Stochastic modeling of conveyor systems

applied to the chain conveyor at

Hollander Barendrecht

Thesis

for the degree of

Master of Science

Erasmus University Rotterdam

Erasmus School of Economics

Econometrics and Management Science

January 8, 2013

Author

R.P. (Rianne) Hulst BSc.

Student number: 364660

Supervisor Erasmus

Dr. A.F. (Adriana) Gabor

Supervisor Hollander

Drs. Ing. J. (John) van den Berg

Algemeen Directeur

Hollander Barendrecht B.V.

Co-reader Erasmus

Dr. T. (Taoying) Farenhorst-Yuan

ERASMUS UNIVERSITEIT ROTTERDAM

Barendrecht hollander

Abstract

The chain conveyor in the distribution center of logistic service provider Hollander stagnates many times each day. In this thesis we investigate if the continuity of the chain conveyor could be improved, using an operational approach. With regard to this improvement, we model the chain conveyor as a queueing network. The queueing model is a general distributed, closed, multi-class queueing network, consisting of FCFS disciplined finite capacity nodes with either transfer or recirculation blocking, allowing head-of-line-priorities and multiple server nodes. We propose the MVABLO-m algorithm, in order to approximate the mean number of jobs for each node in the network. This algorithm is based on the MVABLO algorithm. We conclude that for multiple server node networks, deviations can raise up to 30%, and for multi-class networks with two job classes, deviations can raise up to 80%. Based on these results we conclude that the MVABLO-m algorithm is not suitable for modeling the chain conveyor. Finally, we incorporate the queueing model in an integer program, and solve this integer program using simulation. For all tested scenarios, we show that stagnation can be reduced using implementation of proposed loading strategies. No drastic negative side effects of this implementation are expected.

Keywords: *Optimization, conveyor system, mean value analysis, transfer blocking, simulation.*

Preface

This thesis on *Stochastic modeling of conveyor systems applied to the chain conveyor at Hollander Barendrecht* is the result of my final assignment for the study Econometrics and Management Science with specialization in Operations Research and Quantitative Logistics at the Erasmus University Rotterdam.

At the beginning of this year, logistic service provider Hollander was searching for a mechanical engineer for researching the causes of stagnation of the chain conveyor in their distribution center. Hollander uses this conveyor system to distribute both perishable and semi-perishable goods. After a visit to the distribution center, I offered Hollander to study the operations concerning the chain conveyor and investigate whether its performance could be improved. I would like to thank Hollander for taking this offer, and giving me the opportunity to do an internship with a very practical scope.

Working together with my convivial colleagues at Hollander was in the first place highly insightful, and in the second place very pleasant. Even though my questions were not always easy, everybody was very helpful by taking time to reflect on possible answers. I remember asking for useful data concerning the picking process in the distribution center. Obtaining this data turned out to be harder than expected. However, my colleagues at Hollander managed to hand me the data anyway so that I could continue with my study.

The academic aspects of my study were very educational as well. I quaffed the bitter cup during the programming of simulations and algorithms, and had a lot of fun with studying the math used for my research. Interpreting the obtained results and translate them to a practical use was very defying and fun as well.

In the first place I would like to thank my supervisor Adriana Gabor from Erasmus University Rotterdam. Her encouraging inputs during this research were very motivational and of exceptional good use. Secondly I would like to thank my co-reader Taoying Farenhorst-Yuan from Erasmus University Rotterdam for her accurate feedbacks. My thanks go out as well to my supervisor John van den Berg from Hollander Barendrecht, for sharing his expertise and practical insights regarding logistic management.

Rianne Hulst
Berkel en Rodenrijs, December 2012

Table of Contents

CHAPTER 1

Introduction

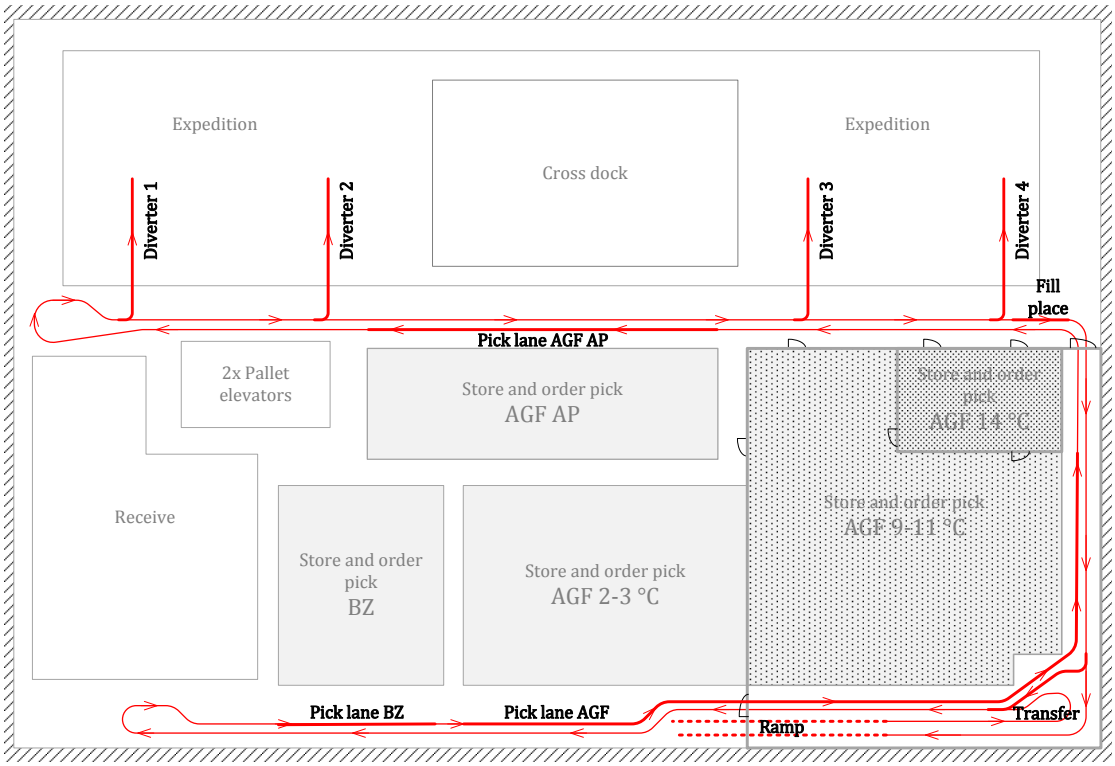
This research focuses on a case study at Hollander. Most of the goods within the distribution center (DC) are transported via a chain conveyor. Unfortunately, the chain conveyor is not operating properly, since it stagnates many times each day. We propose a closed queueing network to model the chain conveyor and the processes around it. Using the model, we investigate whether some operational adjustment will improve the continuity of the chain conveyor.

In Section 1.1 we describe Hollander and its distribution center. We formulate the research questions and give an introduction on the research approach in Section 1.2. In Section 1.3, the outline of this thesis is given.

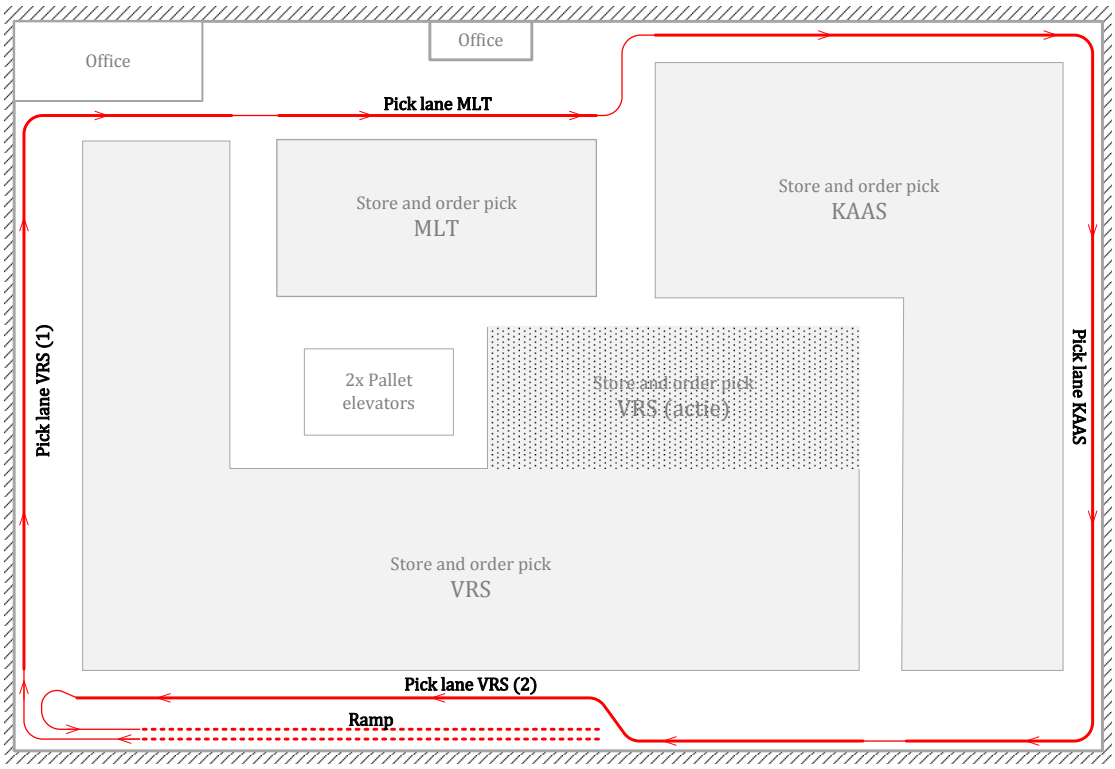
1.1 The distribution center at Hollander

Hollander was founded in the year 1929. Its founder, Siem den Hollander, started with peddling fruit and vegetables through the whole city of Rotterdam. In 1975 the place of establishment became Barendrecht, a small town south of Rotterdam. In 2008 Hollander relocated to a brand new distribution center, also in Barendrecht. Since then, Hollander became a logistic service provider of both perishable (fruit and vegetables) and semi-perishable (dairy, convenience goods) products for retail. Nowadays, Hollander delivers daily chilled products to all their customers in The Netherlands, namely about 270 Plus supermarkets spread over the whole country. Practically all perishables and semi-perishables of Plus are stored at their distribution center, where the stock of these products is managed [Hollander Barendrecht, 2012]. Hollander is a subsidiary of The Greenery, an international trading company in fresh fruit and vegetables [The Greenery, 2012].

The DC at Hollander can be divided into several regions which are spread over the two different floors, see Figure 1.1 for a graphical representation of the DC. Transportation of goods from the first to the second floor is done using two elevators. Pallets are brought to these elevators using multiple forklifts. Goods that have to be transported down from the second to the first floor travel over a ramp on roll containers by the chain conveyor, which is represented by the red, directed lines in the figure. We will refer to roll containers as *carriers*. The track of



(a) First floor.



(b) Second floor.

Figure 1.1: Floor plans of the distribution center at Hollander.

the conveyor system runs through the first floor, over the ramp, through the second floor and back down to the first floor. The main regions of the DC are defined as follows.

Receive Cargo of different vendors arrive at the receive region. Right after the products are unloaded, they are checked on quantity and quality (temperature and 'best before' date). After this check, all products are distributed towards the store regions using forklifts.

Store The products stored in the DC at Hollander are categorized into six different distributions groups. This clustering is based on the layout of the store-shelves at the Plus supermarkets. Each distribution group has its own store region in the DC, see Figure 1.1. The six store regions are semi-perishables (VRS), ready meals (MLT), cheese and meat (KAAS), bulk dairy (BZ), potatoes, vegetables and fruit (AGF), and an action square for potatoes, vegetables and fruit (AGF AP). The overall temperature of the DC is about 2° to 3° Celsius. The part where potatoes, vegetables and fruits are stored consists of three different temperature areas. Next to the overall temperature, there is one area with a temperature of 9° to 11° Celsius and one area with a temperature of 14° Celsius. At the store regions, customer orders are gathered onto carriers by an order pick process using a voice order pick system. Hereby, each order consists of products from solely one store region. After an order is picked, the corresponding carrier is transported towards the expedition region by the chain conveyor.

Expedition Loaded carriers arrive at the expedition region by the chain conveyor. Here, carriers are sorted by delivery route and set up for transport. Next to the orders that arrived by the chain conveyor, the expedition region also receives cross dock orders. The cross dock is an isolated product flow in the DC. The cross dock products are not stored at the store regions, but are set up for transport straight after they arrive from the vendor. Note that the cross dock products are not transported by the chain conveyor.

The chain conveyor is one of the most important equipments of the DC. However, in the current situation at Hollander, the chain conveyor in the DC does not work optimal as it stagnates many times each day. Therefore, Hollander suggested us to investigate whether the continuity of the chain conveyor could be improved. Since the chain conveyor does not run through the receive region, this study solely takes the store regions and the expedition region into account. Furthermore, we ignore the cross dock product flow. In the next sections of this chapter is described how we will use an operational approach in order to reduce stagnation of the chain conveyor.

1.2 Research questions

In this section, the main research question is formulated. We introduce our investigations performed in order to answer the main research question by formulating the sub-questions answered in this study.

As mentioned in the previous section, Hollander suggested us to investigate whether the continuity of the chain conveyor can be improved. Stagnations of the chain conveyor affect the total performance of the DC. The mean total stagnation duration is estimated by the management of Hollander at 30 minutes per day. After observing the processes in the DC, we stated that stagnation is caused by three main reasons, namely manual stagnation due to overwork at the fourth diverter, automatic stagnation due to overloading of the chain, or automatic stagnation due to rubbish in the chain. We choose to ignore stagnations caused by rubbish. Furthermore, Hollander is currently investigating whether it is possible to implement automatic labeling, and asked us to take this into account during our study. Summarizing, we formulate the main research question of this research as follows.

'How can the continuity of the chain conveyor at Hollander be improved by reducing stagnation using an operational approach, and what are the effects of implementing automatic labeling on the performance of the chain conveyor?'

Moreover, the chain conveyor contains a transfer which allows carriers to cut off a part of the conveyor track. However, this transfer is put out of operation. Therefore, we decide to also analyze the effects of putting the transfer in operation. In order to answer the main research question, the following sub-questions are answered in this thesis.

1. Which problems does Hollander cope with regarding the chain conveyor, and what are factors that play an important role?
2. How can we eliminate or reduce the observed occurring problems using a quantitative model?
3. Which methods proposed in literature can be used for modeling the chain conveyor, and which will we use?
4. Can the existing algorithms fully analyze the chain conveyor? If not, is it possible to design more suitable algorithms?
5. Which scenarios should be defined in order to advice Hollander for each situation in the distribution center, and what are values of the corresponding parameters in the model?
6. How can we simulate the chain conveyor?
7. Does putting the transfer in operation improve the performance of the chain conveyor?
8. What is the optimal number of carriers in the chain conveyor system in each defined scenario, and how sensitive are these optimal values?
9. Given the optimal number of carriers stored on the chain conveyor, how should the operations at chain conveyor be adjusted in order to improve its continuity?

The next section describes the structure of this thesis.

1.3 Thesis outline

The remainder of this thesis consists of five parts. The first part contains two chapters. In Chapter 2 we describe the chain conveyor, and discuss the problems Hollander has to cope with regarding the chain conveyor. Based on our observations we formulate an integer program in order to optimize the number of carriers stored on the chain and simultaneously reduce stagnation. Chapter 3 contains a literature research on potential modeling methods for the chain conveyor. The second part consists of two chapters. In Chapter 4 we propose a closed queueing network to model the chain conveyor. In Chapter 5 we first review existing mean value analysis (MVA) algorithms for related networks. Thereafter, we propose an extension to these algorithms to allow analysis of multi-class, finite capacity networks with multiple server nodes. The third part consists of three chapters. Chapter 6 describes the data analysis and defines the studied scenarios. Chapter 7 describes a simulation model for the chain conveyor. Chapter 8 describes the Binary Search algorithm used for solving the integer program and contains the results for the defined scenarios. Furthermore, this chapter describes the effects of optimizing the chain conveyor with regard to the operative transfer and the number of necessary operators at the diverters. In Chapter 9, we define and discuss new operation strategies regarding the storage of carriers on the chain conveyor, and give recommendations for the implementation of these strategies. Finally, the fourth part contains two chapters. All research questions are answered in Chapter 10 and Chapter 11 gives recommendations for future research. In addition, the last part contains three appendices. Appendix A contains the results of the MVA based algorithm proposed in Chapter 5, Appendix B shows the data used in Chapter 6, and Appendix C gives a pseudo code of the simulation described in Chapter 7.

Part I

Problem definition and research approach

CHAPTER 2

Problem description for the chain conveyor

We start our research in this chapter by investigating the current situation regarding the chain conveyor. Causes of both manual and automatic stagnation are described.

Section 2.1 contains a description of the chain conveyor and its purposes. In Section 2.2 we describe the processes concerning the handling of carriers in the conveyor system. In Section 2.3 we describe the problems Hollander copes with, regarding the chain conveyor. We propose an integer program to optimize the number of carriers stored on the chain in Section 2.4. Finally, Section 2.5 gives a summary of all drawn conclusions in this chapter.

2.1 Description of the chain conveyer

The chain conveyor is an iron in-floor chain containing a fixed number of holes equally spaced over its length. The track of the conveyor system contains two closed loops, as shown in Figure 1.1. A carrier can be coupled onto the chain using an iron pin that falls through the carriers into one of the holes, see Figure 2.1a. The chain is set in motion by two motors. If the chain conveyor is running, it follows the same constant speed. We can summarize the purposes of the chain conveyor as follows.

i) *Supply* the store regions with empty carriers.

ii) *Transport* the products stored on carriers towards the expedition region.

iii) *Sort* the loaded carriers at the expedition region.

iv) *Store* both empty and loaded carriers.

At the store and pick regions of the DC, orders are prepared for delivery to the customers of Hollander. These orders are picked and loaded onto carriers, see Figure 2.1b. The loaded carriers are transported towards the expedition region by the chain conveyor, where they are decoupled from the system. The chain conveyor can be divided into several main parts. These different parts are shown in Figure 1.1. Below, the processes concerning each part of the chain conveyor are described.

(a) System to couple and decouple carriers onto the chain.

(b) Loaded carrier coupled onto the chain conveyor.

(c) One of the diverters.

(d) Pushed carrier.

Figure 2.1: Pictures of the chain conveyor system.

Fill place Empty carriers are stored at a storehouse near the DC. During operation hours, these carriers are transported by trucks towards the first floor of the DC. These carriers arrive at the fill place and are manually coupled onto the chain conveyor. Furthermore, loaded carriers are removed from the chain. We will go into more detail about the latter in Section 2.3.

Ramp After a carrier is coupled onto the chain conveyor, it starts its round through the DC. Firstly, the track runs over the ramp. This ramp is located in the DC in order to transport empty carriers from the first floor upwards to the second floor. Both loaded carriers and empty carriers are transported downwards over the ramp. Safety checks are located both at the bottom and at the top of the ramp. These safety checks are installed to detect *pushed carriers*, which are explained in Section 2.3. If a pushed carrier is detected, the chain conveyor stagnates automatically. After manually removing the pushed carrier, the conveyor system is switched back on manually.

Pick lane As described in the previous section, the DC contains different store regions. These regions are located alongside the chain conveyor. Define the pick lane of a store region as the part of the chain conveyor that runs through that store region.

Diverter A diverter is an exit for loaded carriers. Here, they can leave the conveyor system. Each diverter can store a finite number of carriers. There are four diverters at the expedition region. As described above, all loaded carriers are assigned to exactly one diverter. This assignment depends on the delivery route. Each customer is coupled to one or more delivery routes, and each delivery route is coupled to solely one diverter. Right before each diverter, carriers are scanned automatically. Only if the scanned carrier is assigned to the belonging diverter, the carrier is steered onto the diverter by a mechanic switch, see Figure 2.1c. At the end of the diverter, carriers are removed from the system, scanned and set up for transport by the operators. Since the diverters have limited capacity, it is possible that a diverter is full at the moment that an assigned carrier arrives. In that situation, the loaded carrier passes this diverter and starts a new cycle through the DC. Note that if a loaded carrier misses its exit, it does not exit the system at another diverter either. Since the conveyor system is a closed loop, the carrier starts a new round through the DC. Empty carriers always pass all diverters and start a new round through the DC. All diverters have a safety check, analogue to the ones described for the ramp above.

Transfer The transfer of the conveyor system is located after the fill location, right before the ramp, see Figure 1.1a. With this transfer, it is possible for loaded carriers to skip the track around the second floor of the DC. The capacity of the transfer is limited. Loaded carriers that passed all diverters, enter the transfer if free space is available. These carriers can be inserted onto the main track, right after the place where carriers arrive at the first floor, down from the ramp. If a free hole passes by, the carrier is coupled back onto the track. If the transfer is full, arriving loaded carriers pass the transfer, analogue to what happens at the diverters. The latter is not allowed.

Carriers Empty carriers are brought onto the chain at the fill place. They can be replaced by loaded carriers along the pick lanes. Loaded carriers are removed from the chain at both the diverters and the fill place. For each removed loaded carrier, an empty carrier is brought onto the chain at the fill place. Hence, we can assume that the number of carriers in the conveyor system is constant. The handling of carriers in the conveyor system is described in more detail in the next section.

2.2 Process description for the carriers in the conveyor system

The chain conveyor transports carriers through the DC along all store regions, diverters and the transfer. The store regions are located right after each other, succeeded by the diverters. The transfer is located after the diverters, right before the first store region. A carrier can be either assigned to one of the diverters, or not. During each cycle along the chain conveyor, a carrier is loaded in at most one of the store regions. Hence, it is also possible that a carrier has traveled through the whole DC and has not been loaded.

Store regions A team of order pickers is assigned to store region. Solely unassigned carriers traveling over the pick lane can receive service at the store regions. An order picker starts his order by decoupling an empty carrier from the chain conveyor. By scanning the carrier, an order is assigned to the carrier. Furthermore, each order is assigned to exactly one destination, namely one of the diverters. Right after loading an order onto the carrier, the carrier is coupled back onto the chain conveyor. The order picker continues with the next order by decoupling the next empty carrier.

Diverters The diverters solely have to cope with carriers that are assigned to that diverter. If a carrier arrives at its assigned diverter and the diverter is not full, it automatically leaves the loop and is stored at the diverter. Each diverter has its own label printer that is located at the end of the diverter. During service of a carrier, the carrier is manually removed from the diverter and brought to the label printer. There it is scanned, whereby a label is printed. The label is removed from the printer and attached to the carriers. Finally, the carrier is set up for transport. For each carrier that is set up for transport, an unassigned carrier is coupled onto the chain, right after the last diverter. Due to the limited capacity of the diverters, not all carriers are able to enter a diverter. Instead, they travel along and start a new cycle through the DC.

Transfer Since assigned carriers cannot receive service at a store region, it is convenient to let such carrier skip a number of these regions. The transfer creates a short cut for assigned carriers, at a location where two parts of the chain lie close to each other. If an assigned carrier arrives at the transfer and the transfer is not full, then it leaves the loop and is stored at the transfer. The carriers are stored in series, until they can be coupled back onto the chain. Note that carriers can only be coupled back onto the chain if a free hole passes by.

The following section of this chapter describes the operational strategies implemented by the management of Hollander, and the problems Hollander has to cope with.

2.3 Problems Hollander copes with in the current situation

In the current situation, Hollander copes with many problems with regard to the continuity of the chain conveyor. We investigated these problems by interviewing several employees of Hollander, and observing the processes in the DC. Thereby, we will formulate all implemented strategies and noticed problems concerning the continuity of the chain conveyor. We conclude with describing the correlations with regard to these problems and strategies, in order to decide on which problems we will focus in this study.

Full-load-strategy The chain conveyor system is stored with empty carriers up to its maximal capacity. In other words, an empty carrier is coupled at each hole of the chain, so that no free holes 'leave' the fill place. This strategy is referred to as the *full-load-strategy*, see Figure 9.1 at page 96. Using this strategy, two of the four purposes of the chain conveyor are optimized: *supply* and *store*. The former induces that the number of idle order pickers is minimized, namely they are supplied with the the highest possible number of empty carriers.

Switching loaded and empty carriers at the pick lanes A consequence of storing the chain conveyor with maximal capacity is that almost no free holes will pass the pick lanes (or transfer). However, this does not affect the *transport* purpose of the chain conveyor. If a free hole is necessary, one can decouple an empty carrier from the chain. Hence, order pickers firstly decouple an empty carrier from the chain conveyor, before they couple the newly loaded carrier onto the chain. This empty carrier can be used directly to pick the next order.

Transfer not in operation Another consequence of the full-load-strategy is that the management team of Hollander decided to put the transfer out of operation. Once it was operating, the throughput time of the transfer was too high, which caused carriers to arrive too late at the expedition region. The cause of the high throughput time, was that a carrier could not be coupled back on the chain of the conveyor, since no free hole was available. That situation got worse, since the transfer has a limited capacity and other loaded carriers will pass by. These carriers are transported towards the ramp. Loaded carriers are probably to heavy to be transported up the ramp. Furthermore, by traveling the whole track of the chain again, they will arrive too late at the expedition region.

Second cycle not allowed A second cycle through the DC using the transfer has a duration of about 27 minutes, ignoring stagnation. In this study we assume that this extra transport time is allowed for loaded carriers. A second cycle when the transfer is not in operation includes the track on the second floor. Ignoring stagnation, such a cycle has a duration of about 55 minutes. Including stagnation, this is estimated to be over 1 hour. This extra transport time

is not allowed for loaded carriers. Since the transfer is not operating in the current situation, second cycles are not allowed.

Staffing at the diverters Since the transfer is not in operation, one wants to prevent that the diverters are stored with carriers up to the maximal capacity. In that case, assigned carriers miss their exit and start a new cycle in the conveyor system. Therefore, Hollander decided to install enough operators at each diverter, to prevent the diverter from becoming full. This results into the following situation. Just a small part of the total carrier capacity of the diverters is in use. In that case, the carriers are stored 'far away' from the label printer at the diverter. Operators have to walk along the diverter to get the carrier and walk back with the (most of the time very heavy) carrier, to bring it to the label printer. A better situation would be if the diverter is almost full and operators can decouple the carriers from the end of the diverter. In that case, the walking distance is almost halved.

Decoupling loaded carriers at the fill place Another process used to prevent carriers from starting a second cycle through the DC is the following. Each loaded carrier that arrives at the fill place is decoupled by one of the operators that is storing empty carriers on the conveyor system. Thereafter, such carrier is scanned at the nearest diverter (the fourth diverter) so that a label is printed and the operator knows to which delivery route the carrier is assigned. Then, the carrier is manually brought towards its corresponding diverter and set up for transport.

Stagnation Stagnation of the chain conveyor has a drastic effect on three of the four main purposes of the chain conveyor, namely the *supply*, *transport* and *sort* purposes. Furthermore, when the conveyor is put back in operation, all carriers are set in motion. Since loaded carriers can weigh over 300 kg each, this is very energy consuming. The chain conveyor stagnates many times each day, namely up to 2 or 3 times each operating hour. The duration of each stagnation differs between several seconds and about two minutes. The mean total stagnation duration is estimated by the management at 30 minutes per day. The stagnations have different causes. The chain conveyor is stagnated either manually by an operator, or automatically by the system. In the latter case, it is not always possible to detect the exact cause of stagnation. Each time the conveyor stagnates with due to a unknown cause, all operators perform a check along the whole track of the conveyor. When the cause is found, the conveyor is switched back on right after the problem is solved. Otherwise, in the case that nobody detects a possible cause of stagnation, the conveyor is switched back on again as well. Hence, in some cases the cause of stagnation remains undefined.

Pushed carrier A pushed carrier is a carrier that is pushed forward by another carrier instead of being pulled by the chain of the conveyor system, see Figure 2.1d. The carrier on the left hand side pushes the carrier on the right hand side. Remember that a carrier is coupled onto the chain using an iron pin that falls through the carriers into one of the holes on the chain. If the pin falls on the rails, but not into a hole, it can cause a pushed carrier. In the case that the next hole in the chain is free, the pin will fall into that hole and the carrier will safely

continue its journey. Otherwise, the carrier that was not properly attached to the conveyor, will be pushed forward by the carrier in the next hole. The latter is called a pushed carrier. Note that the number of pushed carriers could be reduced by decreasing the number of carriers loaded onto the chain, since the probability that a pushed carrier occurs increases as the total number of carriers in the conveyor system increases and vice versa. Most pushed carriers are caused by mechanical disruptions. For example, in overtaking at the transfer (if in operation) or near the motors of the chain conveyor. In exceptional cases, pushed containers are caused by supine order pickers. A pushed carrier continues along the track until it reaches either the ramp, or one of the diverters. There it is detected by the safety check described in the previous section, so that the conveyor system stagnates. After detecting the cause of that stagnation, the pushed carrier is manually coupled to a free hole, and the conveyor system is put back into operation.

Overwork at the fourth diverter The work pressure at the fourth diverter is too much to handle. This is caused by the extra processes at the fourth diverter, described in paragraph 'decoupling loaded carriers at the fill place'. If the pressure becomes too much, the conveyor system is manually stagnated and put back on after all overwork is eliminated. There are multiple reasons why many loaded carriers arrive at the fill place. Firstly, carriers can miss their assigned diverter due to the limited capacity of the diverter. Secondly, it can occur that one of the diverters is not working properly, due to a mechanical disruption of the switch. In both of these cases, the chain conveyor does not stagnate. The assigned carrier that had to exit the disrupted diverter does not exit, but passes and arrives at the fill place. Furthermore, the operators at the fourth diverter have to cope with so called loaded *ghost carriers*, which are defined below. Note that the overwork at the fourth diverter is a consequence of the full-load-strategy. Due to the full-load-strategy, the transfer is not in operation in the current situation at Hollander. Therefore, carriers are not allowed to travel second cycle through the distribution center. This causes overwork at the fill place, since loaded carriers are decoupled instead of ignored. Since these decoupled carriers are handled at the fourth diverter, the overwork at the fourth diverter is caused by the full-load-strategy.

Ghost carrier A ghost carrier is a loaded carrier without a destination, or an empty carrier with a destination. In the first case, the system does not detect that the ghost carrier has to exit at one of the diverters. Hence, the order loaded on the ghost carrier never reaches the expedition area automatically. In the latter case, an order is assigned to an empty carrier on the chain conveyor. Hence, this order will never be picked and thus never reaches the expedition area automatically. Currently, Hollander does not have a clear problem approach to tackle causes of ghost carriers.

Rubbish in the chain Stagnation of the chain conveyor can have a very simple cause, namely rubbish in the chain. Rubbish, like used packaging, lies around the DC and gets in the chain of the conveyor system which causes stagnation. We choose to ignore this cause of stagnation during our research.

Staffing at the fill place The number of operators working at the fill place is probably higher than necessary, since decoupling loaded carriers at the fill place is very time consuming. If the loaded carrier that arrives at the fill place is assigned to the first diverter, it has to travel about 150 meters to its destination. Hence, the operator that is bringing the carrier has to travel about 300 meters per carrier (back and forth). Assuming that an operator walks 6 km/h, this takes 3 minutes per carrier. Even worse, the floor manager at Hollander estimates this (back and forth) walking time at 5 minutes per carrier on average.

Automatic labeling Nowadays, Hollander is investigating whether it is interesting to implement automatic labeling. At the moment, an operator at the diverter has to scan the carrier manually, get the printed label from the label printer and attach this label to the carrier. After that, the carrier is set up for transport. After implementing automatic labeling, a label machine will be located along the chain conveyor between the last store and order pick region and the expedition region. Hence, after implementation, the manual processing time at each diverter will be reduced.

Summarizing, there are three main reasons that cause stagnation of the chain conveyor, namely pushed carriers, overwork at the fourth diverter, and rubbish in the chain. This study is focused on the former two causes and ignores the latter cause. A flow chart showing the correlations between all strategies and problems described in this section is presented in Figure 2.2. Note that some loaded carriers do not exit at the diverters (and thus arrive at the fill

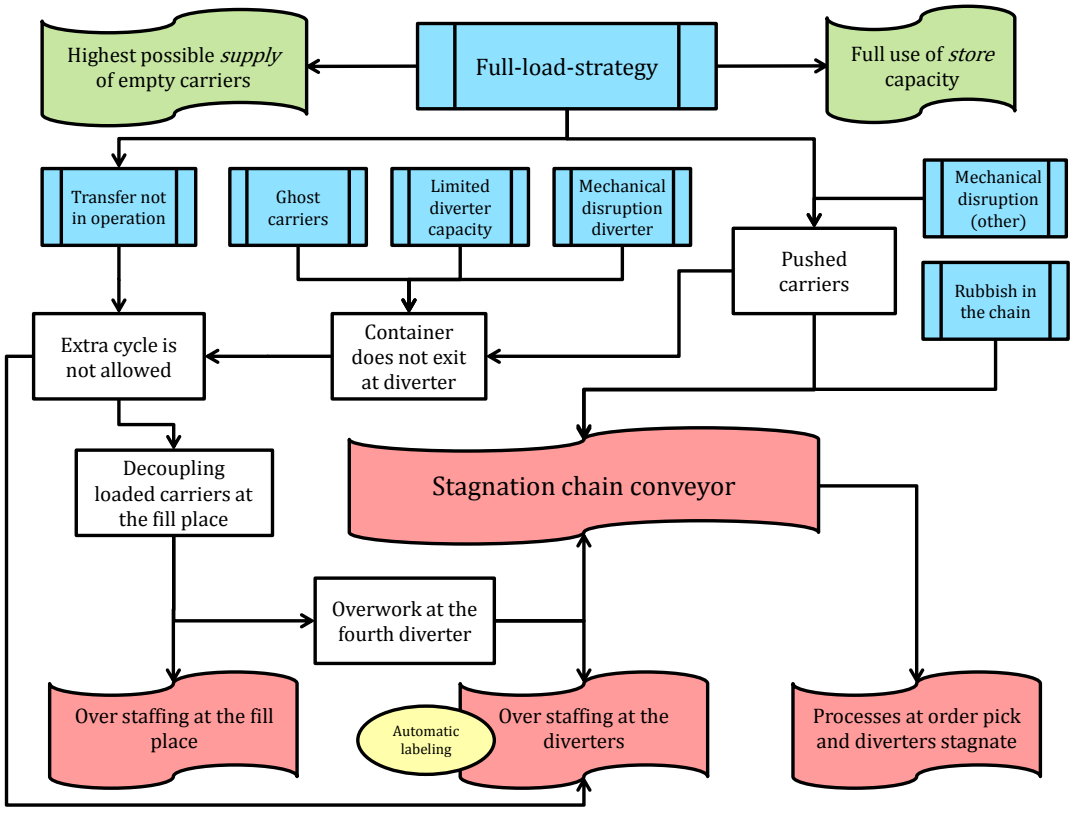


Figure 2.2: Causes of stagnation and consquences of the full-load-strategy.

location). This is caused by four different issues, namely ghost carriers, mechanical disruption, the limited capacity of the diverters and pushed carriers. This study will focus on the latter two causes and will ignore the former two causes. If the number of carriers in the system is reduced, we expect that the transfer can be put in operation. If the transfer is put in operation, then the limited diverter capacity is not a problem any more. Therefore, we choose to investigate the chain conveyor with the transfer in operation. Furthermore, we suspect that the number of pushed carriers will be reduced by reducing the number of carriers in the system. Hence, we will investigate whether the full-load-strategy can be adjusted. The next section describes an integer program, in order to reduce the total number of carriers loaded at the chain conveyor.

