation framework consists of two phases, one varying the RCs, the other varying the WSs. Phase 1 evaluates all 1620 possible RCs on one WS. For phase 1, we compare eight performance measures: (1) unit throughput rate, (2) pick order throughput rate, (3) order turnover time, (4) distance traveled per robot, (5) order offset, (6) fraction of orders that are late, (7) pile-on (8) the pick station idle time. Unit throughput rate is the number of picked units of all SKUs per hour. Pick order throughput rate is the number of pick orders fulfilled per hour. Order turnover time is the average time between submitting a pick order to the backlog and fulfilling it. Order offset is the average time between the due time and the completion time of the pick orders. Thus, a value smaller than zero shows how much in advance pick orders are completed. The rationale behind this is that follow-up processes at the distribution center are not deterministic, hence, pick orders completed earlier may improve the overall service level. The pick station idle time is measured as an average across all pick stations in the system.
Phase 1 selects the RCs with the highest unit throughput rate. However, among these selected best RCs, the variety in the decision rules may be low. For a particular decision problem, all of the selected RCs may use the same decision rule. To ensure more diversity in the RCs in phase 2, we define 6 so-called "benchmark RCs", see Table 5.7.1. The benchmark RCs were chosen such, that all decision rules across all decision problems appear in at

least one of the benchmark RCs. Each benchmark RC has been given a name that reflects a characteristic that the decision rules have most in common.

Table 5.7.1: Benchmark RCs

Benchmark RC	POA	ROA	PPS	RPS	PSA
Demand	Due-Time	Pod-Batch	Demand	Least-Demand	Fixed
Speed	Fast-Lane	Pod-Batch	Lateness	Emptiest	Nearest
Nearest	FCFS	Random	Nearest	Nearest	Nearest
Class	Common-Lines	Pod-Batch	Age	Class	Class
Greedy	Pod-Match	Pod-Batch	Pile-on	Emptiest	Station-Based
Random	Random	Random	Random	Random	Random

Phase 2 evaluates the selected RCs from phase 1 and the benchmark RCs, while varying the warehouse scenarios. Since we are specifically interested in the efficiency of RCs, we neglect layout decisions in this chapter. Thus, we choose one specific layout, using the style described in Section 5.2. The concrete layout instance comprises 1149 pods and 1352 storage locations (85% filled) and is shown in Figure 5.7.1. When varying the number of pick stations during phase 2 we add workstations in the order given in Figure 5.7.1.

5.7.1 Parameters

In the following we describe the used parameters in more detail. The parameters shared for both phases are outlined in Table 5.7.2. We set a continuous simulation horizon of 48 hours in order to decrease the impact of side effects like recurring replenishment overflows, which cause replenishment pauses described previously. Within a duration of 48 hours we observe sufficient repetitions of such patterns to achieve a reasonable mitigation of these side effects.
Furthermore, for each RC and WS combination in phase 1 and in phase 2 we conduct 10 runs to lessen the effect of randomness. To keep the system

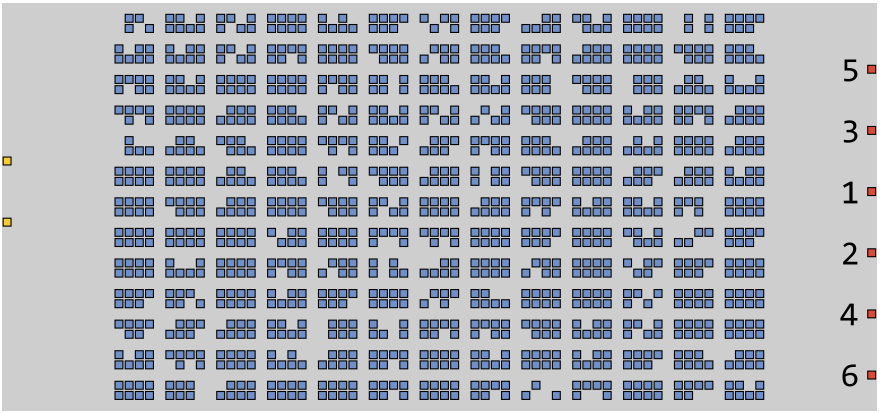


Figure 5.7.1: Top view of the layout, including pick station indices, with the storage area in the middle, replenishment stations to the left, and pick stations to the right

under continuous pressure, like described above, we keep a constant pick and replenishment order backlog of 200 orders each. At simulation start inventory is generated until 70 % overall storage utilization to avoid cold starting the system. This is done using the same process used for generating replenishment orders during simulation and using assignment rules suiting the respective RPS rule in place. The storage capacity of a pod is set to 500 slots while the storage consumption of one SKU unit is drawn from a uniform distribution between 2 and 8 slots, thus, a full pod contains 100 units in average. The popularity of the SKUs is determined by drawing a value from an exponential distribution with parameter $\lambda = \frac{1}{2}$ for each SKU to emulate a typical ABC curve in e-commerce. This popularity is the relative frequency parameter between all SKUs, thus, the frequency (if divided by the sum of all frequencies) is the probability of choosing a particular SKU when generating an order line for both replenishment and pick orders. One replenishment order restocks between 4 and 12 units of one SKU following a uniform distribution. To emulate due times we distinguish between priority

and normal orders that have to be completed in 30 minutes respectively 120 minutes. This reflects the need for preferring important orders.
The movement behavior of the robots is emulated by using a maximum velocity of $1.5\frac{m}{s}$ with acceleration and deceleration rates of $0.5\frac{m}{s^2}$. We set the rotational speed to $\frac{4}{5}\pi\frac{rad}{s}$, i.e., $2.5s$ for a full turn. Turning takes the same amount of time regardless of whether a robot is carrying a pod. The time for lifting and setting down a pod is set to $3s$. This should reflect the capabilities of mobile robots used in similar industry applications reasonably close. For the actual pick operation of one unit at a pick station we assume a constant time of $8s$. The complete time for handling one unit including additional operations, like putting the product unit in the correct pick order tote, is set to $15s$. This distinction is considered to allow for an early release of the robot, such that no unnecessary robot waiting times are caused. This is not distinguished for replenishment operations, since we assume that a robot can only leave after fully completing the put operation to the pod. The time of a put operation of one replenishment order is set to $20s$.

The parameters in Table 5.7.2 are shared across all conducted experiments, while the parameters in Table 5.7.3 are depending on phase and scenario. For the first phase we assess all possible RCs for one fixed warehouse scenario. Note that the RPS rule Nearest and the ROA rule Pod-Batch both wait for the other one to decide first leading to no decision at all, hence, the combination of these rules is forbidden. For the fixed warehouse scenario we set the number of robots to 4 per pick station, i.e. 8 robots in the system at whole. Furthermore, we set the number of pick stations to 2, the number of SKUs to 1000 and exclude the processing of return orders. The order setting is set to Mixed. This means the number of lines per pick order and the number of units per order line are generated following truncated normal distributions with parameters shown in Table 5.7.3. This is done to resemble e-commerce pick order characteristics of generally small orders with occasional larger ones in between.
Equation (5.8) shows that phase 1 has 1620 RCs, and since phase 1 has 1

Table 5.7.2: Parameters shared across all simulations

Parameter	**Value**
Simulation	
Simulated duration of warehouse operations	48 hours
Number of simulation repetitions	10 repetitions
Size of pick order backlog	200 pick orders
Size of repl. order backlog	200 repl. orders
Layout	1149 pods, 1352 storage locations in 2×4 blocks, 12 aisles and 12 cross-aisles
Orders	
Number of units per repl. order	uniform distribution between 4 and 12 units
Amount of priority orders in pick orders	20 %
Priority pick order due time	backlog submission time + 30 min.
Normal pick order due time	backlog submission time + 120 min.
Threshold when pick order generation starts	60% of inventory capacity of the storage area
Threshold when pick order generation stops	10% of inventory capacity of the storage area
Threshold when repl. order generation starts	65% of inventory capacity of the storage area
Threshold when repl. order generation stops	85% of inventory capacity of the storage area
Inventory	
Initial inventory in the storage area	70% of the inventory capacity of the storage area
Space on a pod	500 slots
SKU frequency / popularity	Exponential distribution, $\lambda = \frac{1}{2}$
SKU size	uniform distribution between 2 and 8 slots
Robot movement	
Robot acceleration/deceleration	$0.5 \frac{m}{s^2}$
Robot maximum velocity	$1.5 \frac{m}{s}$
Time needed for a full turn of a robot	$2.5s$
Time needed for lifting and storing a pod	$3s$
Time needed for picking a unit	$8s$
Time needed for handling a unit at pick station	$15s$
Time needed for putting a repl. order on a pod	$20s$
Stations	
Repl. station capacity	two times pod capacity
Pick station capacity	8 pick orders

WS and 10 runs are conducted per RC and WS combination, this results in 16200 simulation runs for phase 2. Phase 2 has 10 RCs (see Table 5.7.3) and Equation (5.8) shows that it has 360 WSs, which together with 10 runs per RC and WS combination leads to 36000 simulation runs for phase 2.

$$\begin{aligned} \#RC \text{ in phase } 1 = \big| \{ROA\} \times \{POA\} \times \{RPS\} \times \{PPS\} \times \{PSA\} \setminus \\ \{(roa, poa, rps, pps, psa) \mid (roa = \text{Pod-Batch}) \wedge (rps = \text{Nearest})\} \big| \\ = 1620 \end{aligned} \tag{5.8}$$

$$\begin{aligned} \#WS \text{ in phase } 2 = \big| \{\#\text{ pick station}\} \times \{\text{robots per pick station}\} \\ \times \{\#\text{ of SKUs}\} \times \{\text{return orders}\} \times \{\text{pick order size}\} \big| \\ = 6 \times 5 \times 2 \times 2 \times 3 = 360 \end{aligned} \tag{5.9}$$

For phase 2 we limit the RCs to the 6 benchmark RCs and the 4 best ones from phase 1, i.e., the 4 RCs with highest throughput rate. Moreover, we vary the number of pick stations from 1 through 6 and the number of robots per pick station from 2 through 6. This leads to a range from 2 robots in the system to 36 robots across all WSs. In addition to WSs with 1000 SKU, we also assess WSs with 10000 SKUs stored in the system. For the order size we define two additional settings of small and large orders. For the Small pick order size, only single line / single unit pick orders are generated. For the Large pick order size, the distributions from the Mixed order setting are used but the min parameter for both is set to 2. Lastly, in WSs where we emulate the processing of return orders, 30 % of the generated replenishment orders are single unit. The total number of RC and WS combinations for the phase 2 is therefore 3600, which leads to 36000 simulation runs.

Table 5.7.3: Varied parameters for phase 1 and 2

Parameter	Phase 1 values	Phase 2 values
Rule configurations (RCs)	1620 RCs	6 Benchmark RCs + 4 best RCs from phase 1
Number of pick stations	2	1, 2, 3, 4, 5, 6
Robots per pick station	4	2, 3, 4, 5, 6
Number of SKUs	1000	1000, 10000
Return orders	0 %	0 %, 30 %
Pick order size	*Mixed* - line & unit distributions: $\mu = 1, \sigma = 1, \min = 1, \max = 4$ $\mu = 1, \sigma = 0.3, \min = 1, \max = 3$	*Small* - line & unit distributions: $\min = 1, \max = 1$ $\min = 1, \max = 1$ *Mixed* - line & unit distributions: $\mu = 1, \sigma = 1, \min = 1, \max = 4$ $\mu = 1, \sigma = 0.3, \min = 1, \max = 3$ *Large* - line & unit distributions: $\mu = 1, \sigma = 1, \min = 2, \max = 4$ $\mu = 1, \sigma = 0.3, \min = 2, \max = 3$
# RC	1620	10
# WS	1	360
# RC×WS	1620	3600
# simulation runs	16200	36000

5.8 Computational Results

This section shows the results from phase 1 and phase 2 of the evaluation framework. Throughout this section, the unit throughput rate is presented as a percentage of the upper bound on the unit throughput rate. The unit throughput rate is presented in this way to facilitate interpretation and comparison of results across experiments. Moreover, the RMFS is supposed

to have high pick rates as it eliminates the need of walking for the workers, while the robots are supposed to supply the pickers with a constant stream of pods to pick from. Presenting the unit throughput rate as a percentage shows clearly to what extent these aims are achieved. The upper bound is discussed in more detail in Appendix 5.A. The length of the confidence intervals is always less than 1% of the mean, based on 10 runs per RC and WS combination, and therefore does not add much information.

5.8.1 Phase 1

Table 5.8.1: Correlations between the different performance measures for first phase

	μ	σ	Unit throughput	Order throughput	Order turnover time	Distance traveled	Order offset	Late orders	Pile-on	Station idle time
Unit throughput	0.556	0.189	-	-	-	-	-	-	-	-
Order throughput	241.963	82.234	1.000	-	-	-	-	-	-	-
Order turnover time	3549.625	1220.445	-0.950	-0.950	-	-	-	-	-	-
Distance traveled	122598.768	19433.860	-0.952	-0.952	0.880	-	-	-	-	-
Order offset	-2565.458	1224.985	-0.950	-0.950	1.000	0.880	-	-	-	-
Late orders	0.187	0.115	-0.590	-0.591	0.685	0.549	0.684	-	-	-
Pile-on	2.438	1.450	0.899	0.899	-0.802	-0.796	-0.802	-0.448	-	-
Station idle time	0.450	0.186	-1.000	-1.000	0.950	0.952	0.950	0.591	-0.899	-

The first phase aims to investigate throughput performance and the impact per decision problem of decision rules on throughput. Furthermore, we assess the behavior of the different output measures depending on decision rule selection. For this, Table 5.8.1 shows how across these simulations the seven previously introduced performance measures correlate with each other. At first, we can observe that as the unit throughput rate score improves, the other performance measures improve as well. As the unit throughput rate score increases, pick order throughput rate and pile-on increase as well,

whereas the order turnover time, the distance that robots travel, the order offset, the fraction of orders that miss their due time, and the station idle time decreases. Although it is not clear what the exact causal relationships are, the correlations suggest that pile-on and the distance traveled by the robots are the main drivers behind these improvements. With higher pile-on, more units are picked per pod, so order lines are fulfilled more quickly and fewer trips are needed to fulfill the pick orders. This also causes longer processing times for each pod at the pick station, which in turn increases the time for the next robot to queue and become ready at the station. In other words: a more continuous input of inventory at the pick station is achieved. Additionally, fewer trips for the pick process free up robots to do more replenishment tasks. With less distance traveled by the robots we expect pods to be presented at the pick stations more continuously. Similar to the pile-on this effect enables more continuous picking, which in turn increases the overall unit throughput rate. Both measures, pile-on and the traveled distance, are intermediate measures affected by the choice of strategy for the different decision problems, i.e., a better score in both helps decreasing the idle time at the stations, which in turn helps increasing the throughput. An increased throughput, in the constant pick order backlog setting of the simulation, also decreases the turnover time of pick orders and the due time offset. Only the number of orders being late is not strongly correlated with the two main throughput drivers. The two main throughput drivers can also be observed when looking at a scatter plot of all simulation runs of the first phase (see Figure 5.8.1). Here we can see the best results in unit throughput rate score are achieved with a high pile-on and less distance traveled per robot. The group of simulation runs with least distance traveled per bot and a pile-on around 4 are RCs involving the Nearest PPS rule, while the simulation runs with highest pile-on (greater 5) at the top of the plot are RCs involving the Demand PPS rule. In both groups we find runs with the highest unit throughput rate score, hence, a higher throughput is not only achieved by a high pile-on. In particular within the top ten RCs in terms of unit throughput rate score the pile-on ranges between 3.84 and 6.36, while

the distance traveled per bot ranges between 68.04 km to 80.36 km. Hence, pile-on and the traveled distance enable higher throughput, but may also compensate for each other. This is particularly interesting, because both come at operational costs. For traveled distance this is energy consumption and robot wear, while for pile-on it may be costs arising from potentially more complex replenishment processes. Furthermore, within both groups better results are obtained with RCs also involving the Pod-Match POA rule, which causes an additional boost in pile-on.

In Figure 5.8.1 we also observe a 'cutoff' of simulation runs in the upper right and bottom left areas. This can be explained by the longer handling time at the station resulting from a higher pile-on. I.e., the longer a robot needs to wait at a station for the picking to finish the less it can travel in the meantime. Thus, rules increasing pile-on may help reducing the necessary travel distance, and by this also robot wear and energy consumption.

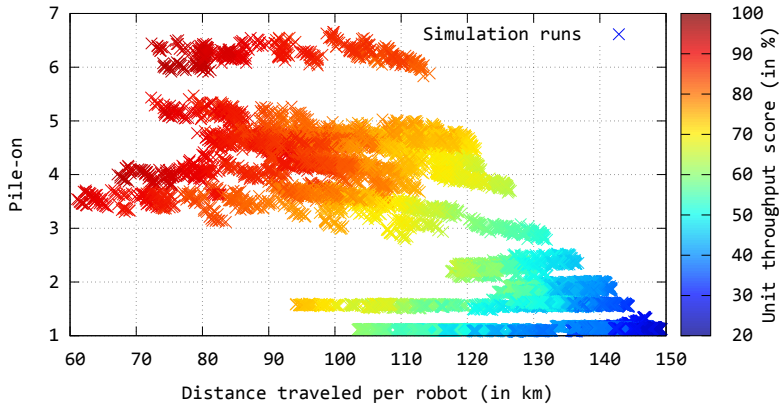


Figure 5.8.1: Scatter plot for pile-on vs. traveled distance per robot colored by the achieved throughput rate score for all simulation runs of the first phase

The pick order throughput rate is neglected completely in the remainder of this chapter, because it almost completely aligns with the unit throughput rate score. The reason for this is the constant backlog of 200 pick orders over

48 hours: with a pick order throughput rate of 241.963 completed orders per hour in average, omitting certain pick orders is almost impossible. Hence, we cannot observe a potential temporary throughput gain by preferring smaller or larger orders. In order to investigate the trade-off between picking many units and completing more pick orders an experiment with a fixed set of backlogged pick orders over a fixed period of time should be devised. For this, the possibly tedious processing of leftover pick orders, which are presumably harder to pick quickly, needs to be investigated.

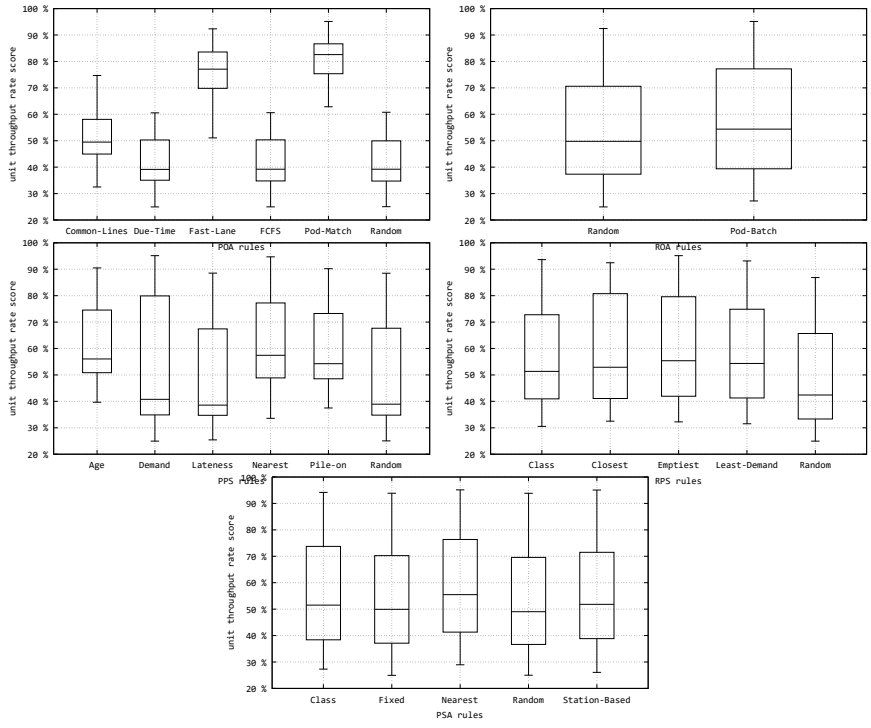
Table 5.8.2: Average unit throughput rates as percentages of the upper bound for all rules, together with the **best** / **worst** performance multiplier per decision problem

							Mult. $\left(\frac{\text{best}}{\text{worst}}\right)$
POA	Common-Lines	Due-Time	Fast-Lane	FCFS	Pod-Match	Random	
	50.93%	41.93%	76.13%	41.81%	**81.18%**	**41.71%**	1.946
ROA	Random	Pod-Batch					
	53.71%	**57.99%**					1.080
PPS	Age	Demand	Lateness	Nearest	Pile-on	Random	
	61.50%	52.70%	**48.63%**	**62.16%**	59.82%	48.88%	1.278
RPS	Class	Nearest	Emptiest	Least-Demand	Random		
	56.16%	58.42%	**59.63%**	57.71%	**47.56%**		1.254
PSA	Class	Fixed	Nearest	Random	Station-Based		
	55.91%	54.08%	**58.79%**	**53.60%**	55.70%		1.097

Table 5.8.2 shows for each decision problem the unit throughput rate score for each of the decision rules, averaged across all simulations in phase 1. We calculate the multiplier by dividing the highest unit throughput rate by the lowest. As the multiplier in unit throughput rates is rather large for the POA decision problem, system integrators and RMFS suppliers may benefit from carefully selecting a POA decision rule and from investigating better decision rules for this decision problem. The multiplier for the Replenishment Order Assignment is near 1, indicating that using a different decision rule does not offer much performance improvements. However, we note that we keep the sequence of incoming replenishment orders fixed at all times, which limits improvement potential. Nevertheless, we expect limited degrees-of-freedom in replenishment operations to be more realistic, because the sequence will

typically be a result of preceding operations or systems. Moreover, the limited number of replenishment stations diminishes the impact of ROA decision rules even more. Furthermore, the impact of the Pod Storage Assignment selection rule seems to be fairly low. This may be a reason of the quite small layout. We expect the impact of PSA decision rules to increase with the size of the instance layout, because the effect on the traveled distance would grow by a large amount.

Figure 5.8.2: Unit throughput rate performance of all runs grouped per rule



In the following we analyze the achieved throughput performance per decision rule. For this, Figure 5.8.2 shows the box-plots of unit throughput rate scores

for each decision problem grouped per decision rule. The boundaries of the boxes are determined by the upper and lower quartile while the line in the middle indicates the median value. The whiskers extend from the boxes to the minimum and maximum values. The first observation is that throughput performance of the RMFS is most sensitive to the choice of POA decision rule among the defined decision rules. This aligns with the previously observed correlations, because the choice of POA immediately affects the pile-on, which is identified as a major performance driver. The best performing POA strategies are FastLane and PodMatch, which both look at the incoming pods at a pick station when assigning new pick orders from the backlog. This suggests that a strategy aligning pick orders with the content of incoming pods seems most promising for throughput efficiency. This backs up the findings of Boysen et al. (2017). Although the Common-Lines rule exploits a similar greedy strategy, it achieves substantially less throughput. Hence, only matching pick orders to each other but not to the content of the pods squanders throughput capabilities of the system. All other POA decision rules achieve similar throughput performance, since they do not consider order characteristics that would affect pile-on or traveled distance.

When looking at the PPS rule box-plots the average best throughput performance with least variance is achieved by the Age, Nearest and Pile-on rules. All of them focus either on maximizing the pile-on or minimizing the traveled distance. Although the Age rule does only indirectly maximize pile-on, it achieves a higher average pile-on of 2.92 among all RCs containing it than the actual Pile-on rule, which achieves an average pile-on of 2.79. The Demand rule has the highest spread across PPS rules with a very low median, but also provides some top performing RCs (see Table 5.8.3). This suggests that the throughput performance of the rule has a higher dependency on the selection of other rules.

Although the variation among the ROA decision rules is small, we observe a slightly better throughput performance by the Pod-Batch rule. This is a reason of the smaller number of trips necessary when batching replenishment orders.

Many of the top performing RCs contain the Emptiest or Closest RPS decision rule. The main reason for the good throughput performance again seems to rely on fewer and shorter trips. The Emptiest rule decreases the number of trips, because more replenishment orders are stored in pods at once until it is full. E.g., only 31.03 % pods need to be brought to replenishment stations in average when compared to the Random rule. The Closest rule benefits from a similar effect since the same (closest) pod is used for further replenishment orders even while it is already approaching. Furthermore, Closest decreases the distance per replenishment trip, because nearer pods are used. The Random rule performs worst for RPS. The main reason for this is that too many trips are caused by randomly selecting pods while only storing few replenishment orders per trip.

Among the PSA decision rules we observe the best throughput performance for the Nearest strategy. This is again mainly caused by the shorter trips for the robots. When comparing the Nearest and the Station-based rule we see an overall benefit from shorter trips for replenishment operations increasing throughputof pick operations. However, this depends on the queue length at stations and the distribution of robots between replenishment and picking. I.e., if longer queue times are expected at replenishment stations, moving pods nearer to the pick stations when returning them to the inventory may improve overall throughput performance. The Fixed and Random decision rules differ little in their performance. The main reason for this is that the storage location per pod in the Fixed rule is randomly selected. Thus, leading to a very similar behavior.

Due to the large sample sizes, the results of ANOVA and Tukey's range tests rejected the hypotheses that the means were equals at the 0.05 significance level across within groups and pair-wise, with five exceptions. The null hypothesis of equal means was not rejected at the 0.05 significance level for POA rules FCFS and Due-Time, for Random and Due-Time, and for Random and FCFS. Furthermore, for PPS rules Random and Lateness the hypothesis of equal means could not be rejected, and for PSA rules Station-Based and Class.

5.8.2 Phase 2

Table 5.8.3: RCs with best throughput score selected from first phase (performance is unit throughput rate score)

RC rank	POA	ROA	PPS	RPS	PSA	performance
1	Pod-Match	Pod-Batch	Demand	Emptiest	Nearest	94.81 %
2	Pod-Match	Pod-Batch	Demand	Emptiest	Station-Based	94.63 %
3	Pod-Match	Pod-Batch	Nearest	Emptiest	Nearest	94.43 %
4	Pod-Match	Pod-Batch	Demand	Emptiest	Class	94.00 %

From the 1620 RCs in phase 1, the four with the highest unit throughput rate (see Table 5.8.3) together with the benchmark RCs form the set of ten RCs used in phase 2. The main purpose of phase 2 is to examine how well the RCs perform under different circumstances. In the following we analyze the results obtained for the 12 warehouse scenarios and 30 resource settings described before (see Section 5.7.1).

Table 5.8.4 shows the results, with the entries being the unit throughput rate as a percentage of the upper bound. In each cell the result of the best performing RC for the respective scenario and station / robot configuration is shown. The unit throughput rate scales well when adding more pick stations, the scaling is (almost) completely independent of the scenario characteristics. However, the necessary number of robots to achieve a given unit throughput rate greatly depends on the scenario characteristics, e.g., for more SKUs more robots are necessary to achieve a high unit throughput rate. The number of SKUs, does have a major impact on performance overall, where the main reason is that pile-on is considerably lower for the 10000 SKU scenarios. A reason for this is the lower likeliness to have a pod with a good combination of SKUs matching the orders of the pick stations available. Thereby, if larger orders have to be processed with the system, this helps mitigating the negative effect of handling lots of SKUs. The main reason for this are the larger number of order lines active at a station when picking larger orders. I.e., more open order lines increase the likeliness of having a well matching pod available for the inventory required at a pick station. Processing return orders has an increased negative effect, if the order size of customer orders

Table 5.8.4: Best unit throughput rate score for all scenarios, robots per pick station and numbers of pick stations. Scenario abbreviations: [SKU count: 1000 (1K), 10000 (10K)]-[Order size: Small (S), Medium (M), Large (L)]-[Return orders: yes (R), no (N)]

Stations	1					2					3					4					5					6				
Robots	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
1K-S-N	44	82	91	97	97	59	89	94	97	98	64	90	95	97	98	60	87	93	97	98	57	87	93	97	98	59	87	94	97	98
1K-S-R	46	82	92	97	97	59	89	94	97	98	63	90	95	97	98	61	88	93	97	98	56	87	93	97	98	55	86	93	97	98
1K-M-N	45	83	92	97	98	60	90	95	98	98	64	90	95	98	98	60	88	93	98	98	57	88	94	98	98	59	88	94	98	98
1K-M-R	45	82	92	97	98	59	89	95	98	98	63	91	96	98	98	62	89	93	98	98	56	88	94	98	98	56	87	94	98	98
1K-L-N	54	83	93	99	99	66	90	97	99	99	68	91	96	99	99	67	88	94	99	99	63	86	94	98	99	63	87	94	99	99
1K-L-R	50	80	92	99	99	64	88	96	99	99	65	90	97	99	99	67	88	94	99	99	62	86	93	98	99	62	84	94	98	99
10K-S-N	21	39	55	68	78	27	44	59	72	81	28	46	61	73	80	29	47	59	69	77	30	47	57	68	77	31	45	58	68	76
10K-S-R	20	40	56	70	80	27	45	61	74	82	29	47	62	74	82	30	48	61	70	78	31	48	58	67	76	31	46	57	66	74
10K-M-N	23	41	58	71	81	28	46	61	75	84	29	48	64	76	83	30	49	61	71	80	31	48	60	71	80	32	47	60	71	79
10K-M-R	21	41	59	73	83	28	48	63	76	85	30	49	65	77	84	31	50	63	72	81	32	49	60	70	79	32	47	59	69	77
10K-L-N	36	63	84	94	98	44	71	89	96	99	46	73	87	94	98	47	69	83	92	97	46	67	84	91	97	45	68	83	91	97
10K-L-R	30	52	76	89	96	37	62	81	92	97	40	64	83	91	96	41	64	76	87	94	42	61	74	84	93	41	59	73	84	92

is large. However, in general, whether return orders are processed has a lesser effect on throughput performance than the other warehouse scenario variations. The reason behind this may be that even though approximately 19.76 % more time is spent on replenishment operations by the robots when compared to the scenarios without return order processing, replenishment operations are overall quick enough to mitigate the effect. Replenishment operations only consume 20.29 % out of the overall time consumed by the robots in average across all phase 2 simulation runs. Furthermore, we can conclude that with 1000 SKUs, the unit throughput rates are close to their theoretical maximum even with relatively few robots per stations.