2.4 Formulation of the integer program for optimizing the number of carriers

We stated that if the total number of carriers in the conveyor system is reduced, then stagnation of the chain conveyor will be reduced as well. On the other hand, the chain conveyor has to supply the store regions with enough empty carriers to prevent idle time of the order pickers. Besides, we suspect that the number of operators at both the diverters and the fill place can be reduced if the transfer is put in operation. In this section, we give the mathematical formulation of the research approach of this study.

The optimization of the total number of carriers in the conveyor system can be formulated as an integer program. There is a trade-off between minimizing the total number of carriers loaded on the chain conveyor and minimizing the idle time of the order pickers. In words, the optimization problem can be formulated as follows. *'Minimize the total number of carriers in the conveyor system with the transfer in operation.' Subject to: 'The supply of empty carriers for the order pickers remains enough.'* The mathematical formulation of this problem is given by

$$\begin{aligned} &\textit{Minimize} && K \\ &\textit{Subject to} && \bar{k}_i(K) \geq M \cdot m_i, && \forall\, i \in \mathbf{N}_l, \\ & && K \in \{0, 1, \ldots, C\}, \end{aligned} \tag{2.1}$$

where K equals the number of carriers in the conveyor system, $\mathbf{N}_l$ is the set of store regions in the DC, $\bar{k}_i(K)$ is the mean number of busy order pickers at store region i with K carriers in the conveyor system, m_i is the number of order pickers at store region i, and C is the maximum number of carriers that the conveyor system can handle. The parameter M will be in the interval $[0, 1]$ and indicates the percentage of required mean number of busy order pickers per time unit. We assume that M equals 0.95. Hence, for the number of carriers in the conveyor system is demanded that at least 95% of the order pickers are supplied with enough empty carriers. Note that this is demanded per load region. Hence, the total number of busy order pickers will be higher than (or in worst case equal to) 95%. We will solve this integer program using a binary search.

2.5 Conclusions

The chain conveyor has four main purposes: *supplying* the store regions with empty carriers, *transporting* picked orders towards the expedition region, *sorting* the loaded carriers by delivery route, and *storing* both empty and loaded carriers. Hollander is investigating whether it is possible to implement automatic labeling. Currently, the chain conveyor does not work properly, since it stagnates many times each day. Furthermore, the transfer is not in operation. There are three main reasons that cause stagnation of the chain conveyor, namely pushed carriers, overwork at the fourth diverter, and rubbish in the chain. This study is focused on the former two causes and ignores the latter cause. Overwork at the fourth diverter is caused by four different issues, namely ghost carriers, mechanical disruptions, the limited capacity of the diverters, and pushed carriers. This study is focused on the latter two causes and ignores the former two causes. We stated that stagnation will be reduced if the currently implemented fill strategy is adapted, since we expect that in that case both of pushed carriers and overload at the fourth diverter is reduced. We will model the transfer being in operation, and thereby suspect that the number of operators at the diverters can be reduced. We formulated an integer program in order to optimize the total number of carriers in the conveyor system, and assumed that for each store region at least 95% of the order pickers should be supplied with enough empty carriers. The integer program will be solved using a binary search. In the next chapter we give a literature review with regard to modeling the chain conveyor at Hollander.

CHAPTER 3

Literature review

Many diverse research approaches concerning the problems arising with the modeling of conveyor systems are described in literature. There are several approaches in the literature to model conveyor systems. This literature study is focused on stochastic modeling using queueing networks.

An overview of potential modeling methods concerning conveyor systems using stochastic modeling is given in Section 3.1. These methods are linked to the chain conveyor. Section 3.2 contains a literature study concerning Mean Value Analysis algorithms for analyzing closed queueing network models. For readers who are not familiar with queueing theory, an introduction will be given in the next chapter. Finally, an overview of the literature used in this study is given in Section 3.3.

3.1 Potential modeling methods

In order to get familiar with the problem environment, several topics should be studied. An introduction to warehousing could turn out to be very useful, since the problem environment at Hollander is a warehouse. A good overview of warehouse and distribution science is given by Bartholdi and Hackman. In their book, Bartholdi and Hackman [2011] discuss several elements of warehousing like equipments, processes, the layout of a warehouse and order-picking processes. An introduction to stochastic models and in particular queueing theory is given by Tijms [2003]. He discusses discrete and continues time Markov chains and several different types of queues. A book more extensive about queueing networks is written by Bolch et al. [2006]. A more theoretical approach is given by Wolff [1989], who does not discuss networks in detail. The remainder of this section gives an overview of published literature concerning stochastic modeling of conveyor systems using queueing networks. All topics concerning queueing theory used in this study are covered in the next chapter of this thesis.

Osorio and Bierlaire [2009] design an open queueing network where jobs are allowed to leave the network and where the external arrivals arise from an infinite population of jobs. The model

is applied to a patient flow network in a hospital, but could be used for a conveyor system as well. Jobs that enter the network at the first queue, are either served or they queue until a server becomes available. After service, the job goes to its next queue, called the target queue. If this target queue is full, then the job is blocked at its current location. Once there is a free place in the target queue, the blocked job proceeds to the target queue. The state of a queue at any point in time is described by three elements: the number of jobs in service, the number of blocked jobs and the number of waiting jobs. The paper could be applicable to this study. Carriers could be modeled as jobs. Furthermore, in this study, there are at least two types of queues. Firstly the pick lanes, these are the queues at which carriers arrive at the queueing network. Jobs are either served or pass the queue. They never queue and are never blocked. Secondly, the queues modeling the diverters. Jobs enter the queue if there is space at the diverter and are blocked otherwise. The difference between the situation at Hollander and the model described by Osorio and Bierlaire, is that blocked jobs at the chain conveyor do not stay at their current location. Instead they start a new cycle through the network. The model of Osorio and Bierlaire supports recirculation, blocking of units, and state-dependent arrival and service rates. The difference with the situation at Hollander is that blocking at a so-called server queue at Hollander is not interesting, but blocking at an exit station is. The study of Osorio and Bierlaire relies on exponential travel times and processing times. However, they expect that their model could also be used for a rough estimation of the performance of systems with other distributions. In the article they discuss a conveyor with two severs, but the model can be easily extended. Each server is modeled as a subsystem consisting one server, one input buffer and one output buffer.

Schmidt and Jackman [2000] present a model that includes recirculation through the system. In their research they model the conveyor system as an open queueing network. Hence, the number of jobs in the system is not constant. Jobs follow a fixed path. Along the conveyor, several servers are present. These servers can represent machines, assembly stations, or possibly multiple stations. A job leaves the conveyor at the first available server. After service, jobs return to the conveyor and proceed to the exit point, where they leave the system. At Hollander, we can model the carriers as jobs and the pick lanes as servers. Jobs leave the system at the first available server. After service (the order picking), they return to the chain conveyor in order to exit the system at one of the assigned diverters. Schmidt and Jackman use several measures to analyze the performance of the conveyor system. The server performances are measured by the steady-state probabilities. These measures include the time the server spends in service and the fraction of time a server is blocked. The stationary probabilities are used to determine the load (job) distribution between servers and the number of recirculations. Furthermore, average number of circuits per job and the system throughput in jobs per time job are measured. Especially the fraction of time a server is blocked and the number of recirculations could be interesting for Hollander.

Bastani [1988] discusses a closed-loop conveyor system having one single loading station and multiple server stations. This model could be useful for Hollander, if we interpret jobs as carriers and servers as pick lanes. Jobs arrive at the loading station and get service at the first available

server. If all servers are busy, then jobs recirculate. The servers have deterministic service times and do never break down. If a job is blocked at the loading station due to a recirculating job, then it is lost. Bastani did not model a queue at the the loading station nor at the servers. This is the case at the store regions of Hollander as well. The difference is that at Hollander, the served jobs enter the conveyor again and are transported towards the exit servers. Secondly, at Hollander, the carriers leave the conveyor system which is not the case at Bastani's study.

Sonderman [1982] describes another model with circulation. He modeled a conveyor system as a closed-loop queueing system with stochastic in- and output. The main objective of his research is to analyze the effect of recirculating loads on the overall behavior of the system. Jobs enter the system at an input station, according to a Poisson process. Then, they are transported by the conveyor and removed from the system at one or multiple stations. Service times at the server stations are exponentially distributed. If the queue of a station is full, jobs will not be removed and thus recirculate back to the input station. Here, they merge with jobs just entering the system. To analyze the model, Sonderman defines three different epoch measures: both potential and actual number of arrived jobs at a server station, and the number of jobs that leave the system. After analyzing and approximating the model, Sonderman proposes a simulation and algorithm comparison. The basic schedule of this model is a perfect fit to model the diverters at Hollander, since carriers arrive at the loading station, and exit a diverter if the queue is not full. In case the queue is full, the carrier merges with new carriers at the arrival station. The difference is that inter-arrival times are not exponential, but deterministic. Furthermore, the model of Sonderman does not take several states of a job into account, like a loaded or empty carrier at Hollander.

Gregory and Litton [1975] describe a discrete model that is very different from the models described above. They introduce a model of a conveyor system with general, bounded service distributions. There are no queues at the service stations and the model does not support recirculation of jobs. The conveyor system transports jobs on a fixed number of individual hooks that are equally spaced along the conveyor and running at a uniform speed. Along the conveyor, multiple ordered servers are located. Jobs are served at the first free server. If no server is available, the job is lost. The similarity with the situation at Hollander is that the conveyor belt consists of hooks that are equally distributed along the conveyor. In this model, the jobs do not represent carriers, but holes in the chain of the conveyor. This approach can be used at Hollander, when different states are assigned to each hole, namely no carrier attached, an empty carrier attached, or a loaded carrier attached with assigned diverter one, two, three or four. For the pick lanes, we probably have to implement recirculation and state changes into the model.

Reiser and Lavenberg [1980] introduced the Mean Value Analysis (MVA) algorithm for closed multi-class queueing networks with a product form solution. They assumed the network consists of closed routing chains where each chain has a fixed number of jobs. Each chain corresponds to exactly one job class and jobs cannot change class. Furthermore, they assume all node capacities

to be infinite. The nodes of the network follow one of the disciplines FCFS, Parallel Server (PS), Last-Come-First-Served with pre-emption (LCFSPR) or an infinite number of servers (D). For the FCFS nodes they assume that all service times are exponentially distributed, allowing queue size dependent service rates. This innovative algorithm works directly with the desired statistics such as the mean number of jobs per node, the mean sojourn times for each job class in each node and the throughputs per job class of the network. The algorithm is innovative, since previous methods such as the convolution algorithm were limited by the computational time. The MVA algorithm is an iterative algorithm where the number of jobs in the network grows with each iteration. Reiser and Lavenberg [1980] proved a relation between the mean sojourn time of a closed queueing network and the mean sojourn time of this same network with one job less. With each iteration they first compute the mean sojourn time for all nodes using this relation and then compute the throughput and mean number of jobs for all nodes using the mean sojourn times. We remarked before that the MVA is designed for solely queueing networks where jobs cannot change class. However, Reiser and Lavenberg [1980, pp. 317] remark that a model with class changes can be mapped trivially into a multi-class model without class changes. This is described by Reiser and Kobayashi [1975].

If we model that an assigned carrier becomes unassigned during service at one of the diverters and enters the loop again after service, right after the diverter where it got served, then we can assume that the total number of carriers in the conveyor system is constant. Therefore, we choose to model the chain conveyor as a closed queueing network. Bolch et al. [2006] describes the Mean Value Analysis algorithm (MVA) amongst others, for analyzing closed queueing networks, and stated that the computation time of the MVA algorithm is very low. Furthermore, there are many studies described in literature concerning extensions of the MVA algorithm. We choose to investigate which extensions of the MVA algorithm could be used for analyzing a closed queueing network which models the chain conveyor. This investigation is described in the next section of this chapter.

3.2 Mean value analysis for closed queueing networks

The MVA algorithm is proposed by Reiser and Lavenberg [1980] for analyzing closed multi-class queueing networks consisting of solely infinite, FCFS disciplined, exponential distributed nodes, allowing multiple server nodes. Thereby, class changes are not allowed. However, they remark that a model with class changes can be mapped trivially into a multi-class model without class changes. This is described by Reiser and Kobayashi [1975]. This section contains a review of studies described in literature concerning extensions of the MVA algorithm, in order to propose a closed queueing network that models the chain conveyor.

Transfer blocking The MVA algorithm [Reiser and Lavenberg, 1980] does not apply for queueing networks with finite capacity nodes. However, this algorithm has been modified by Akyildiz [1988] in order to analyze queueing networks with finite capacity nodes with transfer blocking, which can be explained as follows. If a job wants to jump to a finite capacity node

that is full, it stays in its current server and blocks this server until space becomes free in the finite capacity node. This modified MVA algorithm is called the MVABLO algorithm. The MVABLO algorithm accounts for analyzing single-class, closed queueing networks with solely FCFS disciplined nodes, having finite capacities with transfer blocking, exponentially distributed service times and a single server. However, Akyildiz remarks that the MVABLO can be extended to multiple server nodes as well. The MVABLO algorithm approximates the mean number of jobs in each node, the throughput, and the mean sojourn time of each node. Right after each iteration, it is checked for all nodes if the mean number of jobs does exceed the node capacity. If this is the case for at least one node, the mean sojourn times are adjusted. The new values are used to recompute the throughputs and the mean number of jobs. The MVABLO algorithm is tested for about 150 queueing networks with various number nodes, number of jobs, and network topologies. The results were compared with results gained by simulation. The deviation of the algorithm from the simulation is about 10 percent. These deviations are caused by the modification of the sojourn time when the station capacity is minimally exceeded. In such case, the algorithm introduces large deviations for the mean sojourn times. Furthermore, the cases in which the MVABLO algorithm validates whether the queue capacity is violated, does not always detect blocking. The major advantage of the MVABLO is that its computation time is extremely fast.

Balsamo and Rainero [2000] proposed several approximate analytical methods for closed queueing networks with finite capacity queues. They designed an MVA algorithm for networks with a general topology and $G/M/1/c$ service centers. This algorithm is called the Approximate MVA algorithm (MVA-A). In the MVA-A the relation of the equilibrium mean sojourn times are modified to take transfer blocking into account. The efficiency of the MVA-A is very good, but unfortunately the accuracies of the mean response time and mean number of jobs are poor. The accuracy of the throughput seems fair. Balsamo and Rainero [2000] do not compare the performances of the MVA-A with the MVABLO proposed by Akyildiz [1988].

Recirculation blocking Zhuang and Hindi [1990] model a flexible manufacturing system using a multiple class queueing network with a central service center configuration and finite capacity queues. They extended the MVA algorithm [Reiser and Lavenberg, 1980] to analyze a network with a block and recirculate mechanism. Blocked jobs return towards the central service center that routes the jobs according to a probability routing scheme. All service centers have exponentially distributed service times. Based on several case studies they showed that the results obtained by the proposed algorithm are of good accuracy, compared to results obtained from simulation.

Priority nodes Bryant and Krzesinski [1984] proposed an MVA-based algorithm to compute the mean sojourn time for each node, the throughput and the mean number of jobs for each node for open, closed and mixed queueing networks containing priority nodes with either pre-emptive or non-pre-emptive properties. The nodes follow a FCFS, PS or a infinite server (IS) discipline. The nodes that follow a FCFS discipline are single server nodes. The solutions obtained from the MVA-based algorithm were compared with simulation results. For closed

networks the largest errors were found in the waiting time estimates. However, he concluded that the errors of a MVA-based algorithm are within an acceptable limit for the mean number of jobs. The estimates for the highest priority classes were very accurate, and for the other classes they were within an acceptable limit.

Class dependent service times Schmidt [1997] proposed an approximative, MVA-based algorithm to model queueing networks with class dependent service times. The model supports multiple-server, FCFS disciplined nodes. The departure rates depend on the number of customers of all classes in the queue. It is assumed that class changes of customers are not allowed. The computation time remains very fast, like with the original MVA algorithm. Unfortunately, Rainer was not able to analyze networks with more than 30 customers in eight classes and class dependent service time distributions.

In the previous section we chose to model the chain conveyor as a closed queueing network. Using the MVA-based approximations described in this section, we can analyze closed queueing networks allowing class changes, transfer blocking, recirculation blocking, priorities, or class dependent service times. Therefore, we choose to investigate whether we are able to propose an MVA-based algorithm that can be used to analyze the to be proposed closed queueing network that models the chain conveyor.

3.3 Conclusions

In this chapter, a literature review on potential methods for modeling conveyor systems using queueing network models is performed. Based on this literature review, we chose to model the chain conveyor as a closed queueing network. Furthermore, we chose to investigate whether we are able to propose an MVA-based algorithm that can be used to analyze the to be proposed model. Thereby, we can use a mapping that allows class changes [Reiser and Kobayashi, 1975], and MVA-based approximations for allowing finite capacity nodes with either transfer blocking [Akyildiz, 1988; Balsamo and Rainero, 2000], or recirculation blocking [Zhuang and Hindi, 1990]. Furthermore, we can incorporate priority nodes using [Bryant and Krzesinski, 1984], and class dependent service time distributions using [Schmidt, 1997]. We will propose a closed queueing network model for the chain conveyor at Hollander in the next chapter.

Part II

Stochastic modeling

CHAPTER 4

Modeling the chain conveyor as a closed queueing network

We chose to model the chain conveyor as a closed queueing network with possibly transfer and recirculation blocking, priorities, or class dependent service times. The model proposed in this chapter will be used to determine the mean number of busy order pickers at the store regions, depending on the total number of carriers in the conveyor system. Thereby, the integer program given in (2.1) can be solved.

Section 4.1 contains an introduction on queueing theory for the readers who are not familiar with this subject, and describes all used definitions and terminology used for the proposed model. The queueing network that models the chain conveyor is proposed in Section 4.2. We give a summary the model in Section 4.3.

4.1 Definitions and terminology

Closed queueing networks are defined by a set of service centers (nodes) with a specified service time distribution and discipline for each node, a set of job classes, the network topology defined by the routing probabilities, and the population vector. In Section 4.1.1 service centers are described in detail, in order to describe closed queueing networks in Section 4.1.2. Sections 4.1.3 and 4.1.4 describe networks with finite capacity nodes (and thus with blocking properties) or priorities, respectively.

4.1.1 Service centers

A service center consists of a number of identical servers and one finite or infinite queue. Jobs arrive at the queue and enter one of the servers if available. After service, the jobs depart from the service center, see Figure 4.1. The mean number of jobs that arrive in the service center per time unit is given by arrival rate λ. The mean number of jobs that can be served per time unit is given by service rate μ. For example, think about the the diverters in the DC, where carriers arrive and have to wait in a queue until they receive service from one of the operators. Service of a carrier at a diverter includes decoupling the carrier from the conveyor

and set it up for transport. The number of servers m equals the number of operators working at the diverter. Generally, we can define a service center by four different entries using Kendall's notation $A/B/m/c$. The first entry A specifies the arrival process of the jobs that enter the center. The second entry B gives the service time distribution of the servers. Furthermore, entries m and c represent the number of servers and the job capacity of the center, respectively. The value of c includes the number of servers m. Jobs can only enter the center if the queue is not fully occupied. If m (and thus c as well) are infinite, one writes $A/B/\infty$. In that case all arriving jobs receive service immediately. If solely c is infinite, one writes $A/B/m/\infty$.

The arrival process of jobs in a service center can be defined as a counting process. For example, we can count the number of carriers that arrive at a diverter in the DC. The definition of a counting process can be found in [Bolch et al., 2006, pp. 52]. If the inter-arrival times between two arrivals are exponentially distributed, then we call this counting process a Poisson process.

Definition 4.1 (Poisson process) If the inter-arrival times of a counting process have a common exponential distribution with rate λ, then this process is called a *Poisson process with rate* λ. Hence, for all inter-arrival times X we have that

$$\mathbb{P}(X \leq x) = 1 - e^{-\lambda x}, \qquad \forall\, x \geq 0.$$

The exponential distribution has a very nice property, the memoryless property [Tijms, 2003, pp. 440]. For example, assume that the service time of a carrier in a diverter is exponentially distributed. For a carrier in service, the time until its service is ready has the same exponential distribution at any point during service. We "forget" for how long the carrier is already in service. The following theorem formulates the memoryless property of a Poisson process.

Theorem 4.2 (Memoryless property) Let a Poisson process with rate λ be given. Hence, all inter-arrival times are exponentially distributed with rate λ. For any time $t \geq 0$ we define the random variable γ_t as the waiting time from epoch t until the next arrival. For all $t \geq 0$ the waiting time γ_t satisfies

$$\mathbb{P}(\gamma_t \leq x) = 1 - e^{\lambda x}, \qquad \forall\, x \geq 0,$$

independent of the value of t. Hence, the random variable γ_t has the same exponential distribution as the inter-arrival times. This property is referred to as the *memoryless property*.

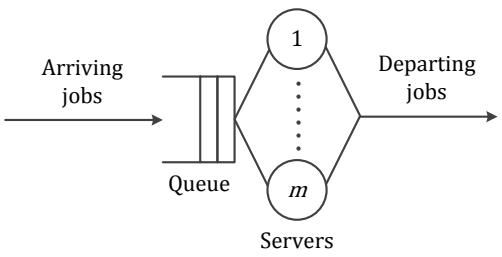


Figure 4.1: Service center with m identical servers.

For a service center with Poisson distributed arrivals, the A in Kendall's notation is substituted by M, referring to memoryless. Another arrival processes is the deterministic one. Here, the inter occurrence times of the arrival are known and constant over time. Deterministic arrival process are denoted by D. If the arrival process is stochastic but not Poisson, then it is denoted by the G of general distribution. Analogue to the arrival process, service times of a center can have a Poisson, deterministic, or general distribution. In that case, the B in Kendall's notation is substituted by M, D, or G, respectively.

Remark 4.3 Due to the memoryless property, it is very nice to work with service centers where customers arrive and get served according to a Poisson process.

Jobs can be categorized into different job classes. The set of job classes is denoted by $\mathbf{R}$. We use different job classes to model different types of jobs. For example, for the chain conveyor, the carriers will be modeled as jobs. The different job classes can correspond to empty carriers, or loaded carriers. In the next section we describe closed queueing networks with multiple job classes.

4.1.2 Closed queueing networks

We focus on queueing networks with multiple job classes, and where all service centers follow a FCFS queueing discipline. The total number of jobs in a closed queueing network is a constant variable, since jobs are not allowed to leave nor enter the network. This section focuses on networks with solely infinite capacity nodes having exponentially distributed service times.

A closed queueing network is defined by a set of nodes (service centers) $\mathbf{N}$, a set of job classes $\mathbf{R}$, the network topology defined by the routing probabilities defined below, and the population vector $\mathbf{K}$ defined below as well. The total number of jobs in the network is given by K. We will write N and R to indicate the number of nodes and the number of job classes in the network, respectively. Each node i is specified by its number of servers m_i and its service rate μ_i.

Definition 4.4 (Routing probability) Let a queueing network be given, with the set of nodes $\mathbf{N}$ and the set of job classes $\mathbf{R}$. For all nodes i, j in $\mathbf{N}$ and job classes r, s in $\mathbf{R}$, the *routing probability* p_{ir}^{js} gives the probability that a class-r job changes into a class-s job and jumps towards node j, given that it just completed its service in node i. Note that all routing probabilities are non-negative, and smaller than or equal to one. Furthermore, they satisfy the equations

$$\sum_{(j,s)\in\mathbf{N}\times\mathbf{R}} p_{ir}^{js} = 1, \qquad \forall\ (i,r) \in \mathbf{N}\times\mathbf{R}.$$

Definition 4.5 (Population vector) Let a closed queueing network with $\mathbf{R}$ the set of R job classes, and K the total number of jobs in the network be given. For all r in $\mathbf{R}$, define $\mathbf{K}_r$ as the number of class-r jobs in the network. The *population vector* of the network is given by

$$\mathbf{K} = (\mathbf{K}_1, \ldots, \mathbf{K}_R),$$

satisfying

$$\sum_{r \in \mathbf{R}} \mathbf{K}_r = K. \tag{4.1}$$

If class changes are not allowed, then the $\mathbf{K}_r$ remain constant. Otherwise, the values of the $\mathbf{K}_r$ vary over time. For single-class queueing networks, we will refer to the population vector as the *population number*.

In order to analyze queueing networks, different variables are used. These are the mean *number of class-r jobs* in a node, the mean *total number of jobs* in a node, the mean *waiting time* in a node, the mean *service time* of a node, the mean *sojourn time* of a node (the sojourn time equals the waiting time plus the service time), and the *throughput* of class-r jobs in the network. Furthermore, for finite capacity networks, the *capacity percentage* of a queueing network is defined in the next section.

4.1.3 Queueing networks with finite capacities

This section focuses on queueing networks with finite capacity nodes. For each finite capacity node i in a network, we write c_i to indicate its capacity.

Definition 4.6 (Network capacity) For closed queueing network with solely finite capacity nodes, let the value of C correspond to the sum of all node capacities. The value of C is referred to by the *network capacity* of the closed queueing network.

Definition 4.7 (Capacity constraint) Consider a closed queueing network with solely finite capacity nodes. The *capacity constraint* defined by

$$K \leq C$$

should be satisfied, such that the number of jobs in the network does not exceed the network capacity.

Definition 4.8 (Capacity percentage) For a closed queueing network with finite network capacity C and total number of jobs in the network K, we define the *capacity percentage* as $K/C \cdot 100\%$.

Consider a queueing network with at least one finite capacity node. If a job wants to jump to this node while it is full, then blocking occurs. We describe two types of blocking based on the literature review, namely transfer blocking and recirculation blocking. More types of blocking are defined in literature, such as blocking after service and rejection blocking.

Definition 4.9 (Transfer blocking) Let i, j in $\mathbf{N}$ be two arbitrary nodes such that node j has finite capacity. Assume that node j is full. If a job in node i completes its service and wants to jump to node j, it stays in the server of node i and blocks this server until capacity in node j becomes free. In the case that multiple servers are blocked due to the finite capacity of node j, these servers are unblocked using the discipline defined for node j. This type of blocking is referred to as *transfer blocking*. Other terms used in literature are blocking after service, type I blocking, production blocking, and non-immediate blocking.

Consider a closed queueing network containing a directed cycle of solely finite capacity nodes with transfer blocking. If the number of jobs in such network is more than or equal to the total capacity of such cycle, then it can occur that all nodes in that cycle are blocked. This is referred to as a *deadlock*. One way to cope with deadlocks, is to assume that in case of deadlock all jobs involved move simultaneously to their destinations. However, Perros and Altiok [1986] stated that this complicates the model, since deadlocks influence the marginal state probabilities. Therefore, we choose to assume that all networks, containing at least one cycle consisting of nodes with transfer blocking, adopt the deadlock property defined below.

Definition 4.10 (Deadlock property) Consider a closed queueing network containing a directed cycle of solely finite capacity nodes with transfer blocking. We say that such network adopts the *deadlock property*, if the number of jobs in the queueing network is less than the total capacity of the smallest directed cycle of solely finite capacity nodes with transfer blocking.

Definition 4.11 (Recirculation blocking, recirculation mapping) Let j in $\mathbf{N}$ be a finite capacity node. We define *recirculation blocking* of class-r jobs in node j as follows. If a class-r job completes its service and wants to jump to full node j, then it will jump to node $\varphi_j(r)$ in class r instead. Node $\varphi_j(r)$ is described by the following mapping. For all nodes j with recirculation blocking, the *recirculation mapping* φ_j should be predefined using

$$\varphi_j : \mathbf{R}' \longrightarrow \mathbf{N},\ r \longmapsto \varphi_j(r)$$

with $\mathbf{R}' \subset \mathbf{R}$ defined as the set of job classes in which jobs can jump to node j. Note that if a class-r job in node i wants to jump to full node j with recirculation blocking in class s, then it jumps to node $\varphi_j(r)$ in class r instead.

Remark 4.12 Recirculation blocking is defined differently in literature. Then, node $\varphi_j(r)$ and the class in which the job arrives at node $\varphi_j(r)$ depend on the routing probabilities. This turned out to be not useful in this study.

Assumption 4.13 In this study it holds that for each finite capacity node j with recirculation blocking and arbitrary job class r, node $\varphi_j(r)$ is not a finite capacity node with recirculation blocking. Node $\varphi_j(r)$ is allowed to be a finite capacity node with transfer blocking.

Definition 4.14 (Blocking probability) The *blocking probability* PB_j of finite capacity node j with recirculation blocking is defined as the probability that an arbitrary job arriving at node j is blocked. Note that the blocking probability of node j equals the probability that node j is fully occupied. Hence, we can define the blocking probability of node j by

$$\mathrm{PB}_j := \mathbb{P}(k_j(K) = c_j), \tag{4.2}$$

where $k_j(K)$ equals the number of jobs in node j for the network with K jobs, and c_j equals the node capacity of node j.

In the next section we describe job class dependent priority properties with regard to jobs in a queueing network.

4.1.4 Queueing networks with priorities

This section focuses on multi-class queueing networks with priority properties. There are two types of priorities described in literature, namely priority with pre-emption and head-of-line priority. Solely the latter type of priority is used in this study.

Definition 4.15 (Priority order) Consider a queueing network with at least two job classes. For the use of priorities in node i, a linear, non-strict order should be predefined on the set of job classes $\mathbf{R}$ for node i as follows. The R elements of the set $\mathbf{R}$ can be represented as

$$r_1 \geq r_2 \geq \cdots \geq r_R. \tag{4.3}$$

At least one of the inequalities above should be a strict inequality, so that the priority property influences the order in which jobs are served in node i. This is referred to as the *priority order* of node i. For r, s two arbitrary job classes with $r > s$ using the representation above, we say that r has a higher priority than s. The relations $r < s$ and $r = s$ can be explained analogously.

Definition 4.16 (Head-of-line priority) Let $\mathbf{R}$ be a set of jobs classes with a predefined priority order. Jobs with equal priority follow a FCFS discipline and jobs of different priorities are served according to the priority order. Jobs with the highest priority are served first. Jobs cannot pre-empt jobs that are all ready in service. We call this *head-of-line priority*. Another term used in literature is priority without pre-emption.

Head-of-line priorities will be used for modeling the chain conveyor. Furthermore, we will use finite capacity nodes with either transfer of recirculation blocking. We propose a closed queueing network for modeling the chain conveyor in the next section of this chapter.

4.2 Model description

In this section we propose a closed queueing network for modeling the chain conveyor at Hollander, adopting the following assumption.

Assumption 4.17 The chain conveyor is modeled as a multi-class closed queueing network with both single server and multiple server nodes. Class changes are allowed. All nodes follow a FCFS discipline and have general distributed service times. The model consists of finite capacity nodes with either transfer blocking or recirculation blocking. Both head-of-line priorities and class-dependent service time distributions are allowed.

Remark 4.18 Remember Remark 4.3 and note that general distributed service times are allowed in the proposed model. However, the MVA-based algorithm for analyzing closed queueing networks will solely support exponential distributed service times. We will investigate whether the to be proposed MVA-based algorithm is suitable for the network that models the chain conveyor.

This section is continued by describing the network topology of the proposed model in Section 4.2.1. Sections 4.2.3 and 4.2.2 describe the network details. We give a summary of the description of the proposed model at the end of this chapter, in Section 4.3.

4.2.1 Network topology

The overall network topology of the proposed queueing network that models the chain conveyor with the transfer in operation is shown in Figure 4.2. All directed edges correspond to routing probabilities that are either positive or equal to zero, depending on the job class. The routing probabilities per job class are specified in Section 4.2.2. The set of nodes $\mathbf{N}$ consists of four different types of service centers, which are defined below. The white nodes correspond to the conveyor nodes. Jobs in the model correspond to carriers in the conveyor system.