Table 5.8.5 shows the unit throughput rate score for the RCs for all combinations of number of robots (n_r) and number of stations (n_s), averaged across WSs and presented as whole percentages. From Table 5.8.5 we can see that the Ranked RCs from phase 1 perform similarly and better than the benchmark RCs. Among the benchmark RCs, the Greedy benchmark outperforms the others consistently across all settings and is the only one whose unit throughput rate scores approached those of the ranked RCs.

Table 5.8.5: Unit throughput rate scores for the RCs in phase 2

	Ranked RCs				Benchmark RCs					
$(n_s\ ,\ n_r)$	1	2	3	4	Demand	Speed	Nearest	Class	Greedy	Random
(1 , 2)	30	28	32	28	14	23	18	25	34	11
(1 , 3)	60	59	61	58	24	40	34	42	57	23
(1 , 4)	76	75	76	74	37	57	49	55	72	23
(1 , 5)	86	86	86	84	47	70	62	66	82	34
(1 , 6)	91	91	91	90	58	79	73	74	87	45
(2 , 2)	42	41	43	39	18	28	25	32	42	11
(2 , 3)	68	67	68	65	29	47	40	47	64	23
(2 , 4)	81	80	80	78	40	63	55	59	76	29
(2 , 5)	88	88	88	86	51	74	67	69	84	37
(2 , 6)	92	92	92	91	62	83	77	76	89	47
(3 , 2)	45	43	46	42	19	32	27	32	44	15
(3 , 3)	70	69	70	66	30	50	43	48	65	23
(3 , 4)	81	81	81	79	42	65	57	60	77	31
(3 , 5)	88	88	88	86	52	75	68	69	84	41
(3 , 6)	92	92	92	90	62	83	77	76	89	50
(4 , 2)	45	44	47	42	20	34	29	34	45	14
(4 , 3)	68	67	69	65	31	51	44	49	64	23
(4 , 4)	78	78	78	76	42	64	56	59	74	34
(4 , 5)	86	86	86	84	51	74	66	68	82	43
(4 , 6)	91	90	90	89	60	81	74	74	87	53
(5 , 2)	43	42	45	40	20	34	29	35	43	14
(5 , 3)	67	66	68	64	31	51	44	48	63	24
(5 , 4)	78	76	77	75	41	63	55	58	73	35
(5 , 5)	85	84	85	83	49	72	63	66	81	45
(5 , 6)	90	90	90	88	58	80	72	73	86	54
(6 , 2)	44	42	45	41	20	35	30	35	43	15
(6 , 3)	66	64	66	63	31	51	43	48	62	25
(6 , 4)	77	76	77	75	40	62	53	58	73	35
(6 , 5)	85	84	85	83	48	71	62	65	80	45
(6 , 6)	90	89	89	88	55	79	70	71	85	53

5.9 Conclusion

In this chapter, we studied the throughput performance of decision rules for multiple decision problems occurring in the control of RMFS. By analyzing a total of eight output measures for a total of 1620 RCs, we found strong correlations between these. Most interestingly a high pile-on and a short distance traveled by the robots together almost immediately account for the success of a decision rule applied to RMFS. Hence, we propose using these two output measures as the key tactics when designing decision strategies for RMFS that aim to achieve high throughput. In the investigated high pressure situation further performance measures like the turnover time of pick orders were also highly correlated with the unit throughput rate, which is why we focused on the throughput itself as the main metric for a successful RMFS.

Furthermore, we found that varying the decision rule used for solving the Pick Order Assignment affected the unit throughput rate the most. The average unit throughput rate was twice as high for the best decision rule as it was for the worst. This finding indicates that system engineers and warehouse operators should pay most attention to the Pick Order Assignment decision problem. Moreover, the unit throughput rate score ranges from 25.24% for the worst RC assessed in phase 1 to 94.81% for the best scoring RC. Hence, the right combination of decision rules plays a crucial role when controlling an RMFS. We propose that future research may assess how to scale beyond the throughput performance of the merely simple decision rules investigated in this chapter. However, we observe some cross-dependencies between different strategies for the core decision problems featured in this chapter, e.g., the Demand PPS rule is part of the best performing and the worst performing RC. Thus, an integrated and realistic evaluation or validation of new decision methods for RMFS is highly important, since dependencies exist and side-effects should not be neglected. Additionally, we found that the number of different SKUs in the system has a strong impact on the unit throughput rate. This finding is probably due to a decrease in pile-on for

a higher number of SKUs. This effect is considerably less for larger orders, presumably because for larger pick orders pile-on tends to be higher. Having to process return orders seems to affect the unit throughput rate more, if the pick orders are large. Moreover, we found that the performance of the "greedy" benchmark consistently came close to the best ranked configurations of decision rules.

This chapter has studied solutions to several operational problems, which lead towards promising directions for future research. Each decision rule in this study has looked at an operational problem in isolation, but heuristics that try to integrate multiple operational problems and optimize these problems jointly could achieve substantial increases in order throughput or reductions in resources used. Investigating rules and heuristics that increase pile-on, i.e. the number of picks per handled pod, would also be of great use to practitioners.

While many decision rules and parameters were varied to deliver insightful results, we expect even more insight when varying the layout itself. For example, we expect a larger impact of the PSA rule selection when facing huge layout instances. Varying the layout was not done in this chapter in order to keep a certain focus and to keep computational resource utilization for the conducted experiments tractable.

Appendix

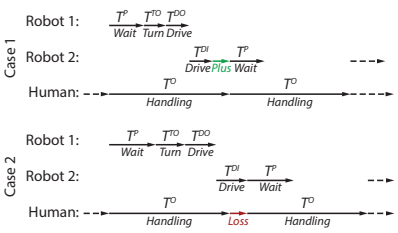
5.A Upper bound on the unit throughput rate

Table 5.A.1: Times for determining the upper bound on unit throughput (all times in seconds)

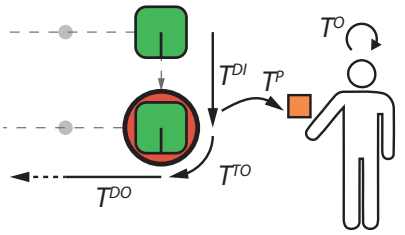
Symbol	Explanation
T^P	Time for picking one unit from the pod (after which the robot can be released, if no further picks are necessary)
T^O	Time for handling one unit at a station (including picking, putting, packing, etc.)
T^{DI}	The time for the robot to move up within the queue to the pick station's waypoint
T^{TO}	The time for the robot to prepare for leaving the station (turning towards exit)
T^{DO}	The time for the robot to clear the station's waypoint (time to cover the minimal distance)

In the following we introduce an upper bound for the number of units picked per hour. This can be done by considering the constant time for picking (T^P) and the constant time for handling (T^O) a unit at a pick station. For an overview of all necessary times see Table 5.A.1. If a robot is queueing in the buffer of a pick station, it is assumed that it already turns the right pick face of the pod towards the side where the picker will be. Since the robot is waiting in the queue, this happens in the best case without any additional loss of time. During the actual pick process, the robot is occupied for T^P

seconds. After this time the robot is allowed to leave the station while the overall handling time for one unit at a station of T^O can be longer.



(a) Sample times for both mentioned cases



(b) Times of the different steps involved in picking units

There are two cases to distinguish for obtaining a performance upper bound. First, if the time for picking a unit from the pod plus the time for moving up the next robot (i.e. the time to turn and drive away from the station and the time for the next robot to approach the station from the queue area) is smaller than the overall handling time of a unit at the station, there is a surplus of time available on the system side and the performance is limited by the handling time of the picker (see case 1 in Figure 5.A.1a). In the second case we face a longer time for moving up the next robot, hence, in this case we have a loss of time on the system's side and the system is limiting the throughput performance of the picker (see case 2 in Figure 5.A.1a). For the sake of clarity we define the time for moving up the next robot in queue as $T^{MU} := T^{DI} + T^{TO} + T^{DO}$.

$$UB := \begin{cases} \left|\mathcal{M}^O\right| \dfrac{3600}{T^O} & T^P + T^{MU} \leq T^O \\ \left|\mathcal{M}^O\right| \mathrm{IPO} \dfrac{3600}{T^P + T^{MU} - T^O + \mathrm{IPO} T^O} & \text{else} \end{cases} \tag{5.10}$$

Considering both cases we can determine an upper bound on the unit throughput rate, i.e. the number of units picked per hour (see Equation 5.10). For the first case we only need to consider the unit handling time of the picker

to determine the maximum throughput rate of one station and multiply it with the overall count of pick stations $\left|\mathcal{M}^O\right|$. The second case is slightly more complicated, because we also need to consider the pile-on. In the denominator we first calculate the loss of time seen in Figure 5.A.1a and add it to the average handling time of a pod based on the estimated number of picks from it (IPO). We calculate how many pods are handled in one hour and multiply this by the IPO and the number of stations overall to get the overall upper bound.

We recognize that this upper bound on the unit throughput rate relies on some heavy assumptions for real systems, but still propose it as a rule-of-thumb for practitioners, since it is useful for implementations where the time for moving the next robot, the handling times and the pile-on can be estimated. It is a natural limit of the system's performance that the system cannot exceed, even if the number of robots is more than sufficient to supply a continuous stream of pods and all rules are performing well. For the work in this chapter the upper bound is correct, because all mentioned times can be accurately determined or are constant and are not subject to random influence within the simulation.

6 Conclusions and future outlook

6.1 Conclusions

This dissertation studies how to optimize the performance of robotic mobile fulfillment systems. Chapter 2 creates queueing networks and uses them to show where to place workstations and how to size the storage area in order to maximize the order throughput.
Chapter 3 examines how to balance the workloads at the pick process and the replenishment process by exploring the trade-offs between three decisions: across how many pods to spread the inventory of a SKU, how many workstations to assign to picking and to replenishment, and when to send a pod for replenishment. This chapter expands on the queueing networks from Chapter 2 and creates a new type of queueing network, the cross-class matching multi-class SOQNs. Interestingly, the stability conditions for the cross-class matching multi-class SOQNs shows that the network can become unstable if the order arrival rates for certain classes of orders becomes too low.
Chapter 4 also studies the balance between the pick process and the replenishment process, but instead focuses on dynamic, real-time continual reallocation of robots and workers across both processes, that is executed in response to time-varying demand. This chapter describes a queueing network that is larger than the ones used in chapters 2 and 3 and that contains different kinds of tokens. This network is made more compact and subsequently integrated into a Markov Decision Process (MDP). The MDP

is then used to compare benchmark policies from practice with each other and with the optimal policy.
Lastly, Chapter 5 uses a detailed and realistic simulation of the RMFS to understand for several decision problems, which decision rule optimizes performance for each decision problem. This is done in two phases, where the first phases calculates all possible combinations of decision rules across decision problems and the second phase uses the best performing decision rule combinations from the first phase and applies them to a variety of warehouse scenarios. Performance is measured in eight different ways, but the measures turn out to be highly correlated, meaning that one performance measure, the unit throughput, suffices.

6.1.1 Chapter 2

The aim of this chapter is to obtain new insights for the warehouse design of RMFSs. This chapter develops queueing networks that are capable of modeling accurate driving behavior of robots, storage zoning, and multi-line orders. Furthermore, this chapter derive analytical expressions for the distributions of the robot travel times and uses them in the queueing networks. The models allow us in a short amount of time to analytically calculate the maximum order throughput, the robot utilization, and order cycle time for a given RMFS warehouse design.
As mentioned in Section 1.3, the results of the experiments lead to the following insights: (1) the analytical models accurately estimate the average order cycle time, and robot utilization, (2) the maximum order throughput appears to be quite insensitive to the length-to-width ratio of the storage area, and (3) the maximum order throughput is affected by the location of the workstations around the storage area.
Two limitations of this study are that congestion and robot switching between workstations have not been included in the model. In practice however, congestion should only have a small effect on the order throughput and

workstation utilization, since the system is designed with the aim that the picker is kept busy to achieve high pick rates. Congestion may then cause a robot to enter the queue at the workstation a little later but the bottleneck in the system is the picker and not transportation. Also, congestion is unlikely to happen, as the number of robots is typically small relative to the space in which they drive and robots can travel underneath the racks when they are not carrying a pod. This makes it unlikely that the robots run into multiple other robots in the adjacent space around them. Robot switching could be beneficial if the workstations have a low utilization.

6.1.2 Chapter 3

This chapter examines three decision variables jointly: the number of pods across which the inventory of a SKU should be spread, ratio of the number of pick stations to the number of replenishment stations, and the replenishment level. The results show that each of these decision variables can be optimized in order to minize order throughput time. The order throughput time appears lowest when inventory is spread across as many pods as possible, when the ratio of the number of pick stations to replenishment stations is 4 to 2, and when the replenishment level is 50%. The relationship between order throughput time and the ratio of the number of pick stations to replenishment stations has a skewed U shape. Moreover, the replenishment level has a strong influence on the utilization of the replenishment stations, which can become quite low.

6.1.3 Chapter 4

This chapter explores how resources, in this case robots and workers, should be allocated across both the pick process and replenishment process in order to minimize costs. This is particularly important in online retail environments, which suffer from high demand peaks and which promise short response times

to their customers. If resources are allocated ineffectually, the warehouse may be running low on inventory within the storage area at the beginning of the peak demand period, the period where most resources should be allocated to the pick process in order to minimize response times. In an RMFS, resources such as robots and workers can be reallocated between both processes in a short amount of time (with no or little setup). We build an MDP model that embeds a queuing network and that can find the optimal policy, i.e., find which resource allocation in any situation leads to minimizing costs. Costs in our model include costs related to customer waiting times and related to lost sales. We compare several benchmark policies from practice with the optimal policy for small instances, and find that the benchmark policies perform similarly to the optimal policy. We can therefore use these benchmark policies in for larger, real-life size instances. Four benchmark policies from practice are evaluated. The fixed policy keeps the resource allocation fixed, whereas the demand-dependent policy has two different resource allocations, one for high demand periods and one for low demand periods. The cost-dependent policy also has two different resource allocations, depending on the ratio of picking to replenishment costs. Finally, the state-dependent policy uses the transition probabilities of the MDP model to estimate the expected cost of an allocation and chooses the one with lowest cost. The cost-dependent and state-dependent policies outperform the fixed and the demand-dependent policies. Continually reallocating resources based on the state of the system (order demand rate, allocation, and number of waiting orders) appears to bring substantial cost savings. The benefits are even more pronounced when the number of robots is small, because then the cost-dependent and/or state-dependent policies can sharply reduce costs compared to the fixed and demand-dependent policies. We also show that the characteristics of peak demand have a strong effect on the costs. Given a fixed average order arrival rate across all demand phases, we varied the length of the peak demand phase and the height of the peak, and found that this influences both the average costs but also the type of policy that minimizes costs. In order to

minimize costs in an RMFS, the resource reallocation policies must be chosen based to the expected duration and height of peak demand.