The set of job classes $\mathbf{R}$ can be subdivided into three types of classes. First, $\mathbf{R}_0$ is the set of the one job class that represents empty carriers. Secondly, we define the set of four job classes $\mathbf{R}_1$. There is a one-to-one correspondence between this set and the diverters in the DC as follows. A class-r job with r in $\mathbf{R}_1$ corresponds to a loaded carrier assigned to diverter r. We will refer to the set $\mathbf{R}_0 \cup \mathbf{R}_1$ as the set of *basis job classes*. Finally, $\mathbf{R}_2$ is a set of job classes also representing carriers that are assigned to diverters. These job classes are introduced to model certain priority properties and class dependent service time distributions. We will refer to the set of job classes $\mathbf{R}_2$ as the set of *extra job classes*. The one-to-one correspondence ψ between

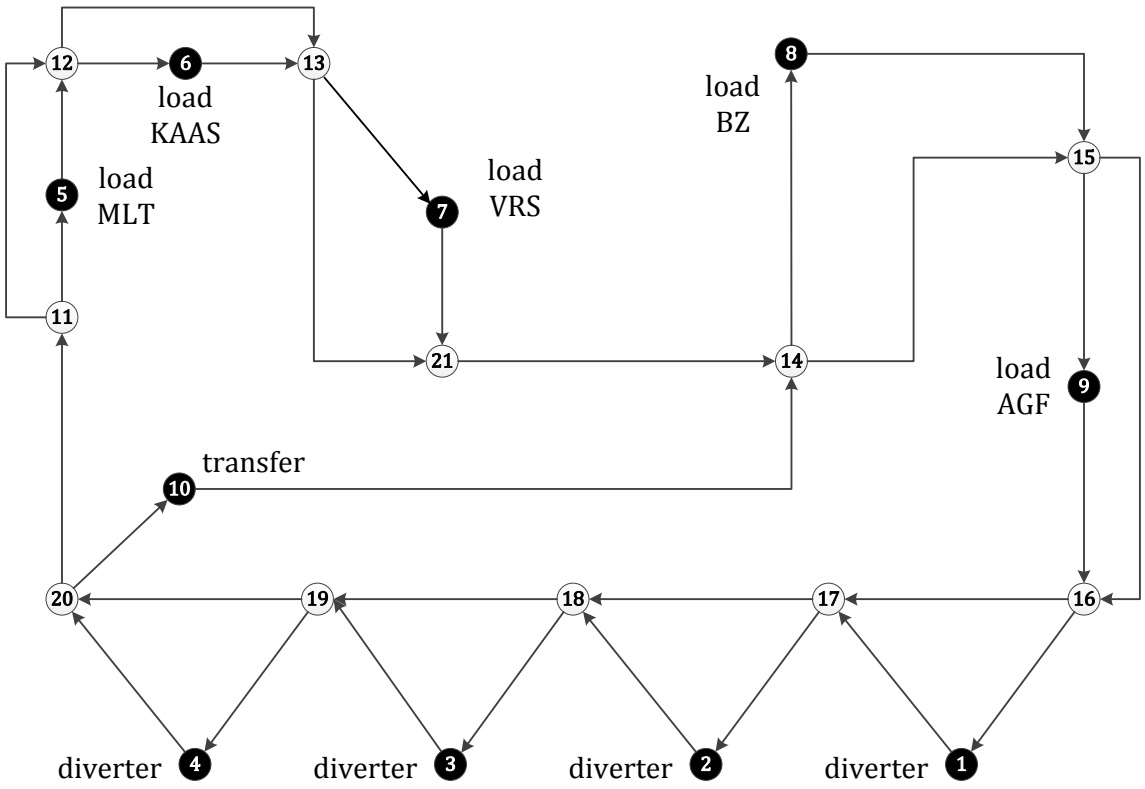


Figure 4.2: Topology of the proposed closed queueing network.

the job classes in $\mathbf{R}_1$ and the job classes in $\mathbf{R}_2$ is defined by

$$\psi : \mathbf{R}_1 \longrightarrow \mathbf{R}_2,\ r \longmapsto \psi(r), \tag{4.4}$$

such that for all job classes r in $\mathbf{R}$, the job classes r and $\psi(r)$ represent the same diverter. The remainder of this section is used to discuss all types of nodes in the node set $\mathbf{N}$, as shown in Figure 4.2.

Diverter node $-/G/m/c$ Nodes 1 till 4 are referred to as diverter nodes. For all diverter nodes, the number of servers equals the number of operators at the diverter. The diverters contain a queue where assigned carriers are stored. Therefore, the node capacity is strictly greater than the number of servers. The service time distribution is a general continuous distribution. Service at a diverter node corresponds to removing a carrier from the diverter and set it up for transport.

Load node $-/G/m/m$ Nodes 5 till 9 are referred to as load nodes. The load nodes correspond to the pick lanes in the DC. For all load nodes, the number of servers is equal to the number of order pickers in the corresponding store regions. All load nodes have no queue. Hence, for all load nodes the node capacity equals the number of servers. The service time distribution is a general continuous distribution. Service at a load node is removing a carrier from the chain, load the carrier, and couple it back onto the chain. Note that the model contains solely five load nodes, even though the distribution center contains six store regions. This is caused by data limitations, which will be explained in Section 6.2.

Transfer node $-/D/1/c$ Node 10 is referred to as the transfer node. For the transfer node, the node capacity is equal to the number of carriers that can be stored at the transfer. There is only one server and the service time distribution is deterministic. The service times are equal to the time duration between two subsequent holes that pass the transfer. Service at the transfer is inserting a loaded carrier back on the chain conveyor.

Conveyor node $-/G/m/m$ Nodes 11 till 21 are referred to as conveyor nodes. The conveyor nodes are introduced to model the travel time of the closed loop of the chain conveyor. For all conveyor nodes the number of servers equals the number of holes in the chain between the foregoing node and subsequent node in the system. There is no queue, therefore the node capacity equals the number of servers. Jobs that arrive at a conveyor node from a foregoing conveyor node, diverter node, or the transfer node travel along the whole conveyor part corresponding to the conveyor node. Hence, their service time is deterministic and equals the travel time. Jobs that arrive at a conveyor node from a load node correspond to carriers that have been loaded in the corresponding store region. The location where the carrier is coupled back onto the chain conveyor is variable. For these jobs the service time distribution is a discrete random distribution.

In the following section we describe the routing probabilities specified for each job class.

4.2.2 Routing probabilities

Ignoring the different job classes, the routing probabilities are shown in Figure 4.2. In this section, the purpose of defining the different job classes is discussed in more detail and the routing probabilities are specified per job class for each node in the queueing network.

Diverter nodes For all diverter nodes i equal to 1, 2, 3, and 4 with corresponding basic job class r_1, r_2, r_3, and r_4 in $\mathbf{R}_1$, and subsequent conveyor node j equal to 17, 18, 19, and 20, respectively, the routing probabilities are defined by

$$p_{ir_i}^{ms} = \begin{cases} 1 & \text{if } m = j \text{ and } s \in \mathbf{R}_0, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, jobs always jump to the conveyor node in the job class corresponding to empty carriers. Jobs in the diverter nodes are never of class r with r in $\mathbf{R}_0 \cup \mathbf{R}_2$.

Load nodes For all load nodes i equal to 5, 6, 7, 8, and 9 with conveyor nodes j equal to 12, 13, 21, 15, and 16, respectively, the routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = j \text{ and } s = \psi(r), \\ 0 & \text{otherwise,} \end{cases}$$

for all basic job classes r in $\mathbf{R}_1$. Hence, jobs always jump to the conveyor node in the corresponding extra job class. Jobs in the load nodes are never of class r with r in $\mathbf{R}_0 \cup \mathbf{R}_2$.

Transfer node For the transfer node $i = 10$, the conveyor node $j = 14$ and all basic job classes r in $\mathbf{R}_1$ the routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = j \text{ and } s = \psi(r), \\ 0 & \text{otherwise.} \end{cases}$$

Hence, jobs in the basic job classes jump to the conveyor node in the corresponding extra job class. Jobs in the transfer node are never of class r with r in $\mathbf{R}_0 \cup \mathbf{R}_2$.

Conveyor nodes For all conveyor nodes i equal to 11, 12, 13, 14, and 15, with subsequent load nodes j equal to 5, 6, 7, 8, and 9, and subsequent conveyor nodes k equal to 12, 13, 21, 15, and 16, respectively, the routing probabilities for basic job class r in $\mathbf{R}_0$ are defined by

$$p_{ir}^{ms} = \begin{cases} \alpha_{ms} & \text{if } m = j \text{ and } s \in \mathbf{R}_1 \\ 0 & \text{otherwise,} \end{cases} \tag{4.5}$$

where α_{ms} equals the probability that an unassigned carrier is assigned to the diverter corresponding to job class s, right before the service in node m. Note that the sum $\sum_{s \in \mathbf{R}_1} \alpha_{ms}$

should be equal to one for all conveyor nodes m mentioned above. For all basic job classes r in $\mathbf{R}_1$ these routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = k \text{ and } s = r \\ 0 & \text{otherwise,} \end{cases}$$

and for all extra job classes r in $\mathbf{R}_2$ these routing probabilities are given by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = k \text{ and } s = \psi^{-1}(r), \\ 0 & \text{otherwise.} \end{cases}$$

Hence, jobs corresponding to empty carriers jump to the load node and become assigned to a diverter. Jobs corresponding to assigned carriers always jump to the conveyor node in the corresponding basic job class.

For all conveyor nodes i equal to 16, 17, 18 and 19, with subsequent diverter nodes j equal to 1, 2, 3 and 4, and subsequent conveyor nodes k equal to 17, 18, 19, and 20, respectively, let r_j in $\mathbf{R}_1$ be the basis job class corresponding to diverter j. The routing probabilities of all conveyor nodes i described above are defined by

$$\begin{aligned} p_{ir}^{ms} &= \begin{cases} 1 & \text{if } r \in \{r_j, \psi(r_j)\},\ m = j \text{ and } s = r_j, \\ 0 & \text{otherwise,} \end{cases} \\ p_{ir}^{ms} &= \begin{cases} 1 & \text{if } r \in (\mathbf{R}_0 \cup \mathbf{R}_1) \backslash \{r_j\},\ m = k \text{ and } s = r, \\ 0 & \text{otherwise,} \end{cases} \\ p_{ir}^{ms} &= \begin{cases} 1 & \text{if } r \in \mathbf{R}_2 \backslash \{\psi(r_j)\},\ m = k \text{ and } s = \psi^{-1}(r), \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Hence, jobs in one of the classes corresponding to the diverter always jump to that diverter node. All other jobs jump to the conveyor node in one of the basic job classes.

For conveyor node $i = 20$, conveyor node $j = 11$, the transfer node $k = 10$, and the basic job class r in $\mathbf{R}_0$ the routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = j \text{ and } s = r, \\ 0 & \text{otherwise.} \end{cases}$$

For the basic job classes r in $\mathbf{R}_1$ these routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = k \text{ and } s = r, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, jobs corresponding to empty carriers jump to the conveyor node and jobs corresponding to assigned carriers jump to the transfer node. Jobs in conveyor node 20 are never of class r with r in the set of extra job classes $\mathbf{R}_2$.

Finally, for conveyor node $i = 21$ with subsequent conveyor node $j = 14$ and all basic job class r in $\mathbf{R}_0 \cup \mathbf{R}_1$ the routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } m = j \text{ and } s = r, \\ 0 & \text{otherwise.} \end{cases}$$

For all extra job classes r in $\mathbf{R}_2$, these routing probabilities are defined by

$$p_{ir}^{ms} = \begin{cases} 1 & \text{if } j = m \text{ and } s = \psi^{-1}(r), \\ 0 & \text{otherwise.} \end{cases}$$

Hence, all jobs arriving in conveyor node 14 that arrive from the load node have one of the basic job classes. The network topology description is completed in the next section by describing the blocking properties, priority properties, and the class dependent service time distributions of the model.

4.2.3 Class dependent service time distributions, finite capacity nodes, and priority properties

The proposed model consists of solely finite capacity nodes with either recirculation blocking or transfer blocking. Furthermore, some nodes of the network have priority properties. As mentioned at the description of the conveyor nodes, some conveyor nodes have class dependent service time distributions.

Blocking properties We define the set of nodes with recirculation blocking as the set of all diverter nodes, load nodes and the transfer node. The set of nodes with transfer blocking consists of all conveyor nodes. For each node with recirculation blocking, we define the recirculation mapping as follows. For diverter node j with corresponding job class d, the recirculation mapping is defined by

$$\varphi_j : \{d\} \longrightarrow \mathbf{N},\ d \longmapsto \varphi_j(d),$$

where $\varphi_j(d)$ equals node 17, 18, 19, and 20 for j equal to node 1, 2, 3, and 4 respectively. For load node j the recirculation mapping is defined by

$$\varphi_j : \mathbf{R}_0 \longrightarrow \mathbf{N},\ r \longmapsto \varphi_j(r),$$

where $\varphi_j(r)$ equals node 12, 13, 21, 15, and 16 for j equal to node 5, 6, 7, 8, and 9, respectively. For the transfer node, the recirculation mapping is defined by

$$\varphi_{10} : \mathbf{R}_1 \longrightarrow \mathbf{N},\ r \longmapsto 11.$$

Remark 4.19 Remember Assumption 4.13. Note that the model does not contain a pair of subsequent nodes which are both finite capacity nodes with recirculation blocking.

Priority properties Jobs that arrive from a diverter node correspond to carriers that have been set up for transport. In practice, these carriers leave the DC. However, for each carrier that leaves the DC, an empty carrier is coupled onto the conveyor at the fill place. We model that assigned carriers enter the closed loop of the chain conveyor right after the diverter they are assigned to. They change into empty carriers. Since this is not the case in real world, these carriers should not influence the process of carriers that actually travel along the conveyor system. Therefore, we model conveyor nodes 17 till 20 with head-of-line priorities. Note that the conveyor nodes have no queue. Hence, the priorities are modeled with regard to the order in which blocked servers (due to transfer blocking) become unblocked. Jobs arriving in these nodes either arrive from a conveyor node or from a diverter node, where the former jobs should be priory served. Hence, for these conveyor nodes we define the priority order $r > s$, for all job classes r in $\mathbf{R}_1 \cup \mathbf{R}_2$ and s in $\mathbf{R}_0$. All pairs of jobs both of a class in $\mathbf{R}_1 \cup \mathbf{R}_2$ have equal priority. Analogously, all pairs of jobs both of the class in $\mathbf{R}_0$ have equal priority.

Jobs that arrive from the transfer node in conveyor node 14 correspond to carriers that were stored onto the transfer. These carriers can solely insert back onto the chain if a free hole passes by. Therefore, these carriers do not influence the carriers on the chain. Hence, jobs arriving from the transfer node should not influence jobs that arrive from conveyor node 21. For conveyor node 14 we define the priority order $r > s$ for all basic job classes r in $\mathbf{R}_0 \cup \mathbf{R}_1$ and all extra job classes s in $\mathbf{R}_2$. All pairs of jobs both of a basic class in have equal priority. Analogously, all pairs of jobs both of an extra class have equal priority.

Class dependent service time distributions Conveyor nodes 12, 13, 21, 15 and 16 model parts of the chain conveyor containing a pick lane. Therefore, these conveyor nodes have class dependent service time distributions. Jobs arriving in one of the basic classes in $\mathbf{R}_0 \cup \mathbf{R}_1$ correspond to jobs that arrive from a conveyor node. Therefore, these job classes have a deterministic service time. Jobs arriving in one of the extra job classes in $\mathbf{R}_2$ correspond to jobs that arrive from a load node. Hence, these jobs follow a discrete service time distribution.

4.3 Conclusions

We modeled the chain conveyor as a multi-class closed queueing network, consisting of both single and multiple server nodes. All nodes follow a FCFS discipline and have general distributed service times. Class changes are allowed. All nodes are finite capacity nodes with either transfer blocking or recirculation blocking. Both of head-of-line priorities and class-dependent service time distributions are used. In the next chapter we propose an MVA-based algorithm and evaluate whether this algorithm is suitable for analyzing the proposed model.

CHAPTER 5

Mean value analysis for closed queueing networks

In this chapter we propose the MVA-based MVABLO-m algorithm in order to analyze the queueing model of the chain conveyor. The MVA [Reiser and Lavenberg, 1980] algorithm is known for its fast computation time. It allows multi-class closed queueing network models with FCFS disciplined infinite capacity nodes with exponential distributed service times and transfer blocking. No class changes are allowed. Both single and multiple server nodes are allowed. The MVA algorithm is used to determine the mean number of jobs per job class in each node. Furthermore, it computes the mean sojourn time per node, and the throughput of the network per job class. The MVABLO algorithm [Akyildiz, 1988] allows closed queueing networks consisting of solely single server nodes, and one single job class. Transfer blocking occurs at all finite capacity nodes. The MVABLO-m algorithm will allow both multiple server nodes and multiple job classes, including class changes. Recirculation blocking, priorities and class dependent service time distributions are ignored. The notation used in is this study is presented in Table 5.1. The values of $\bar{t}_{ir}$, $\bar{k}_i$ and λ_r are referred to as the *performance measures* of the network.

In Section 5.1, we discuss the MVA algorithm [Reiser and Lavenberg, 1980] and extend it for networks allowing class changes [Reiser and Kobayashi, 1975]. In Section 5.2, we discuss the MVABLO algorithm allowing transfer blocking [Akyildiz, 1988] and extend it into the MVABLO-m algorithm. The MVABLO-m algorithm will be verified in Section 5.3 and evaluated in Section 5.4. In the former section, the simulation with pseudo code shown in Appendix C will be verified as well for networks with transfer blocking. In Section 5.5 we discuss the suitability of the MVABLO-m algorithm for analyzing the queueing model of the chain conveyor. This chapter will be summarized in Section 5.6.

5.1 MVA algorithm allowing class changes

In this section, the MVA algorithm proposed by Reiser and Lavenberg [1980] is discussed. The MVA algorithm is used to model multi-class closed queueing networks with solely FCFS disciplined nodes, allowing multiple server nodes. All service time distributions are exponential.

Table 5.1: Notation used for the mean value analysis algorithms.

Symbol	Interpretation
$\mathbf{N}$	Set of nodes.
N	Total number of nodes.
$\mathbf{R}$	Set of job classes.
R	Total number of job classes.
K	Total number of jobs.
$\mathbf{K}_r$	Total number of class-r jobs.
$\mathbf{K}$	Population vector equal to $(\mathbf{K}_1, \ldots, \mathbf{K}_R)$.
$\mathbf{1}_r$	Unit vector of length R in direction r.
$\mathbf{0}$	Zero vector of length R.
C	Capacity of the queueing network (solely used in finite capacity networks).
c_i	Capacity of node i (solely used in finite capacity networks).
m_i	Number of servers in node i.
μ_i	Exponential service rate at node i.
p_{ir}^{js}	Routing probability for all i, j in $\mathbf{N}$ and s, r in $\mathbf{R}$.
k_{ir}	Random variable: *number of class-r jobs* in node i.
k_i	Random variable: *total number of jobs* in node i.
$\pi_i(n)$	*Marginal state probability* of node i: the probability that k_i equals n. To indicate that the marginal state probablity depends on the population vector, one writes $\pi_i(n \mid \mathbf{K})$.
w_{ir}	Random variable: *waiting time* of a class-r job in node i.
τ_{ir}	Random variable: *service time* of a class-r job in node i.
t_{ir}	Random variable: *sojourn time* of a class-r job in node i. The sojourn time equals the waiting time plus the service time of a job.
b_i	Random variable: *blocking time* of node i (solely used in finite capacity networks).
λ_r	The *throughput* of class-r jobs in the network. The throughput is the rate at which class-r jobs are served in the network. Hence, $(1/\lambda_r)$ equals the mean number of class-r jobs that depart from a node, each time unit.
ρ_{ir}	Traffic intensity of class-r jobs in node i.
e_{ir}	Mean number of visits of class-r jobs to node i.
$\bar{\cdot}$	The expected value $\mathrm{E}[\,\cdot\,]$ of a random variable $\cdot$.
$\cdot(\mathbf{K})$	Random variable $\cdot$ depending on the population vector $\mathbf{K}$.

Class changes are not allowed. However, in this section, the MVA will be adjusted using a mapping proposed by Reiser and Kobayashi [1975] so that class changes are allowed. Hence, the following assumption is adopted for the MVA algorithm.

Assumption 5.1 All queueing networks are closed and have solely FCFS disciplined nodes with exponentially distributed service times. Both single and multiple server nodes are allowed. Multiple job classes with class changes are allowed. Finite capacity nodes are not allowed.

The fundamental relation used in the MVA algorithm [Reiser and Lavenberg, 1980] is based on the arrival theorem for closed product form networks. For Poisson processes, the arrival theorem is referred to using the *PASTA property* [Wolff, 1982].

Theorem 5.2 (Arrival theorem) Consider a closed queueing network. If a job arrives at a node, then that job observes the queueing network as the same network but with one job less.

The MVA algorithm [Reiser and Lavenberg, 1980] is an iterative process where the performance measures are computed using these same performance measures of the same network, but with one job less. Hence, if these performance values are initialized for a network with no jobs, all performance measures can be computed iteratively. Reiser and Lavenberg [1980] designed the MVA algorithm to solve closed multi-class queueing networks where jobs cannot change of class. In order to be able to use this MVA algorithm for a queueing network with

class changes, the mapping proposed by Reiser and Kobayashi [1975] can be used to map the multi-class queueing network allowing class changes, trivially into a multi-class model without class changes. Thereby, instead of using the set of chains as input for the algorithm, the set of job classes $\mathbf{R}$ is used. Furthermore, the definition of the relative workload intensity given by Reiser and Kobayashi [1975] is used for the traffic intensity ρ_{ir} [Reiser and Lavenberg, 1980]. These relative workload intensities are defined as the traffic intensities given below.

Definition 5.3 (Traffic intensity) Let a closed multi-class queueing network be given. The *traffic intensity* ρ_{ir} of class-r jobs in node i is the measure of congestion given by

$$\rho_{ir} = \frac{e_{ir}}{\mu_i}$$

where the *mean number of visits* e_{ir} of r-class jobs to node i is defined by the system of linear equations

$$\begin{aligned} e_{ir} &= \sum_{(j,s)\in\mathbf{N}\times\mathbf{R}} e_{js} p_{js}^{ir} \qquad \forall (i,r) \in \mathbf{N}\times\mathbf{R} \\ e_{11} &= 1 \end{aligned}$$

for all classes r in $\mathbf{R}$ and nodes i in $\mathbf{N}$.

According to Reiser and Kobayashi [1975], class changes are allowed using the MVA algorithm, if the traffic intensities defined above are used, instead of the defined traffic intensities in [Reiser and Lavenberg, 1980]. The following proposition contains the fundamental property of the MVA algorithm, namely a relation between the mean sojourn time of a node in the network and the mean number of jobs in that node in the same network with one job less. We give a proof of the proposition, since the proof proposed by Reiser and Lavenberg [1980] is not clearly formulated.

Proposition 5.4 Let a closed multi-class queueing network be given with population vector $\mathbf{K}$. The mean sojourn time of a class-r customer in a single server node i satisfies

$$\bar{t}_{ir}(\mathbf{K}) = \frac{1}{\mu_i}\left(1 + \bar{k}_i(\mathbf{K} - \mathbf{1}_r)\right). \tag{5.1}$$

Proof. The sojourn time of an arriving job equals its waiting time plus its service time, which can be denoted by

$$\bar{t}_{ir}(\mathbf{K}) = \mathrm{E}[\, w_{ir}(\mathbf{K}) + \tau_{ir}(\mathbf{K}) \,] = \bar{w}_{ir}(\mathbf{K}) + \bar{\tau}_{ir}(\mathbf{K}). \tag{5.2}$$

Since exponentially distributed service times are assumed, the mean service time satisfies

$$\bar{\tau}_{ir}(\mathbf{K}) = \frac{1}{\mu_i}. \tag{5.3}$$

Furthermore, the mean waiting time equals the mean service time, times the mean number of customers in node i. From the PASTA property it follows that the state of a node as seen by an arriving class-r job (not including this job), is the same state of that node at an arbitrary

moment with with one class-r job less in the network. Hence, the mean number of jobs in node i at the moment the class-r job arrives, equals the mean number of jobs in node i in the network with one class-r job less, at an arbitrary moment. Summarizing, the mean waiting time satisfies

$$\bar{w}_{ir}(\mathbf{K}) = \frac{1}{\mu_i} \cdot \bar{k}_i(\mathbf{K} - \mathbf{1}_r). \tag{5.4}$$

From equations (5.2), (5.3) and (5.4) it follows that the mean sojourn time of a class-r job at single server node i satisfies equation (5.1). □

The equation for the mean sojourn time of a class-r job at a single server node given in Proposition 5.4 can be extended for multiple server nodes. Reiser and Lavenberg [1980] did not propose a proper proof of the proposition below.

Proposition 5.5 Let a closed multi-class queueing network be given, with population vector $\mathbf{K}$. The mean sojourn time of a class-r job in a multiple server node i with m_i servers satisfies

$$\bar{t}_{ir}(\mathbf{K}) = \frac{1}{m_i \mu_i} \left[1 + \bar{k}_i(\mathbf{K} - \mathbf{1}_r) + \sum_{n=0}^{m_i - 2} (m_i - n - 1)\pi_i(n \mid \mathbf{K} - \mathbf{1}_r) \right]. \tag{5.5}$$

Note that for m_i equal to one, the sojourn time computed using equation (5.5) equals the sojourn time given by equation (5.1).

Proof. Let node i be a multiple server node with m_i servers in the given queueing network with population vector $\mathbf{K}$. For simplicity, we write t_{ir} and w_{ir} instead of $t_{ir}(\mathbf{K})$ and $w_{ir}(\mathbf{K})$, respectively. Define A_i as the event where all m_i servers in node i are busy, and p_i as the probability that event A_i happens. Hence, the complementary event A_i^C is the event that an arriving job immediately gets served at node i, with belonging probability $(1 - p_i)$. It follows that the mean sojourn time of an arriving class-r job in node i can be written as

$$\bar{t}_{ir} = \mathrm{E}[\, t_{ir} \mid A_i \,] \cdot p_i + \mathrm{E}[\, t_{ir} \mid A_i^C \,] \cdot (1 - p_i). \tag{5.6}$$

From equation (5.2) it follows that

$$\mathrm{E}[\, t_{ir} \mid A_i^C \,] = 0 + \frac{1}{\mu_i}, \tag{5.7}$$

and

$$\mathrm{E}[\, t_{ir} \mid A_i \,] = \mathrm{E}[\, w_{ir} \mid A_i \,] + \frac{1}{\mu_i}. \tag{5.8}$$

From equations, (5.6) (5.7) and (5.8) is follows that

$$\bar{t}_{ir} = \frac{1}{\mu_i} + \mathrm{E}[\, w_{ir} \mid A_i \,] \cdot p_i. \tag{5.9}$$

The mean waiting time of an arriving class-r job at node i satisfies

$$\begin{aligned}
\mathrm{E}[\, w_{ir} \mid A_i \,] \cdot p_i &= \sum_{j=m_i}^{K-1} \frac{1}{m_i \mu_i} (j + 1 - m_i) \cdot \mathbb{P}[k_i = j \mid A_i] \cdot p_i \\
&= \sum_{j=m_i}^{K-1} \frac{1}{m_i \mu_i} (j + 1 - m_i) \cdot \mathbb{P}[k_i = j, A_i] \\
&= \sum_{j=m_i}^{K-1} \frac{1}{m_i \mu_i} (j + 1 - m_i) \cdot \pi_i(j) \\
&= \frac{1}{m_i \mu_i} \left[\sum_{j=m_i}^{K-1} \pi_i(j) + \sum_{j=m_i}^{K-1} (j - m_i) \pi_i(j) \right].
\end{aligned} \tag{5.10}$$

From equations (5.9) and (5.10) it follows that

$$\bar{t}_{ir} = \frac{1}{m_i \mu_i} \left[m_i + \sum_{j=m_i}^{K-1} \pi_i(j) + \sum_{j=m_i}^{K-1} (j - m_i) \pi_i(j) \right]. \tag{5.11}$$

The second sum in equation (5.11) equals the mean size of the waiting line at node i. Furthermore, the mean waiting line size equals the mean number of jobs at node i, minus the mean number of busy servers at node i. From the PASTA property it follows that the mean number of jobs at node i at the moment a class-r customer arrives, equals $\bar{k}_i(\mathbf{K} - \mathbf{1}_r)$. The mean number of busy servers at node i equals

$$\sum_{j=1}^{m_i - 1} j \pi_i(j) + \sum_{j=mi}^{K-1} m_i \pi_i(j).$$

Hence, the mean waiting line size of node i satisfies

$$\sum_{j=m_i}^{K-1} (j - m_i) \pi_i(j) = \bar{k}_i(\mathbf{K} - \mathbf{1}_r) - \left[\sum_{j=1}^{m_i - 1} j \pi_i(j) + \sum_{j=m_i}^{K-1} m_i \pi_i(j) \right]. \tag{5.12}$$

From equations (5.11) and (5.12) it follows that the mean sojourn time satisfies

$$\bar{t}_{ir} = \frac{1}{m_i \mu_i} \left[m_i + \sum_{j=m_i}^{K-1} \pi(j) + \bar{k}_i(\mathbf{K} - \mathbf{1}_r) - \sum_{j=1}^{m_i - 1} j \pi_i(j) - m_i \sum_{j=m_i}^{K-1} \pi_i(j) \right]. \tag{5.13}$$

Using that the first and the third sums in equation (5.13) are probabilities, and that for j equal to zero the term in the second sum equals zero, it follows that

$$\bar{t}_{ir} = \frac{1}{m_i \mu_i} \left[m_i + \left(1 - \sum_{j=0}^{m_i - 1} \pi(j) \right) + \bar{k}_i(\mathbf{K} - \mathbf{1}_r) - \sum_{j=0}^{m_i - 1} j \pi_i(j) - m_i \left(1 - \sum_{j=0}^{m_i - 1} \pi_i(j) \right) \right]. \tag{5.14}$$

Merging the three sums in equation (5.14) results into equation

$$\bar{t}_{ir} = \frac{1}{m_i \mu_i} \left[1 + \bar{k}_i(\mathbf{K} - \mathbf{1}_r) + \sum_{j=0}^{m_i - 1} (m_i - 1 - j)\pi_i(j) \right]. \tag{5.15}$$

Since the term of the sum in equation (5.15) equals zero for j equal to $m_i - 1$, Proposition 5.5 is proved by this equation. □

Lemma 5.6 Let a closed multi-class queueing network be given. For the throughput of class-r jobs we have that the equation

$$\lambda_r(\mathbf{K}) = \frac{\mathbf{K}_r}{\sum_{i \in N} e_{ir} \bar{t}_{ir}(\mathbf{K})}$$

holds for all r in $\mathbf{R}$. As indicated by Reiser and Lavenberg [1980], the proof of Lemma 5.6 follows directly from Little's law [Little, 1961].

Lemma 5.7 Let a closed multi-class queueing network be given. The mean number of class-r jobs at node i satisfies

$$\bar{k}_{ir}(\mathbf{K}) = \lambda_r(\mathbf{K}) e_{ir} \bar{t}_{ir}(\mathbf{K})$$

for all nodes i in $\mathbf{N}$ and job classes r in $\mathbf{R}$. Hence, the mean number of jobs in node i satisfies

$$\bar{k}_i(\mathbf{K}) = \sum_{r \in \mathbf{R}} \lambda_r(\mathbf{K}) e_{ir} \bar{t}_{ir}(\mathbf{K}).$$

As indicated by Reiser and Lavenberg [1980], the proof of Lemma 5.7 follows directly from Little's equation for service centers [Little, 1961].

Proposition 5.8 Let a closed multi-class queueing network be given and let $\mathbf{K}$ be the population vector of the queueing network. The marginal state probabilities satisfy

$$\pi_i(n \mid \mathbf{K}) = \frac{1}{n} \sum_{r \in \mathbf{R}} \lambda_r(\mathbf{K}) \rho_{ir} \pi_i(n - 1 \mid \mathbf{K} - \mathbf{1}_r),$$

for all i in $\mathbf{N}$ with $n = 1, \ldots, m_i - 1$. Furthermore, for all i in $\mathbf{N}$ the marginal state probabilities satisfy

$$\pi_i(0 \mid \mathbf{K}) = 1 - \frac{1}{m_i} \left[\sum_{r \in \mathbf{R}} \lambda_r(\mathbf{K}) \rho_{ir} + \sum_{n=1}^{m_i - 1} (m_i - n) \pi_i(n \mid \mathbf{K}) \right].$$

A proof of Proposition 5.8 is given by Reiser and Lavenberg [1980].

Given the marginal state probability and the mean number of jobs for each node of a queueing network under Assumption 5.1 with no jobs, the performance measures of that queueing networks can be computed iteratively for any population vector $\mathbf{K}$, using the equations described above. Consider such queueing network. The $\bar{k}_i(\mathbf{0})$ equals zero for all nodes i. Hence, the marginal state probabilities $\pi_i(0 \mid \mathbf{0})$ equal one, and the remaining marginal state probabilities $\pi_i(n \mid \mathbf{0})$ equal zero for nodes i and all n greater than zero.

The MVA algorithm for queueing networks under Assumption 5.1 is given in Algorithm 5.9. Using this algorithm, the following performance measures can be computed: the mean number of jobs in each node, the mean sojourn time in each node, and the throughputs of the network.

Algorithm 5.9 (MVA algorithm) Given a queueing network under Assumption 5.1, with the set of nodes $\mathbf{N}$, the set of job classes $\mathbf{R}$ and the population vector $\mathbf{K}$, this algorithm computes the exact values of the mean number of jobs $\bar{k}_i$, throughput λ_r, and the mean sojourn time $\bar{t}_{ir}$ for all nodes i and job classes r.

Initialization

For all i in $\mathbf{N}$ and r in $\mathbf{R}$
- $\bar{k}_{ir}(\mathbf{0}) \longleftarrow 0$;
- $\pi_i(0 \mid \mathbf{0}) \longleftarrow 1$;
- **For** all $k = 1, 2, \ldots, m_i - 1$
  - $\pi_i(k \mid \mathbf{0}) \longleftarrow 0$;
- **end**

end

Main iteration

For all $\mathbf{k} \leq \mathbf{K}$ with $\mathbf{k} = (\mathbf{k}_1, \ldots, \mathbf{k}_R)$
- **For** all i in $\mathbf{N}$ and r in $\mathbf{R}$
  - 1. $\bar{t}_{ir}(\mathbf{k}) \longleftarrow \dfrac{1}{m_i \mu_i}\left[1 + \sum_{r \in \mathbf{R}} \bar{k}_{ir}(\mathbf{k} - \mathbf{1}_r) + \sum_{n=0}^{m_i - 2} (m_i - n - 1)\pi_i(n \mid \mathbf{k} - \mathbf{1}_r)\right]$;
- **end**
- **For** all r in $\mathbf{R}$
  - 2. $\lambda_r(\mathbf{k}) \longleftarrow \dfrac{\mathbf{k}_r}{\sum_{i \in \mathbf{N}} e_{ir} \bar{t}_{ir}(\mathbf{k})}$;
- **end**
- **For** all i in $\mathbf{N}$ and r in $\mathbf{R}$
  - 3. $\bar{k}_{ir}(\mathbf{k}) \longleftarrow \lambda_r(\mathbf{k}) e_{ir} \bar{t}_{ir}(\mathbf{k})$;
- **end**
- **For** all i in $\mathbf{N}$ and $n = 1, \ldots, m_i - 1$
  - 4. $\pi_i(n \mid \mathbf{k}) \longleftarrow \dfrac{1}{n} \sum_{r \in \mathbf{R}} \lambda_r(\mathbf{k}) \rho_{ir} \pi_i(n - 1 \mid \mathbf{k} - \mathbf{1}_r)$;
  - 5. $\pi_i(0 \mid \mathbf{k}) \longleftarrow 1 - \dfrac{1}{m_i}\left[\sum_{r \in \mathbf{R}} \lambda_r(\mathbf{k}) \rho_{ir} + \sum_{n=1}^{m_i - 1} (m_i - n)\pi_i(n \mid \mathbf{k})\right]$;
- **end**

end

In the next section of this chapter, we propose and discuss the MVABLO-m algorithm. Unlike the MVA algorithm, the MVABLO-m algorithm allows finite capacity nodes with transfer blocking.