6.1.4 Chapter 5

In this chapter we studied the throughput performance of decision rules for multiple decision problems occurring in the control of an RMFS. We analyzed a total of eight output measures for a total of 1620 RCs, and we found strong correlations between these output measures. Most interestingly, two output measures, namely a high pile-on and a short robot travel distance, were the most important determinant for whether a decision rule turned out to be better than other decision rules for the same decision problem. Hence, we propose using these two output measures as the key performance indicators when designing decision strategies for an RMFS that aims to achieve a high throughput. In the investigated high demand situation, other performance measures such as the turnover time of pick orders are also highly correlated with the unit throughput rate, which is why we focused on the unit throughput itself as the main metric for performance.
Furthermore, we found that varying the decision rule used for solving the Pick Order Assignment affected the unit throughput rate the most. The average unit throughput rate was twice as high for the best decision rule as it was for the worst. This finding indicates that system engineers and warehouse operators should pay most attention to the Pick Order Assignment decision problem. Moreover, the unit throughput rate score ranges from 25.24% for the worst RC assessed in phase 1 of the analysis, to 94.81% for the best scoring RC. Hence, the right combination of decision rules plays a crucial role when controlling an RMFS. In addition, we observe some cross-dependencies between different strategies for the core decision problems featured in this chapter, e.g., the Demand PPS rule is part of the best performing and the worst performing RC. Thus, an integrated and realistic evaluation or validation of new decision methods for RMFSs is highly

important, since dependencies exist and side-effects should not be neglected. Additionally, we found that the number of different SKUs in the system has a strong impact on the unit throughput rate. This finding is probably due to a decrease in pile-on for a higher number of SKUs. This effect is considerably less for larger orders, presumably because for larger pick orders pile-on tends to be higher. Having to process return orders seems to affect the unit throughput rate more, if the pick orders are large. Moreover, we found that the performance of the "greedy" benchmark consistently comes close to the most highly ranked RCs.

6.2 Future outlook

This dissertation studies a wide variety concepts and solves a wide range of decision problems related to RMFSs. Nonetheless, RMFSs offer such a rich context in which to explore warehousing concepts and decision problems that no single dissertation could ever hope to cover them all. More specifically, Section 1.2 identified five interesting concepts, namely: (1) "pile-on", (2) "well-sortedness", (3) priority zoning, (4) dynamic resource allocation, and (5) the centralization-decentralization trade-off. Of these five concepts, "pile-on" and "dynamic resource reallocation" are investigated, in Chapters 5 and 4, respectively. Investigating the other concepts would require additional chapters to this dissertation, because these concepts are specific to RMFSs and have not been modeled yet. Modeling them would, however, not only shed light on aspects of RMFSs, but also lead to insights for other material handling systems. We already saw this phenomenon in Chapter 3, where a new queueing network is created in order to model SKUs on pods, which led to new stability conditions and a new class of semi-open queueing network that can be used to model (aspects of) other material handling systems. With regard to the unexplored concepts, we expect that addressing them will lead to new insights for other systems as well. The concept of "well-sortedness" is closely related to the concept of pre-marshalling in for example a container

terminal context. However, in RMFSs, the inventory is continually sorted throughout operations, whereas pre-marshalling is typically done only before operations and not during operations. Studying well-sortedness in RMFSs may therefore lead to new insights and solution methods that may also benefit systems that have a form of pre-marshalling. Priority zoning is similar to the concept of a "forward area", with the two main differences being that priority zoning is done within one contiguous area rather than across multiple areas within the warehouse, and that the time span differs with regard to SKU allocation. Which SKUs are allocated to the forward area is typically decided once every few months or years, whereas for priority zoning this decision may be taken multiple times per day. Studying priority zoning may yield new insights with regard to the use of a forward area within a warehouse.
Lastly, an interesting concept to explore further is the centralization - decentralization trade-off. The RMFS has multiple decision problems on the operational level that intersect and that can benefit from being optimized jointly, as explained in Chapter 5. However, these problems can usually be solved in both a centralized and decentralized manner. The path planning of the robots can be done by a central computer, but in most implementations the robots plan their own paths, i.e. path planning is done decentralized rather than centralized. Which decision problems should be solved in a centralized manner and which ones in a decentralized manner is largely unexplored territory. Moreover, the effect that solving a certain decision problem in a centralized or decentralized manner has on the other decision problems is unexplored as well. Game theory, in particular auction theory, may lend itself well to solving problems in an RMFS in a decentralized manner, as argued in Wurman & Enright (2011) and Wurman et al. (2008). Solving some problems in a centralized manner and others in a decentralized manner may therefore lead to new approaches where solution methods from different disciplines are combined and integrated.

Bibliography

Agatz, N. A. H., Fleischmann, M., & van Nunen, J. A. E. E. (2008). E-fulfillment and multi-channel distribution: a review. *European Journal of Operational Research*, 187, 339–356.

Archibald, T. W. (2007). Modelling replenishment and transshipment decisions in periodic review multilocation inventory systems. *The Journal of the Operational Research Society*, 58, 948–956.

Azadeh, K., de Koster, M. B. M., & Roy, D. (2017). Robotized warehouse systems: Developments and research opportunities. Available at SSRN: `https://ssrn.com/abstract=2977779`.

Beckschäfer, M., Malberg, S., Tierney, K., & Weskamp, C. (2017). Simulating storage policies for an automated grid-based warehouse system. In T. Bektas, S. Coniglio, A. Martinez-Sykora, & S. Voß (Eds.), *Computational logistics*, Lecture Notes in Computer Science (pp. 468–482). Cham: Springer.

Bhat, S. & Krishnamurthy, A. (2012). Production rate strategies for manufacturing systems with seasonal demands. In *Proceedings of the 2012 Industrial and Systems Engineering Research Conference*.

Bhat, S. & Krishnamurthy, A. (2013). Flexible policies for multi-class systems with seasonal demands. In *Proceedings of the 2013 Industrial and Systems Engineering Research Conference* (pp. 2846–2854).

Bhat, S. & Krishnamurthy, A. (2015). Value of capacity flexibility in manufacturing systems with seasonal demands. *IIE Transactions*, 47, 693–714.

Bhat, S. & Krishnamurthy, A. (2016a). Interactive effects of seasonal-demand characteristics on manufacturing systems. *International Journal of Production Research*, 54(10), 2951–2964.

Bhat, S. & Krishnamurthy, A. (2016b). Production control policies to maintain service levels in different seasons. *Journal of Manufacturing Systems*, 41, 31–44.

Bolch, G., Greiner, S., de Meer, H., & Trivedi, K. S. (2006). *Queueing Networks and Markov Chains*. Wiley Publishing Inc.

Bond, J. (2016). 3PLs bid on a new proposal. *Modern Materials Handling*, 9, 41–44.

Boysen, N., Briskorn, D., & Emde, S. (2017). Parts-to-picker based order processing in a rack-moving mobile robots environment. *European Journal of Operational Research*, 262(2), 550–562.

Buitenhek, R., Van Houtum, G.-J., & Zijm, H. (2000). AMVA-based solution procedures for open queueing networks with population constraints. *Annals of Operations Research*, 93, 15–40.

Business Insider (2016). `http://www.businessinsider.in/Amazon-is-now-using-a-whole-lot-more-of-the-robots-from-the-company-it-bought-for-775-million/articleshow/49499530.cms`. [Accessed December 18th, 2016].

Business Wire (2015). Amazon unveils its eighth generation fulfillment center. `www.businesswire.com/multimedia/home/20141130005031/en`. [Accessed September 23rd, 2016].

Chadès, I., Chapron, G., Cros, M.-J., Garcia, F., & Sabbadin, R. (2014). Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography*, 37(9), 916–920.

Chen, C. M., Gong, Y., de Koster, M. B. M., & van Nunen, J. A. E. E. (2010). A flexible evaluative framework for order picking systems. *Production and Operations Management*, 19(1), 70–82.

Ching, W. K., Chan, R. H., & Zhou, X. Y. (1997). Circulant preconditioners for markov-modulated poisson processes and their applications to manufacturing systems. *Siam Journal of Matrix Analytical Applications*, 18(2), 464–481.

Crabil, T. B. (1974). Optimal control of a maintenance system with variable service rates. *Operations Research*, 22(4), 736–745.

Dhingra, V., Kumawat, G. L., Roy, D., & de Koster, R. (2017). Solving semi-open queuing networks with time-varying arrivals: An application in container terminal landside operations. *European Journal of Operational Research*.

E-commerce Europe (2016). European b2c e-commerce report 2016. `https://www.ecommercewiki.org/wikis/www.ecommercewiki.org/images/2/25/European_B2C_Ecommerce_Report_2016.pdf`.

Ekren, B. Y., Heragu, S. S., Krishnamurthy, A., & Malmborg, C. J. (2014). Matrix-geometric solution for semi-open queuing network model of autonomous vehicle storage and retrieval system. *Computers & Industrial Engineering*, 68, 78–86.

Fischer, W. & Meier-Hellstern, K. (1992). The markov-modulated poisson process (MMPP) cookbook. *Performance Evaluation*, 18, 149–171.

Frazzoli, E., Dahleh, M. A., & Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control and Dynamics*, 25(1), 116–129.

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

Grey Orange (2016). `http://www.greyorange.com/`. [Accessed December 18th, 2016].

Gue, K., Akcali, E., Erera, A., Ferrell, B., & Forger, G. (2014). Material Handling & Logistics: US Roadmap. `http://www.mhlroadmap.org/roadmap.html`. [Accessed January 10th, 2017].

Hazard, C. J., Wurman, P. R., & D'Andrea, R. (2006a). Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *Proceedings of AAAI Workshop on Auction Mechanisms for Robot Coordination* (pp. 23–30).: Citeseer.

Hazard, C. J., Wurman, P. R., & D'Andrea, R. (2006b). Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *Proceedings of AAAI Workshop on Auction Mechanisms for Robot Coordination* (pp. 23–30).: Citeseer.

Heragu, S. S., Cai, X., Krishnamurthy, A., & Malmborg, C. J. (2011). Analytical models for analysis of automated warehouse material handling systems. *International Journal of Production Research*, 49(22), 6833–6861.

Jia, J. (2005). *Solving Semi-Open Queuing Networks*. PhD thesis, Rensselaer Polytechnic Institute.

Jünemann, R. (1989). *Materialfluß und Logistik Systemtechnische Grundlagen mit Praxisbeispielen*. Springer-Verlag.

Kiva Systems (2010). How kiva systems and warehouse management systems interact. White Paper.

Kuo, P.-H., Krishnamurthy, A., & Malmborg, C. J. (2007). Design models for unit load storage and retrieval systems using autonomous vehicle technology and resource conserving storage and dwell point policies. *Applied Mathematical Modelling*, 31, 2332–2346.

Lamballais, T., Merschformann, M., Roy, D., Suhl, L., & De Koster, M. B. M. (2018a). Dynamic policies for resource reallocation in a robotic mobile fulfillment system with time-varying demand.

Lamballais, T., Roy, D., & de Koster, M. B. M. (2017). Estimating performance in a robotic mobile fulfillment system. *European Journal of Operations Research*, 256, 976–990.

Lamballais, T., Roy, D., & De Koster, M. B. M. (2018b). Inventory allocation in robotic mobile fulfillment systems. Forthcoming in IISE Transactions.

Lazowska, E. D., Zahorjan, J., Graham, G. S., & Sevcik, K. C. (1984). *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc.

Li, X. (2013). Managing dynamic inventory systems with product returns: A markov decision process. *Journal of Optimization Theory and Applications*, 157, 577–592.

Lu, W., McFarlane, D., Giannikas, V., & Zhang, Q. (2016). An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, 248, 107–122.

Marchet, G., Melacini, M., Perotti, S., & Tappia, E. (2013). Development of a framework for the design of autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, 51(14), 4365–4387.

Merschformann, M., Lamballais, T., De Koster, M. B. M., & Suhl, L. (2018). Decision rules for robotic mobile fulfillment systems.

Merschformann, M., Xie, L., & Erdmann, D. (2017a). Path planning for robotic mobile fulfillment systems. Working paper, available at arXiv, `https://arxiv.org/abs/1706.09347`.

Merschformann, M., Xie, L., & Li, H. (2017b). RAWSim-O: A simulation framework for robotic mobile fulfillment systems. Working Paper, available at arXiv, https://arxiv.org/abs/1710.04726.

MHI & Deloitte (2014). *The 2014 MHI Annual Industry Report: Innovations that drive supply chains*. Technical report, MHI. [Available at `https://www.mhi.org/publications/report`, Accessed January 10th, 2017].

MHI & Deloitte (2015). *The 2015 MHI Annual Industry Report: Supply chain innovation - Making the impossible possible.* Technical report, MHI. [Available at `https://www.mhi.org/publications/report`, Accessed January 10th, 2017].

MHI & Deloitte (2016). *The 2016 MHI Annual Industry Report: Accelerating change: How innovation is driving digital, always-on supply chains.* Technical report, MHI. [Available at `https://www.mhi.org/publications/report`, Accessed January 10th, 2017].

Min, H. (2007). Examining sources of warehouse employee turnover. *International Journal Of Physical Distribution & Logistics Management*, 37(5), 375–388.

MiR (2016). `http://mobile-industrial-robots.com/en/mir100-2/`. [Accessed December 18th, 2016].

Morris, J. (2015). The evolution of e-commerce warehouses: e-commerce is transforming DCs into the retail stores of the future as consumers increasingly purchase merchandise online. *Material Handling & Logistics*, 9, 18–21.

Mountz, M. (2012). Kiva the disrupter. *Harvard Business Review*, 90, 74–80.

Nigam, S., Roy, D., de Koster, M. B. M., & Adan, I. J. B. F. (2014). Analysis of class-based storage strategies for the mobile shelf-based order pick system. In *Progress in Material Handling Research: 2014*.

Olender, A. (2012). Warehouse design for e-commerce. *MHD Supply Chain Solutions*, 42(6), 42–44.

Patil, H. & Divekar, B. R. (2014). Inventory management challenges for B2C e-commerce retailers. *Procedia Economics and Finance*, 11, 561–571.

Prabhu, N. U. & Zhu, Y. (1989). Markov modulated queueing systems. *Queueing Systems*, 5, 215–246.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Publishing Inc.

Roodbergen, K. J., Vis, I. F., & Taylor, G. D. (2014). Simultaneous determination of warehouse layout and control policies. *International Journal of Production Research*, 53(11), 3306–3326.

Roy, D. (2016). Semi-open queuing networks: a review of stochastic models, solution methods and new research areas. *International Journal of Production Research*, 54(6), 1735–1752.

Roy, D., Krishnamurthy, A., Heragu, S. S., & Malmborg, C. J. (2012). Performance analysis and design trade-offs in warehouses with autonomous vehicle technology. *IIE Transactions*, 44, 1045–1060.