5.2 MVABLO-m algorithm allowing multiple job classes and multiple server nodes

In this section, the MVABLO algorithm proposed by Akyildiz [1988] is discussed. Unlike the MVA algorithm, the MVABLO algorithm allows finite capacity nodes with transfer blocking. However, the MVABLO does not allow multiple server nodes and multiple job classes. We extend the MVABLO algorithm into the MVABLO-m algorithm, for which multiple server nodes and multiple job classes are allowed, including class changes. Hence, the following assumption is adopted for the MVABLO-m algorithm.

Assumption 5.10 All queueing networks are closed and have solely FCFS disciplined nodes with exponentially distributed service times. Both single and multiple server nodes are allowed. Multiple job classes with class changes are allowed. Transfer blocking occurs at all finite capacity nodes. The total number of jobs in the network satisfies the capacity constraint and adopts the deadlock property.

Consider an arbitrary, finite capacity node j of a given queueing network and let node i be a node of the network for which there exist r, s in $\mathbf{R}$ such that $p_{ir}^{js} > 0$. In other words, node j is subsequent to node i. Akyildiz [1988] adjusted the MVA algorithm for single-class networks with solely single server nodes, modeling finite capacity queues with transfer blocking. This resulted into the MVABLO algorithm. The adjustments are based on the following two basic properties of transfer blocking.

(i) Node i is blocked if node j is fully occupied.

(ii) Node j cannot accept any new jobs if it is fully occupied.

In this study, queueing networks with both multiple job classes and multiple server nodes are allowed. Hence, the MVABLO algorithm should be adjusted. This section is continued by describing our adjustments of the MVA algorithm given in Algorithm 5.9 into the MVABLO-m algorithm, using the properties holding for the MVABLO algorithm described above.

Imagine that the MVA algorithm is executed for a queueing network with finite capacity node j and a foregoing node i. Possibly, there will be a moment that the computed mean number of jobs at node j becomes greater than its node capacity c_j. Hence, then it is very likely that blocking occurs. From property (i) it follows that the mean sojourn time of node i will increase.

Definition 5.11 (Mean blocking time) Let i in $\mathbf{N}$ be an arbitrary node of a given queueing network. Define b_i as the random variable corresponding to the time with which the sojourn time of jobs in node i increases, due to blocked jobs at the subsequent nodes. Hence, $\bar{b}_i$ equals the mean time duration a job is in node i, while at least one of the the servers of i is blocked. This is referred to using the *mean blocking time* of node i. For the mean blocking time at node i due to blocked jobs at node j, write $\bar{b}_{ij}$. Hence, the sum over all full nodes j of $\bar{b}_{ij}$ equals $\bar{b}_i$.

Remember that the notation $\bar{b}_i(\mathbf{K})$ is used to indicate that $\bar{b}_i$ depends on the population vector $\mathbf{K}$. Given the mean blocking time of an arbitrary node i in $\mathbf{N}$ of a queueing network, the mean sojourn time of class-r jobs at node i given in (5.5) can be substituted by

$$\bar{t}_{ir}(\mathbf{K}) = \frac{1}{m_i \mu_i} \left[1 + \bar{k}_i(\mathbf{K} - \mathbf{1}_r) + \sum_{n=0}^{m_i - 2} (m_i - n - 1)\pi_i(n \mid \mathbf{K} - \mathbf{1}_r) \right] + \bar{b}_i(\mathbf{K}). \tag{5.16}$$

Lemma 5.12 Let j be an arbitrary, finite capacity node with exponentially distributed service times and assume that transfer blocking occurs. Define the random variable γ_j as the time duration between the arrival of a job at full node j, till the moment the next server in node j finishes its service. The $\bar{\gamma}_j$ is referred to using the *mean block duration* of a job jumping to node j. The mean block duration of a blocked job at node j is given by

$$\bar{\gamma}_j = \frac{1}{m_j \mu_j}.$$

The proof of Lemma 5.12 follows directly from the memoryless property of the exponential distribution described in Theorem 4.2.

Akyildiz [1988] approximated the mean blocking time of an arbitrary blocked node i, due to blocked jobs at the subsequent full node j, for queueing networks with solely single server nodes and single job classes using

$$\bar{b}_{ij} = \frac{1}{\mu_j} \left[\frac{p_{ij} e_i}{e_j} \right]. \tag{5.17}$$

Thereby, p_{ij} equals the probability that a job in node i proceeds to node j, and e_i equals the mean number of visits at node i (analogue for e_j). Hence, these are the routing probability and the mean number of visits defined for single-class networks, respectively. Allowing multiple server nodes, $1/\mu_j$ in (5.17) is substituted by the mean block duration $\bar{\gamma}_j$ defined in Lemma 5.12. Furthermore, for queueing networks with $R > 1$ job classes, the mean number of visits to arbitrary node i satisfies

$$e_i = \sum_{r \in \mathbf{R}} e_{ir}. \tag{5.18}$$

The probability that a job in node i proceeds to node j is given by

$$\begin{aligned} p_{ij} &= \sum_{r \in \mathbf{R}} \mathbb{P}\left[\, p_{ij} \mid \text{class-}r \text{ job in node } i \,\right] \cdot \mathbb{P}\left[\, \text{class-}r \text{ job in node } i \,\right] \\ &= \sum_{r \in \mathbf{R}} \left[\sum_{s \in \mathbf{R}} p_{ir}^{js} \right] \cdot \frac{e_{ir}}{\sum_{s \in \mathbf{R}} e_{is}} = \frac{\sum_{r \in \mathbf{R}} \sum_{s \in \mathbf{R}} p_{ir}^{js} e_{ir}}{\sum_{s \in \mathbf{R}} e_{is}}. \end{aligned} \tag{5.19}$$

Substituting (5.18) for e_i and e_j and (5.19) for p_{ij} in equation (5.17) results into the mean blocking time of node i due to blocked jobs at full multiple server node j in a queueing network

with multiple job classes. This mean blocking time is given by

$$\bar{b}_{ij} = \frac{\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{R}} p_{ir}^{js} e_{ir}}{m_j\mu_j \sum_{r\in\mathbf{R}} e_{jr}}. \tag{5.20}$$

From property (ii) it follows that at the moment the number of jobs in node j becomes equal to the node capacity c_j, node j cannot accept any new jobs. Consider the moment that node j is full, no servers are blocked due to node j, and a job in node i completes its service and wants to jump towards node j. In equation (5.5) is used that the mean sojourn time of an arriving job equals its own service time plus its own waiting time. Since the job cannot enter node j, it will spend part of its waiting time in node i. This waiting time spend in node i is exactly the same time as the mean block duration, which is given in Lemma 5.12. Hence, to compute the expected sojourn time of the job, the mean sojourn time given in equation (5.5) should be deducted by $1/m_j\mu_j$. Therefore, the mean sojourn time of class-r customers in node j satisfies

$$\bar{t}_{jr}(\mathbf{K}) = \frac{1}{m_j\mu_j}\left[\bar{k}_j(\mathbf{K}-\mathbf{1}_r) + \sum_{n=0}^{m_j-2}(m_j-n-1)\pi_j(n \mid \mathbf{K}-\mathbf{1}_r)\right]. \tag{5.21}$$

Remark 5.13 For both the mean blocking time given in equation (5.20) and the mean sojourn time given in equation (5.21), the mean block duration given in Lemma 5.12 is used. However, the mean block duration only holds when is assumed that solely one job is blocked at a time. Consider an arbitrary moment at which node j is full, and assume that there are more than one jobs that have finished service and want to jump to node j. The servers are unblocked following a FCFS discipline. For the first server, we have that the expected blocking time is given by (5.20). However, for the x-th blocked server, with $x > 1$, it holds that $x-1$ servers unblock first. Hence, for the x-th blocked server, the job has to wait x times the value of the $\bar{b}_{ij}$ in equation (5.20). For the x-th blocked server, equation (5.20) holds if and only if the right-hand side of the equation is multiplied by x. Taking this into account using the MVABLO-m algorithm, the probability distribution of the number of blocked servers due to the finite capacity of node j should be determined. We choose to ignore this inaccuracy.

Combining equations (5.16) and (5.21), the mean sojourn time of an arbitrary node i in a queueing network under Assumption 5.10 is approximated by

$$\bar{t}_{ir}(\mathbf{K}) = \frac{1}{m_i\mu_i}\left[z_i(\mathbf{K}) + \bar{k}_i(\mathbf{K}-\mathbf{1}_r) + \sum_{n=0}^{m_i-2}(m_i-n-1)\pi_i(n \mid \mathbf{K}-\mathbf{1}_r)\right] + \bar{b}_i(\mathbf{K}),$$

where $z_i(\mathbf{K})$ is a binary indicator for all nodes i. This binary indicator equals zero if and only if the mean number of customers at node i as computed by the MVA algorithm is greater than the capacity of node i.

The MVABLO-m algorithm for queueing networks under Assumption 5.10 is given in Algorithm 5.14 below. Using this algorithm, the following performance measures can be approxi-

mated: the mean number of jobs in each node, the mean sojourn time of each node, the mean blocking time of each node, and the throughput of the network. Analogue to the MVA algorithm it holds that, given the marginal state probability and the mean number of jobs for all nodes of a queueing network with no jobs, the performance measures of that queueing networks can be computed iteratively. For the MVABLO-m algorithm we need three additional initializations. For all nodes i, the binary indicators $z_i(\mathbf{k})$ are equal to one, for all population vectors $\mathbf{k} \leq \mathbf{K}$, and the mean blocking times $\bar{b}_i(\mathbf{0})$ are equal to zero. Furthermore, an extra binary indicator $y_i(\mathbf{k})$ is introduced for all nodes i and population vectors $\mathbf{k} \leq \mathbf{K}$. For each iteration with population vector $\mathbf{k}$, this indicator equals one, until there is detected that the mean number of jobs $\bar{k}_i(\mathbf{k})$ is less than equal to the node capacity c_i, for all nodes i.

Algorithm 5.14 (MVABLO-m algorithm) Given a queueing network under Assumption 5.10, with the nodes $\mathbf{N}$, the set of job classes $\mathbf{R}$, the population vector $\mathbf{K}$, and for all nodes i in $\mathbf{N}$ the node capacity c_i, this algorithm approximates the exact values of the mean number of jobs $\bar{k}_i$, throughput λ_r, the mean sojourn time $\bar{t}_{ir}$, and the mean blocking time $\bar{b}_i$ for all nodes i and job classes r.

Initialization

For all i in $\mathbf{N}$ and r in $\mathbf{R}$
- $\bar{k}_{ir}(\mathbf{0}) \longleftarrow 0$;
- $\pi_i(0 \mid \mathbf{0}) \longleftarrow 1$;
- **For** all $k = 1, 2, \ldots, m_i - 1$
  - $\pi_i(k \mid \mathbf{0}) \longleftarrow 0$;
- **end**
- $\bar{b}_i(\mathbf{0}) \longleftarrow 0$;
- **For** all $\mathbf{k} \leq \mathbf{K}$
  - $z_i(\mathbf{k}) \longleftarrow 1$;
  - $y_i(\mathbf{k}) \longleftarrow 1$;
- **end**

end

Main iteration

For all $\mathbf{k} \leq \mathbf{K}$ with $\mathbf{k} = (\mathbf{k}_1, \ldots, \mathbf{k}_R)$
- **While** $\sum_{i \in \mathbf{N}} y_i(\mathbf{k}) \geq 1$
  - **For** all i in $\mathbf{N}$ and r in $\mathbf{R}$
    - 1. $\bar{t}_{ir}(\mathbf{k}) \longleftarrow \dfrac{1}{m_i \mu_i} \left[z_i(\mathbf{k}) + \sum_{r \in \mathbf{R}} \bar{k}_{ir}(\mathbf{k} - \mathbf{1}_r) + \sum_{n=0}^{m_i - 2} (m_i - n - 1) \pi_i(n \mid \mathbf{k} - \mathbf{1}_r) \right] + \bar{b}_i(\mathbf{k})$;
  - **end**
  - **For** all r in $\mathbf{R}$
    - 2. $\lambda_r(\mathbf{k}) \longleftarrow \mathbf{k}_r / \sum_{i \in \mathbf{N}} e_{ir} \bar{t}_{ir}(\mathbf{k})$;
  - **end**
  - **For** all i in $\mathbf{N}$ and r in $\mathbf{R}$
    - 3. $\bar{k}_{ir}(\mathbf{k}) \longleftarrow \lambda_r(\mathbf{k}) e_{ir} \bar{t}_{ir}(\mathbf{k})$;
  - **end**

For all i in $\mathbf{N}$
If $\bar{k}_i(\mathbf{k}) < c_i$
For all r in $\mathbf{R}$
4. $z_i(\mathbf{k}+\mathbf{1}_r) \longleftarrow z_i(\mathbf{k})$;
5. $\bar{b}_i(\mathbf{k}+\mathbf{1}_r) \longleftarrow \bar{b}_i(\mathbf{k})$;
6. $y_i(\mathbf{k}) \longleftarrow 0$; (1.1)
end
Else
For all j in $\mathbf{N}$
7. $\bar{b}_j(\mathbf{k}) \longleftarrow \bar{b}_j(\mathbf{k}) + \dfrac{\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{R}} p_{jr}^{is} e_{jr}}{m_i\mu_i \sum_{r\in\mathbf{R}} e_{ir}}$;
end
8. $z_i(\mathbf{k}) \longleftarrow 0$;
9. $y_i(\mathbf{k}) \longleftarrow 1$;
end
end
end
For all i in $\mathbf{N}$ and $n = 1, \ldots, m_i - 1$
10. $\pi_i(n \mid \mathbf{k}) \longleftarrow \dfrac{1}{n}\sum_{r\in\mathbf{R}} \lambda_r(\mathbf{k})\rho_{ir}\pi_i(n-1 \mid \mathbf{k}-\mathbf{1}_r)$;
11. $\pi_i(0 \mid k) \longleftarrow 1 - \dfrac{1}{m_i}\left[\sum_{r\in\mathbf{R}} \lambda_r(\mathbf{k})\rho_{ir} + \sum_{n=1}^{m_i-1}(m_i - n)\pi_i(n \mid \mathbf{k})\right]$;
end
end

In the following two sections of this chapter, the MVABLO-m algorithm will be verified and evaluated.

5.3 Verifications

In order to evaluate the MVABLO-m algorithm, a benchmark value is needed for the results. In this study, a simulation is used as the benchmark. The pseudo code of the simulation can be found in Appendix C. Note that the simulation allows recirculation blocking and priorities. This can be ignored for the evaluation of the MVABLO-m algorithm. Both the simulation and the MVABLO-m algorithm are programmed using Matlab. The simulation will be verified using exact values obtained by numerical analysis in Section 5.3.1. For the verification of the MVABLO-m algorithm in Section 5.3.2, the results of the MVABLO algorithm presented by Akyildiz [1988] will be used.

5.3.1 Verification of the simulation for evaluating the MVABLO-m algorithm

In this section, the simulation with pseudo code given in Appendix C is verified for single-class queueing networks under Assumpion 5.10, with solely single server nodes. The simulation is programmed using Matlab. Hereby, the simulation results are compared to exact values

Table 5.2: Examples of single-class queueing networks with solely single server nodes.

Ex.	N	$(c_1, \ldots, c_N)$	$(\tau_1, \ldots, \tau_N)$	Network topology	$(e_1, \ldots, e_N)$	**K**
A	2	$(4, 7)$	$(10, 2.5)$	Serially switched	$(1, 1)$	8
B	2	$(10, 10)$	$(1.111, 1.5)$	Serially switched	$(1, 1)$	15
C	2	$(18, 10)$	$(2, 3.33)$	Serially switched	$(1, 1)$	20
D	2	$(18, 13)$	$(0.8, 0.5)$	Serially switched	$(1, 1)$	25
E	2	$(31, 24)$	$(4, 2)$	Serially switched	$(1, 1)$	50
0	3	$(12, 10, 14)$	$(1, 0.5, 0.333)$	Serially switched	$(1, 1, 1)$	27–30, 32
1	3	$(12, 14, 8)$	$(1, 0.5, 0.333)$	Serially switched	$(1, 1, 1)$	15, 20, 33
2	3	$(4, 5, 5)$	$(1.5, 2.0, 1.0)$	Serially switched	$(1, 1, 1)$	10, 12
3	3	$(8, 7, 6)$	$(0.2, 1.2, 1.4)$	Central server model	$(1, 0.5, 0.5)$	10, 12
4	3	$(6, 8, 6)$	$(2.5, 1.2, 1.0)$	Complete network	$(1, 0.714, 0.714)$	8, 10, 11
5	3	$(6, 6, 6)$	$(2.5, 1.2, 1)$	Complete network	$(1, 0.945, 0.484)$	8, 10, 11
6	4	$(4, 4, 4, 4)$	$(1.8, 2.6, 2.8, 2.4)$	Serially switched	$(1, 1, 1, 1)$	10, 12, 14
7	4	$(3, 6, 7, 5)$	$(1.8, 2.6, 2.8, 2.4)$	Serially switched	$(1, 1, 1, 1)$	15, 18
8	5	$(2, 4, 3, 4, 2)$	$(1.2, 1.5, 1.8, 1.6)$	Serially switched	$(1, 1, 1, 1, 1)$	10, 12, 14
9	3	$(35, 30, 40)$	$(10, 100, 50)$	Serially switched	$(1, 1, 1)$	50, 75, 100
10	4	$(20, 30, 40, 15)$	$(10, 5, 20, 15)$	Serially switched	$(1, 1, 1, 1)$	50, 75, 100

obtained by numerical analysis, which are used by Akyildiz [1988] as well. As a performance measure of the verification, the deviation δ(Simulation) definition below is computed.

Definition 5.15 (Deviation) For x the benchmark value and x' the simulation value to be evaluated, the *deviation* δ(Simulation) of x' from x is defined by

$$\delta(\text{Simulation}) := 100 \cdot \frac{|x - x'|}{x}. \tag{5.22}$$

The deviation δ(MVABLO-m) can be defined analogously.

All networks for which Akyildiz [1988] presented results of the MVABLO algorithm are presented in Table 5.2. Since all networks consist solely one job class, the routing probabilities p_{i1}^{j1} can be written as p_{ij}. Example 3 has routing probabilities $p_{1j} = 0.5$ and $p_{j1} = 1$ for $j = 2, 3$. Example 4 has routing probabilities $p_{12} = p_{13} = 0.5$, $p_{21} = p_{31} = 0.7$, and $p_{23} = p_{32} = 0.3$. Example 5 has routing probabilities $p_{12} = 0.8$, $p_{21} = p_{31} = 0.7$, $p_{13} = 0.2$, and $p_{23} = p_{32} = 0.3$. The mean number of visits at each node is computed using Definition 5.3. For Examples A up to E and 0, Akyildiz [1988] presented exact values computed using numerical analysis [Stewart, 1978]. Amongst others, these exact values are presented in Table A.1 in Appendix A. The simulation used for the evaluation of the MVABLO-m algorithm is verified using these exact values as a benchmark.

The queueing networks described above are simulated over a time horizon of 100,000 where solely the last 90% of the simulation is taken into account. Hence, the verified time horizon equals 90,000. The simulation results are also presented in Table A.1 together with the deviations from the exact values. The mean deviation of the simulations of all networks described above are estimated to be 0.5% for the mean sojourn time, and 0.7% for the mean number of jobs. Based on these two values we conclude that the simulation performs sufficiently for the evaluation of the MVABLO-m algorithm.

5.3.2 Verification of the MVABLO-m algorithm

The MVABLO-m will be verified for single-class queueing networks under Assumpion 5.10 with solely single server nodes. The MVABLO-m algorithm is programmed using MATLAB. Hereby, the results of the MVABLO algorithm presented by Akyildiz [1988] are used as a benchmark.

The results of the MVABLO-m are compared to the results of the MVABLO [Akyildiz, 1988] for all single-class queueing networks under Assumption 5.10 presented in Table 5.2. The results of both the MVABLO and the MVABLO-m are presented in Table A.1. Note that the value of δ(MVABLO-m) indicates the deviation of the MVABLO-m algorithm, using the simulation as a benchmark. Hence, this deviation is not used for the verification of the MVABLO-m algorithm. It will be used for the evaluation of the algorithm.

Using all examples defined by Akyildiz, the mean sojourn time $\bar{t}_i$ is compared for 124 different nodes i, and the mean number of jobs $\bar{k}_i$ is compared for 112 different nodes i. Hereby, it is assumed that it is sufficient if the value computed using the MVABLO-m algorithm, differs 0.01 or less from the value obtained by the MVABLO algorithm. It follows that 94% (221 out of 236) of the values computed by the MVABLO-m algorithm are correct. The values that are not correct are discussed below.

For Example C, both the sojourn time $\bar{t}_2 = 31.70$ and the mean number of jobs $\bar{k}_1 = 10.39$ do not agree with the values computed by the MVABLO algorithm. Note that the sum of the mean number of jobs $\bar{k}_1 + \bar{k}_2$ computed by the MVABLO algorithm equals $11.02 + 9.62 = 20.64$, which is not equal to the population number $\mathbf{K}$. Using the results of the MVABLO-m algorithm, this sum equals $10.39 + 9.61 = 20.00$, which equals the population number $\mathbf{K}$. Hence, it is more likely for the MVABLO-m results to be correct. Furthermore, all queueing networks of Example 1, Example 2 with population number $\mathbf{K} = 10$, and Example 9 with population number $\mathbf{K} = 100$ have similar situations as Example C. One or two of the values obtained by the MVABLO-m algorithm do not match the corresponding values obtained by the MVABLO algorithm. For these examples as well, the equation $\bar{k}_1 + \bar{k}_2 + \bar{k}_3 = \mathbf{K}$ holds for MVALBO-m results, and does not hold for the MVABLO results. For these examples, the MVABLO-m results are more likely to be correct as well.

The only remarkable results of the MVABLO-m verification are obtained for Example 8 with population number $\mathbf{K} = 14$. For Example 8 with $\mathbf{K}$ equal to 10 or 12, all MVABLO-m results agree with the MVABLO results. With continuing the MVABLO-m algorithm to the network with 14 jobs, the mean sojourn times explode. However, the computed mean number of jobs agree the MVABLO results. For all i in $\mathbf{N}$ define the proportion of sojourn time $\bar{t}'_i$ as $\bar{t}_i / \sum_{j=1}^{5} \bar{t}_j$. We studied the proportions of sojourn times shown in Table 5.3. Notice that these proportions are similar for the MVABLO and the MVABLO-m algorithm. Unfortunately, the cause of these results is not detected in this study.

Assuming that the deviations in Examples C, 1, 2 and 9 are caused by errors of the MVABLO algorithm, it follows that 98% (231 out of 236) of the values computed by the MVABLO-m algo-

Table 5.3: Proportions of sojourn times for Example 8 with **K** equal to 14 jobs in the network.

	$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{t}_5$	$\bar{t}'_1$	$\bar{t}'_2$	$\bar{t}'_3$	$\bar{t}'_4$	$\bar{t}'_5$
MVABLO	5.271	10.018	7.286	10.111	5.448	0.1382	0.2627	0.1911	0.2651	0.1429
MVABLO-m	1039.7	1975.8	1437.0	1994.1	1074.4	0.1382	0.2627	0.1911	0.2651	0.1429

rithm are correct. Disregarding the exploded mean sojourn times for Example 8 with $\mathbf{K} = 14$, it can be said that the verification of the MVABLO-m algorithm showed that the MVABLO-m algorithm works correctly. This chapter is continued with the evaluation of the MVABLO-m algorithm.

5.4 Evaluation of the MVABLO-m algorithm

The MVABLO-m algorithm will be evaluated for different queueing networks using simulation results. The number of nodes N will be varied between 2 and 5, the number of job classes R equals either 1 or 2, and the number of servers in a node is increased up to the node capacity of that node. The examples shown in Table 5.2 will be used for all evaluations. All networks adopt Assumption 5.10. The simulation verified in the previous section is used as a benchmark for the evaluation of the MVABLO-m algorithm. For the queueing networks corresponding to Examples 9 and 10 a time horizon of 1,000,000 is used. For all remaining queueing networks, the simulation is executed over a time horizon of 100,000. The results are presented in Tables A.1, A.2 and A.3 in Appendix A. The values of δ(MVABLO-m) shown in the mentioned tables indicate the deviation defined in Definition 5.15 for the MVABLO-m algorithm. For the networks where exact values are available (the networks in Examples A till C and 0), these exact values are used for the benchmark values of δ(MVABLO-m). For all other networks, the simulation results are used for the benchmark values.

The evaluation of single-class queueing networks with solely single server nodes is discussed in Section 5.4.1. Multi-class queueing networks with solely single server nodes are evaluated in Section 5.4.2. Single-class queueing networks with both single and multiple server nodes are evaluated in Section 5.4.3. The results for the evaluation of the MVABLO-m algorithm are summarized in Section 5.4.4.

5.4.1 Single-class networks with solely single server nodes

The MVABLO-m algorithm is evaluated for single-class queueing networks under Assumption 5.10, with solely single server nodes. As mentioned above, both exact values and simulation results are used as a benchmark. The MVABLO-m results and the simulation results for the single-class networks with solely single server nodes are presented in Table A.1 in Appendix A. Furthermore, the results of the MVABLO algorithm [Akyildiz, 1988] are adopted in this evaluation. Note that for the queueing networks evaluated in this section, the MVABLO-m is the same algorithm as the MVABLO algorithm presented by Akyildiz [1988]. Akyildiz evaluated the MVABLO algorithm for about 150 queueing networks with transfer blocking. The number of jobs were varied from 5 to 100 and the number of nodes from 2 to 6. We will verify his results.

We evaluate the MVABLO-m algorithm for all 16 networks shown in Table 5.2, where various population numbers and numbers of nodes are used. Taking the various population numbers into account, the MVABLO-m algorithm is evaluated for 36 different single-class queueing networks with solely single server nodes. Akyildiz already mentioned that a major advantage of the MVABLO algorithm in comparison with a simulation based methodology, is that the computation time is very short. In this study, the computation time of the simulation is about 60 times longer than the computation time of the MVABLO-m. Akyildiz mentioned that the MVABLO algorithm differs about 10% from the exact numerical results or the simulation results. This study gives a mean deviation of 9.9%. Hence, for single-class networks with solely single server nodes, the MVABLO algorithm and the MVABLO-m algorithm have similar accuracy. This section continues discussing the computation mean blocking times by the MVABLO-m algorithm, using Example 3 and 6.

Example 3 is a queueing network with three nodes and a central server topology. Remarkable is that the deviations of the $\bar{k}_i$ are large for $\mathbf{K} = 10$, and after continuing the MVABLO-m, they become low again for $\mathbf{K} = 12$. First, consider Example 3 with population number $\mathbf{K} = 10$. Running the MVA algorithm shows that the first node for which the capacity is exceeded is node 3. The mean blocking time equals $\bar{k}_3(10) = 6.19$, where the node capacity equals $m_3 = 6$. As a result, the value of $\bar{b}_1(10)$ is increased up to 1.40. Simulation results show that the actual value is given by $\bar{b}_1(10) = 0.92$. Hence, the MVABLO-m algorithm approximated the mean blocking time to high. This can be explained by the minimal exceeded value of $\bar{k}_3(10)$. The real exceeded value is negligible, which causes an overcompensation by the mean blocking time $\bar{b}_1(10)$. Hence, the value of $\bar{t}_1(10) = 1.974$ becomes too high, and the values of $\bar{b}_2(10) = 5.007$ and $\bar{b}_3(10) = 7.640$ become too low in Step 1 of the MVABLO-m algorithm. This causes the large deviations for the $\bar{k}_i(10)$. Secondly, consider Example 3 with population number $\mathbf{K} = 12$. The mean blocking times $\bar{b}_2(12)$ and $\bar{b}_3(12)$ remain equal to zero, and $\bar{b}_1(12) = 1.40$ remains of the same value as $\bar{b}_1(10)$ as well. This implies that, using the MVABLO-m algorithm, the $\bar{k}_i(\mathbf{K})$ never exceeded the node capacities for population numbers $\mathbf{K} = 11,\ 12$. However, the simulation value of the mean blocking time is increased from $\bar{b}_1(10) = 0.92$ up to $\bar{b}_1(12) = 1.77$. Hence, the $\bar{b}_1(12)$ is approximated slightly too low. This results in a slightly too low approximation of $\bar{t}_1(12)$. The impact on the $\bar{k}_i(12)$ is remarkably lower than for the $\bar{k}_i(10)$, since the approximation of the $\bar{b}_1(12)$ is is more accurate than the approximation of $\bar{b}_1(10)$. Finally, consider Example 6 with population number $\mathbf{K} = 10$. This is a serially switched queueing network with four nodes. The MVABLO-m algorithm results into mean blocking times $\bar{b}_i(10)$ that are equal to zero, for all four nodes. The simulation indicates that all $\bar{b}_i(10)$ are positive. Hence, checking the violation of the capacity constraints does not always detect blocking, as Akyildiz [1988] already mentioned.

Summarazing, checking the violation of the capacity constraints using the MVABLO-m algorithm, possibly results into detecting negligible exceeded values values of the $\bar{k}_i$. However, this violation is compensated by increasing the mean blocking times. This increase does not depend on the degree of violation. Hence, negligible violations can cause large compensations at the mean blocking times. This can result into large deviations for the MVABLO-m algorithm.

Furthermore, checking the violation of the capacity constraints does not always detect blocking. The MVABLO algorithm and the MVABLO-m algorithm show similar results for single-class closed queueing networks with solely finite capacity, single server nodes with transfer blocking.

5.4.2 Multi-class networks with solely single server nodes

The MVABLO-m algorithm is evaluated for multi-class queueing networks under Assumption 5.10, with solely single server nodes. The evaluation is performed using four examples from Table 5.2, which are exteded to multi-class networks with two job classes, namely Examples 0, 3, 5 and 7. For these examples, the routing probabilities will be redefined. The MVABLO-m results and the simulation results for the multi-class networks with solely single server nodes are presented in Table A.2 in Appendix A. The MVABLO-m results are evaluated using the deviations δ(MVABLO-m, $R = 2$) with the simulation results as a benchmark. The simulations are executed over a time horizon of 100,000. Since the service times are not class dependent, it is assumed that the mean sojourn times of a multi-class network do not vary per job class. Therefore, solely the mean number of jobs and the mean blocking time of each node are taken into account.

The routing probabilities are redefined such that for all job classes r in $\mathbf{R}$, and all nodes i, j in $\mathbf{N}$, the total routing probability $\sum_{s\in\mathbf{R}} p_{ir}^{js}$ equals the corresponding routing probability p_{ij} given in Section 5.3.1. Furthermore, for all evaluated population vectors $\mathbf{K} = (\mathbf{K}_1, \mathbf{K}_2)$ it holds that the sum $\mathbf{K}_1 + \mathbf{K}_2$ equals one of the population numbers of the corresponding example, shown in Table 5.2. Hence, the results of the MVABLO-m algorithm for the multi-class queueing networks can be compared to the corresponding single-class queueing networks. The results for the single-class networks are also presented in Table A.2. In this study, the influence of varying the distribution of jobs over the initial population vector will be evaluated. We continue this section with redefining the routing probabilities for the evaluated examples.

Example 0 with two job classes The serially switched single-class network with three nodes is extended to a multi-class network with two job classes. The routing probabilities defined for the single-class example are substituted by

$$\begin{array}{lllll} p_{11}^{21} = 0.4, & p_{11}^{22} = 0.6, & p_{12}^{21} = 0.7, & p_{12}^{22} = 0.3, & p_{21}^{31} = 1, \\ p_{22}^{32} = 1, & p_{31}^{11} = 0.4, & p_{31}^{12} = 0.6, & p_{32}^{11} = 0.7, & p_{32}^{12} = 0.3. \end{array}$$

The mean number of visits are given by $e_{11} = e_{21} = e_{31} = 1$ and $e_{12} = e_{22} = e_{32} = 0.857$. This multi-class example will be evaluated for population vectors $\mathbf{K}$ satisfying $\mathbf{K}_1 + \mathbf{K}_2 = 27$.

Example 3 with two job classes The central server single-class network with three nodes is extended to a multi-class network with two job classes. The routing probabilities defined for the single-class example are substituted by

$$\begin{array}{llllll} p_{11}^{21} = 0.15, & p_{11}^{22} = 0.35, & p_{11}^{31} = 0.3, & p_{11}^{32} = 0.2, & p_{12}^{21} = 0.3, & p_{12}^{22} = 0.2, \\ p_{12}^{31} = 0.15, & p_{12}^{32} = 0.35, & p_{21}^{11} = 1, & p_{22}^{12} = 1, & p_{31}^{11} = 1, & p_{32}^{12} = 1. \end{array}$$

It follows that the mean number of visits for all nodes and job classes are given by $e_{11} = 1$, $e_{21} = 0.517$, $e_{31} = 0.483$, $e_{12} = 1.222$, $e_{22} = 0.594$, and $e_{32} = 0.628$. This multi-class example will be evaluated for population vectors $\mathbf{K}$ satisfying $\mathbf{K}_1 + \mathbf{K}_2 = 12$.

Example 5 with two job classes The complete single-class network with three nodes is extended to a multi-class network with two job classes. The routing probabilities defined for the single-class example are substituted by

$$\begin{array}{llllll} p_{11}^{21}=0.3, & p_{11}^{22}=0.5, & p_{11}^{31}=0.1, & p_{11}^{32}=0.1, & p_{12}^{21}=0.5, & p_{12}^{22}=0.3, \\ p_{12}^{31}=0.1, & p_{12}^{32}=0.1, & p_{21}^{11}=0.7, & p_{21}^{31}=0.3, & p_{22}^{12}=0.7, & p_{22}^{32}=0.3, \\ p_{31}^{11}=0.7, & p_{31}^{21}=0.3, & p_{32}^{12}=0.7, & p_{32}^{22}=0.3, & & \end{array}$$

It follows that the mean number of visits for all nodes and job classes are given by $e_{11} = e_{12} = 1$, $e_{21} = e_{22} = 0.945$, and $e_{31} = e_{32} = 0.484$. This multi-class example will be evaluated for population vectors $\mathbf{K}$ satisfying $\mathbf{K}_1 + \mathbf{K}_2 = 11$.