Roy, D., Krishnamurthy, A., Heragu, S. S., & Malmborg, C. J. (2013). Blocking effects in warehouse systems with autonomous vehicles. *IEEE Transactions on Automation Science and Engineering*, 99, 1–13.

Roy, D., Krishnamurthy, A., Heragu, S. S., & Malmborg, C. J. (2015a). Queuing models to analyze dwell-point and cross-aisle location in autonomous vehicle-based warehouse systems. *European Journal of Operational Research*, 242, 72–87.

Roy, D., Krishnamurthy, A., Heragu, S. S., & Malmborg, C. J. (2015b). Stochastic models for unit-load operations in warehouse systems with autonomous vehicles. *Annals of Operations Research*, 231, 129–155.

Roy, D., Krishnamurthy, A., Heragu, S. S., & Malmborg, C. J. (2016). A simulation framework for studying blocking effects in warehouse systems with autonomous vehicles. *European Journal of Industrial Engineering*, 10(1), 51–80.

Scallog (2016). `http://www.scallog.com/en_US/goods-to-man/scallog-system/`. [Accessed December 18th, 2016].

Schleyer, M. & Gue, K. R. (2012). Throughput time distribution analysis for a one-block warehouse. *Transportation Research Part E*, 48, 652–666.

Seidscher, A. & Minner, S. (2013). A semi-markov decision problem for proactive and reactive transshipments between multiple warehouses. *European Journal of Operational Research*, 230, 42–52.

Singularity Hub (2016). `https://singularityhub.com/2009/05/08/kiva-robots-continue-to-conquer-warehouses/`. [Accessed December 18th, 2016].

Swisslog (2016). `http://www.swisslog.com/carrypick`. [Accessed December 18th, 2016].

Tappia, E., Roy, D., De Koster, M. B. M., & Melacini, M. (2016). Modeling, analysis, and design insights for shuttle-based compact storage systems. *Transportation Science*, 51, 269–295.

Tunc, H., Kilic, O. A., Tarim, S. A., & Eksioglu, B. (2011). The cost of using stationary inventory policies when demand is non-stationary. *Omega*, 39, 410–415.

van Hoorn, M. H. & Seelen, L. P. (1983). The SPP/G/1 queue: a single server queue with a switched poisson process as input process. *O.R. Spektrum*, 5, 207–218.

van Nieuwenhuyse, I. & De Koster, M. B. M. (2009). Evaluating order throughput time in 2-block warehouses with time window batching. *International Journal of Production Economics*, 121, 654–664.

Viswanadham, N. & Narahari, Y. (1992). *Performance modeling of automated manufacturing systems*. Prentice Hall Englewood Cliffs, NJ.

Walker Sands Communications (2016). Reinventing retail: four predictions for 2016 and beyond. `http://www.walkersands.com/images/files/image/pdf/Walker-Sands-2016-Future-of-Retail-Four-Key-Takeaways-for-Retailers-in-2016-and-Beyond-Whitepaper.pdf`.

Wang, K. H. C. & Botea, A. (2008). Fast and memory-efficient multi-agent pathfinding. *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, (pp. 380–387).

Wulfraat, M. (2012). Is Kiva systems a good fit for your distribution center? An unbiased distribution consultant evaluation. `http://www.mwpvl.com/html/kiva_systems.html`.

Wurman, P. R., D'Andrea, R., & Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1), 9–19.

Wurman, P. R. & Enright, J. J. (2011). Optimization and coordinated autonomy in mobile fulfillment systems. *Working paper*.

Zou, B., Gong, Y., Xu, X., & Yuan, Z. (2017). Assignment rules in robotic mobile fulfilment systems for online retailers. *International Journal of Production Research*, 55(20), 6175–6192.

Zou, B., Xu, X., Gong, Y., & De Koster, M. B. M. (2016). Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. Working Paper.

Zou, B., Xu, X., Gong, Y. Y., & de Koster, R. (2018). Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. *European Journal of Operational Research*, 267(2), 733–753.

About the author

Tim Lamballais Tessensohn was born on February 13th 1988 in Capelle aan den IJssel, the Netherlands. After graduation in 2006 from Marnix Gymnasium he spent two years studying Architecture in Delft before entering the Bachelor program of Econometrics and Operations Research in Rotterdam. In 2012 he graduated cum laude in the master program Econometrics and Management Science with a specialization in Operations Research and Quantitative Logistics. Before his graduation he worked for two years as a research assistant in the field of Entrepreneurship at the department of Applied Economics, where he conducted research on the relationship between the business cycle and entrepreneurship. From 2012 to 2016 he was a PhD candidate at the Rotterdam School of Management, at the department of Management of Technology and Innovation, where his research focused on the Stochastic Modeling of Material Handling Systems. As part of two collaboration projects he spent three months in Paderborn, Germany, in the group of Professor Leena Suhl.

Portfolio

Publications

Publications in journals:

Lamballais, T., Roy, D., & de Koster, M. B. M. (2017). Estimating performance in a robotic mobile fulfillment system. *European Journal of Operations Research*, 256, 976–990

Working papers:

Lamballais, T., Roy, D., & De Koster, M. B. M. (2018b). Inventory allocation in robotic mobile fulfillment systems. Forthcoming in IISE Transactions

Lamballais, T., Merschformann, M., Roy, D., Suhl, L., & De Koster, M. B. M. (2018a). Dynamic policies for resource reallocation in a robotic mobile fulfillment system with time-varying demand

Merschformann, M., Lamballais, T., De Koster, M. B. M., & Suhl, L. (2018). Decision rules for robotic mobile fulfillment systems

PhD Courses

Algorithmic Methods in Queueing Theory	Statistical Methods
Advanced Statistical Methods	Cooperative Games
Teaching, Presenting, and Writing in English	OR-Games
Publishing Strategy	Scientific Integrity
Algoritmic Game Theory	Noncooperative Games
Stochastic Dynamic Optimization	

Teaching

Lecturer:
Research Training and Bachelor Thesis 2014-2015

Tutorial lecturer:
Primaire Processen 2013-2016
Operations Management 2013-2016
OR Method 2015-2017
SCM Methods and Skills 2014-2015
Facility Logistics Management 2012-2016
Verdiepingsvak O&SCM 2013-2014
SELS Logistics Academy 2014-2015
Global Supply Chain Management, CEMS track 2014-2015

Conferences attended

LNMB Conference 2013, Lunteren, The Netherlands
EURO 2013, Rome, Italy
OR 2013, Rotterdam, The Netherlands
LNMB Conference 2014, Lunteren, The Netherlands
ELA 2014, Hamburg, Germany
CEMS Saint Louis 2015, Brussels, Belgium
EURO 2015, Glasgow, The United Kingdom
OR 2015, Vienna, Austria
ICCL 2015, Delft, The Netherlands
BNAIC 2015, Hasselt, Belgium
IMHRC 2016, Karlsruhe, Germany
ISSL 2016, Karlsruhe, Germany
EURO 2016, Poznan, Poland

Summary

Consumers are increasingly ordering online. The rise of e-commerce has created new pressures and challenges for warehouse operations. The Robotic Mobile Fulfillment System (RMFS) is a novel automated parts-to-picker material handling system designed for warehouses processing e-commerce orders. The RMFS is flexible with regard to the shape and size of the storage area and in its capacity, as resources such as storage racks, robots, and workstations are modular and can be added or removed in a short amount of time. This flexibility makes it suitable for e-commerce companies with high average walking distances for pickers and e-commerce companies facing rapid and unpredictable growth. This thesis studies decision problems at the strategic, tactical, and operational level in an RMFS.

Chapter 2 presents queueing networks for modeling the picking process in an RMFS. These queueing networks can accurately estimate the order throughput capacity, the average order throughput time, and the robot utilization. The modeling contributes to the literature by including realistic robot movement, storage zones and multi-line orders. These models can be used to study warehouse design problems at the strategic level, such as workstation placement and storage area sizing. The results show that the optimal workstation placement depends on storage zoning and that the maximum order throughput is relatively insensitive to the width-to-length ratio of the storage area.

Chapter 3 presents queueing networks that model both the picking process and the replenishment process. These queueing networks analyze how to minimize the order throughput time by jointly optimizing three key decision

variables at the tactical level, namely the number of pods over which the inventory of a SKU should be distributed, the ratio of the number of pick stations to the number of replenishment stations, and the replenishment level. This chapter contributes to the scientific literature by providing counter-intuitive insights for practice with respect to order arrival rates, by developing a new type of queueing network, and by introducing a new modeling technique.

Chapter 4 studies dynamic resource reallocation across picking and replenishment activities in response to strong demand fluctuations. This chapter presents a combination of Markov Decision Process (MDP) models and queueing networks that can model multiple resources that are continually reallocated across both the pick process and the replenishment process. Several benchmark policies from practice are studied with different time-varying demand patterns. We find that the optimal policy choice depends on the characteristics of the peak demand period(s), that policies with continual resource reallocation outperform those without, and that if the number of robots is relatively low, continual resource reallocation can greatly reduce costs.

Chapter 5 examines several decision problems at the operational level. These decision problems have a large solution space and must to be solved in real time, which means that the solutions have to be computed in a short period of time. To achieve short computation times, simple decision rules are used. This chapter investigates several decision rules per decision problem, and uses realistic and detailed simulations to estimate the performance of these decision rules. For all combinations of decision rules, multiple performance measures are calculated across a large number of scenarios. The experiments indicate that optimizing the decision rule for the pick order assignment decision problem increases the throughput capacity the most. Moreover, warehouses with more SKUs need more robots for the same throughput capacity, even if warehouse dimensions remain unaltered.

Samenvatting (Summary in Dutch)

Consumenten bestellen steeds meer online. De opkomst van e-commerce heeft magazijnen voor nieuwe uitdagingen en eisen gesteld. Het heeft ook geleid tot nieuwe systemen om orders van klanten te verzamelen. Eén van die systemen is het "Robotgeleide-Mobiele order-Formerings-Systeem" (RMFS), een geautomatiseerd magazijnsysteem waarbij verrijdbare kasten met goederen naar werkers worden vervoerd door robots. Door de nieuwe manier waarop robots goederen vervoeren, beschikt een RMFS over verschillende soorten van flexibiliteit, die het bijzonder geschikt maakt voor e-commerce bedrijven. Dit geldt met name voor bedrijven waarbij de pickers anders gemiddeld veel zouden moeten lopen en voor bedrijven die kampen met snelle en onvoorspelbare groei.

Deze dissertatie bestudeert beslissingsproblemen op het strategische, tactische en operationele niveau in een RMFS. Hoofdstuk 2 gebruikt wachtrij-netwerken voor het modelleren van het verzamelenproces in een RMFS. Deze wachtrij-netwerken kunnen nauwkeurig de maximale doorzet, de gemiddelde doorzet en de bezettingsgraad van de robots schatten. Dit hoofdstuk draagt bij aan de wetenschappelijke literatuur, door nieuwe aspecten te modelleren, zoals realistisch rijgedrag van robots, product oplsagzones binnen de opslagruimte en bestellingen voor meerdere SKUs. De modellen kunnen gebruikt worden om het proces van het ontwerpen en inpassen van RMFS systemen in magazijnen te verbeteren (het strategische niveau), bijvoorbeeld door aan te

geven waar werkplekken het beste rondom de opslagruimte neergezet kunnen worden en wat de beste dimensionering van de opslagruimte is. De resultaten tonen aan dat de beste manier om werkplekken rondom de opslagruimte te plaatsen afhangt van hoe de opslagruimte verdeeld is in zones. De resultaten tonen ook aan dat de maximale doorzet van bestellingen niet of nauwelijks beïnvloed wordt door de lengte-breedte verhouding van de opslagruimte.
Hoofdstuk 3 bevat wachtrij-netwerken die zowel het verzamelproces als het herbevoorradingsproces modelleren. Deze wachtrij-netwerken analyseren hoe drie cruciale beslissingsvariabelen op het tactische niveau gebruikt kunnen worden om de doorlooptijd van bestellingen zo laag mogelijk te houden. Deze beslissingsvariabelen zijn ten eerste het aantal verrijdbare kasten waarin een bepaald product bewaard wordt, ten tweede het aantal werkplekken voor verzamelen ten opzichte van het aantal werkplekken voor herbevoorrading en ten derde het herbevoorradingsniveau. Dit hoofdstuk draagt bij aan de wetenschappelijke literatuur doordat het een nieuw type wachtrij-netwerk introduceert en doordat het een nieuwe manier laat zien om opdrachtklassen te gebruiken binnen wachtrij-netwerken in het algemeen. Het draagt ook bij aan de praktijk door een tegenintuïtief inzicht te bieden over de vraagfrequenties van SKUs ten opzichte van elkaar.
Hoofdstuk 4 bestudeert hoeveel robots en werkers toegewezen moeten worden aan het verzamelproces en hoeveel aan het herbevoorradingsproces en bestudeert hoe deze toewijzing het beste veranderd kan worden gedurende een shift. Het houdt daarbij rekening met sterke schommelingen in de vraag die zich gedurende een shift voor kunnen doen. Het gebruikte model combineert Markov Decision Process (MDP) modellering met wachtrij-netwerken en is daardoor in staat meerdere beslissingsvariabelen, zoals aantallen robots en werkers, te optimaliseren over meerdere processen, zoals het verzamelproces en het herbevoorradingsproces, terwijl het daarbij rekening blijft houden met veranderende vraag naar producten. We vergelijken dit model met beslisregels uit de praktijk en we bestuderen ook een aantal variaties in de manier waarop de vraag naar producten schommelt door de tijd heen. De resultaten laten zien dat de beslisregel die het beste presteert samenhangt

met hoe de vraag naar producten verandert. Verder zien we dat beslisregels die vaker robots en werkers herverdelen over het verzamelproces en het herbevoorradingsproces in het algemeen beter presteren en zien we ook dat wanneer een magazijn over weinig robots beschikt, het voortdurend herverdelen van robots over de beide processen de kosten aanzienlijk kan drukken. Hoofdstuk 5 bekijkt beslissingsproblemen op het operationele niveau. Elk van de beslissingsproblemen heeft een groot aantal mogelijk oplossingen en de oplossingen moeten in real-time gevonden worden terwijl het systeem draait. Dit betekent dat het systeem slechts kort de tijd heeft om een goede oplossing te vinden. In dit hoofdstuk wordt er daarom beslisregels bestudeerd en wordt er geen optimaliseringsmodel opgesteld. Dit hoofdstuk onderzoekt meerdere beslisregels per beslissingsprobleem en gebruikt realistische en gedetailleerde simulaties om de prestaties van deze beslisregels in te schatten. Voor alle combinaties van beslisregels worden meerdere prestatiematen uitgerekend voor een groot aantal scenario's. De experimenten geven aan dat voor het beslissingsprobleem waarbij orders aan werkplekken worden toegewezen, het veranderen van de gebruikte beslisregel een groter effect heeft op de doorlooptijd van bestellingen dan het veranderen van de beslisregel die gebruikt wordt voor andere beslissingsproblemen. Dit is dan ook het beslissingsprobleem waarbij het zich het meeste loont om een bij het magazijn passende beslisregel te vinden. Als laatste laten de resultaten ook zien dat distributiecentra met een stijgend aantal SKUs meer robots nodig hebben om de doorzet van bestellingen gelijk te houden, zelfs als ze hetzelfde aantal bestellingen verwerken en de opslagruimte hetzelfde blijft.