Example 7 with two job classes The serially switched single-class network with four nodes is extended to a multi-class network with two job classes. The routing probabilities defined for the single-class example are substituted by

$$\begin{array}{llllll} p_{11}^{21}=0.4, & p_{11}^{22}=0.6, & p_{12}^{21}=0.7, & p_{12}^{22}=0.3, & p_{21}^{31}=1, & p_{22}^{32}=1, \\ p_{31}^{41}=0.4, & p_{31}^{42}=0.6, & p_{32}^{41}=0.7, & p_{32}^{42}=0.3, & p_{41}^{11}=1 & p_{42}^{12}=1, \end{array}$$

The mean number of visits for all nodes and job classes are given by $e_{11} = e_{21} = e_{31} = e_{41} = 1$, and $e_{12} = e_{22} = e_{32} = e_{42} = 0.857$. This multi-class Example 7 is evaluated for population vectors $\mathbf{K}$ satisfying $\mathbf{K}_1 + \mathbf{K}_2 = 18$.

This section continues with a discussion on the mean deviations for all tested multi-class queueing networks with solely single server nodes, presented in Figure 5.1. As mentioned before, solely the values of the mean number of jobs and the mean blocking duration are presented. The figure shows the mean deviations for the mean number of jobs, separated per evaluated

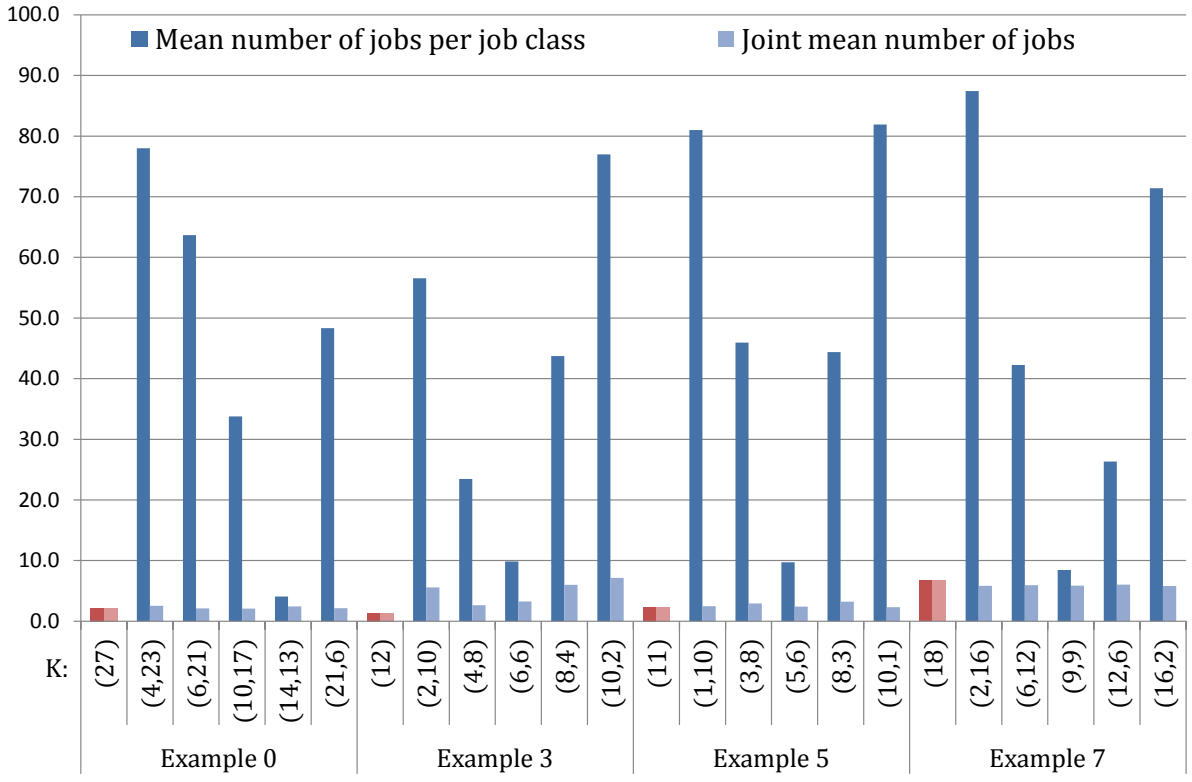


Figure 5.1: Mean deviations for multi-class queueing networks with solely single server nodes.

population vector. The mean deviations of the corresponding single-class examples are shown as well. The population vectors $\mathbf{K}$ are varied. The dark-colored bars indicate the mean deviations of the joint mean number of jobs $\bar{k}_i$, and the light-colored bars correspond to the deviations of the mean number of jobs $\bar{k}_{ir}$. The results specified per node are presented in Table A.2 in Appendix A.

As we expected, the deviations of the joint mean number of jobs are similar to the single-class networks. However, for Example 3, the deviations of the multi-class networks are slightly higher than the deviations of the single-class networks. We conclude from Table A.2 that all values of the $\bar{k}_i$ are identical for the single- and multi-class networks, except for the networks belonging to Example 3. Note that Example 3 is the only evaluated example for which the multi-class network has widely various values for the mean number of visits e_{ir}. Future research should determine whether this is the cause of the larger deviations.

Furthermore, the deviations of the separate mean number of jobs $\bar{k}_{ir}$ are very high for population vectors with large values of $|\mathbf{K}_1 - \mathbf{K}_2|$. The mean deviation for the mean number of jobs per job class $\bar{k}_{ir}$ equals 46.9%. The 20% of the examples with the highest mean deviations, has a mean deviation of 82.1%. We conclude that the MVABLO-m is not suitable for determining the mean number of jobs $\bar{k}_{ir}$ for queueing networks with multiple job classes and class changes. Remember the mapping proposed by Reiser and Kobayashi [1975] to map a multi-class queueing network allowing class changes, trivially into a multi-class model without class changes, see Section 5.1. Future research should determine whether the use of this mapping causes these large deviations, or that large deviations occor as well for queueing networks without class changes.

5.4.3 Single-class networks with multiple server nodes

The MVABLO-m algorithm is evaluated for single-class queueing networks under Assumption 5.10, with both single and multiple server nodes. The evaluation is performed using the same four examples from Table 5.2 as in the previous section, which are now extend to single-class networks with multiple server nodes. For these examples, the number of servers is raised for some nodes of the network, keeping the same node capacities and population numbers. The MVABLO-m results and the simulation results for the single-class networks with both single and multiple server nodes are presented in Table A.3 in Appendix A. The MVABLO-m results are evaluated using the deviations δ(MVABLO-m) with the simulation results as a benchmark. The simulations are executed over a time horizon of 100,000. The mean sojourn time and the mean number of jobs are taken into account.

We continue this section by discussing the results presented in Figure 5.2. This figure shows the mean deviations of the mean sojourn time and the mean number of jobs, separated per evaluated server vector $m := (m_1, \ldots, m_N)$. The exact computed values for each separate variable can be found in Table A.3. For each example, all parameters are as given in Table 5.2, except for the number of servers in each node. These are presented by the server vector, see Figure 5.2.

The serially switched network belonging to Example 0 is studied with population number $\mathbf{K}$

equal to 27. Remember that the node capacities equal $c = (12, 10, 14)$. For the single server case, the mean deviations equal 3.2% and 2.2% (with a mean of 2.7%) for the mean sojourn times $\bar{t}_i$ and the mean number of jobs $\bar{k}_i$ respectively. For the multiple server cases, the mean deviations equal 10.7% and 9.9% (with a mean of 10.3%) respectively. Example 0 indicates that the MVABLO-m algorithm can be more than three times less accurate for multiple server node networks, then for single server node networks. Condider the cases of Example 0 for which solely the number of servers in node 1 is raised (this is the node with the lowest mean service time). The mean deviations increase to 27.7%. This can be explained by the computed values of the $\bar{b}_i$. For the single server case, the mean blocking times are given by $\bar{b}_1 = 0.00$, $\bar{b}_2 (= \bar{b}_{23}) = 0.67$ and $\bar{b}_3 (= \bar{b}_{31}) = 10.00$. When m_1 is raised, we expect that the mean blocking time of node 3 decreases, since the departure rate of node 1 increases. For the simulation, this is actually the case, namely no blocking occurs in nodes 2 and 3. However, the MVABLO-m detects blocking in node 3. Furthermore, after raising m_1, the simulation shows that blocking occurs in node 1. The MVABLO-m does detect this blocking, but the values of the $\bar{b}_1$ are approximated much higher than the simulation results. The mean blocking time $\bar{b}_{12}$ is overcompensated. Raising all of Example 0 m_i simultaneously, the deviations increase as well. However, these deviation are much lower than for the other cases. The highest deviation equals 8.0%, and for the situation with server vector $m = (8, 8, 8)$ and $c = m$ the mean deviations are rarely low. Simulation shows that almost no blocking occurs for these situations, which explains these low deviations. The MVABLO-m detects small amounts of blocking. This follows as well from the definition of the $\bar{b}_{ij}$ shown in equation 5.20, where is divided through by the m_i. Hence, the mean blocking times are overcompensated by a factor m_i smaller amount.

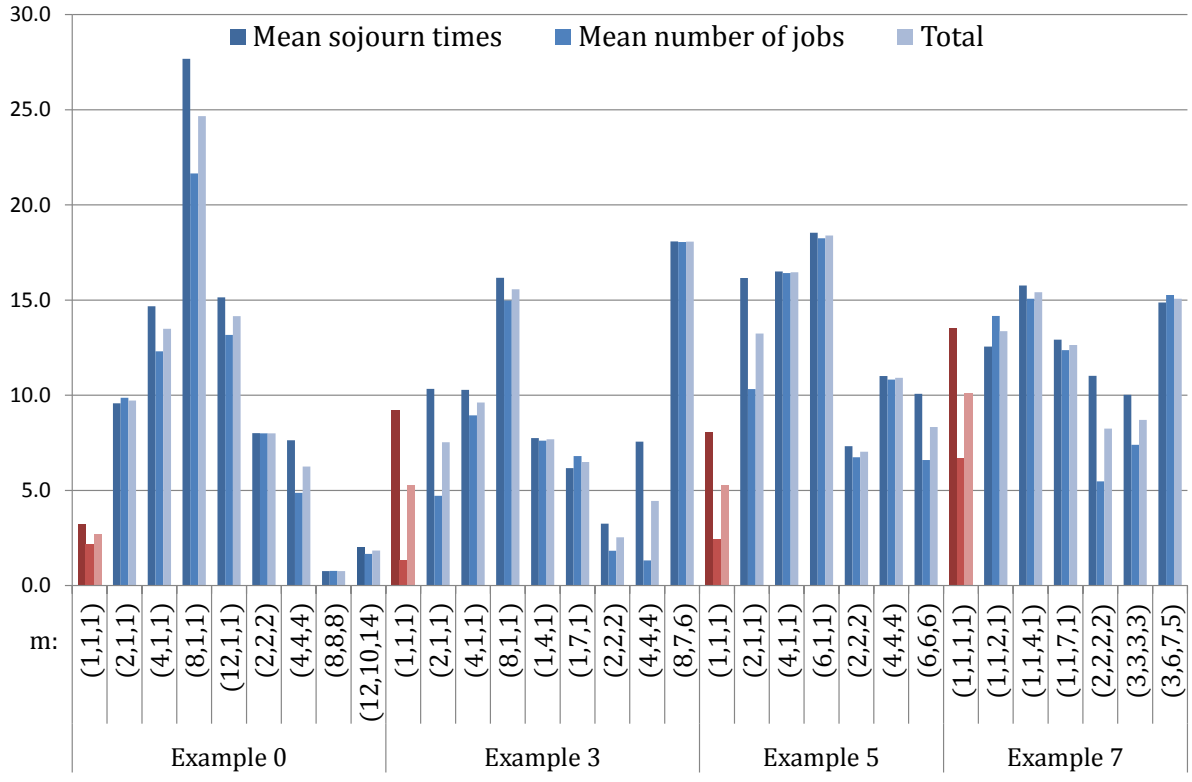


Figure 5.2: Mean deviations for queueing networks with multiple server nodes.

The central server network belonging to Example 3 is studied with population number $\mathbf{K}$ equal to 12, and the serially switched network belonging to Example 7 is studied with population number $\mathbf{K}$ equal to 18. Remember that for Example 3, the node capacities equal $c = (8, 7, 6)$. For the single server case, the mean deviation equals 9.2% and 1.3% (with a mean of 5.3%) for the mean sojourn time $\bar{t}_i$ and the mean number of jobs $\bar{k}_i$ respectively. For the multiple server cases, the mean deviations equal 9.9% and 8.0% (with a mean of 9.0%) respectively. The node capacities for the network in Example 7 equal $c = (3, 6, 7, 5)$. For the single server case, the mean deviation equals 13.5% and 6.7% (with a mean of 10.1%) for the mean sojourn time $\bar{t}_i$ and the mean number of jobs $\bar{k}_i$, respectively. For the multiple server cases, the mean deviations equal 12.9% and 11.6% (with a mean of 12.2%), respectively. Only the deviations obtained for the server vector m equal to (1,1,8) or (8,7,6) in Example 3 are quite larger than for the single server case. Furthermore, for Example 7 with m equal to (1,1,7,1) we see from Table A.3 that the mean blocking time $\bar{b}_3 = 19.20$ is very high. However, the mean deviations are similar as for the single server case. Examples 3 and 7 suggest that the MVABLO-m has a similar accuracy for single-class queueing networks with solely single server nodes, and single-class queueing networks with both single and multiple server nodes.

The complete network belonging to Example 5 is studied with population number $\mathbf{K}$ equal to 11. Remember that the node capacities equal $c = (6, 6, 6)$. For the single server case, the mean deviations equal 8.1% and 2.4% (with a mean of 5.2%) for the mean sojourn time $\bar{t}_i$, and the mean number of jobs $\bar{k}_i$ respectively. For the multiple server cases, the mean deviations equal 13.3% and 11.5% (with a mean of 12.4%) respectively. Hence, Example 5 indicates that the MVABLO-m is more than two times less accurate for single-class queueing networks with both single and multiple server nodes, in comparison with single-class queueing networks with solely single server nodes. As in Example 0, the deviations mainly increase when the number of servers is raised for solely one node of the network. If all m_i are raised simultaneously, the MVABLO-m algorithm computes the mean blocking times with the same accuracy as for the single server nodes case.

Summarizing, the MVABLO-m algorithm shows various accuracies for multiple server node networks. The deviations can be more than two times or even three times less accurate than for single server node networks. Furthermore, we obtained that the MVABLO-m can have a similar accuracy for single-class queueing networks with solely single server nodes, and single-class queueing networks with both single and multiple server nodes. The deviations mainly increase for networks where the number of servers is higher than one for solely one node of the network. If the number of servers is raised for all nodes of a network simultaneously, then the MVABLO-m algorithm computes the mean blocking times with the same accuracy as for the single server nodes case.

5.4.4 Results for the evaluation of the MVABLO-m algorithm

The MVABLO-m is evaluated for different queueing networks under Assumption 5.10. The evaluation is divided over three types of networks: single-class networks with solely single server nodes; multi-class networks with solely single server nodes; and single-class networks with both

single and multiple server nodes. For single-class networks with solely single server nodes, the MVABLO-m algorithm performs similar to the MVABLO algorithm. The mean deviations are approximately 10%. For multi-class networks with solely single server nodes, the deviations of the joint mean number of jobs are slightly higher than the deviations in single-class networks. However, we showed that the MVABLO-m algorithm is not suitable for determining the mean number of jobs per job class, for multi-class queueing networks where class changes are allowed. The mean deviations for such networks are estimated at about 80%. The single-class networks with both single and multiple server nodes showed various results. For some examples we concluded that the MVABLO-m algorithm performs more than three times less accurate, in comparison to the MVABLO-m algorithm used for single server node networks. This results into mean deviations of approximately 30%. However, some examples showed similar results, in comparison to single server node networks. The deviations mainly increase for networks where solely one node of the network has multiple servers. If the number of servers is raised for all nodes simultaneously, the MVABLO-m shows the same accuracy as for the single server nodes case. We discuss the suitability of the MVABLO-m algorithm for the proposed queueing model in the next section of this chapter.

5.5 Suitability of the MVABLO-m algorithm regarding the proposed model

The MVABLO-m algorithm is evaluated in the previous section. It shows a deviation of 80% for multi-class networks where class changes are allowed. For single-class networks with solely one multiple server node, the MVABLO-m algorithm showes a deviation of 30%. Firstly, the proposed queueing model contains both multiple server nodes and single server nodes. Secondly, class changes are allowed. Besides, the model allows recirculation blocking, priority properties, and class dependent service time distributions, unlike the MVABLO-m algorithm. Therefore, it can be said that the MVABLO-m algorithm is not suitable for analyzing the model of the chain conveyor. Therefore, we stagnate our research concerning MVA algorithms here, and choose not to investigate MVA algorithms that support recirculation blocking, priorities, or class dependent service time distributions. We will analyze the proposed model using simulations.

In addition, note that the queueing network modeling the chain conveyor contains general service times and recirculation blocking. Note that, in order to ignore recirculation blocking for analyzing the proposed model, the routing probabilities of the model should be adjusted as follows. Let j be a finite capacity node with recirculation blocking. A job that arrives at full node j will jump towards node $\varphi(j)$. The blocking probability BP_j defined in (4.2) equals the probability that an arriving job at node j will jump towards node $\varphi(j)$ instead. These blocking probabilities could be used to adjust the routing probabilities of the proposed queueing model in order to analyze the model ignoring recirculation blocking. Furthermore, instead of general service time distributions, exponential distributions should be implemented.

5.6 Conclusions

In this chapter we proposed a modification of the MVABLO to fit the proposed network that models the chain conveyor. Firstly, the MVA proposed by Reiser and Lavenberg [1980] for analyzing multi-class networks with solely infinite capacity nodes is adapted. Thereby, we used a mapping proposed by Reiser and Kobayashi [1975] in order to allow class changes. This MVA algorithm is extended for queueing networks with finite capacity nodes with transfer blocking, using the MVABLO proposed by Akyildiz [1988]. Akyildiz studied single-class networks with solely single server nodes. In this study, the MVABLO algorithm is extended into the MVABLO-m algorithm, for which multi-class networks with multiple server nodes are allowed. For single-class networks with solely single server nodes, the MVABLO-m algorithm performs similar to the MVABLO algorithm. The mean deviations are approximately 10%. For multi-class networks with solely single server nodes, the deviations of the joint mean number of jobs are slightly higher than the deviations in single-class networks. However, the MVABLO-m algorithm is not suitable for determining the mean number of jobs per job class, for multi-class queueing networks where class changes are allowed. The mean deviations for such networks can raise up to 80%. For single-class networks with both single and multiple server nodes, the MVABLO-m showed various results. The mean deviation raise up to about 30%. The deviations are mainly high for networks where solely one node of the network has multiple servers. We chose not to use mean value analysis algorithms for analyzing the proposed queueing model. Instead, we will use simulations. The parameters of the queueing network modeling the chain conveyor are given in the next chapter.

Part III

Case study

CHAPTER 6

Data analysis

In order to analyze the chain conveyor, all parameters of the proposed model are determined in this chapter. At the start of this study, there were no available data sets ready for use. Working together with the managers of the warehouse management system MLS, data sets were defined and obtained from MLS. However, not all necessary data was available. Fortunately, Hollander is implementing a new software package named Qlikview. Using Qlikview, we were able to replenish more necessary data from MLS. The last data gaps were filled by performing some measurements in the DC.

Section 6.1 contains the description of the used data sets. All parameters of the proposed model will be determined in Section 6.2. Section 6.3 contains the definition of several scenarios, for which the optimal values of the number of carriers in the conveyor system will be researched. A summary is given in Section 6.4.

6.1 Data sets

As described above, two different data sets are used. First, the Pick Documents, obtained from the warehouse management system MLS over the period from the 19th of March till the 4th of May 2012. Since the DC at Hollander is in operation 6 days per week (not on Sundays), the total number of days in this period equals 42. Secondly, data is taken from MLS using Qlikview. This data is gained from the 19th till the 21th of March 2012.

The examples in the Pick Documents are referred to as pick docs. Each pick doc corresponds to one order that is picked and loaded onto one carrier. Amongst others, the Pick Documents include the following information.

Employee number
: The unique number assigned the order picker that picked the order corresponding to the pick doc.

Work area
: The store region where the order corresponding to the pick doc is picked: MLT, KAAS, VRS, BZ, AGF or AGF AP.

Relation number

The unique number corresponding to the Plus customer for whom the order is picked. Each Plus customer belongs to one or more fixed delivery routes. Each delivery route is assigned to exactly one of the diverters in the expedition region. Hence, the assigned diverter corresponding to the loaded carrier can be determined.

Departure date & time

The date and time the carrier corresponding to the pick doc has to be delivered at the Plus customer.

See Table B.2 in Appendix B for examples of the pick docs.

Each example in the Qlikview data corresponds to one customer order that can consist of multiple pick orders. For example, one Plus customer orders an amount of AGF products that does not fit onto one carrier. Then, the example consists of more than one pick orders. Note that customer orders do not contain products from different work areas. The data set from Qlikview contains the following information.

Number of carriers

The total number of carriers belonging to the same customer order.

Work area

The store region where the order corresponding to the pick doc is picked: MLT, KAAS, VRS, BZ or AGF.

Date & time first pick

Date and time that the first item is picked onto the carrier.

Locations first pick

Exact location of the first picked item, consisting of a lane, column and position number.

Date & time last pick

Date and time that the last item is picked onto the carrier.

Location last pick

Exact location of the last picked item, consisting of a lane, column and position number.

Mutation date

Date that the order is completed in the warehouse management system used by Hollander.

See Table B.3 in Appendix B for examples of the Qlikview data.

To complete the available data, some measurements are performed at the DC. Firstly, the speed of the chain conveyor is measured by the management of Hollander. They took the average over 32 time measures, taken over different distances along the chain conveyor, with an accuracy of 0.1 seconds. Secondly, we obtained the lengths of the chain conveyor from blue prints, and verified these by measurements. It turned out that the lengths at the blue prints were a factor 0.5 less than the real lengths. Finally, we measured the service times at the diverters. This will be explained in Section 6.2.1.

6.2 Determination of the parameters for the proposed model

In this section we will determine all parameters necessary for modeling the chain conveyor. Firstly, some general parameters of the chain conveyor system are described. In Section 6.2.1 the parameters of the diverters are discussed, including a description of the data gained by measurements. In Sections 6.2.3, 6.2.2, and 6.2.4 the parameters of the transfer node, load nodes and the conveyor nodes are discussed, respectively.

The speed of the chain conveyor in the DC at Hollander equals 23.73 meter per minute. The distance between two holes in the chain equals 2.134 meters. The total length of the chain conveyor (including the diverters and the transfer) equals 1352.0 meters. Hence, the conveyor system can store about $1352.0/2.134 = 633$ carriers.

The products stored at the DC at Hollander are subdivided over six different store regions, as shown in Figure 1.1. However, the Qlikview data set distinguishes only five different work areas, namely the AGF and the AGF AP regions are merged together. Therefore, the queueing network model describing the chain conveyor contains five different load nodes instead of six. Furthermore, we have four diverter nodes, one transfer node and eleven conveyor nodes. As shown in Figure 4.2, the proposed model consists of $N = 21$ nodes. Remember that the set of job classes $\mathbf{R}$ consists of three merged sets of job classes: the set $\mathbf{R}_0$ with the one job class corresponding to the empty carriers; the set $\mathbf{R}_1$ with four job classes, corresponding to loaded carriers that are assigned to one of the four diverters; and finally $\mathbf{R}_2$, the set of four extra job classes, each corresponding to one of the job classes in $\mathbf{R}_1$. The extra job classes are introduced to model priority properties and class dependent service time distributions. Summarizing, the model contains $R = 9$ job classes.

This section is continued by determining the number of servers, the node capacity and the service time distributions for each node in the network. The parameters are determined separately for the four different types of nodes in the proposed queueing network (the diverter nodes, transfer node, load nodes, and conveyor nodes).

6.2.1 Parameters for the diverters

The diverters are modeled as diverter nodes 1, 2, 3 and 4 of the proposed queueing network, see Figure 4.2. The diverter are $-/G/m/c$ service centers. In the current situation, there are two operators working at each diverter. Each diverter can store six carriers. Hence, for all diverter nodes i, the number of servers m_i equals 2 and the node capacity c_i equals 8. The remainder of this section is used to determine the service time distributions at the diverters. Two different scenario's are defined, namely before and after the implementation of automatic labeling.

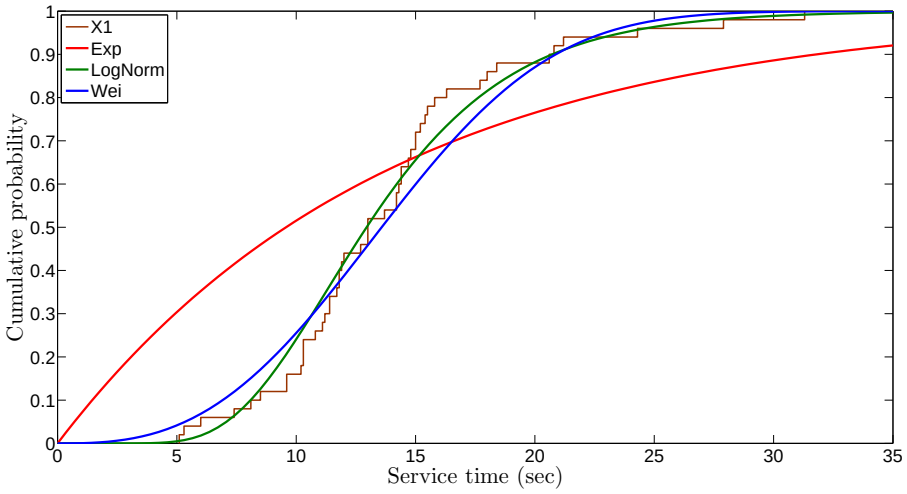
To determine the distribution of the service times at the diverter nodes, some measurements are done at the DC. We assumed that the service time distribution is the same for all four diverters. Service of one carrier at a diverter covers decoupling the carrier, label the carrier,

and set it up for transport. The floor managers of the expedition area require the operators at the diverters to handle solely one carrier at a time. However, they commonly choose to handle multiple carriers simultaneously. Firstly, they decouple a number of carriers, and then move two at a time towards the label printer. Right after they stored a number of carriers in front of the printer, they scan all carriers so that all labels are printed. Then they gain the labels from the printer and attach these to the carriers. After each carrier got a label, they move one or two carriers at a time and set them ready for transport. Since they commonly do not handle one carrier at a time, it is hard to gain accurate measures of the total service time per carrier. Therefore, we choose to split the measures into three time intervals and add these durations later.

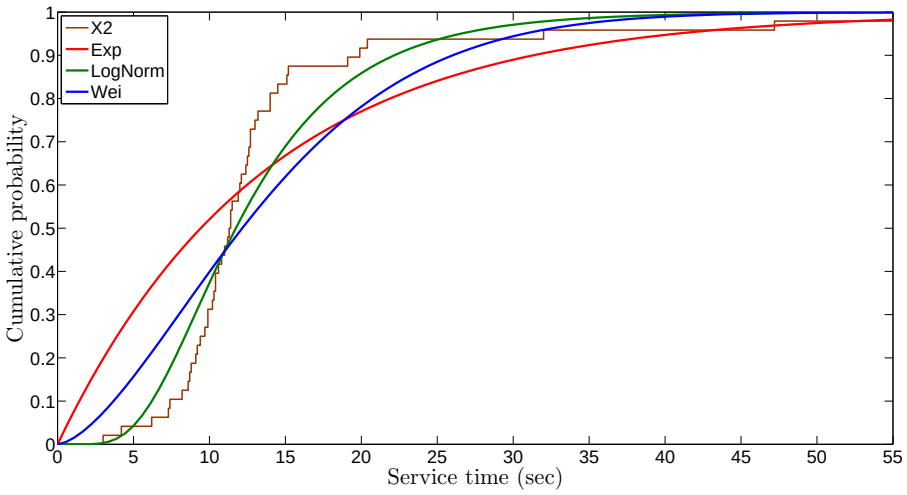
In Table B.1 of Appendix B all measurements are presented. The measurements were made during the night of August the 30th and August the 31st, divided into two periods: period 1 from 21:00h to 23:00h, and period 2 from 01:00h to 03:00h. These times correspond to some of the most busy hours of the day, at one of the less busy days of the week. Since operators attend to work harder when observed, we can assume these measurements hold for the most busy hours of the day during the most busy days of the week. Each row in the table contains three or less measured time intervals. The first interval is the duration of decoupling carriers from the chain conveyor and bring it towards the label printer. The second measure is the time is takes to scan and label the carriers. The third time epoch contains the time it takes to move labeled carriers towards its destination, where they are stored before being transported. This time includes the time to walk back to the diverter. Finally, the number of carriers the operator handled in this time period is noted. The three columns at the right side contain the measured times scaled per carrier.

Define the random variable X_1 as the time duration to decouple and move a carrier towards the printer, X_2 as the time duration to scan and label the carrier, and X_3 as the duration to transport the carrier and walk back towards the diverter. To determine the service time distributions of the diverters for the current situation, the cumulative distribution functions of X_1, X_2 and X_3 are determined, using the times presented in the most right three columns of Table B.1. Below, the cumulative distribution function of the random variable $Y := X_1+X_2+X_3$ will be determined. The random variable Y represents the service time of the diverters in the current situation. The `dfittool` function in MATLAB is used to determine the best fits to the measured times of X_1, X_2 and X_3, see Figure 6.1. For all three random variables, the LogNormal distribution shows the best fit, see Table 6.1a for the detailed parameters.

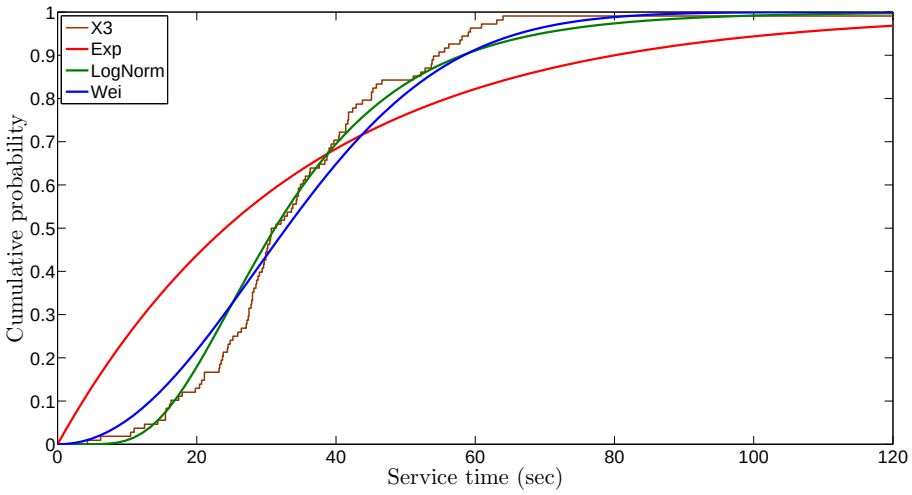
Define Y_{fut} as the random variable representing the service time of the diverters in future situation, after implementation of automatic labeling. In future situation, the carriers are labeled before they enter one of the diverters. Therefore, the time duration of labeling a carrier can be ignored for this situation. Furthermore, the carriers have to travel a shorter distance between the diverter and the transport destination. Another time consuming factor in this process is bringing the carrier to a standstill and bringing it back into motion before and after labeling. This is time consuming, since the weight of one carrier can rise to 300 kg. Hence, the sum of variables X_1 and X_3 probably gives a too high estimation of the service times in future situation. However, in the current situation labeling the carrier is a physical break for



(a) Fit distribution to X_1.

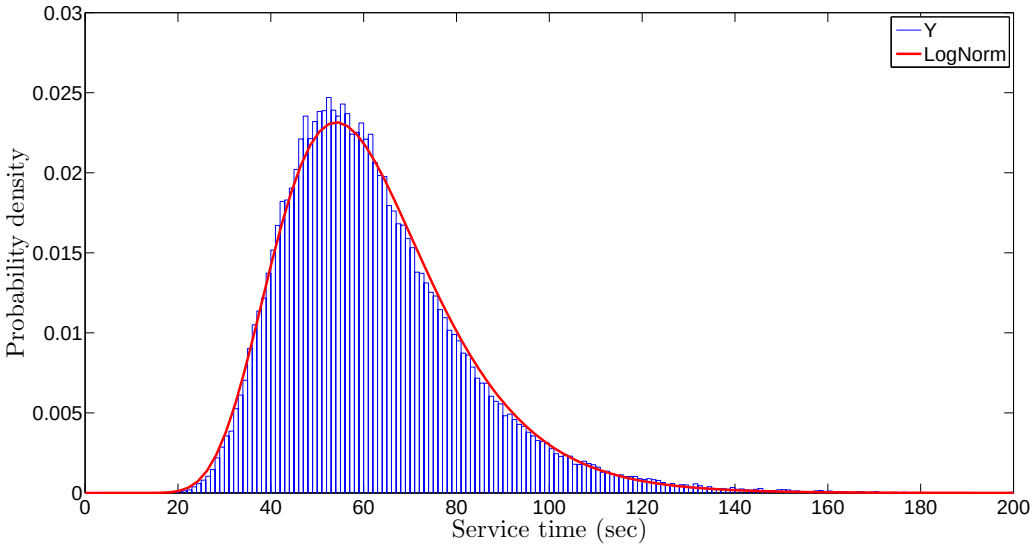


(b) Fit distribution to X_2.

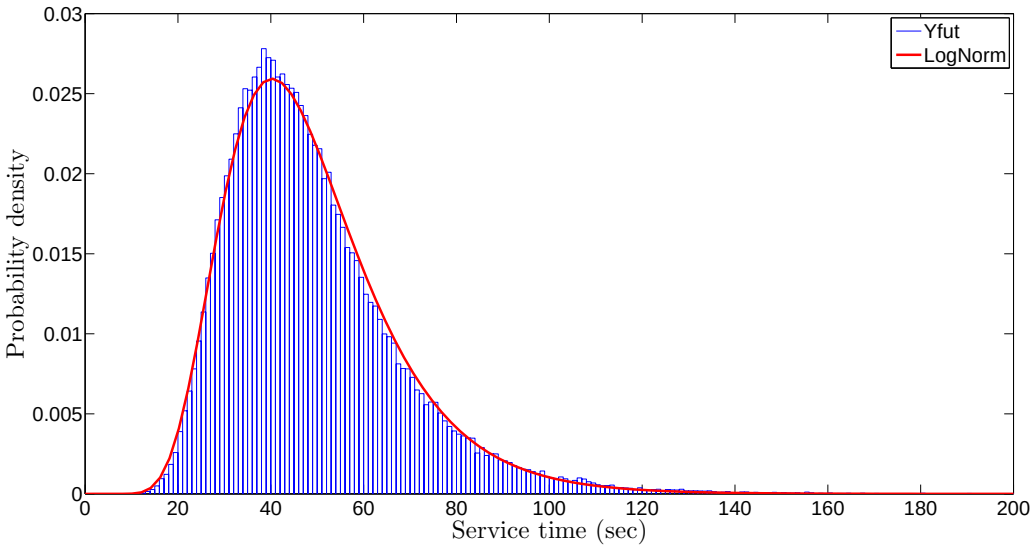


(c) Fit distribution to X_3.