ERIM Ph.D. Series Research in Management

The ERIM PhD Series contains PhD dissertations in the field of Research in Management defended at Erasmus University Rotterdam and supervised by senior researchers affiliated to the Erasmus Research Institute of Management (ERIM). All dissertations in the ERIM PhD Series are available in full text through the ERIM Electronic Series Portal: `http://repub.eur.nl/pub`. ERIM is the joint research institute of the Rotterdam School of Management (RSM) and the Erasmus School of Economics at the Erasmus University Rotterdam (EUR).

Dissertations last five years

Abbink, E.J., *Crew Management in Passenger Rail Transport*, Promotor(s): Prof.dr. L.G. Kroon & Prof.dr. A.P.M. Wagelmans, EPS-2014-325-LIS, `http://repub.eur.nl/pub/76927`

Acar, O.A., *Crowdsourcing for Innovation: Unpacking Motivational, Knowledge and Relational Mechanisms of Innovative Behavior in Crowdsourcing Platforms*, Promotor(s): Prof.dr.ir. J.C.M. van den Ende, EPS-2014-321-LIS, `http://repub.eur.nl/pub/76076`

Akin Ates, M., *Purchasing and Supply Management at the Purchase Category Level: strategy, structure and performance*, Promotor(s): Prof.dr. J.Y.F. Wynstra & Dr. E.M. van Raaij, EPS-2014-300-LIS, http://repub.eur.nl/pub/50283

Akpinar, E., *Consumer Information Sharing*, Promotor(s): Prof.dr.ir. A. Smidts, EPS- 2013-297-MKT, http://repub.eur.nl/pub/50140

Alexander, L., *People, Politics, and Innovation: A Process Perspective*, Promotor(s): Prof.dr. H.G. Barkema & Prof.dr. D.L. van Knippenberg, EPS-2014-331-S&E, http://repub.eur.nl/pub/77209

Alexiou, A., *Management of Emerging Technologies and the Learning Organization : Lessons from the Cloud and Serious Games Technology*, Promotor(s): Prof. S.J. Magala, Prof. M.C. Schippers & Dr. I. Oshri, EPS-2016-404-ORG, http://repub.eur.nl/pub/93818

Almeida e Santos Nogueira, R.J. de, *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*, Promotor(s): Prof.dr.ir. U. Kaymak & Prof.dr. J.M.C. Sousa, EPS-2014-310-LIS, http://repub.eur.nl/pub/51560

Bannouh, K., *Measuring and Forecasting Financial Market Volatility using High-frequency Data*, Promotor(s): Prof.dr. D.J.C. van Dijk, EPS-2013-273-F&A, http://repub.eur.nl/pub/38240

Ben-Menahem, S.M., *Strategic Timing and Proactiveness of Organizations*, Promotor(s): Prof.dr. H.W. Volberda & Prof.dr.ing. F.A.J. van den Bosch, EPS-2013-278-S&E, http://repub.eur.nl/pub/39128

Benning, T.M., *A Consumer Perspective on Flexibility in Health Care: Priority Access Pricing and Customized Care*, Promotor(s): Prof.dr.ir. B.G.C. Dellaert, EPS-2011-241-MKT, http://repub.eur.nl/pub/23670

Benschop, N, *Biases in Project Escalation: Names, frames & construal levels*, Promotors: Prof.dr. K.I.M. Rhode, Prof.dr. H.R. Commandeur, Prof.dr. M.Keil & Dr. A.L.P. Nuijten, EPS-2015-375-S&E, http://repub.eur.nl/pub/79408

Berg, W.E. van den, *Understanding Salesforce Behavior using Genetic Association Studies*, Promotor(s): Prof.dr. W.J.M.I. Verbeke, EPS-2014-311-MKT, http://repub.eur.nl/pub/51440

Betancourt, N.E., *Typical Atypicality: Formal and Informal Institutional Conformity, Deviance, and Dynamics*, Promotor(s): Prof.dr. B. Krug, EPS-2012-262-ORG, http://repub.eur.nl/pub/32345

Beusichem, H.C. van, *Firms and Financial Markets: Empirical Studies on the Informational Value of Dividends, Governance and Financial Reporting*, Promotor(s): Prof. A. de Jong & Dr. G. Westerhuis, EPS-2016-378-F&A,http://repub.eur.nl/pub/93079

Bliek, R. de, *Empirical Studies on the Economic Impact of Trust*, Promotor(s): Prof.dr. J. Veenman & Prof.dr. Ph.H.B.F. Franses, EPS-2015-324-ORG, http://repub.eur.nl/pub/78159

Blitz, D.C., *Benchmarking Benchmarks*, Promotor(s): Prof.dr. A.G.Z. Kemna & Prof.dr. W.F.C. Verschoor, EPS-2011-225-F&A, http://repub.eur.nl/pub/22624

Boons, M., *Working Together Alone in the Online Crowd: The Effects of Social Motivations and Individual Knowledge Backgrounds on the Participation and Performance of Members of Online Crowdsourcing Platforms*, Promotor(s): Prof.dr. H.G. Barkema & Dr. D.A. Stam, EPS-2014-306-S&E, http://repub.eur.nl/pub/50711

Brazys, J., *Aggregated Marcoeconomic News and Price Discovery*, Promotor(s): Prof.dr. W.F.C. Verschoor, EPS-2015-351-F&A, http://repub.eur.nl/pub/78243

Burger, M.J., *Structure and Cooptition in Urban Networks*, Promotor(s): Prof.dr. G.A. van der Knaap & Prof.dr. H.R. Commandeur, EPS-2011-243-ORG, http://repub.eur.nl/pub/26178

Byington, E., *Exploring Coworker Relationships: Antecedents and Dimensions of Interpersonal Fit, Coworker Satisfaction, and Relational Models*,

Promotor(s): Prof.dr. D.L. van Knippenberg, EPS-2013-292-ORG, http://repub.eur.nl/pub/41508

Camacho, N.M., *Health and Marketing: Essays on Physician and Patient Decision-Making*, Promotor(s): Prof.dr. S. Stremersch, EPS-2011-237-MKT, http://repub.eur.nl/pub/23604

Cancurtaran, P., *Essays on Accelerated Product Development*, Promotor(s): Prof.dr. F. Langerak & Prof.dr.ir. G.H. van Bruggen, EPS-2014-317-MKT, http://repub.eur.nl/pub/76074

Caron, E.A.M., *Explanation of Exceptional Values in Multi-dimensional Business Databases*, Promotor(s): Prof.dr.ir. H.A.M. Daniels & Prof.dr. G.W.J. Hendrikse, EPS-2013-296-LIS, http://repub.eur.nl/pub/50005

Carvalho, L. de, *Knowledge Locations in Cities: Emergence and Development Dynamics*, Promotor(s): Prof.dr. L. Berg, EPS-2013-274-S&E, http://repub.eur.nl/pub/38449

Cranenburgh, K.C. van, *Money or Ethics: Multinational corporations and religious organisations operating in an era of corporate responsibility*, Prof. L.C.P.M. Meijs, Prof. R.J.M. van Tulder & Dr D. Arenas, EPS-2016-385-ORG, *http://repub.eur.nl/pub/93104*

Consiglio, I., *Others: Essays on Interpersonal and Consumer Behavior*, Promotor: Prof.dr. S.M.J. van Osselaer, EPS-2016-366-MKT, http://repub.eur.nl/pub/79820

Cox, R.H.G.M., *To Own, To Finance, and To Insure - Residential Real Estate Revealed*, Promotor(s): Prof.dr. D. Brounen, EPS-2013-290-F&A, http://repub.eur.nl/pub/40964

Darnihamedani, P., *Individual Characteristics, Contextual Factors and Entrepreneurial Behavior*, Promotor(s): Prof. A.R. Thurik & S.J.A. Hessels, EPS-2016-360-S&E, http://repub.eur.nl/pub/93280

Deichmann, D., *Idea Management: Perspectives from Leadership, Learning, and Network Theory*, Promotor(s): Prof.dr.ir. J.C.M. van den Ende, EPS-2012-255-ORG, http://repub.eur.nl/pub/31174

Deng, W., *Social Capital and Diversification of Cooperatives*, Promotor(s): Prof.dr. G.W.J. Hendrikse, EPS-2015-341-ORG, http://repub.eur.nl/pub/77449

Depecik, B.E., *Revitalizing brands and brand: Essays on Brand and Brand Portfolio Management Strategies*, Promotor(s): Prof. G.H. van Bruggen, dr Y.M. van Everdingen and Dr M.B. Ataman, EPS-2016406-MKT, http://repub.eur.nl/pub/93507

Desmet, P.T.M., *In Money we Trust? Trust Repair and the Psychology of Financial Compensations*, Promotor(s): Prof.dr. D. de Cremer, EPS-2011-232-ORG, http://repub.eur.nl/pub/23268

Dollevoet, T.A.B., *Delay Management and Dispatching in Railways*, Promotor(s): Prof.dr. A.P.M. Wagelmans, EPS-2013-272-LIS, http://repub.eur.nl/pub/38241

Doorn, S. van, *Managing Entrepreneurial Orientation*, Promotor(s): Prof.dr. J.J.P. Jansen, Prof.dr.ing. F.A.J. van den Bosch, & Prof.dr. H.W. Volberda, EPS-2012-258- STR, http://repub.eur.nl/pub/32166

Douwens-Zonneveld, M.G., *Animal Spirits and Extreme Confidence: No Guts, No Glory?* Promotor(s): Prof.dr. W.F.C. Verschoor, EPS-2012-257-F&A, http://repub.eur.nl/pub/31914

Duca, E., *The Impact of Investor Demand on Security Offerings*, Promotor(s): Prof.dr. A. de Jong, EPS-2011-240-F&A, http://repub.eur.nl/pub/26041

Duyvesteyn, J.G. *Empirical Studies on Sovereign Fixed Income Markets*, Promotor(s): Prof.dr P.Verwijmeren & Prof.dr. M.P.E. Martens, EPS-2015-361-F&A, http://repub.eur.nl/pub/79033

Duursema, H., *Strategic Leadership: Moving Beyond the Leader-Follower Dyad*, Promotor(s): Prof.dr. R.J.M. van Tulder, EPS-2013-279-ORG, http://repub.eur.nl/pub/39129

Eck, N.J. van, *Methodological Advances in Bibliometric Mapping of Science*, Promotor(s): Prof.dr.ir. R. Dekker, EPS-2011-247-LIS, http://repub.eur.nl/pub/26509

Elemes, A, *Studies on Determinants and Consequences of Financial Reporting Quality*, Promotor: Prof.dr. E.Peek, EPS-2015-354-F&A, http://repub.eur.nl/pub/79037

Ellen, S. ter, *Measurement, Dynamics, and Implications of Heterogeneous Beliefs in Financial Markets*, Promotor(s): Prof.dr. W.F.C. Verschoor, EPS-2015-343-F&A, http://repub.eur.nl/pub/78191

Erlemann, C., *Gender and Leadership Aspiration: The Impact of the Organizational Environment*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2016-376-ORG, http://repub.eur.nl/pub/79409

Eskenazi, P.I., *The Accountable Animal*, Promotor(s): Prof.dr. F.G.H. Hartmann, EPS- 2015-355-F&A, http://repub.eur.nl/pub/78300

Essen, M. van, *An Institution-Based View of Ownership*, Promotor(s): Prof.dr. J. van Oosterhout & Prof.dr. G.M.H. Mertens, EPS-2011-226-ORG, http://repub.eur.nl/pub/22643

Evangelidis, I., *Preference Construction under Prominence*, Promotor(s): Prof.dr. S.M.J. van Osselaer, EPS-2015-340-MKT, http://repub.eur.nl/pub/78202

Faber, N., *Structuring Warehouse Management*, Promotor(s): Prof.dr. MB.M. de Koster, Prof.dr. Ale Smidts, EPS-2015-336-LIS, http://repub.eur.nl/pub/78603

Fernald, K., *The Waves of Biotechnological Innovation in Medicine: Interfirm Cooperation Effects and a Venture Capital Perspective*, Promotor(s): Prof.dr.

E.Claassen, Prof.dr. H.P.G.Pennings & Prof.dr. H.R. Commandeur, EPS-2015-371-S&E, http://repub.eur.nl/pub/79120

Fliers, P.T., *Essays on Financing and Performance: The role of firms, banks and board*, Promotor(s): Prof. A. de Jong & Prof P.G.J. Roosenboom, EPS-2016-388-F&A, http://repub.eur.nl/pub/93019

Fourne, S.P., *Managing Organizational Tensions: A Multi-Level Perspective on Exploration, Exploitation and Ambidexterity*, Promotor(s): Prof.dr. J.J.P. Jansen & Prof.dr. S.J. Magala, EPS-2014-318-S&E, http://repub.eur.nl/pub/76075

Gaast, J.P. van der, *Stochastic Models for Order Picking Systems*, Promotor(s): Prof. M.B.M de Koster & Prof. I.J.B.F. Adan, EPS-2016-398-LIS, http://repub.eur.nl/pub/93222

Gharehgozli, A.H., *Developing New Methods for Efficient Container Stacking Operations*, Promotor(s): Prof.dr.ir. M.B.M. de Koster, EPS-2012-269-LIS, http://repub.eur.nl/pub/37779

Gils, S. van, *Morality in Interactions: On the Display of Moral Behavior by Leaders and Employees*, Promotor(s): Prof.dr. D.L. van Knippenberg, EPS-2012-270-ORG, http://repub.eur.nl/pub/38027

Ginkel-Bieshaar, M.N.G. van, *The Impact of Abstract versus Concrete Product Communications on Consumer Decision-making Processes*, Promotor(s): Prof.dr.ir. B.G.C. Dellaert, EPS-2012-256-MKT, http://repub.eur.nl/pub/31913

Gkougkousi, X., *Empirical Studies in Financial Accounting*, Promotor(s): Prof.dr. G.M.H. Mertens & Prof.dr. E. Peek, EPS-2012-264-F&A, http://repub.eur.nl/pub/37170

Glorie, K.M., *Clearing Barter Exchange Markets: Kidney Exchange and Beyond*, Promotor(s): Prof.dr. A.P.M. Wagelmans & Prof.dr. J.J. van de Klundert, EPS-2014-329-LIS, http://repub.eur.nl/pub/77183

Hekimoglu, M., *Spare Parts Management of Aging Capital Products*, Promotor: Prof.dr.ir. R. Dekker, EPS-2015-368-LIS, http://repub.eur.nl/pub/79092

Heij, C.V., *Innovating beyond Technology. Studies on how management innovation, co-creation and business model innovation contribute to firm's (innovation) performance*, Promotor(s): Prof.dr.ing. F.A.J. van den Bosch & Prof.dr. H.W. Volberda, EPS-2012-370-STR, http://repub.eur.nl/pub/78651