Figure 6.1: Fit distribution of the service times at the diverter nodes.



(a) Fit distribution to the service time at a diverter in the current situation Y.



(b) Fit distribution to the service time at a diverter in the future situation Y_{fut}.

Figure 6.2: Fit of distributions.

Table 6.1: Fits of the LogNormal distribution to the service times in the diverter nodes.

(a) Variables X_1, X_2 and X_3

	X_1	X_2	X_3
Mean	13.8498	13.2746	35.135
Variance	27.6585	49.1886	327.643
Parameter	(μ, σ)	(μ, σ)	(μ, σ)
Estimate	$(2.56, 0.367)$	$(2.46, 0.496)$	$(3.44, 0.485)$
Std. Err. ($\times 10^{-2}$)	$(5.19, 3.73)$	$(7.16, 5.15)$	$(4.67, 3.32)$

(b) Variables $Y = X_1 + X_2 + X_3$ and $Y_{\text{fut}} = X_1 + X_3$

	Y	Y_{fut}
Mean	62.1198	48.7982
Variance	374.126	326.325
Parameter	(μ, σ)	(μ, σ)
Estimate	$(4.08, 0.304)$	$(3.82, 0.358)$
Std. Err. ($\times 10^{-4}$)	$(9.62, 6.80)$	$(11.3, 8.01)$

the operators. Hence, due to the high weights of the carriers we expect that the time duration of moving the carriers will raise without the break during the labeling. We assume that the random variable $Y_{\text{fut}} := X_1 + X_2$ gives a good estimation of the service times in future situation.

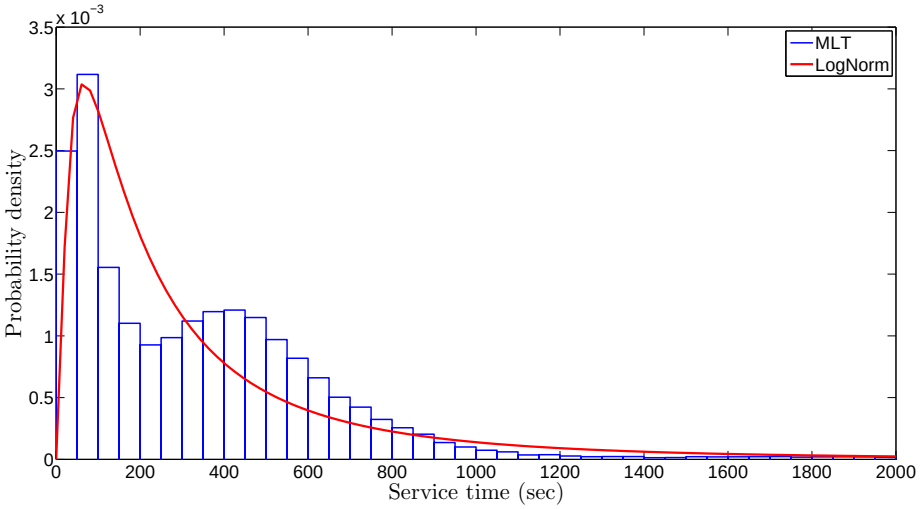
In order to determine the distribution of random variables Y and Y_{fut}, a dataset with values that represent the total service time of one of the diverters is created. We took 100,000 random values of X_1, X_2 and X_3 each, using the LogNormal distribution with the parameters presented in Table 6.1a. Using this created data set and the `dfittool` function in MATLAB, the best fits to both random random variables is determined, namely the LogNormal distribution. See Figures 6.2a and 6.2b for the LogNormal probability density fits of the random variables Y and Y_{fut}, respectively. In Table 6.1b the belonging parameters are presented. The domain of the LogNormal distribution is $(0, \infty)$, and the service times are in seconds.

6.2.2 Parameters for the store regions

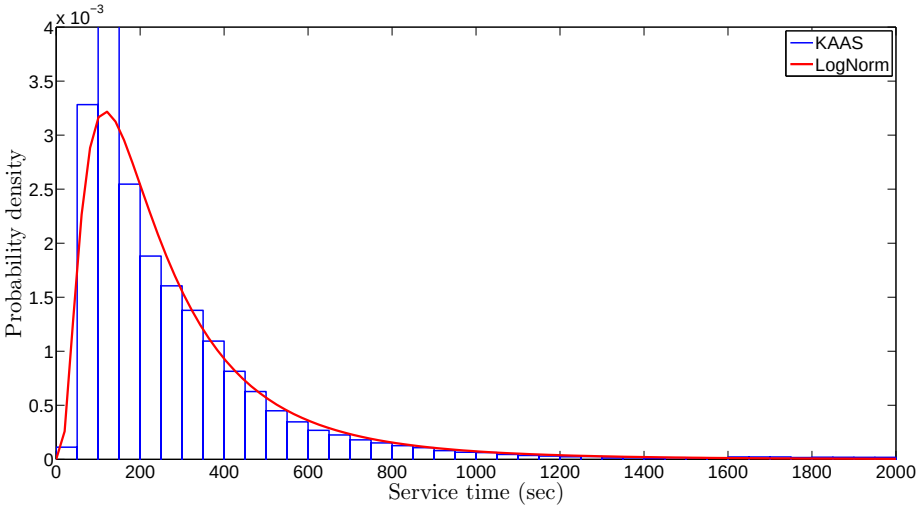
The store regions are modeled as load nodes 5, 6, 7, 8 and 9 of the proposed queueing network, see Figure 4.2. These load nodes represent the store regions MLT, KAAS, VRS, BZ and AGF, respectively, were the load node corresponding to region AGF includes store region AGF AP. The load nodes are $-/G/m/m$ service centers. These nodes are multiple server nodes, where the node capacity equals the number of servers (order pickers) in the corresponding store region. The location of a load node in the network, is the starting point of a load region along the chain conveyor. The number of operators for each store region could not be obtained from the available data. Therefore, the floor manager made an estimation for the number of active order pickers at each store region, see Table 6.2. Since the distribution groups AGF and AGF AP are merged into node 9 of the network, we add the values for these pick regions to obtain the value of m_9.

To model the service times at the load nodes, the service time distribution should be determined for each load node separately. Service at a load node consist of three steps: the proceedings before the actual picking, loading the carrier, and the proceedings after the actual picking. The duration of the middle procedure is referred to as the *pick time*.

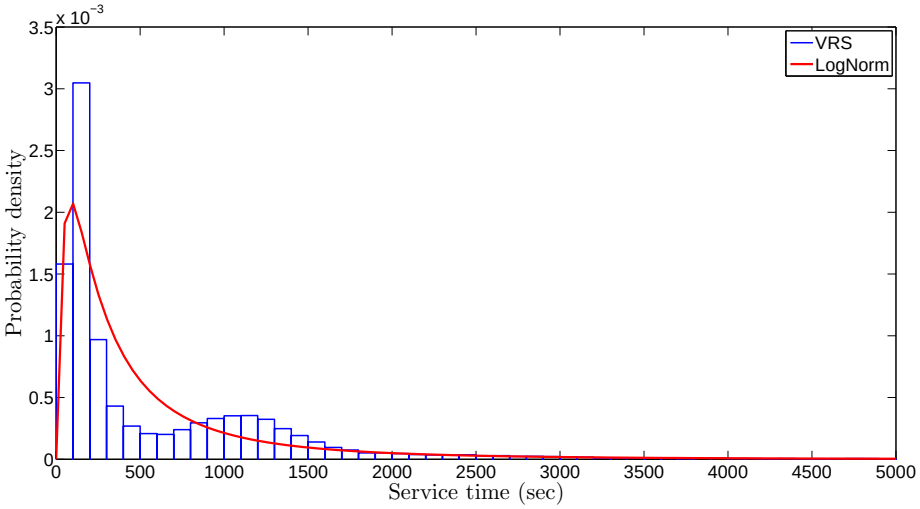
To determine the distribution of the pick time, the Qlikview data set is used. Each example of the Qlikview data set contains both the time of the first pick and the time of the last



(a) Fit distribution to the service times of load node MLT.

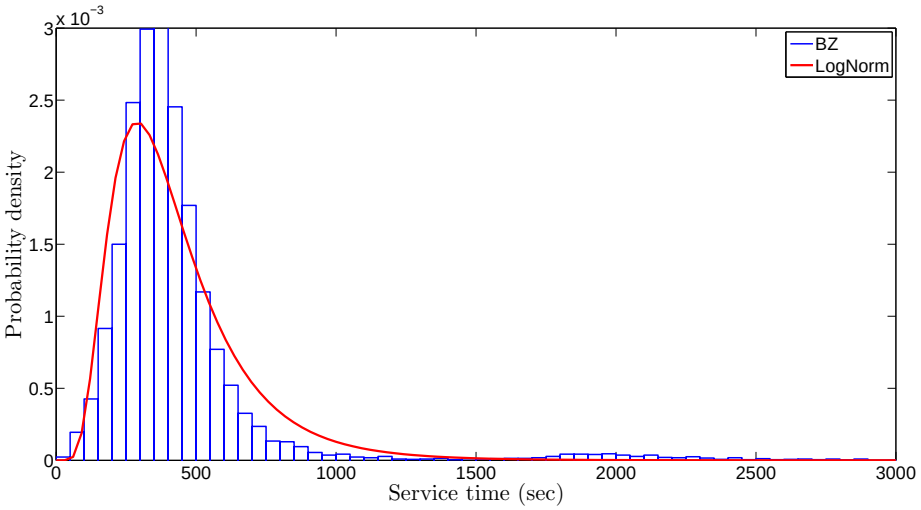


(b) Fit distribution to the service times of load node KAAS.

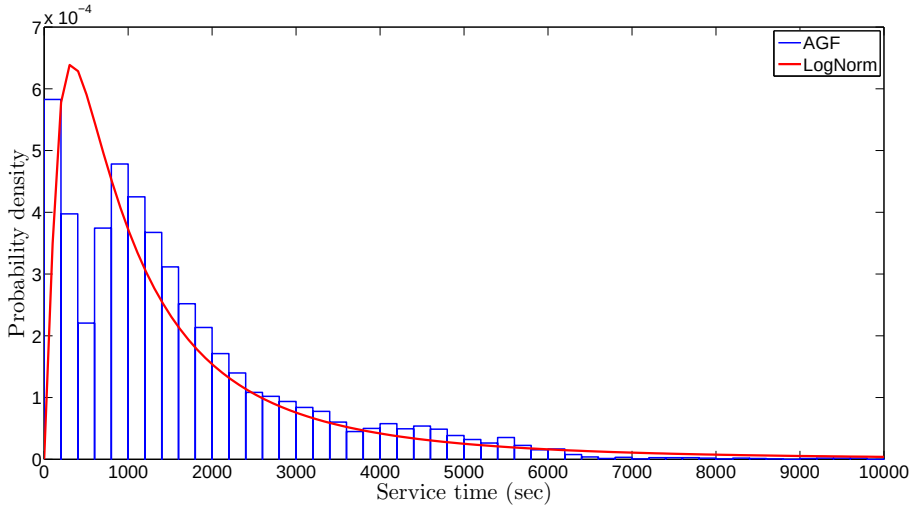


(c) Fit distribution to the service times of load node VRS.

Figure 6.3: Fit of distributions.



(d) Fit distribution to the service times of load node BZ.



(e) Fit distribution to the service times of load node AGF.

Figure 6.3: Fits of the LogNormal distribution.

Table 6.2: Number of order pickers in each store region.

	MLT	KAAS	VRS	BZ	AGF	*Total*
Morning (5:00am - 12:00am)	3-4	6-10	8-13	3-5	6-7	*26-39*
Evening (06:00pm - 04:00am)	5-8	5-8	8-14	3	13-16	*34-49*

Table 6.3: Parameter values of the best fit to measurements in the DC.

	MLT		KAAS		VRS		AGF		*BZ*	
	Before	After	Before	After	Before	After	Before	After	*Before*	*After*
μ	2.86	3.12	3.59	3.92	3.72	4.06	3.52	4.82	3.39	3.83
σ	0.78	0.72	0.55	0.34	0.55	0.56	0.22	0.60	0.70	0.72

Table 6.4: Fits of the LogNormal distribution to the service times in the load nodes.

	MLT	KAAS	VRS	AGF	BZ
Mean	399	292	630	1830	430
Variance	372313	71754.9	1132550	7182630	56234
Parameter	(μ, σ)	(μ, σ)	(μ, σ)	(μ, σ)	(μ, σ)
Estimate	$(5.39, 1.10)$	$(5.370, 0.782)$	$(5.77, 1.16)$	$(6.94, 1.07)$	$(5.93, 0.515)$
Std. Err. ($\times 10^{-3}$)	$(6.26, 4.43)$	$(3.49, 2.47)$	$(6.91, 4.89)$	$(12.1, 8.57)$	$(5.48, 3.88)$

pick. Hence, pick time can be determined. The data is split into five parts, one part for each distribution group. Solely the examples that correspond to an order of one carrier are taken into account. Furthermore, all examples for which the first pick and last pick have different dates are eliminated, since the system cannot handle these examples. The proceedings that are performed before the first pick, consist of decoupling a carrier from the chain conveyor, and transport it manually towards the first pick location. After the last pick, most carriers have to be sealed to keep the products in their desired temperature before they are coupled back onto the chain conveyor. There is no exact data about the time durations of the proceedings before and after loading the carriers. However, two interns at Hollander[1] performed some measures at the DC. Using the `dfittool` in MATLAB is obtained that the best fit to these procedure times is the LogNormal distribution. The belonging parameters are presented in Table 6.3. The measures of value *Before* consists of decoupling a carrier from the chain conveyor and transport it towards the first pick location. The measures of value *After* consists of a count check, stacking the load of multiple carriers onto one carrier (if necessary), seal the carrier (if necessary), transport the carrier and couple it onto the chain conveyor. The fits are in seconds. Since no measures are available for the BZ store region, all available data of the other store regions is used to determine the best fit for BZ.

The distributions presented in Table 6.3 are used to add two random values to each example of the Qlickview data. Using the `dfittool` in MATLAB, the best fit of the service times in all five load nodes is determined. It appears that the best fit for the service times is the LogNormal distribution. See Figure 6.3 and Table 6.4 for the belonging probability density functions and the corresponding parameters, respectively. The domain of the LogNormal distribution is $(0, \infty)$ and the service times are in seconds.

[1] Meeuwis Veldhoen and Eric Versteeg

Table 6.5: Partition of the chain conveyor over the conveyor nodes queueing network model.

Node	From	To	Distance (m)	# carriers	Time (min)
20	diverter 4	transfer	89.6	42	3.776
11	transfer	MLT	189.8	89	7.998
12	first location MLT MLT	last location MLT KAAS	83.9 0.0	39	3.535
13	first location KAAS KAAS	last location KAAS VRS	162.3 0.0	76	6.839
21	first location VRS VRS	last location VRS transfer	93.4 122.4	101	9.094
14	transfer	BZ	159.9	75	6.738
15	first location BZ BZ	last location BZ AGF	34.0 0.0	16	1.433
16	first location AGF AGF	last location AGF diverter 1	196.0 46.5	114	10.219
17	diverter 1	diverter 2	24.0	11	1.011
18	diverter 2	diverter 3	60.2	28	2.537
19	diverter 3	diverter 4	24.0	11	1.011
Total			**1286.0**	**602**	**54.190**

6.2.3 Parameters for the transfer

The transfer is modeled as the transfer node, node 10, of the proposed model, see Figure 4.2. The transfer node is a $-/D/1/c$ service center. Solely one carrier can be coupled back onto the chain conveyor at the time. Therefore, the number of servers m_{10} equals 1. The transfer can store at most 5 carriers. Hence, the node capacity c_{10} equals 5. The service times at the transfer are deterministic and equal to the time it takes for the chain conveyor to travel the distance between two holes in the chain. Hence, the service times equal $2.134/23.72 = 0.09$ minutes, or 5.40 seconds.

6.2.4 Parameters for the conveyor

The different parts of the chain along the track of the conveyor system are modeled as conveyor nodes 11 to 21 of the proposed queueing network, see Figure 4.2. The conveyor nodes are $-/G/m/m$ service centers. We will determine the lengths of the conveyor that each conveyor node in the network covers for. The necessary data is gained by measuring the lengths manually. In Table 6.5, all distances of the conveyor loop are presented. The measured distances from the chain conveyor are presented in the fourth column. We present the maximum number of carriers that can be stored on the part chain conveyor corresponding to the node in the fifth column. Thereby, we used the distance between two carriers. The most right column represents the time it takes for one carrier to travel along this part of the chain conveyor, assuming the conveyor is running at a constant speed. For each conveyor node i, the number of servers m_i (and thus the node capacity c_i) equals the maximum number of carriers.

For all conveyor nodes that are not right after a load node, the service time of the conveyor node is deterministic. These service times equal the time given in Table 6.5. This holds for the conveyor nodes 11, 14, 17, 18, 19 and 20. For the remaining conveyor nodes, the ones that are located right after a load node, the service time is variable. Namely, carriers that arrive

from the foregoing store region are coupled back onto the chain conveyor at a variable place along the belonging pick lane. In Appendix B, two floor plans of the DC are presented. On these floor maps, the lane numbers and the column numbers as given in the Qlikview data are presented. For example, in Figure B.1b it is shown that lane V06 belongs to store region MLT, and that lane V06 has column locations 1 till 24. Consider a carrier that is decoupled from the chain conveyor at the beginning of the MLT pick lane, this is near lane V01. If the last pick location is in lane V06, then the travel time of the carrier along the chain conveyor towards the KAAS region, will probably be longer than if the last pick location is in lane V11. Therefore, the Qlikview data is used to determine a discrete distribution for the service times of nodes 12, 13, 21, 15 and 16. In Table B.4, the possible distances a loaded carrier can travel along a pick lane right after being loaded (according to the Qlikview data) are presented, together with the probability distributions.

The parameter α_{ms} is defined at equation (4.5) for all load nodes m and job classes s in $\mathbf{R}_1$. This is the probability that a class-r job with r in $\mathbf{R}_0$ jumping towards node m will become a class-s job. This probabilities correspond to the probability that a carrier loaded in store region m is assigned to diverter s. The number of carriers that is assigned to each diverter is determined per pick region, using the Pick Documents. In Figure 6.4 we present the corresponding probabilities. Hence, the values of the α_{ms} follow directly from Figure 6.4.

All parameters of the queueing network modeling the chain conveyor are determined. There are two scenarios concerning the service time distributions of the diverter nodes: before and after implementation of automatic labeling. Furthermore, in Table 6.2 it is shown that the number order pickers in the distribution center is variable and has large deviations. Different scenarios will be defined in the next section of this chapter, to cover for most of the possible situations regarding the chain conveyor.

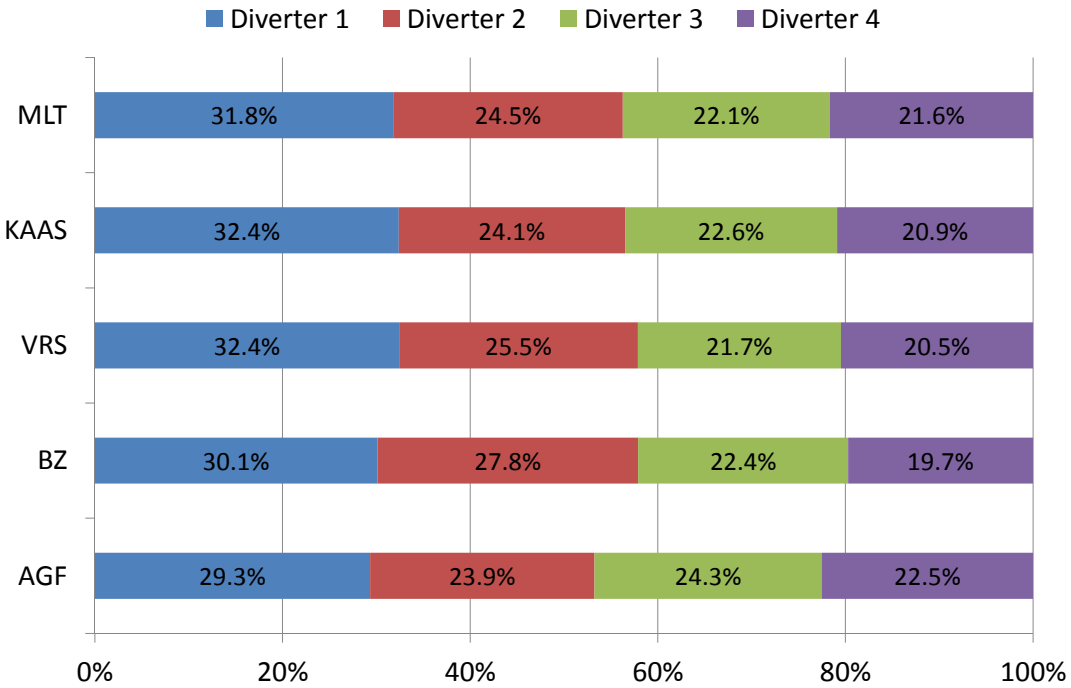


Figure 6.4: Probability distributions of the assigned diverters per store region.

6.3 Scenario definitions

In order to give an advise that is applicable for multiple situations at Hollander, twelve scenarios will be taken into account. Firstly, we distinguish between the morning and the evening shift, since the distribution of order pickers over the store regions differ for these two parts of the day. Secondly, both the current situation and the future situation are taken into account, regarding future implementation of automatic labeling. The service times at the diverters will be decreased for the future scenarios. Summarizing, four main scenarios are defined: morning in current situation, evening in current situation, morning in future situation, and evening in future situation. The total number of order pickers in the DC has large deviations as well. Therefore, three sub-scenarios are taken into account for each main scenario, where the total number of order pickers in the DC is varied. The sub-scenario with the highest number of order pickers is referred to as the most busy scenario. The other sub-scenarios are referred to as the medium and the quiet sub-scenarios, analogously.

The parameters for the proposed queueing model are presented in Table 6.6. For the parameters that are similar for each scenario, see Table 6.6a. For the parameters that vary per scenario, see Table 6.6b. The LogNormal distribution with μ the mean parameter and σ the deviation parameter is referred to as LogN(μ, σ). These parameters are in seconds. Remember that $\mathbf{R}_2$ is the set of extra job classes and $\mathbf{R}\backslash\mathbf{R}_2$ represents the set of basic job classes. Deterministic service times with a duration of λ are referred to as Deterministic(λ). These distributions are in minutes. The discrete service time distributions of conveyor nodes 12, 13, 15, 16 and 21 are specified in Table B.4 of Appendix B.

The optimal value of the total number of carriers in the conveyor system will be determined for each defined scenario, together with the corresponding mean numbers of busy order pickers. The scenario results will be compared, in order to perform a sensitivity analysis regarding the variation of number of order pickers in the DC, the distribution of order pickers over the store regions, and the implementation of automatic labeling.

6.4 Conclusions

We used the Pic Documents data to determine the routing probabilities of class-r jobs with r in $\mathbf{R}_0$ in the conveyor nodes foregoing to the load nodes. Furthermore, we used the Qlikview data to determine the service time distributions at the load nodes, and the discrete service time distributions of some conveyor nodes. Measurements are used to determine the service time distributions of the diverter nodes. Other measurements are used to determine the number of servers at each conveyor node. In order to give an advise that is applicable for Hollander in each situation, twelve scenarios are taken into account. The four main scenarios correspond to the morning and evening shifts, before or after implementation of automatic labeling. Three sub-scenarios are defined for each main scenario, with an increasing total number order pickers in the DC. An overview of all parameters specified per scenario is given in Table 6.6. We will propose a simulation in the next chapter for analyzing the model for the chain conveyor.

Table 6.6: Parameters for the queueing network that models the chain conveyor.
(a) Parameters that are similar for each scenario.

Node i	m_i	c_i	Service time distribution	Description
1	2	8	Specified per scenario	Diverter 1
2	2	8	Specified per scenario	Diverter 2
3	2	8	Specified per scenario	Diverter 3
4	2	8	Specified per scenario	Diverter 4
5	Specified per scenario		LogN(5.39, 1.10)	Start load region MLT
6	Specified per scenario		LogN(5.37, 0.782)	Start load region KAAS
7	Specified per scenario		LogN(5.77, 1.16)	Start load region VRS
8	Specified per scenario		LogN(5.93, 0.515)	Start load region BZ
9	Specified per scenario		LogN(6.94, 1.07)	Start load region AGF
10	1	5	Deterministic(0.090)	Transfer
11	89	89	Deterministic(7.998)	Conveyor between nodes 10 and 5
12	39	39	*Class dependent* $\mathbf{R}_2$: Discrete $\mathbf{R}\backslash\mathbf{R}_2$: Deterministic(3.535)	Conveyor between nodes 5 and 6
13	76	76	*Class dependent* $\mathbf{R}_2$: Discrete $\mathbf{R}\backslash\mathbf{R}_2$: Deterministic(6.839)	Conveyor between nodes 6 and 7
14	75	75	Deterministic(6.738)	Conveyor between nodes 10 and 8
15	16	16	*Class dependent* $\mathbf{R}_2$: Discrete $\mathbf{R}\backslash\mathbf{R}_2$: Deterministic(1.433)	Conveyor between nodes 8 and 9
16	114	114	*Class dependent* $\mathbf{R}_2$: Discrete $\mathbf{R}\backslash\mathbf{R}_2$: Deterministic(10.219)	Conveyor between nodes 9 and 1
17	11	11	Deterministic(1.011)	Conveyor between nodes 1 and 2
18	28	28	Deterministic(2.537)	Conveyor between nodes 2 and 3
19	11	11	Deterministic(1.011)	Conveyor between nodes 3 and 4
20	42	42	Deterministic(3.776)	Conveyor between nodes 4 and 10
21	101	101	*Class dependent* $\mathbf{R}_2$: Discrete $\mathbf{R}\backslash\mathbf{R}_2$: Deterministic(9.094)	Conveyor between nodes 7 and 10

(b) Parameters that vary per scenario.

Scenario:	**Service time distribution diverter nodes** 1, 2, 3, 4	**Load nodes** **MLT** m_5	**KAAS** m_6	**VRS** m_7	**BZ** m_8	**AGF** m_9	*Total number of order pickers*	*Total network capacity*
Scenario 1 corresponds to morning shifts in the current situation								
1.a	LogN(4.08, 0.304)	3	6	8	3	6	26	665
1.b	LogN(4.08, 0.304)	4	8	11	4	6	33	672
1.c	LogN(4.08, 0.304)	4	10	13	5	7	39	678
Scenario 2 corresponds to evening shifts in the current situation								
2.a	LogN(4.08, 0.304)	5	5	8	3	13	34	673
2.b	LogN(4.08, 0.304)	7	7	11	3	14	42	681
2.c	LogN(4.08, 0.304)	8	8	14	3	16	49	688
Scenario 3 corresponds to morning shifts in the future situation								
3.a	LogN(3.82, 0.358)	3	6	8	3	6	26	665
3.b	LogN(3.82, 0.358)	4	8	11	4	6	33	672
3.c	LogN(3.82, 0.358)	4	10	13	5	7	39	678
Scenario 4 corresponds to evening shifts in the future situation								
4.a	LogN(3.82, 0.358)	5	5	8	3	13	34	673
4.b	LogN(3.82, 0.358)	7	7	11	3	14	42	681
4.c	LogN(3.82, 0.358)	8	8	14	3	16	49	688

CHAPTER 7

Simulation of the chain conveyor

Based on the evaluation of the MVABLO-m algorithm we concluded that this algorithm is not suitable for analyzing the model of the chain conveyor. Therefore, we will simulate the proposed model in order to determine the mean number of busy order pickers for each store region in the distribution center at Hollander.

In Section 7.1 we propose the simulation including the determination of the stopping criteria. The simulation will be validated for the chain conveyor in Section 7.2. This chapter is summarized in Section 7.3. In addition, the pseudo code of the simulation is given in Appendix C.

7.1 Description of the simulation

We will optimize the chain conveyor using a binary search algorithm. For each iteration of the algorithm, the mean number of busy order pickers is approximated using the proposed queueing model with a fixed number of jobs in the network. The model is a multi-class, closed queueing network consisting of both single and multiple finite capacity nodes with either transfer or recirculation blocking. All nodes follow a FCFS discipline with possible head-of-line priorities and have general distributed service times which are possibly class dependent. Class changes are allowed. Furthermore, the model satisfies the capacity constraint and adopts the deadlock property. The simulation described in this section allows closed queuing networks such as the proposed model.

The simulation of the chain conveyor needs the following input with regard to the proposed model: the number of nodes, the number of job classes, the number of jobs, the routing probabilities, and for each node its capacity, number of servers, service time distribution, type of blocking, and priority order. The network is initialized by placing all jobs in the class corresponding to an empty carrier at a random place in the smallest directed cycle of solely finite capacity nodes with transfer blocking. This cycle corresponds to the conveyor nodes of the proposed model. For all conveyor nodes i with capacity c_i and deterministic service time λ_i, the initial service times are pairwise distinct positive multiples of λ_i/c_i.

The number of jobs in the network is fixed. However, the distribution of jobs over the job classes varies, since class changes are allowed. For all pairs of one node and one job class, we keep track on the mean number of jobs, the mean sojourn time, and the mean number of visits per time unit. Using the former values, we can determine the distribution of jobs over the classes. Furthermore, for all nodes with transfer blocking, we keep track on the number of blocked servers and the blocking durations per blocked server. These measures are used to determine the mean sojourn times described above. As well, for all nodes with recirculation blocking we keep track on the number of blocked jobs per time unit. Together with the mean number of visits we determine the blocking probability for each node with recirculation blocking.

We use the stopping criteria proposed by Ross [2006, p.p. 121]. The simulation is continued until the standard deviations of the chosen estimators, are smaller than a predefined acceptable value of d. The method can be explained as follows. Consider the sequence of data values $D_1, D_2, \ldots, D_j, \ldots$ where D_j equals the data value at time j. Given a predefined value of d, continue the simulation until the following inequality is satisfied.

$$S_j/\sqrt{j} < d, \tag{7.1}$$

where for $j = 2, 3, 4, \ldots$ the S_j are defined using the recursive formula

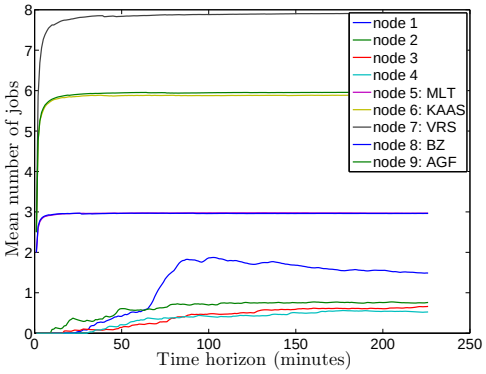
$$S_j{}^2 = \left(\frac{j-2}{j-1}\right) S_{j-1}^2 + j\big(\overline{D}_j - \overline{D}_{j-1}\big)^2.$$

This recursive formula is initialized using $S_1^2 = 0$ and $\overline{D}_0 = 0$. For $j = 1, 2, \ldots$ the $\overline{D}_j$ are defined using

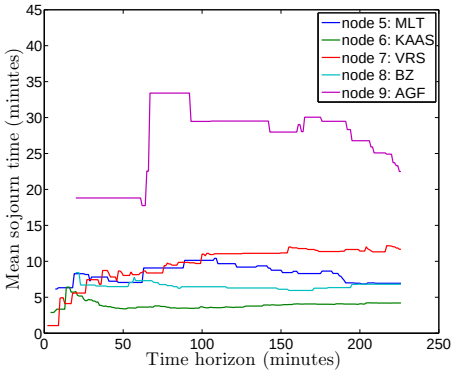
$$\overline{D}_j = \overline{D}_{j-1} + \frac{D_j - \overline{D}_{j-1}}{j}.$$

The following random variables are used as estimators: the mean number of jobs $\bar{k}_i$ for all nodes i in $\mathbf{N}$; the mean sojourn times $\bar{t}_{ir}$ for all diverter nodes i and job classes $r = i$; the mean sojourn times $\bar{t}_{i1}$ for all load nodes and the transfer node i; the mean sojourn times $\bar{t}_{i\psi(1)}$ for all conveyor nodes i; the mean number of class-r jobs in the queueing network $\mathbf{K}_r$ for the job class r in $\mathbf{R}_0$ (empty carriers); and for all job classes r in $\mathbf{R}_1$ the value of the mean number of jobs $\mathbf{K}_r + \mathbf{K}_{\psi(r)}$ in the queueing network. The latter corresponds to the total number of carriers assigned to diverter r, for all diverters r.

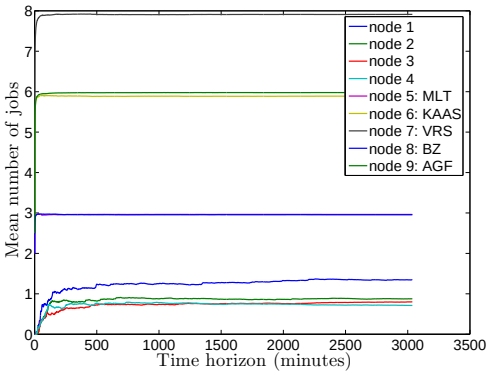
The capacity percentage of a network is an increasing function for increasing number of jobs K in the network, or a decreasing function for increasing network capacity C. Remember the scenario definition specified in Table 6.6. The smallest value of C is obtained for Scenario 1.a. We expect that the standard deviations increase, for increasing the capacity percentage. Therefore, we will predefine the value of d using simulations of Scenario 1.a with a high value of the total number of jobs in the network, $K = 600$. We expect that the standard deviations of most estimators in other scenarios will be less than the predefined value d, due to equilibriums that will be reached for much lower values than the value of d. The predefined value of d is based on a trade-off in lowering the run time of the simulation, and lowering the standard deviations of the estimators simultaneously. Since the network percentage is high for Scenario 1.a, we expect



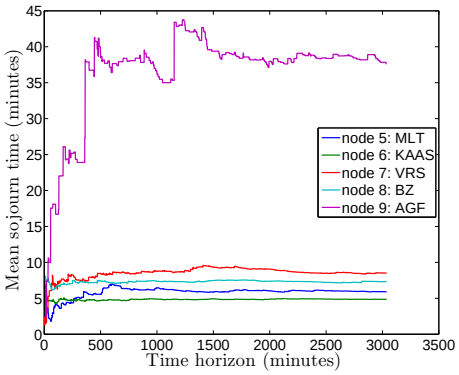
(a) Mean number of jobs for $d = 0.1$.



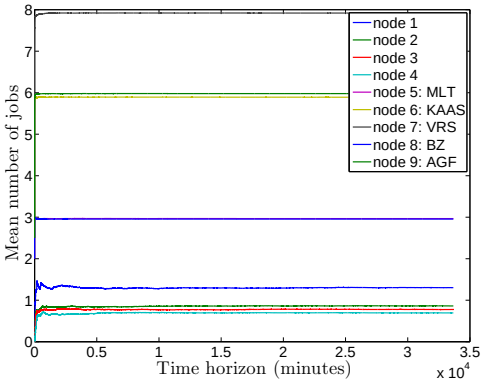
(b) Mean sojourn times for $d = 0.1$.



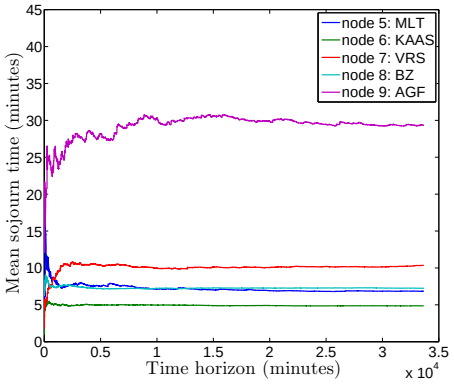
(c) Mean number of jobs for $d = 0.01$.



(d) Mean sojourn times for $d = 0.01$.



(e) Mean number of jobs for $d = 0.001$.



(f) Mean sojourn times for $d = 0.001$.

Figure 7.1: Simulation results of Scenario 1.a with 600 carriers.

Table 7.1: Validation results for the simulation.

		Mean number of carriers that leave the distribution center.					
		Total	MLT	KAAS	VRS	BZ	AGF
Real world	Per day	**3359±67**	431±11	536±27	642±23	369±9	1381±40
	Per hour	**240±4.8**	31±0.8	38±2.0	46±1.6	26±0.7	99±2.8
Simulation	Per hour	**179**	26	72	46	24	11

that the run time of other scenarios will be higher. Scenario 1.a is simulated for values of d equal to 1, 0.1, 0.01 and 0.001. In Figure 7.1, some results are presented for d-values of 0.1, 0.01 and 0.001. The run time for d equal to $0.1, 0.01$ and 0.001 was about 0.3, 3 and 30 minutes respectively. Since the optimization problem given in (2.1) is based on the mean number of jobs in the load nodes, the stability of these estimators is most important. We see from Figure 7.1c that for a value of $d = 0.01$, these estimators reach equilibrium. Besides, the running time of the simulation with this d-value is sufficiently low to be run several times by the Binary Search algorithm described later in this thesis. We validate the simulation for the chain conveyor in the next section of this chapter.

7.2 Validation of the simulation for analyzing the chain conveyor

With regard to the validation of the simulation, the simulation results of Scenario 1.a with K equal to 600 carriers and with a d-value of 0.01 are used. In real world, this scenario could correspond to a Tuesday morning in the current situation of Hollander. Note that in the current situation, the transfer is not in operation. However, for the model is assumed that the transfer is operative. Presently, the full-load strategy is implemented. Therefore, we assume that the order pickers have no idle time in the current situation. The simulation results shown in Figure 7.1c for Scenario 1.a with 600 carriers, indicate no idle time for the order pickers neither. Therefore, we assume that this scenario corresponds to the current situation at Hollander.

In Table 7.1, the mean number of carriers that leave the DC per day in the real world situation are presented with a 95% confidence interval. These values are based on all Tuesdays of the year 2011. Assuming that the chain conveyor is operative 14 hours per day, we presented the mean number of carriers that leave the DC per hour as well with a 95% confidence interval. For the simulation of the chain conveyor, the departure of jobs in the load nodes are counted and also presented in Table 7.1. Assuming that all jobs departing from the load nodes will be served at one of the diverter nodes, these values correspond to the carriers that leave the DC. For all of the MLT, VRS and BZ store regions, the simulated mean number of carriers that leave the DC each hour is close to the real world situation. For the KAAS and the AGF store regions, high inaccuracies are obtained. Both inaccuracies can be explained by the heavy variation of the number of order pickers in the store regions between the morning and evening shifts. For simulated Scenario 1.a, the AGF region contains 6 order pickers, which results in a mean number of 11 carriers per hour. In real world situation, AGF handles 99 carriers per hour, averaged over the total day (taking into account that the chain conveyor is operative 14

hours per day). During the evening shift, the number of order pickers can rise up to 16. Since the mean number of carriers is taken over the total day, we can assume that this difference is not caused by inaccuracies of the simulation. The difference from the real world for the KAAS region can be explained analogously. Furthermore, note that for the remaining store regions the simulated mean number of carriers leaving the distribution center is lower than the real values. Stagnation causes increased throughputs for the conveyor system. The difference in number of carriers leaving the DC can be explained by this increase of the throughput.

Based on the validation described above, we state that the simulation results will be close to the real world situation. Future research should determine the sensitivity of the model with regard to the throughput of the network.

7.3 Conclusions

We proposed a simulation for analyzing the model of the chain conveyor. The simulation allows multi-class, closed queueing networks with class changes, consisting of both single server and multiple server nodes following a FCFS discipline. Furthermore, class dependent, general distributed service times, finite capacity nodes with either transfer blocking or recirculation blocking, and head-of-line priorities are allowed. Simulated networks should satisfy the capacity constraint and adopt the deadlock property. The simulation will continue until the standard deviations of the chosen estimators are all smaller than $d = 0.01$. We validated the simulation for the chain conveyor using the mean number of carriers that leave the distribution center. The simulation results will be close to the real world situation. In the next chapter, we will use the simulation to analyze the proposed model in order to optimize the number of carriers stored at the chain conveyor.

CHAPTER 8

Results for optimizing the chain conveyor

The problem of improving the continuity of the conveyor chain at Hollander is formulated as an integer program, which will be solved using a binary search. For each iteration of the binary search, the chain conveyor will be modeled using the proposed queueing network, and analyzed using the proposed simulation. We will investigate the optimal number of carriers stored on the chain conveyor, decide whether the transfer can be put in operation, and discuss the possibility of lowering the number of operators at the diverters.

In Section 8.1 we describe the Binary Search algorithm that will be used for solving the integer program. In Section 8.2 the chain conveyor is optimized for all scenarios defined in Section 6.3. The effects of putting the transfer in operation are discussed in Section 8.3, and Section 8.4 contains the results of six extra scenarios, for which implementation of automatic labeling is assumed and solely one operator is installed at each diverter. The results are summarized in Section 8.5.

8.1 Binary Search algorithm for solving the integer program

Remember the integer program corresponding to the optimization of the number of carriers stored on the chain conveyor:

$$\begin{aligned} &\textit{Minimize} && K \\ &\textit{Subject to} && \bar{k}_i(K) \geq M \cdot m_i, && \forall\, i \in \mathbf{N}_l, \\ &&& K \in \{0, 1, \ldots, C\}, \end{aligned} \tag{2.1}$$

where K equals the number of carriers in the conveyor system, $\mathbf{N}_l$ is the set of store regions in the DC, $\bar{k}_i(K)$ is the mean number of busy order pickers at store region i with K carriers in the conveyor system, m_i is the number of order pickers at store region i, and C is the maximum number of carriers that the conveyor system can handle. The parameter M corresponds to the percentage of required mean number of busy order pickers per time unit per store region and is assumed to be 95%. Note that the integer program consists of solely one decision variable,

namely the number of carriers K. We will solve the proposed integer program using the Binary Search algorithm given below, which iterates over the possible values of the decision variable.

Algorithm 8.1 (Binary search) This algorithm solves integer programs as formulated above. Given two integers a and b with $0 \leq a \leq b \leq C$, define the interval $I = [a, b]$. Furthermore, define the length of that interval by $|I| = b - a + 1$.

Initialization

$a \longleftarrow 0$;
$b \longleftarrow C$;
$K^* \longleftarrow C$;
$K \longleftarrow a + \lfloor \frac{1}{2}|I| \rfloor$;

Main iteration

While $K \neq K^*$
  1. Compute the mean number busy order pickers $\bar{k}_i(\mathbf{K})$ in all store regions i.
  If $\bar{k}_i(K) \geq M \cdot m_i$ for store regions i
    `% Mean number of required busy order pickers at store regions satisfied`
    2. $b \longleftarrow K$;
    3. $K^* \longleftarrow K$;
  Else
    `% Mean number of required busy order pickers at store regions not satisfied`
    4. $a \longleftarrow K + 1$;
  end
  5. $K \longleftarrow a + \lfloor \frac{1}{2}|I| \rfloor$;
end

The major advantage is that the Binary Search algorithm has few iterations. The number of possible solutions equals $C+1$. With each iteration we eliminate half of the solutions. Hence, for the second iteration, the number of possible solutions is less than or equal to $\lceil \frac{1}{2}(C+1) \rceil$, for the third iteration we have a maximum of $\lceil (\frac{1}{2})^2(C+1) \rceil$ possible solutions, and so on. Hence, the maximum number of necessary iterations to solve program (2.1) is given by

$$\text{Maximum number of iterations } = \lceil \log_2(C+1) \rceil. \tag{8.1}$$

The value of C is equal to the maximum number of carriers in the conveyor system. According to the adopted deadlock property, the number of carriers K is assumed to be less than the total capacity of the smallest directed cycle of solely finite capacity nodes with transfer blocking. This is the cycle consisting of all conveyor nodes. Hence, from Table 6.5 it follows that C is given by

$$C = \sum_{i=11}^{21} c_i = 602.$$

It follows from equation (8.1) that the maximum number of iterations by the Binary Search algorithm equals $\lceil \log_2(602+1) \rceil = 10$. Each iteration of the Binary Search algorithm contains the determination of the mean number of busy order pickers, for all load regions. We will estimate the mean number of busy order pickers in each load region using the simulation described

in the previous chapter for analyzing the proposed model of the chain conveyor.

8.2 Optimizing the number of carriers in the conveyor system

We optimize the chain conveyor for four main scenarios, containing three sub-scenarios each. Remember the four main scenarios defined as

1. morning shifts with manual labeling at the diverters,
2. evening shifts with manual labeling at the diverters,
3. morning shifts after implementation of automatic labeling, and
4. evening shifts after implementation of automatic labeling.

We refer to the former two scenarios as the current situation, and to the latter two as the future situation. The distribution of the order pickers over the store regions differs for the morning and evening shifts. For each scenario it is assumed that the transfer is operative. For all main scenarios, the total number of order pickers in the store regions is varied per shift in sub-scenarios a, b and c, increasing respectively. The parameters of the model are specified per scenario in Table 6.6. The scenario results are compared in this section, in order to perform a sensitivity analysis regarding the order pickers in the load regions, and the implementation of automatic labeling.

The results for the optimization of the chain conveyor are presented in Table 8.1 per scenario. The second column contains the optimal value of K for each scenario presented in the first column. Note that for each scenario, the total network capacity differs, since the total number of order pickers in the DC varies. The third column contains the percentage of the optimal number of carriers, from the total network capacity. The corresponding network capacities are given in Table 6.6b. The mean percentage of busy order pickers is presented per store region, and also for the total distribution center. The last two columns show the simulated time horizon and the run time of the simulation. This section continues with a discussion of the results.

For the the morning shift in the current situation the difference between the optimal values of K in the first two sub-scenarios and the last two sub-scenarios equals $476 - 427 = 40$ and $499 - 467 = 32$ carriers, respectively. For the the evening shift in the current situation these differences equal $491 - 439 = 52$ and $516 - 491 = 25$. For the the morning shift in the future situation these differences equal $470 - 433 = 37$ and $502 - 470 = 32$. Finally, for the the evening shift in the future situation these differences equal $495 - 443 = 52$ and $518 - 495 = 23$. Using the total conveyor node capacity of 602 as a benchmark, the deviations are 6.6% and 5.3% for main scenario 1, 8.6% and 4.2% for main scenario 2, 6.1% 5.3% for main scenario 3, and 8.6% and 3.8% for main scenario 4. This gives a mean deviation from the average optimal number of carriers in the conveyor system of 6.1%. This corresponds to about 36 carriers in the conveyor system. Performing the same sensitivity analysis, but for the difference between the first and last sub-scenarios of each main scenario, we obtain a maximum deviation of 12.8% for the

Table 8.1: Results for optimizing the total number of carriers in the conveyor system.

Scenario	**Optimal nr of carriers**	*Capacity percentage*	*Mean percentage of busy order pickers per store region* MLT	KAAS	VRS	BZ	AGF	*Total DC*	Time horizon (hours)	Run time (min)
Scenario 1 corresponds to morning shifts in the current situation										
1.a	**427**	64.2	97.7	96.0	97.3	95.2	98.6	97.1	378	115
1.b	**467**	69.5	98.0	96.5	97.5	95.1	98.7	97.2	303	89
1.c	**499**	73.6	98.2	96.7	97.5	95.0	98.6	97.3	201	74
Scenario 2 corresponds to evening shifts in the current situation										
2.a	**439**	65.2	97.7	95.9	97.4	95.2	98.5	97.4	360	107
2.b	**491**	72.1	98.1	96.4	97.5	95.4	98.6	97.6	245	101
2.c	**516**	75.0	98.2	96.5	97.5	95.5	98.7	97.7	149	78
Scenario 3 corresponds to morning shifts in the future situation										
3.a	**433**	65.1	97.7	96.0	97.3	95.0	98.6	97.1	364	106
3.b	**470**	69.9	98.0	96.5	97.5	95.2	98.7	97.3	294	105
3.c	**502**	74.0	98.2	96.7	97.5	95.1	98.5	97.2	186	91
Scenario 4 corresponds to evening shifts in the future situation										
4.a	**443**	65.8	97.8	96.0	97.4	95.2	98.6	97.5	363	104
4.b	**495**	72.7	98.0	96.3	97.5	95.3	98.7	97.7	229	78
4.c	**518**	75.3	98.2	96.5	97.4	95.0	98.6	97.6	145	68

evening shift in the current situation. Hence, in worst case, the optimal number of carriers in the conveyor system has a deviation of 12.8% within one main scenario. Taking all scenarios into account, the highest difference between optimal values of K is obtained for sub-scenarios 1.a (morning shifts in the current situation with the smallest number of order pickers) and 4.c (evening shifts in the future situation with the highest number of order pickers). The difference equals $518 - 427 = 91$, which corresponds to a deviation of 15.1% using the total conveyor node capacity of 602 as a benchmark. Hence, in worst case, the optimal number of carriers in the conveyor system has a deviation of 15.1%. It can be said that the optimal number of carriers in the conveyor system is not sensitive for variation of the distribution of the order pickers over the load regions, but is sensitive to variation of the total number of order pickers in the DC.

Consider the differences of the optimal values of K for the scenarios before and after implementation of automatic labeling. Hence, Scenario 1.a is compared to Scenario 3.a, Scenario 1.b is compared to Scenario 3.b, Scenario 2.a is compared to Scenario 4.a, and so on. The mean deviation equals 0.6%. This suggests that implementation of automatic labeling does not influence the optimal value of K, assuming that the transfer is in operation both before and after implementation of automatic labeling.

Summarizing, the optimal value of K is solely sensitive for varying the total number of order pickers, and not for varying the distribution of the order pickers, nor for implementation of automatic labeling. Since the capacity percentage depends on the total number of order pickers and the value of K, we will continue this section with a discussion on the results using the capacity percentages corresponding to the optimal values of K. Thereby, we will solely take the future scenarios into account.

We will propose three new loading strategies in the next chapter, whereby the percentage of the used capacity of the chain conveyor can be controlled by the operators at the fill place. We refer to this percentage as the *loading percentage of the conveyor system*. Each of the to be defined strategies corresponds to a loading percentage of the conveyor system 67%, 75%, or 80%.

Table 8.2: Comparison of the capacity percentages and the load percentages.

Scenario	Capacity percentage	Load percentage of 67%	75%	80%
3.a, morning shift with 26 order pickers	65.1	**1.6**	9.9	14.9
3.b, morning shift with 33 orderpickers	69.9	**3.2**	5.1	10.1
4.a, evening shift with 34 order pickers	65.8	**0.9**	9.2	14.2
3.c, morning shift with 39 order pickers	74.0	7.3	**1.0**	6.0
4.b, evening shift with 42 order pickers	72.7	6.0	**2.3**	7.3
4.c, evening shift with 49 order pickers	75.3	8.6	**0.3**	4.7

The capacity percentages corresponding to the optimal values of K for the future scenarios are repeated in Table 8.2, sorted by increasing order of the total number of order pickers in the DC. We compare these network percentages to the load percentages mentioned above. The differences are shown in the most right three columns of the table. We can conclude that total the number of order pickers is a good basis to determine which strategy is best to implement. The capacity percentages seem to increase as the total number of order pickers increases. For the three scenarios with the lowest total number of operators, the capacity percentages are closest to 67%. For the other three scenarios this holds for the load percentage 75%. Note that the increase of the capacity percentage is not strict for increasing the total number of order pickers. Since all capacity percentages are lower for the evening shift compared to the morning shift, this could be explained by the different distribution of order pickers over the store regions.

The mean percentages of busy order pickers in the total DC shown in Table 8.1 suggest that using an M-value of 95%, the mean number of busy order pickers in the DC is at least 97%, if the optimal value of K could be implemented. If the capacity percentage is lower than the implemented load percentage, then the number of carriers in the system will be higher than the optimal number. Regarding the integer program given in (2.1), we expect the implemented strategy results in a percentage of busy order pickers higher than 71%. However, the drawback could be that the number of carriers in the conveyor system is higher than necessary. Since this overload is in the order of 30 to 40 carriers, we assume that implementing a strategy with a higher capacity percentage does not have drastic consequences. Whether this assumption holds could be investigated in future research. On the other hand, if the capacity percentage is higher than the implemented load percentage, then the number of carriers in the conveyor system will be lower than the optimal value. This could be interpreted as a not allowed number of carriers with regard to the integer program given in (2.1). Hence, there will be a store region for which the percentage of busy order pickers is less than 95%. From Table 8.1 we conclude that for each optimal solution, the BZ store region has the lowest percentage of busy order pickers, followed by the KAAS region. Hence, we expect that the capacity constraints for the BZ and KAAS store regions in (2.1) will be violated, and thus the order pickers in those store regions will have an idle time higher than the allowed 5%. Future research should investigate the consequences of implementing a too low capacity percentage for these store regions in more detail. We assumed that the transfer is operative for all scenarios, while this is not the case in the current situation at Hollander. Therefore, we will discuss the effects of putting the transfer in operation in the next section of this chapter.

8.3 Effects of putting the transfer in operation

We discuss the performance of the conveyor system regarding the transfer in this section. In Chapter 2 the current usage of the conveyor system and the problems Hollander copes with were described. We noticed that currently, the transfer is put out of operation which is compensated by manually removing carriers from the chain conveyor at the fill place. Therefore, the operators at the fourth diverter face overwork. This problem could be solved if the transfer was put in operation. We assumed that the extra travel time of 30 minutes, of a carrier making an extra cycle through the DC using the transfer, is allowed. However, it is not allowed that loaded carriers are blocked by the transfer.

In Figure 8.1 the probability distributions of the number of carriers at the diverter nodes are shown. These values are based on the simulations for the optimal values of K, presented in Table 8.1. For all scenarios, diverter 1 is most occupied. This could be explained as a result of the determined probabilities corresponding to assigning carriers to diverters in the load region, as shown in Figure 6.4. Furthermore, from all shown scenarios in Figure 8.1 we conclude that during more than 95% of the time the conveyor is operating, the number of carriers is less than or equal to 6 carriers (where the capacity equals eight carriers for each diverter). The blocking probabilities for the diverter nodes are shown in the red framed part of Figure 8.1. Using these probabilities, the expected number of carriers that travel an extra round through the DC due to the finite capacities of the diverter can be computed. Consider the evening scenario with the highest number of order pickers in the current situation, Scenario 2.c. The blocking probabilities at the diverters are given by $PB_1 = 0.03127$, $PB_2 = 0.0007$, $PB_3 = 0.0012$, and $PB_4 = 0.0004$. The mean number of carriers that leave the DC at a busy day is estimated at 4500. Using the distributions shown in Figure 6.4, we conclude that the number of carriers that leave the DC through diverter 1, 2, 3 and 4 equal about 1400, 1100, 1000 and 1000, respectively. Using the blocking probabilities, the expected number of blocked carriers are approximately

$$0.03127 \cdot 1400 + 0.0007 \cdot 1100 + (0.0012 + 0.0012) \cdot 1000 = 46$$

at a busy day. This corresponds to 1% of the total loaded carriers distributed by the DC. This is assumed to be an acceptable percentage of carriers.

The capacity of the transfer equals five carriers. Figure 8.2 shows the approximated probability that the transfer is loaded with one, two, or three carriers. The probability that the number of carriers at the transfer equals four or five is approximately zero for each tested scenario. Figure 8.2 induces that the probability that there are three or even two carriers are stored at the transfer is negligible for all tested scenarios. The approximated mean sojourn times at the transfer given in Table 8.3 are all close to 0.1 minutes. The travel time of an extra cycle through the first floor of the DC is estimated to be less than 30 minutes. This suggests that the sojourn time at the transfer is negligible for computing the travel time of such cycle.

Summarizing, it is worth to put the transfer in operation with the number of carriers in the conveyor system close to optimal, and the number of operators is two per diverter. Thereby, we expected that about 1% of all loaded carriers will travel an extra cycle through the DC.

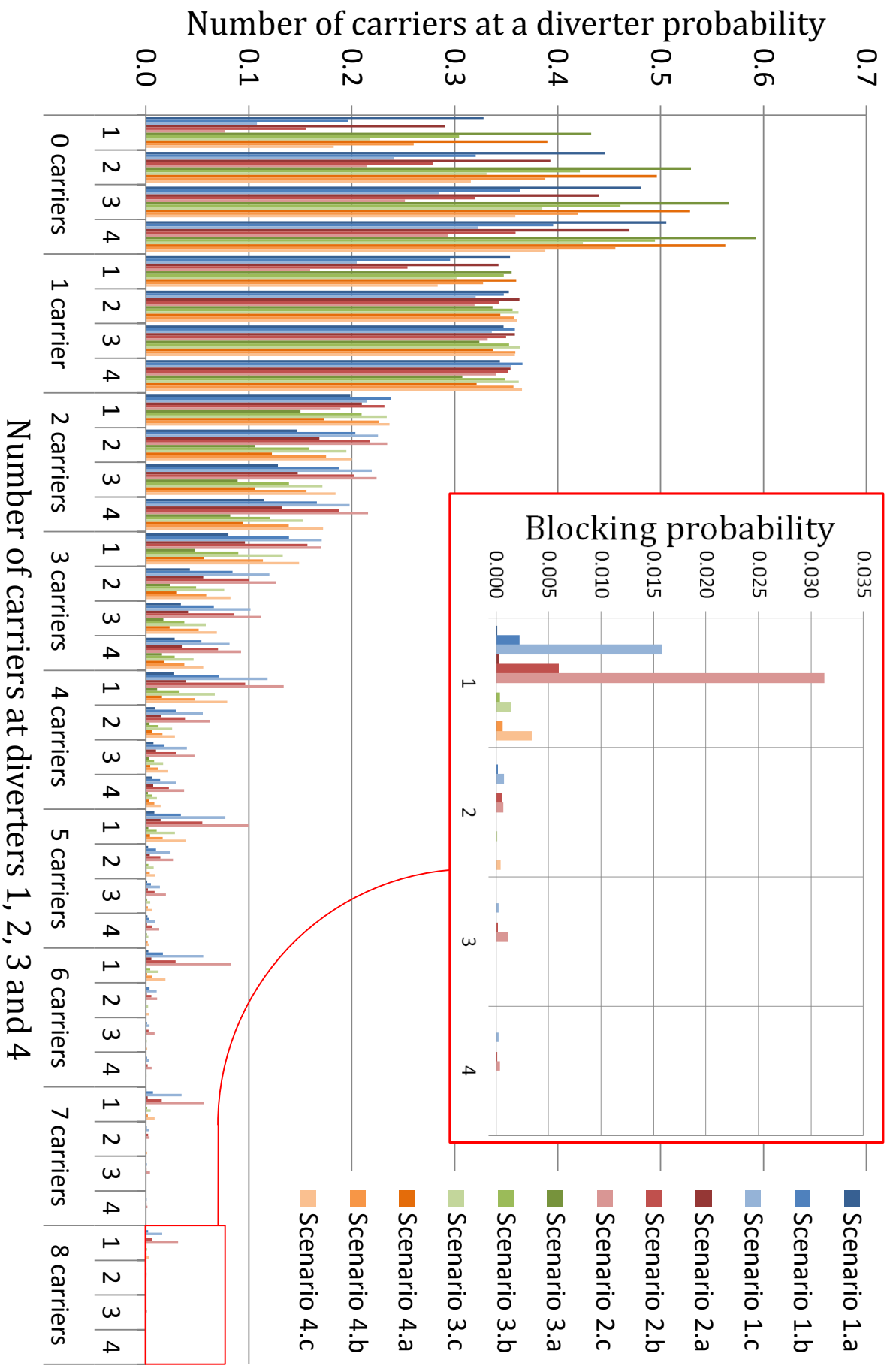


Figure 8.1: Probability distributions of the number of jobs at the diverter nodes.

Table 8.3: Mean sojourn times in minutes at the transfer of the chain conveyor.

Scenario	1.a	1.b	1.c	2.a	2.b	2.c	3.a	3.b	3.c	4.a	4.b	4.c
Mean sojourn time	–	0.09	0.11	0.09	0.09	0.10	–	0.09	0.09	0.09	0.11	0.10

Such extra cycle has a travel time of about 30 minutes. With the operating transfer, loaded carriers do not have to be manually removed from the chain conveyor any more. Therefore, we expect that the number of operators at both the fill location and the fourth diverter can be reduced. Furthermore, the probability that carriers are blocked by the transfer is negligible. The following section contains a small study regarding the consequences of lowering the number of operators at all diverters.

8.4 Decreasing the number of operators at the diverters after implementation of automatic labeling

We concluded that automatic labeling does not influence the optimal number of carriers in the conveyor system, and that the transfer never blocks carriers. These conclusions are drawn assuming that the number of operators at each diverter equals two. Figure 8.1 indicated that for all future scenarios, the probability that an arbitrary diverter is stored with seven or more carriers is less than 0.01. Thereby, we estimate that if the diverter nodes all have a capacity of seven, then the percentage of blocked carriers at a diverter will be less than 1%. Furthermore, in the previous section we assumed that 1% of the carriers being blocked by the diverters is allowed. In this section we will discuss the effect of decreasing the number of operators at each

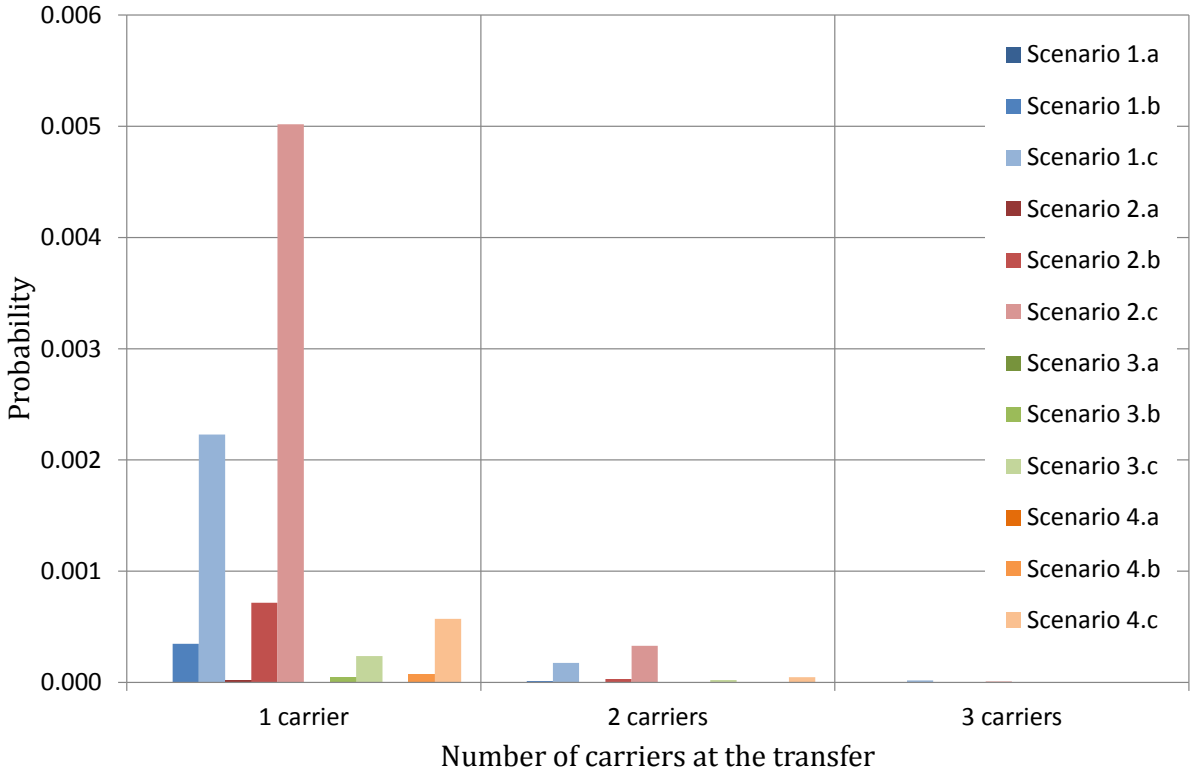


Figure 8.2: Probabilities for the number of carriers loaded on the transfer.

Table 8.4: Results for the total number of carriers in the conveyor system for the reduced future scenarios.

Scenario	Nr of carriers	*Capacity percentage*	*Mean percentage of busy order pickers per store region* MLT	KAAS	VRS	BZ	AGF	Time horizon (hours)
3.a	**476**	71.6	97.9	96.2	97.5	95.4	98.7	28
3.b	**516**	78.8	98.2	96.5	97.6	95.1	98.8	17
3.c	*602*	*88.8*	95.9	92.5	82.9	45.0	36.9	74
4.a	**478**	71.0	97.7	96.1	97.6	95.1	98.8	29
4.b	*602*	*88.4*	95.6	90.4	83.3	50.6	42.4	77
4.c	*602*	*87.5*	95.4	87.9	71.1	27.3	18.6	67

diverter to one. Note that in that case, the diverter node capacities in the model decease from 8 to 7. We will refer to these scenarios as the *reduced future scenarios*.

The reduced future scenarios are analyzed using the simulation adopting a d-value of 0.1. See Table 8.4 for the results. For the two least busy morning scenarios, Scenarios 3.a and 3.b, and the least busy evening scenario, Scenario 4.a shown in the table, the optimal solutions obtained by the Binary Search algorithm are presented. We will refer to these scenarios as *feasible scenarios*. For the most busy future morning scenario, Scenario 3.c, and the two most busy evening scenarios, Scenario 4.b and 4.c shown in the table the binary search did not find a solution. Instead, the results for the simulation with $C = 602$ carriers are presented. We will refer to these scenarios as *infeasible scenarios*.

For the infeasible scenarios, the percentages of busy order pickers at the KAAS and VRS region are quite acceptable. However, for the BZ and AGF region, these percentages are all less than 51%. Furthermore, the corresponding blocking probabilities PB_1, PB_2, PB_3 and PB_4 for the diverters, and the blocking probability PB_{10} of the transfer are presented in Figure 8.3. For the infeasible scenarios, the blocking probability of Diverter 1 raises op to 0.9 and the blocking probability of the transfer raises up to 0.5. Note that blocking at the diverter causes more loaded carriers arriving at the transfer, which causes a higher blocking probability at the transfer, and

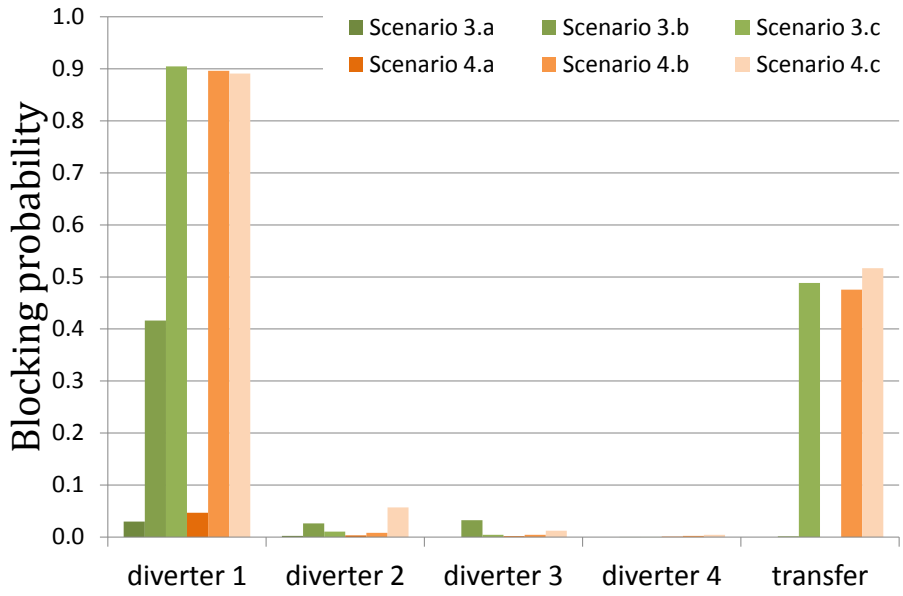


Figure 8.3: Blocking probabilities for the reduced future scenarios.

more loaded carriers in the cycle 'from the transfer along the BZ region and the AFG region, back to the transfer'. The latter causes more blocking at the diverter. This explains the very high blocking probabilities. We conclude that for the morning and evening shifts in the future situation corresponding to these scenarios, Hollander should not reduce all number of operators at the diverters to one.