Heyde Fernandes, D. von der, *The Functions and Dysfunctions of Reminders*, Promotor(s): Prof.dr. S.M.J. van Osselaer, EPS-2013-295-MKT, http://repub.eur.nl/pub/41514

Heyden, M.L.M., *Essays on Upper Echelons & Strategic Renewal: A Multilevel Contingency Approach*, Promotor(s): Prof.dr.ing. F.A.J. van den Bosch & Prof.dr. H.W. Volberda, EPS-2012-259-STR, http://repub.eur.nl/pub/32167

Hoever, I.J., *Diversity and Creativity*, Promotor(s): Prof.dr. D.L. van Knippenberg, EPS-2012-267-ORG, http://repub.eur.nl/pub/37392

Hogenboom, A.C., *Sentiment Analysis of Text Guided by Semantics and Structure*, Promotor(s):Prof.dr.ir. U.Kaymak & Prof.dr. F.M.G. de Jong, EPS-2015-369-LIS, http://repub.eur.nl/pub/79034

Hogenboom, F.P., *Automated Detection of Financial Events in News Text*, Promotor(s): Prof.dr.ir. U. Kaymak & Prof.dr. F.M.G. de Jong, EPS-2014-326-LIS, http://repub.eur.nl/pub/77237

Hollen, R.M.A., *Exploratory Studies into Strategies to Enhance Innovation-Driven International Competitiveness in a Port Context: Toward Ambidextrous Ports*, Promotor(s) Prof.dr.ing. F.A.J. Van Den Bosch & Prof.dr. H.W.Volberda, EPS-2015-372-S&E, http://repub.eur.nl/pub/78881

Hoogendoorn, B., *Social Entrepreneurship in the Modern Economy: Warm Glow, Cold Feet*, Promotor(s): Prof.dr. H.P.G. Pennings & Prof.dr. A.R. Thurik, EPS-2011-246-STR, http://repub.eur.nl/pub/26447

Hoogervorst, N., *On The Psychology of Displaying Ethical Leadership: A Behavioral Ethics Approach*, Promotor(s): Prof.dr. D. de Cremer & Dr. M. van Dijke, EPS-2011- 244-ORG, http://repub.eur.nl/pub/26228

Hout, D.H. van, *Measuring Meaningful Differences: Sensory Testing Based Decision Making in an Industrial Context; Applications of Signal Detection Theory and Thurstonian Modelling*, Promotor(s): Prof.dr. P.J.F. Groenen & Prof.dr. G.B. Dijksterhuis, EPS-2014-304-MKT, http://repub.eur.nl/pub/50387

Houwelingen, G.G. van, *Something To Rely On*, Promotor(s): Prof.dr. D. de Cremer & Prof.dr. M.H. van Dijke, EPS-2014-335-ORG, http://repub.eur.nl/pub/77320

Hurk, E. van der, *Passengers, Information, and Disruption*s, Promotor(s): Prof.dr. L.G. Kroon & Prof.mr.dr. P.H.M. Vervest, EPS-2015-345-LIS, http://repub.eur.nl/pub/78275

Hytonen, K.A., *Context Effects in Valuation, Judgment and Choice: A Neuroscientific Approach*, Promotor(s): Prof.dr.ir. A. Smidts, EPS-2011-252-MKT, http://repub.eur.nl/pub/30668

Iseger, P. den, *Fourier and Laplace Transform Inversion with Applications in Finance*, Promotor(s): Prof.dr.ir. R. Dekker, EPS-2014-322-LIS, http://repub.eur.nl/pub/76954

Jaarsveld, W.L. van, *Maintenance Centered Service Parts Inventory Control*, Promotor(s): Prof.dr.ir. R. Dekker, EPS-2013-288-LIS, http://repub.eur.nl/pub/39933

Jalil, M.N., *Customer Information Driven After Sales Service Management: Lessons from Spare Parts Logistics*, Promotor(s): Prof.dr. L.G. Kroon, EPS-2011-222-LIS, http://repub.eur.nl/pub/22156

Kappe, E.R., *The Effectiveness of Pharmaceutical Marketing*, Promotor(s): Prof.dr. S. Stremersch, EPS-2011-239-MKT, http://repub.eur.nl/pub/23610

Karreman, B., *Financial Services and Emerging Markets*, Promotor(s): Prof.dr. G.A. van der Knaap & Prof.dr. H.P.G. Pennings, EPS-2011-223-ORG, http://repub.eur.nl/pub/22280

Khanagha, S., *Dynamic Capabilities for Managing Emerging Technologies*, Promotor(s): Prof.dr. H.W. Volberda, EPS-2014-339-S&E, http://repub.eur.nl/pub/77319

Kil, J., *Acquisitions Through a Behavioral and Real Options Lens*, Promotor(s): Prof.dr. H.T.J. Smit, EPS-2013-298-F&A, http://repub.eur.nl/pub/50142

Klooster, E. van 't, *Travel to Learn: the Influence of Cultural Distance on Competence Development in Educational Travel*, Promotor(s): Prof.dr. F.M. Go & Prof.dr. P.J. van Baalen, EPS-2014-312-MKT, http://repub.eur.nl/pub/51462

Koendjbiharie, S.R., *The Information-Based View on Business Network Performance: Revealing the Performance of Interorganizational Networks*, Promotor(s): Prof.dr.ir. H.W.G.M. van Heck & Prof.mr.dr. P.H.M. Vervest, EPS-2014-315-LIS, http://repub.eur.nl/pub/51751

Koning, M., *The Financial Reporting Environment: The Role of the Media, Regulators and Auditors*, Promotor(s): Prof.dr. G.M.H. Mertens & Prof.dr. P.G.J. Roosenboom, EPS-2014-330-F&A, http://repub.eur.nl/pub/77154

Konter, D.J., *Crossing Borders with HRM: An Inquiry of the Influence of Contextual Differences in the Adoption and Effectiveness of HRM*, Promotor(s): Prof.dr. J. Paauwe & Dr. L.H. Hoeksema, EPS-2014-305-ORG, http://repub.eur.nl/pub/50388

Korkmaz, E., *Bridging Models and Business: Understanding Heterogeneity in Hidden Drivers of Customer Purchase Behavior*, Promotor(s): Prof.dr. S.L. van de Velde & Prof.dr. D. Fok, EPS-2014-316-LIS, http://repub.eur.nl/pub/76008

Kroezen, J.J., *The Renewal of Mature Industries: An Examination of the Revival of the Dutch Beer Brewing Industry*, Promotor(s): Prof.dr. P.P.M.A.R. Heugens, EPS-2014- 333-S&E, http://repub.eur.nl/pub/77042

Kysucky, V., *Access to Finance in a Cros-Country Context*, Promotor(s): Prof.dr. L. Norden, EPS-2015-350-F&A, http://repub.eur.nl/pub/78225

Lam, K.Y., *Reliability and Rankings*, Promotor(s): Prof.dr. Ph.H.B.F. Franses, EPS- 2011-230-MKT, http://repub.eur.nl/pub/22977

Lander, M.W., *Profits or Professionalism? On Designing Professional Service Firms*, Promotor(s): Prof.dr. J. van Oosterhout & Prof.dr. P.P.M.A.R. Heugens, EPS-2012-253- ORG, http://repub.eur.nl/pub/30682

Langhe, B. de, *Contingencies: Learning Numerical and Emotional Associations in an Uncertain World*, Promotor(s): Prof.dr.ir. B. Wierenga & Prof.dr. S.M.J. van Osselaer, EPS-2011-236-MKT, http://repub.eur.nl/pub/23504

Lee, C.I.S.G, *Big Data in Management Research: Exploring New Avenues*, Promotor(s): Prof.dr. S.J. Magala & Dr W.A. Felps, EPS-2016-365-ORG, http://repub.eur.nl/pub/79818

Legault-Tremblay, P.O., *Corporate Governance During Market Transition: Heterogeneous responses to Institution Tensions in China*, Promotor: Prof.dr. B. Krug, EPS-2015-362-ORG, http://repub.eur.nl/pub/78649

Lenoir, A.S. *Are You Talking to Me? Addressing Consumers in a Globalised World*, Promotor(s) Prof.dr. S. Puntoni & Prof.dr. S.M.J. van Osselaer, EPS-2015-363-MKT, http://repub.eur.nl/pub/79036

Leunissen, J.M., *All Apologies: On the Willingness of Perpetrators to Apologize*, Promotor(s): Prof.dr. D. de Cremer & Dr. M. van Dijke, EPS-2014-301-ORG, http://repub.eur.nl/pub/50318

Li, D., *Supply Chain Contracting for After-sales Service and Product Support*, Promotor(s): Prof.dr.ir. M.B.M. de Koster, EPS-2015-347-LIS, http://repub.eur.nl/pub/78526

Li, Z., *Irrationality: What, Why and How*, Promotor(s): Prof.dr. H. Bleichrodt, Prof.dr. P.P. Wakker, & Prof.dr. K.I.M. Rohde, EPS-2014-338-MKT, http://repub.eur.nl/pub/77205

Liang, Q.X., *Governance, CEO Identity, and Quality Provision of Farmer Cooperatives*, Promotor(s): Prof.dr. G.W.J. Hendrikse, EPS-2013-281-ORG, http://repub.eur.nl/pub/39253

Liket, K., *Why 'Doing Good' is not Good Enough: Essays on Social Impact Measurement*, Promotor(s): Prof.dr. H.R. Commandeur & Dr. K.E.H. Maas, EPS-2014-307-STR, http://repub.eur.nl/pub/51130

Loos, M.J.H.M. van der, *Molecular Genetics and Hormones: New Frontiers in Entrepreneurship Research*, Promotor(s): Prof.dr. A.R. Thurik, Prof.dr. P.J.F. Groenen, & Prof.dr. A. Hofman, EPS-2013-287-S&E, http://repub.eur.nl/pub/40081

Lovric, M., *Behavioral Finance and Agent-Based Artificial Markets*, Promotor(s): Prof.dr. J. Spronk & Prof.dr.ir. U. Kaymak, EPS-2011-229-F&A, http://repub.eur.nl/pub/22814

Lu, Y., *Data-Driven Decision Making in Auction Markets*, Promotor(s): Prof.dr.ir. H.W.G.M. van Heck & Prof.dr. W. Ketter, EPS-2014-314-LIS, http://repub.eur.nl/pub/51543

Ma, Y., *The Use of Advanced Transportation Monitoring Data for Official Statistics*, Promotors: Prof. L.G. Kroon and Dr Jan van Dalen, EPS-2016-391-LIS, http://repub.eur.nl/pub/80174

Manders, B., *Implementation and Impact of ISO 9001*, Promotor(s): Prof.dr. K. Blind, EPS-2014-337-LIS, http://repub.eur.nl/pub/77412

Markwat, T.D., *Extreme Dependence in Asset Markets Around the Globe*, Promotor(s): Prof.dr. D.J.C. van Dijk, EPS-2011-227-F&A, http://repub.eur.nl/pub/22744

Mees, H., *Changing Fortunes: How China's Boom Caused the Financial Crisis*, Promotor(s): Prof.dr. Ph.H.B.F. Franses, EPS-2012-266-MKT, http://repub.eur.nl/pub/34930

Mell, J.N., *Connecting Minds: On The Role of Metaknowledge in Knowledge Coordination*, Promotor: Prof.dr.D.L. van Knippenberg, EPS-2015-359-ORG, http://repub.eur.nl/pub/78951

Meuer, J., *Configurations of Inter-firm Relations in Management Innovation: A Study in China's Biopharmaceutical Industry*, Promotor(s): Prof.dr. B. Krug, EPS-2011-228-ORG, http://repub.eur.nl/pub/22745

Micheli, M.R., *Business Model Innovation: A Journey across Managers' Attention and Inter-Organizational Networks*, Promotor(s): Prof.dr. J.J.P. Jansen, EPS-2015-344-S&E, http://repub.eur.nl/pub/78241

Mihalache, O.R., *Stimulating Firm Innovativeness: Probing the Interrelations between Managerial and Organizational Determinants*, Promotor(s): Prof.dr. J.J.P. Jansen, Prof.dr.ing. F.A.J. van den Bosch, & Prof.dr. H.W. Volberda, EPS-2012-260-S&E, http://repub.eur.nl/pub/32343

Milea, V., *News Analytics for Financial Decision Support*, Promotor(s): Prof.dr.ir. U. Kaymak, EPS-2013-275-LIS, http://repub.eur.nl/pub/38673

Moniz, A, *Textual Analysis of Intangible Information*, Promotor(s): Prof C.B.M. van Riel, Prof. F.M.G de Jong & Dr G.A.J.M. Berens, EPS-2016-393-ORG, http://repub.eur.nl/pub/93001

Mulder, J., *Network design and robust scheduling in liner shipping*, Promotor(s): Prof. R. Dekker & Dr W.L. van Jaarsveld, Eps-2016-384-LIS, http://repub.eur.nl/pub/80258

Naumovska, I., *Socially Situated Financial Markets: A Neo-Behavioral Perspective on Firms, Investors and Practices*, Promotor(s): Prof.dr. P.P.M.A.R. Heugens & Prof.dr. A. de Jong, EPS-2014-319-S&E, http://repub.eur.nl/pub/76084

Neerijnen, P., *The Adaptive Organization: the socio-cognitive antecedents of ambidexterity and individual exploration*, Promotor(s): Prof. J.J.P. Jansen, P.P.M.A.R. Heugens & Dr T.J.M. Mom, EPS-2016-358-S&E, http://repub.eur.nl/pub/93274

Nielsen, L.K., *Rolling Stock Rescheduling in Passenger Railways: Applications in short term planning and in disruption management*, Promotor(s): Prof.dr. L.G. Kroon, EPS- 2011-224-LIS, http://repub.eur.nl/pub/22444

Nuijten, A.L.P., *Deaf Effect for Risk Warnings: A Causal Examination applied to Information Systems Projects*, Promotor(s): Prof.dr. G.J. van der Pijl, Prof.dr. H.R. Commandeur & Prof.dr. M. Keil, EPS-2012-263-S&E, http://repub.eur.nl/pub/34928

Oord, J.A. van, *Essays on Momentum Strategies in Finance*, Promotor: Prof. H.K. van Dijk, EPS-2016-380-F&A, http://repub.eur.nl/pub/80036

Osadchiy, S.E., *The Dynamics of Formal Organization: Essays on bureaucracy and formal rules*, Promotor(s): Prof.dr. P.P.M.A.R. Heugens, EPS-2011-231-ORG, http://repub.eur.nl/pub/23250

Ozdemir, M.N., *Project-level Governance, Monetary Incentives, and Performance in Strategic R&D Alliances*, Promotor(s): Prof.dr.ir. J.C.M. van den Ende, EPS-2011-235-LIS, http://repub.eur.nl/pub/23550

Peers, Y., *Econometric Advances in Diffusion Models*, Promotor(s): Prof.dr. Ph.H.B.F. Franses, EPS-2011-251-MKT, http://repub.eur.nl/pub/30586

Peters, M., *Machine Learning Algorithms for Smart Electricity Markets*, Promotor(s): Prof.dr. W. Ketter, EPS-2014-332-LIS, http://repub.eur.nl/pub/77413

Porck, J., *No Team is an Island: An Integrative View of Strategic Consensus between Groups*, Promotor(s): Prof.dr. P.J.F. Groenen & Prof.dr. D.L. van Knippenberg, EPS- 2013-299-ORG, http://repub.eur.nl/pub/50141

Porras Prado, M., *The Long and Short Side of Real Estate, Real Estate Stocks, and Equity*, Promotor(s): Prof.dr. M.J.C.M. Verbeek, EPS-2012-254-F&A, http://repub.eur.nl/pub/30848

Poruthiyil, P.V., *Steering Through: How organizations negotiate permanent uncertainty and unresolvable choices*, Promotor(s): Prof.dr. P.P.M.A.R. Heugens & Prof.dr. S.J. Magala, EPS-2011-245-ORG, http://repub.eur.nl/pub/26392

Pourakbar, M., *End-of-Life Inventory Decisions of Service Parts*, Promotor(s): Prof.dr.ir. R. Dekker, EPS-2011-249-LIS, http://repub.eur.nl/pub/30584

Pronker, E.S., *Innovation Paradox in Vaccine Target Selection*, Promotor(s): Prof.dr. H.J.H.M. Claassen & Prof.dr. H.R. Commandeur, EPS-2013-282-S&E, http://repub.eur.nl/pub/39654

Protzner, S. *Mind the gap between demand and supply: A behavioral perspective on demand forecasting*, Promotor(s): Prof.dr. S.L. van de Velde & Dr. L. Rook, EPS-2015-364-LIS, http://repub.eur.nl/pub/79355

Pruijssers, J.K., *An Organizational Perspective on Auditor Conduct*, Promotor(s): Prof.dr. J. van Oosterhout & Prof.dr. P.P.M.A.R. Heugens, EPS-2015-342-S&E, http://repub.eur.nl/pub/78192

Retel Helmrich, M.J., *Green Lot-Sizing*, Promotor(s): Prof.dr. A.P.M. Wagelmans, EPS- 2013-291-LIS, http://repub.eur.nl/pub/41330

Rietdijk, W.J.R. *The Use of Cognitive Factors for Explaining Entrepreneurship*, Promotor(s): Prof.dr. A.R. Thurik & Prof.dr. I.H.A. Franken, EPS-2015-356-S&E, http://repub.eur.nl/pub/79817

Rietveld, N., *Essays on the Intersection of Economics and Biology*, Promotor(s): Prof.dr. A.R. Thurik, Prof.dr. Ph.D. Koellinger, Prof.dr. P.J.F. Groenen, & Prof.dr. A. Hofman, EPS-2014-320-S&E, http://repub.eur.nl/pub/76907

Rijsenbilt, J.A., *CEO Narcissism: Measurement and Impact*, Promotor(s): Prof.dr. A.G.Z. Kemna & Prof.dr. H.R. Commandeur, EPS-2011-238-STR, http://repub.eur.nl/pub/23554

Rösch, D. *Market Efficiency and Liquidity*, Promotor: Prof.dr. M.A. van Dijk, EPS-2015-353-F&A, http://repub.eur.nl/pub/79121/

Roza, L., *Employee Engagement In Corporate Social Responsibility: A collection of essays*, Promotor: L.C.P.M. Meijs, EPS-2016-396-ORG, http://repub.eur.nl/pub/93254

Roza-van Vuren, M.W., *The Relationship between Offshoring Strategies and Firm Performance: Impact of innovation, absorptive capacity and firm size*, Promotor(s): Prof.dr. H.W. Volberda & Prof.dr.ing. F.A.J. van den Bosch, EPS-2011-214-STR, http://repub.eur.nl/pub/22155

Rubbaniy, G., *Investment Behaviour of Institutional Investors*, Promotor(s): Prof.dr. W.F.C. Verschoor, EPS-2013-284-F&A, http://repub.eur.nl/pub/40068

Schoonees, P. *Methods for Modelling Response Styles*, Promotor: Prof.dr P.J.F. Groenen, EPS-2015-348-MKT, http://repub.eur.nl/pub/79327

Schouten, M.E., *The Ups and Downs of Hierarchy: the causes and consequences of hierarchy struggles and positional loss*, Promotors; Prof. D.L. van Knippenberg & Dr L.L. Greer, EPS-2016-386-ORG, http://repub.eur.nl/pub/80059

Shahzad, K., *Credit Rating Agencies, Financial Regulations and the Capital Markets*, Promotor(s): Prof.dr. G.M.H. Mertens, EPS-2013-283-F&A, http://repub.eur.nl/pub/39655

Smit, J., *Unlocking Business Model Innovation: A look through the keyhole at the inner workings of Business Model Innovation*, Promotor: H.G. Barkema, EPS-2016-399-S&E, http://repub.eur.nl/pub/93211

Sousa, M.J.C. de, *Servant Leadership to the Test: New Perspectives and Insights*, Promotor(s): Prof.dr. D.L. van Knippenberg & Dr. D. van Dierendonck, EPS-2014-313-ORG, http://repub.eur.nl/pub/51537

Spliet, R., *Vehicle Routing with Uncertain Demand*, Promotor(s): Prof.dr.ir. R. Dekker, EPS-2013-293-LIS, http://repub.eur.nl/pub/41513

Staadt, J.L., *Leading Public Housing Organisation in a Problematic Situation: A Critical Soft Systems Methodology Approach*, Promotor(s): Prof.dr. S.J. Magala, EPS-2014-308- ORG, http://repub.eur.nl/pub/50712

Stallen, M., *Social Context Effects on Decision-Making: A Neurobiological Approach*, Promotor(s): Prof.dr.ir. A. Smidts, EPS-2013-285-MKT, http://repub.eur.nl/pub/39931

Tarakci, M., *Behavioral Strategy: Strategic Consensus, Power and Networks*, Promotor(s): Prof.dr. D.L. van Knippenberg & Prof.dr. P.J.F. Groenen, EPS-2013-280-ORG, http://repub.eur.nl/pub/39130

Teixeira de Vasconcelos, M., *Agency Costs, Firm Value, and Corporate Investment*, Promotor(s): Prof.dr. P.G.J. Roosenboom, EPS-2012-265-F&A, http://repub.eur.nl/pub/37265

Troster, C., *Nationality Heterogeneity and Interpersonal Relationships at Work*, Promotor(s): Prof.dr. D.L. van Knippenberg, EPS-2011-233-ORG, http://repub.eur.nl/pub/23298

Tsekouras, D., *No Pain No Gain: The Beneficial Role of Consumer Effort in Decision-Making*, Promotor(s): Prof.dr.ir. B.G.C. Dellaert, EPS-2012-268-MKT, http://repub.eur.nl/pub/37542

Tuijl, E. van, *Upgrading across Organisational and Geographical Configurations*, Promotor(s): Prof.dr. L. van den Berg, EPS-2015-349-S&E, http://repub.eur.nl/pub/78224

Tuncdoğan, A., *Decision Making and Behavioral Strategy: The Role of Regulatory Focus in Corporate Innovation Processes*, Promotor(s): Prof.dr.ing.

F.A.J. van den Bosch, Prof.dr. H.W. Volberda, & Prof.dr. T.J.M. Mom, EPS-2014-334-S&E, http://repub.eur.nl/pub/76978

Uijl, S. den, *The Emergence of De-facto Standards*, Promotor(s): Prof.dr. K. Blind, EPS-2014-328-LIS, http://repub.eur.nl/pub/77382

Vagias, D., *Liquidity, Investors and International Capital Markets*, Promotor(s): Prof.dr. M.A. van Dijk, EPS-2013-294-F&A, http://repub.eur.nl/pub/41511

Valogianni, K., *Sustainable Electric Vehicle Management using Coordinated Machine Learning*, Promotor(s): Prof. H.W.G.M. van Heck & Prof. W. Ketter, Eps-2016-387-LIS, http://repub.eur.nl/pub/93018

Veelenturf, L.P., *Disruption Management in Passenger Railways: Models for Timetable, Rolling Stock and Crew Rescheduling*, Promotor(s): Prof.dr. L.G. Kroon, EPS-2014-327-LIS, http://repub.eur.nl/pub/77155

Venus, M., *Demystifying Visionary Leadership: In search of the essence of effective vision communication*, Promotor(s): Prof.dr. D.L. van Knippenberg, EPS-2013-289-ORG, http://repub.eur.nl/pub/40079

Vermeer, W., *Propagation in Networks: The impact of information processing at the actor level on system-wide propagation dynamics*, Promotor: Prof.mr.dr. P.H.M.Vervest, EPS-2015-373-LIS, http://repub.eur.nl/pub/79325

Versluis, I., *Prevention of the Portion Size Effect*, Promotors: Prof. Ph.H.B.F. Franses & Dr E.K. Papies, EPS-2016-382-MKT, http://repub.eur.nl/pub/79880

Vishwanathan, P., *Governing for Stakeholders: How Organizations May Create or Destroy Value for their Stakeholders*, Promotors: Prof. J. van Oosterhout & Prof. L.C.P. M. Meijs, EPS-2016-377-ORG, http://repub.eur.nl/pub/93016

Visser, V.A., *Leader Affect and Leadership Effectiveness: How leader affective displays influence follower outcomes*, Promotor(s): Prof.dr. D.L. van Knippenberg, EPS-2013- 286-ORG, http://repub.eur.nl/pub/40076

Vlam, A.J., *Customer First? The Relationship between Advisors and Consumers of Financial Products*, Promotor(s): Prof.dr. Ph.H.B.F. Franses, EPS-2011-250-MKT, http://repub.eur.nl/pub/30585

Vries, J. de, *Behavioral Operations in Logistics*, Promotor(s): Prof.dr M.B.M de Koster & Prof.dr. D.A. Stam, EPS-2015-374-LIS, http://repub.eur.nl/pub/79705

Wagenaar, J.C., *Practice Oriented Algorithmic Disruption Management in Passenger Railways*, Prof. L.G. Kroon & Prof. A.P.M. Wagelmans, EPS-2016-390-LIS, http://repub.eur.nl/pub/93177

Waltman, L., *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*, Promotor(s): Prof.dr.ir. R. Dekker & Prof.dr.ir. U. Kaymak, EPS-2011-248- LIS, http://repub.eur.nl/pub/26564

Wang, P., *Innovations, status, and networks*, Promotors: Prof. J.J.P. Jansen & Dr V.J.A. van de Vrande, EPS-2016-381-S&E, http://repub.eur.nl/pub/93176

Wang, T., *Essays in Banking and Corporate Finance*, Promotor(s): Prof.dr. L. Norden & Prof.dr. P.G.J. Roosenboom, EPS-2015-352-F&A, http://repub.eur.nl/pub/78301

Wang, Y., *Information Content of Mutual Fund Portfolio Disclosure*, Promotor(s): Prof.dr. M.J.C.M. Verbeek, EPS-2011-242-F&A, http://repub.eur.nl/pub/26066

Wang, Y., *Corporate Reputation Management: Reaching Out to Financial Stakeholders*, Promotor(s): Prof.dr. C.B.M. van Riel, EPS-2013-271-ORG, http://repub.eur.nl/pub/38675

Weenen, T.C., *On the Origin and Development of the Medical Nutrition Industry*, Promotor(s): Prof.dr. H.R. Commandeur & Prof.dr. H.J.H.M. Claassen, EPS-2014-309-S&E, http://repub.eur.nl/pub/51134

Wolfswinkel, M., *Corporate Governance, Firm Risk and Shareholder Value*, Promotor(s): Prof.dr. A. de Jong, EPS-2013-277-F&A, http://repub.eur.nl/pub/39127

Yang, S., *Information Aggregation Efficiency of Prediction Markets*, Promotor(s): Prof.dr.ir. H.W.G.M. van Heck, EPS-2014-323-LIS, http://repub.eur.nl/pub/77184

Yuferova, D., *Price Discovery, Liquidity Provision, and Low-Latency Trading*, Promotors: Prof. M.A. van Dijk & Dr D.G.J. Bongaerts, EPS-2016-379-F&A, http://repub.eur.nl/pub/93017

Zaerpour, N., *Efficient Management of Compact Storage Systems*, Promotor(s): Prof.dr.ir. M.B.M. de Koster, EPS-2013-276-LIS, http://repub.eur.nl/pub/38766

Zhang, D., *Essays in Executive Compensation*, Promotor(s): Prof.dr. I. Dittmann, EPS- 2012-261-F&A, http://repub.eur.nl/pub/32344

Zwan, P.W. van der, *The Entrepreneurial Process: An International Analysis of Entry and Exit*, Promotor(s): Prof.dr. A.R. Thurik & Prof.dr. P.J.F. Groenen, EPS-2011-234-ORG, http://repub.eur.nl/pub/23422

A robotic mobile fulfillment system is a novel type of automated part-to-picker material handling system. In this type of system, robots transport mobile shelves, called pods, containing items between the storage area and the workstations. It is well suited to e-commerce, due to its modularity and it's ability to adapt to changing orders patterns. Robots can nearly instantaneously switch between inbound and outbound tasks, pods can be continually repositioned to allow for automatic sorting of the inventory, pods can contain many different types of items, and unloaded robots can drive underneath pods, allowing them to use completely different routes than loaded robots.

This thesis studies the performance of robotic mobile fulfillment systems by solving decision problems related to warehouse design, inventory and resource allocation, and real-time operations. For warehouse design, a new queueing network is developed that incorporates realistic robot movement, storage zones, and multi-line orders. For inventory allocation, we develop a new type of queueing network, the cross-class matching multi-class semi-open queueing network, which can be applied to other systems as well. Resource (re)allocation is modeled by combining queueing networks with Markov decision processes while including time-varying demand. This model compares benchmark policies from practice with the optimal policy. Lastly, we study decision rules for real-time operations by using detailed, large scale simulations.

ERiM

The Erasmus Research Institute of Management (ERIM) is the Research School (Onderzoekschool) in the field of management of the Erasmus University Rotterdam. The founding participants of ERIM are the Rotterdam School of Management (RSM), and the Erasmus School of Economics (ESE). ERIM was founded in 1999 and is officially accredited by the Royal Netherlands Academy of Arts and Sciences (KNAW). The research undertaken by ERIM is focused on the management of the firm in its environment, its intra- and interfirm relations, and its business processes in their interdependent connections.

The objective of ERIM is to carry out first rate research in management, and to offer an advanced doctoral programme in Research in Management. Within ERIM, over three hundred senior researchers and PhD candidates are active in the different research programmes. From a variety of academic backgrounds and expertises, the ERIM community is united in striving for excellence and working at the forefront of creating new business knowledge.

ERiM

ERIM PhD Series
Research in Management

Erasmus University Rotterdam (EUR)
Erasmus Research Institute of Management
Mandeville (T) Building
Burgemeester Oudlaan 50
3062 PA Rotterdam, The Netherlands

P.O. Box 1738
3000 DR Rotterdam, The Netherlands
T +31 10 408 1182
E info@erim.eur.nl
W www.erim.eur.nl