The feasible scenarios show optimal solutions for which the capacity percentages remain less than 75% or 80%. Hence, one of the fill strageties mentioned in Section 8.2 could be implemented for these scenarios. Furthermore, the blocking probabilities for the transfer, shown in Figure 8.3, are neglicible. However, the approximated blocking probability at Diverter 1 corresponding to medium busy morning scenario raises over 0.4. This will result into to many loaded carriers that have to travel multiple cycles through the DC. For the other two feasible scenarios, we expect that the number of loaded carriers that have to travel an extra cycle through the DC will be acceptable. Therefore, we conclude that solely during the most quiet shifts of the DC, the number of operators at all diverters can be reduced to one.

Summarizing, solely for Diverters 2, 3 and 4 one operator per diverter is sufficient for the blocking probabilities to remain within proportions for each of the reduced future scenarios. Solely during the most quiet times at the DC, Hollander can reduce the number of operators at Diverter 1 as well. Thereby, we take the risk that the order pickers at the both of the BZ region and the AGF region are not supplied with enough empty carriers to prevent idle time. In addition, we assume that the results for the throughput time of the the operating transfer described in the previous section, also hold for the reduced future scenario described in this section.

8.5 Conclusions

The Binary Search algorithm gave a solution for each defined scenario with two operators at each diverter. For all scenarios is assumed that the transfer is put in operation. We concluded that it is worth to put the transfer in operation, since we approximated the blocking probability of the transfer at zero, and the percentage of carriers that travel multiple cycles through the DC is estimated at 1%. Thereby, the mean percentage of busy order pickers in the DC is approximated to be higher than 97%. The optimal number of carriers in the conveyor system is not sensitive for variation of the distribution of the order pickers over the load regions. However, the optimal number of carriers is sensitive to variation of the total number of order pickers in the DC. This deviation can raise up to 15%. Furthermore, implementation of automatic labeling does not influence the optimal number of carriers. In addition, the chain conveyor is simulated for the future scenarios with solely one operator at each diverter. During the most quiet shifts at the DC, the number of operators at all diverters can be reduced to one per diverter. Thereby, for all other shifts we expect that Hollander can reduce the number of operators at Diverter 2, 3 and 4, but not at Diverter 1. For all scenarios for which we obtained an optimal solution, the capacity percentages are approximately 67%, 75% or 80%. We will implement the latter result in the following chapter by defining fill strategies for which the load capacities are approximately 67%, 75% or 80% as well.

CHAPTER 9

Recommendations for Hollander

In this chapter we will formulate all recommendations for Hollander with regard to improving the continuity of the chain conveyor.

In Section 9.1 we will propose and discuss three new fill strategies to substitute the currently implemented full-load-strategy. We discuss the recommendations for implementation of these new fill strategies in Section 9.2. The recommendations for Hollander are summarized in Section 9.3.

9.1 Definition and discussion of new fill strategies

In the current situation, the *full-load-strategy* is implemented at the chain conveyor at Hollander, as described in Section 2.3. In this section we describe and discuss three new fill strategies, namely the *two-one-strategy*, the *three-one-strategy* and *the four-one-strategy*, see Figure 9.1. Using the two-one-strategy, the operator at the fill place of the chain conveyor should store the chain with carriers as follows. Alternate two holes of the chain storing a carrier and one free hole of the chain. The three-one-strategy and the four-one-strategy are defined analogously.

The order pickers should try to remain the alternating stored and empty holes closest to one of the fill strategies when storing the chain with loaded carriers. However, they are always allowed to couple a loaded carrier onto the chain. Thereby, the transportation of loaded carriers towards the diverters is not disturbed. We noticed that for each loaded carrier that an order picker stores on the chain, this carrier is compensated by removing an empty carrier from the chain, see Section 2.3. Therefore, we expect that the mean number of carriers on the chain remains constant over time after implementing one of the new fill strategies.

Implementation of automatic labeling We concluded in Section 8.2 that implementation of automatic labeling does not affect the optimal number of carriers in the DC. Therefore, all results for scenarios after implementation of automatic labeling hold for scenarios before implementation of automatic labeling as well.

Transfer in operation In Section 2.3 we assumed that the travel time of a second cycle through the DC of 30 minutes is allowed. Furthermore, we described that by implementation of the full-load-strategy, the throughput time of the transfer is too high. This causes blocked carriers at the transfer and loaded carriers that arrive too late at the expedition region. Remember that we obtained the following conclusion in Section 8.3. Consider the chain conveyor with a loading percentage of the conveyor system that corresponds to the optimal number of carriers in the conveyor system. Furthermore, assume that there are exactly two operators at each diverter, and that the transfer is operative. In this situation, the throughput time of carriers in the transfer is negligible in comparison to the travel time of a second cycle through the first floor in the DC. Thereby, we expect that 1% of the loaded carriers will pass by the fourth diverter and none of these carriers will be blocked by the transfer. Since we assumed that 1% of the carriers passing by the fourth diverter is allowed, the transfer can be put in operation by implementation of a loading percentage of the conveyor system corresponding to the optimal number of carriers in the chain conveyor and two operators at each diverter. Analogously, we showed in Section 8.4 that the transfer can be put in operation with the use of a new fill strategy in the following situations as well. During the most quiet shifts in the DC with solely one operator installed per diverter. During the more busy shifts in the DC with two operators installed at Diverter 1 and solely one operator installed per each of the other diverters. Overall, the transfer can be put in operation without affecting the travel times of loaded carriers through the DC.

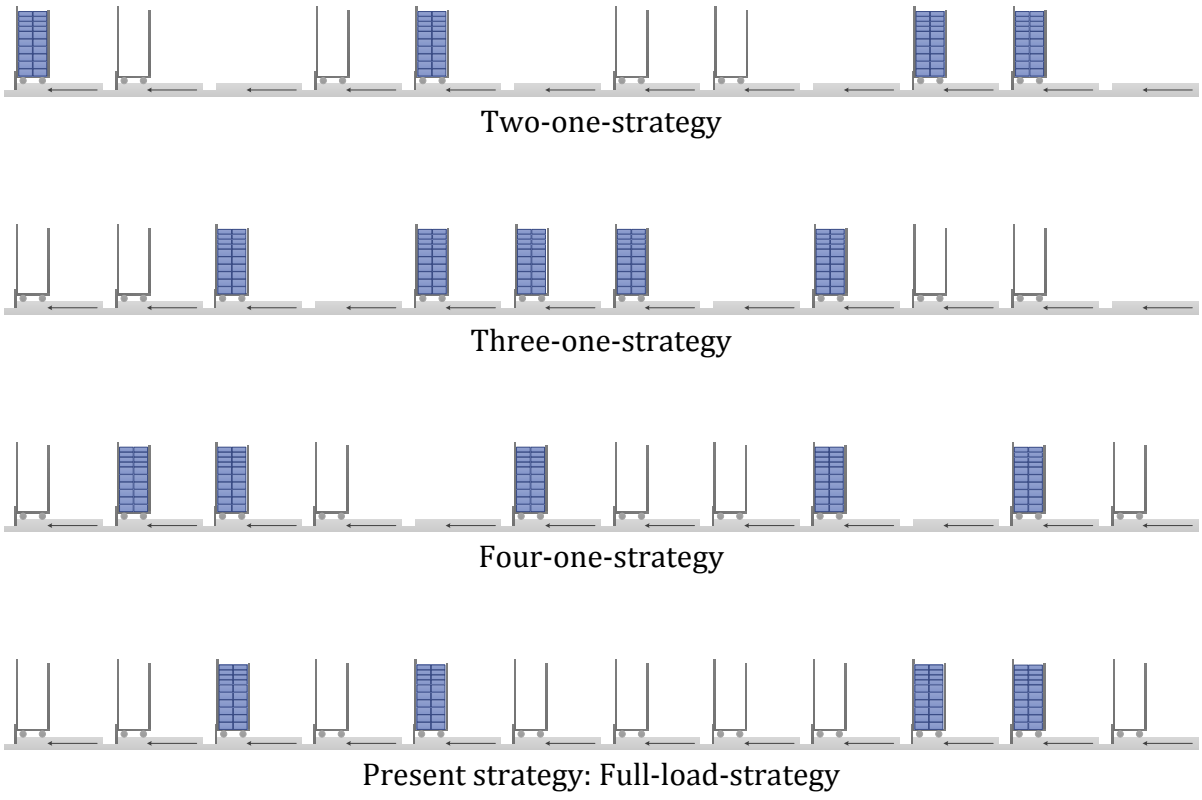


Figure 9.1: Fill strategies for storing the chain conveyor with carriers.

Reducing stagnation caused by pushed carriers By implementation of the two-one-strategy, 67% of the holes in the chain conveyor will be stored with a carrier. For the three-one-strategy and the four-one-strategy, 75% and 80% of the holes in the the chain conveyor will be stored with a carrier, respectively. These percentage are refered to as the *loading percentage of the fill strategy*. In Section 8.2 we defined the loading percentage of the conveyor system. This corresponds to the total number of carriers in the conveyor system, including the ones stored on the transfer or one of the diverters, the ones that are loaded by an order picker, and the ones that are handled by the operators at the diverter. Assume that before the start of a shift in the DC, the conveyor is stored with empty carriers according to the implemented fill strategy, and all order pickers have an empty carrier to load their first order on. It follows that for the value of the loading percentage of an arbitrary implemented loading strategy, the loading percentage of the conveyor system will be higher than this value. In Section 2.3 we concluded that stagnation is caused by pushed carriers, amongst others. Thereby, pushed carriers are caused by the combination of mechanical disruptions and the full-load-strategy. A mechanical disruption for a carrier on the chain causes a pushed carrier if and only if that carrier is followed by another carrier. By implementation of the two-load-strategy, 50% of the carriers stored on the chain is directly followed by another carrier. The other 50% of the carriers is directly followed by an empty hole of the chain. Hence, we expect that by implementation of the two-one-strategy stagnation caused by pushed carriers will be reduced with 50%. Analoguesly, we expect that by implementation of the three-one-strategy or the four-one-strategy stagnation will be reduced with 33% or 25%, respectively.

Eliminating stagnation due to overwork at the fourth diverter In Section 2.3 we showed that stagnation is caused by overwork at the fourth diverter. The full-load-strategy causes that the throughput time of carriers in transfer is too high. Therefore, Hollander putted the transfer out of operation. It follows that loaded carriers are not allowed to pass the fourth diverter, which causes the overwork at the diverter. Above we mentioned that the transfer can be put in operation. Therefore, after implementation of the new fill strategy with a loading percentage corresponding to the optimal number of carriers in the DC, we expect that overwork at the fourth diverter will never occur and thus stagnation due to overwork at the fourth diverter will be eliminated.

Remaining a high percentage of busy order pickers In Section 8.2 we showed that by implementation of a loading percentage of the conveyor system corresponding to the optimal number of carriers in the DC and two operators at each diverter, the percentage of busy order pickers in the DC will be at least 97%. Above we mentioned that for the value of the loading percentage of an arbitrary implemented loading strategy, the loading percentage of the conveyor system will be higher than this value. Hence, by implementation of a loading percentage of a fill strategy equal to the loading percentage of the conveyor system corresponding to the optimal number of carriers and two operators at each diverter, we expect that the mean percentage of busy order pickers in the DC is at least 97%. In Section 8.4 we said that this also holds for solely one operator at each diverter during the most quiet shifts in the DC, and for two operations at Diverter 1, and solely one operator at each of the other diverters during the more busy shifts in

the DC. Summarizing, after putting the transfer in operation and implementation of the new fill strategy with a loading percentage corresponding to the optimal number of carriers in the DC, we expect that the percentage of busy order pickers in the DC is at least 97%.

Eliminating risks at the store regions In Section 8.4 we mentioned that the risk of lowering the number of operators at the diverters is that the order pickers at both BZ region and AGF region are not supplied with enough empty carriers. This risk does not apply for all other store regions. These risks can cause too high percentages of idle order pickers for these regions. To compensate the low percentages, Hollander could supply these store regions could with a buffer of empty carriers. These could be stored near the receive area, in the lower left area shown at the floor plan in Figure 1.1a. Both of the store regions BZ and AGF are located at the first floor, right after the transfer. Hence, implementing the full-load-strategy using empty carriers between the transfer and the BZ region, lowers the risk of idle times for the order pickers. Given that the operator at the fill place holds to one of the other fill strategies, this extra fill place will not affect the throughput time of the transfer. Hence, during busy shifts in the DC, Hollander can locate an extra operator at the receive area who will fill the chain using the full-load-strategy. Thereby, the risks at the store regions will be eliminated and the throughput time of the transfer is not affected.

Risk with regard to ghost carriers In Section 2.3 we described the problem of ghost carriers. Currently, all ghost carriers are detected at the fill place. This will not be the case if the operators at the fill location let all loaded carriers pass by. This side effect is not investigated in this study. However, we would like to share some insights obtained during our observations in the DC. We suspect that ghost carriers are caused by the following reasons, amongst others.

1. *Interchanging the current and next carrier.* In the current situation, the conveyor system is filled according to the full-load-strategy. Hence, if an order picker wants to couple his loaded carrier back on the chain, he first has to remove the next (empty) carrier. If he scans the carriers in the wrong order, this causes a ghost carrier.

2. *Heavy load.* Loaded carriers can weigh up to 300 kg. Hence, they are very heavy to push forward during the order picking process. A smart order picker brings two empty carriers during the picking process to spread the heavy load. After finishing the order, he merges the two carriers. If he scans one carrier before the process and the other one after, this causes a ghost carrier.

We recommend Hollander to perform more research on the causes of ghost carriers.

Sensitivity with regard to deviating loading percentages Above, we described that we expect the following. By implementation of a loading percentage of a fill strategy equal to the loading percentage of the conveyor system corresponding to the optimal number of carriers and two operators at each diverter, the mean percentage of busy order pickers in the DC is at least 97%. We recommend Hollander to perform more research on the sensitivity

of the mean percentage of busy order pickers. This sensitivity regards the implementation of fill strategies with deviating loading percentages of the conveyor system from the loading percentages corresponding to the optimal number of carriers. By giving our recommendations for Hollander in the next section, we will assume that a deviaton of approximately 3% is allowed. Hereby, the deviation is defined as the difference between the implemented loading strategy and the loading strategy corresponding to the optimal number of carriers. Implemented loading percentages higher than loading percentages corresponding to optimal values are always allowed, since this does not reduce the mean number of busy order pickers in the DC.

Reducing the number of operators in the DC In Chapter 6 we studied the mean process time of a carrier at an arbitrary diverter, see Table 6.1b. The middle column corresponds to the current situation and the most right column correspond to the situation where automatic labeling is implemented. We obtain that the mean process time in the current situation is approximately 62 seconds, and with automatic labeling approximately 49 seconds. Hence, by implementation of automatic labeling the mean process time of a carrier is reduced by 13 seconds. Assuming that Hollander processes about one million carriers on a yearly basis, this indicates a theoretical saving of 3611 hours each year. Assuming that 1 FTE (Full-time equivalent) corresponds to $52 \times 36 = 1872$ hours per year, Hollander can save approximately 2 FTE per year.

On the other hand, in the next section we will recommend Hollander for three scenarios with regard to the number of operators at the diverters. Presently, Hollander installs two operators at each diverter. Hence, if the number of operators at three diverters is reduced to one, Hollander saves 3 FTE each year. Analogously, Hollander saves 4 FTE each year if the number of operators is reduced at each diverter. Furthermore, in Section 2.3 we described that in the current situation the number of operators at the fill place is higher than necessary. We expect that Hollander can save 1 FTE at the fill place. Summarizing, we expect that Hollander can save 3 to 4 FTE per year after implementation of the new fill strategies.

In the next section, we give recommendations with regard to the implementation of the new fill strategies depending on the total number of order pickers in the DC.

9.2 Implementation of the new fill strategies

With regard to improving the continuity of the chain conveyor, we recommend Hollander to put the transfer in operation and implement new fill strategies as described in this section.

We recommend Hollander to replace the currently implemented full-load-strategy by the two-one-strategy, the three-one-strategy and the four-one-strategy as shown in Table 9.1. The busy, medium, and quiet morning shifts correspond to morning shifts in the DC, whereby the total number of order pickers in the DC is approximately 26, 33 and 39, respectively. The busy, medium, and quiet evening shifts correspond to evening shifts in the DC, whereby the total number of order pickers in the DC is approximately 34, 42 and 49, respectively. Furthermore, the recommendations are specified for three scenarios with regard to the number of operators

Table 9.1: Recommendations for implementing new fill strategies.

Number of operators		Morning shifts Busy	Morning shifts Medium	Morning shifts Quiet	Evening shifts Busy	Evening shifts Medium	Evening shifts Quiet
Div 1:	*2*	three-one	two-one or three-one	two-one	three-one	three-one	two-one
Div 2:	*2*						
Div 3:	*2*						
Div 4:	*2*						
Div 1:	*2*	three-one or four-one	three-one or four-one	two-one or three-one	three-one or four-one	three-one or four-one	two-one or three-one
Div 2:	*1*						
Div 3:	*1*						
Div 4:	*1*						
Div 1:	*1*	-	four-one	three-one	-	-	three-one
Div 2:	*1*						
Div 3:	*1*						
Div 4:	*1*						

at the diverters. There is a trade-off between decreasing the number of carriers stored on the chain conveyor and reducing the number of operators at the diverters. Based on these recommendations, Hollander can choose a fill strategy with regard to this trade-off.

In Chapter 8 we stated that a mean number of busy order pickers in the DC of 97% is sufficient. In the previous section we concluded the following. By implementation of a loading percentage of a fill strategy equal to the loading percentage of the conveyor system corresponding to the optimal number of carriers and two operators at each diverter, the mean percentage of busy order pickers in the DC is at least 97%. Furthermore, we assume that a deviaton of approximately 3% is allowed, where the deviation is defined as the difference between the implemented loading strategy and the loading strategy corresponding to the optimal number of carriers. Implemented loading percentages higher than loading percentages corresponding to optimal values are always allowed. Therefore, the recommendations are based on the loading percentage of the conveyor system corresponding to the optimal number of carriers determined in Chapter 8. Thereby, we expect that the mean number of busy order pickers in the DC will be at least 97%.

Firstly, we discuss the recommendations whereby all diverters have two operators. The loading percentages corresponding to the optimal values of carriers are presented in the second column of Table 8.2. The deviations from the loading percentages of the two-one-, three-one-, and four-one-strategies are presented in columns three, four, and five respectively. Based on these values, we recommend to implement the two-one-strategy during the quiet evening shifts, and during both of the quiet and medium morning shifts. During the latter shift, Hollander can choose for the three-one strategy to lower the risk of idle order pickers. Secondly, we recommend to implement the three-one-strategy during the busy morning shift, and during both of the busy and medium evening shifts.

Secondly, we discuss the recommendations whereby all diverters have one operator. The loading percentages corresponding to the investigated number of carriers are presented in the third column of Table 8.4. The deviations from the loading percentages of the two-one-, three-one-, and four-one-strategies are as follows. In the quiet morning scenario, the deviations are $71.6 - 67 = 5\%$, $75 - 71.6 = 3\%$, and $80 - 71.6 = 8\%$, respectively. Therefore, we recom-

mend to implement the three-one-strategy. In the medium morning scenario, the deviations are $78.8 - 67 = 12\%$, $78.8 - 75 = 6\%$, and $80 - 78.8 = 1\%$, respectively. Therefore, we recommend to implement the four-one-strategy. In the quiet evening scenario, the deviations are $71.0 - 67 = 4\%$, $75 - 71.0 = 4\%$, and $80 - 71.0 = 9\%$, respectively. Therefore, we recommend to implement the three-one-strategy. Furthermore, we recommend Hollander to install two operators at the first diverter during the remaining shifts, as described below.

Finally, we discuss the recommendations whereby Diverter 1 has two operators and the other diverters have one operator each. For this scenario we did not perform any simulations. However, based on the discussion in Section 8.4 we can say that for Diverters 2, 3 and 4, solely one operator is sufficient for the blocking probabilities to remain within proportions. We recommend Hollander to implement the same fill strategies as for the scenario where each diverters has one operator. To lower the risk of idle order pickers, Hollander can choose to implement one of fill strategies with a higher loading percentage, as indicated in Table 9.1.

9.3 Conclusions

We discussed the implementation of new fill strategies with regard to improving the continuity of the chain conveyor. We recommend Hollander to put the transfer in operation and implement new fill strategies as indicated in Table 9.1. Thereby, we expect the mean percentage of busy order pickers in the DC to be at least 97%. The recommendations hold for both before and after implementation of automatic labeling. There is a trade-off between decreasing the number of carriers stored on the chain conveyor and reducing the number of operators at the diverters. Based on the recommendations in the table, Hollander can choose a fill strategy with regard to this trade-off. We expect that 1% of the loaded carriers will pass the fourth diverter and travel a second cycle of 30 minutes through the first floor of the DC. Thereby, no carriers will be blocked by the transfer. By implementing the two-one-, three-one, or four-one-strategy, 50%, 33%, or 25% of the stagnations caused by pushed carriers are eliminated, respectively. For all new fill-strategies, all stagnation due to overwork at the fourth diverter is eliminated. To eliminate the risk of unsufficient supply of empty carriers at the store regions, Hollander can create a new fill place between the transfer and the BZ store region. Thereby, the currently implemented full-load-strategy can solely be implemented between the transfer and the original fill place. Alongside improving the continuity of the chain conveyor, Hollander will save 3 to 4 FTE per year by implementating the recommended fill strategies.

Part IV

Concluding remarks

CHAPTER 10

Conclusions

The main scope of this study was to improve the continuity of the chain conveyor at Hollander. After performing observations in the distribution center at Hollander, the main research question was formulated as follows.

'How can the continuity of the chain conveyor at Hollander be improved by reducing stagnation using an operational approach, and what are the effects of implementing automatic labeling on the performance of the chain conveyor?'

In order to answer the main research question, the sub-questions formulated in Chapter 1 are answered.

In Chapter 2 we investigated the causes of stagnation of the chain conveyor and formulated a mathematical research approach. During the investigation, additional problems Hollander copes with regard to the chain conveyor were noticed. We chose to focus on the following topics: storing the chain, supplying order pickers, usage of transfer, pushed carriers, overwork at the fourth diverter, over staffing, and limited diverter capacities. The mathematical problem definition based on the correlations regarding these issues can be found in Section 2.4. We suspected that by optimizing the number of carriers in the conveyor system stagnation will be reduced. Thereby, the transfer should be put in operation. For the optimization of the chain conveyor we assumed that at least 95% of the order pickers in each store region should be supplied with sufficient empty carriers. This answers both of Sub-questions 1 and 2.

We performed a literature research on potential modeling methods, chose to model the chain conveyor as a closed queueing network, and analyzed the model using mean value analysis. This is reported in Chapter 3. In Chapter 4 is described how we modeled the chain conveyor as a queueing model. The model is a multi-class, closed queueing network consisting of both single and multiple, finite capacity, FCFS disciplined nodes where either recirculation blocking or transfer blocking occurs. Service times are generally distributed and class dependent, and both of class changes and head-of-line priorities are allowed. Hereby, Sub-question 3 is answered.

We proposed the MVABLO-m algorithm by adjusting the MVABLO algorithm Akyildiz [1988] in Chapter 5. The MVABLO-m algorithm is used for analyzing multi-class, closed queue-

ing networks consisting of both single and multiple server finite capacity nodes with transfer blocking and exponentially distributed service times. All nodes follow a FCFS discipline and class changes are allowed. The algorithm approximates the mean number jobs in a node, per job class; the mean sojourn time of a node, per job class; the throughput, per job class; and the mean blocking time of arbitrary jobs in a node. The way we modified the MVABLO algorithm is described in Sections 5.1 and 5.2. Firstly, we modified the MVA algorithm [Reiser and Lavenberg, 1980] using a mapping proposed in [Reiser and Kobayashi, 1975] in order to allow class changes. Thereafter, we modified the MVABLO algorithm so that it supports both of multi-class networks and networks with multiple server nodes. The evaluation of the MVABLO-m algorithm, using simulation results as a benchmark, is given in Section 5.4. For single-class networks with solely single server nodes, the MVABLO-m algorithm performs similar to the MVABLO algorithm. The deviations are estimated to be about 10%. For multi-class networks with solely single server nodes, the deviations of the joint mean number of jobs are slightly higher than the deviations in single-class networks. However, for multi-class queueing networks where class changes are allowed, the deviations raise to about 80%. For single-class networks with both single and multiple server node the deviations raise to 30%. The deviations are mainly high for networks where solely one node of the network has multiple servers. Based on the evaluation, we suggested in Section 5.5 that mean value analysis algorithms are not suitable for analyzing the proposed model. Summarizing, Sub-question 4 is answered.

We performed a data analysis and defined twelve scenarios in Chapter 6. The former in order to determine the parameters of the proposed model, and the latter in order to give an advise that is applicable for Hollander in each situation. Four main scenarios are defined in Section 6.3. First, we distinguished between the morning and the evening shifts in the DC with regard to the varying distribution of order pickers over the store regions in the DC. Secondly, both the current situation and the future situation were taken into account, regarding future implementation of automatic labeling. The total number of order pickers in the DC varies as well. Therefore, three sub-scenarios are taken into account for each main scenario. The parameters of all defined scenarios can be found in Table 6.6. Sub-question 5 is answered.

Since we showed that the MVABLO-m algorithm is not suitable for analyzing the chain conveyor, we proposed a simulation for closed queueing networks such as the proposed model in Chapter 7. The pseudo code of the simulation can be found in Appendix C. The stopping criteria for simulating the chain conveyor are specified in Section 7.1. The verification of the simulation can be found in Section 5.3.1, and the validation of the simulation for the chain conveyor can be found in Section 7.2. We suggested that the simulation results are close to the real world situation. This answers Sub-question 6.

The chain conveyor with the transfer in operation is optimized in Chapter 8. Thereby, we solved the proposed integer program using the Binary Search algorithm described in Section 8.1. The algorithm iterates over the number of carriers in the conveyor system. Each iteration, the algorithm uses the mean number of busy order pickers for each store region in the distribution center, given the number of carriers in the conveyor system. We used the simulation to approximate the mean number of busy order pickers in each store region. The optimal number of carriers in the conveyor system are presented for different scenarios in Tables 8.1, and 8.4.

These tables correspond to the situations where either two operators or one operator is installed per diverter, respectively. In Section 8.2 we suggested that the optimal number of carriers in the conveyor system is not sensitive for variation of the distribution of the order pickers over the load regions. However, the optimal number of carriers is sensitive to variation of the total number of order pickers in the DC. This deviation can raise up to 15%. Furthermore, we concluded that implementation of automatic labeling does not influence the optimal number of carriers. Finally, the results shown in Section 8.3 indicated that the transfer can be put in operation. Thereby, the blocking probability of the transfer is approximately zero, and the percentage of carriers that travel multiple cycles through the DC is estimated at 1%. For all optimizations, the mean percentage of busy order pickers in the DC is approximated to be higher than 97%. Hereby, Sub-questions 7 and 8 are answered.

Finally, in Chapter 9 we discussed the implementation of new fill strategies with regard to improving the continuity of the chain conveyor. We proposed three new fill strategies, the two-one-, three-one, and four-one-strategy. The strategies are shown in Figure 9.1. By implementation of these strategies, the chain conveyor will be stored using 67%, 75% and 80% of its capacity, respectively. We recommend Hollander to put the transfer in operation and implement new fill strategies as indicated in Table 9.1. There is a trade-off between decreasing the number of carriers stored on the chain conveyor and reducing the number of operators at the diverters. Based on the recommendations in the table, Hollander can choose a fill strategy with regard to this trade-off. The recommendations hold for both before and after implementation of automatic labeling. We expect the mean percentage of busy order pickers in the DC to be at least 97%, and that 1% of the loaded carriers will travel a second cycle through the first floor of the DC. Thereby, no carriers will be blocked by the transfer. By implementing the new fill strategies, 50%, 33%, or 25% of the stagnations caused by pushed carriers are eliminated, respectively. For all new fill-strategies, all stagnation due to overwork at the fourth diverter is eliminated. To eliminate the risk of unsufficient supply of empty carriers at the store regions, Hollander can create a new fill place whereby the currently implemented full-load-strategy can solely be implemented between the transfer and the original fill place. Alongside improving the continuity of the chain conveyor, Hollander will save 3 to 4 FTE per year by implementating the recommended fill strategies. We answered Sub-question 9.

Summarizing, the main research question of this study is answered as follows.

'We can improve the continuity of the chain conveyor at Hollander by putting the transfer in operation, and substituting the currently implemented fill strategy with the two-one-strategy, the three-one-strategy, and the four-one-strategy, depending on the shifts in the DC and the number of operators at the diverters. Automatic labeling allows Hollander to reduce the number of operators at the diverters without affecting the performance of the chain conveyor.'

CHAPTER 11

Recommendations for future research

A lot more research could be done with regard to improving the continuity of the chain conveyor at Hollander, and the proposed MVABLO-m algorithm. Below, recommendations for future research are given.

I. Firstly, evaluation of the MVABLO-m algorithm indicates that multi-class queueing networks have higher deviations than similar single-class networks, for networks where the mean number of visits varies for different nodes of the network. We recommend to investigate whether there is a correlation between these deviations and mean number of visits, using random determination of the routing probabilities.

II. Secondly, evaluation of the MVABLO-m algorithm suggested that multi-class queueing networks allowing class changes have very high deviations using initial population vectors with large differences for the number of jobs per job class. We recommend to investigate whether this is caused by the used mapping for allowing class changes.

III. We showed that the MVABLO-m algorithm is not suitable for analyzing the chain conveyor. We recommend Hollander to investigate whether one of the other potential modeling methods is suitable for analyzing the chain conveyor, since running simulations is more time consuming.

IV. Three new fill strategies are recommended with respect to the storage of empty carriers at the chain conveyor. We assumed that a deviation of 3% is allowed, regarding the implementation of fill strategies with deviating loading percentages of the conveyor system from the loading percentages corresponding to the optimal number of carriers. We recommend Hollander to perform more research on the sensitivity of the proposed fill strategies.

V. The chain conveyor is modeled ignoring stagnation times with respect to the throughput of the system, and mechanical disruptions of the diverters. We recommend Hollander to perform further research on the sensitivity of modeling the chain conveyor, with regard to ignoring stagnation times for the determination of the parameters in the proposed model and mechanical disruptions of the diverters.

VI. Firstly, during our observations in the distribution center at Hollander, we noticed two causes of ghost carriers. These causes are formulated as 'interchanging the current and next carrier' and 'heavy load'. We recommend Hollander to perform more research with respect to elimination of ghost carriers.

VII. Secondly, during our observations in the distribution center at Hollander, we noticed that stagnation can be caused by rubbish in the chain. We recommend Hollander to investigate how this can be prevented.

VIII. Finally, during our observations in the distribution center at Hollander, we noticed that pushed carriers are caused by disruptions along the track of the chain conveyor. We recommend Hollander to study the causes of these disruptions.

Bibliography

I. F. Akyildiz. Mean value analysis for blocking queueing networks. *IEEE Transactions on Software Engineering*, 14(1):418–429, 1988.

S. Balsamo and A. Rainero. Closed queueing networks with finite capacity queues: approximate analysis. *Proceedings of the ESM 2000, SCS European Simulation Multiconference, Ghent, Belgium, May 22–26*, 2000.

J. J. Bartholdi and S. T. Hackman. *Warehouse & Distrubution Science*. The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta GA 30332-0205 USA, 2011.

A. S. Bastani. Analytical solution of closed-loop conveyor systems with discrete and deterministic material flow. *European Journal of Operational Research*, 35:187–192, 1988.

G. Bolch, S. Greiner, H. d. Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. Wiley-Interscience, 2006.

R. M. Bryant and A. E. Krzesinski. The mva priority approximation. *ACM Transcactions on Computer Systems*, 2(4):335–359, Nov. 1984.

G. Gregory and C. D. Litton. A markovian analysis of a single conveyor system. *Management Science*, 22(3):371–375, 1975.

Hollander Barendrecht. Homepage hollander, Mar. 2012. URL `http://www.hollander.nl/`.

J. Little. A proof of the queueing formula $l = \lambda w$. *Operations Research*, 9(3):383–387, May 1961.

C. Osorio and M. Bierlaire. An analytic finite capacity queueing network model capturing the propagation of congestion and blocking. *European Journal of Operational Research*, 196(3): 996–1007, Nov. 2009.

H. G. Perros and T. Altiok. Approximate analysis of open networks of queues with blocking: Tandem configurations. *IEEE Transactions on Software Engineering*, SE-12(3):450–461, 1986.

M. Reiser and H. Kobayashi. Queueing networks with multiple closed chains: Theory and computational algorithms. *IBM J Res and Develop*, 19, 1975.

M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queueing networks. *Journal of the Association for Computing Machinery*, 27(2):313–322, Apr. 1980.

S. M. Ross. *Simulation*. Elsevier Academic Press, 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA, 2006.

L. C. Schmidt and J. Jackman. Modeling recirculating conveyors with blocking. *European Journal of Operational Research*, 124:422–436, 2000.

R. Schmidt. An approximate mva algorithm for exponential, class-dependent multiple servers. *Performance Evaluation*, 29, 1997.

D. Sonderman. An analytical model for recirculating conveyors with stochastic inputs and outputs. *International Journal of Production Research*, 20(5):591–605, 1982.

W. Stewart. A comparison of numerical techniques in markov modeling. *Communications of the ACM*, 21(2):144–152, 1978.

The Greenery. Homepage the greenery, Mar. 2012. URL `http://www.thegreenery.com/`.

H. C. Tijms. *A First Course in Stochastic Models*. Wiley, Vrije Universiteit, Amsterdam, The Netherlands, 2003.

R. W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982.

R. W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, Englewood Cliffs, N.J., 1989.

L. Zhuang and K. Hindi. Mean value analysis for multiclass closed queueing network models of flexible manufacturing systems with limited buffers. *European Journal of Operational Research*, 46, 1990.

Part V

Appendices

APPENDIX *A*

Results for the evaluation of the MVABLO-m algorithm

Table A.1: Results for queueing networks with exponentially distributed service times, FCFS disciplined single server nodes and single job classes.

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{t}_5$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{k}_5$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$	$\bar{b}_5$
Ex. A	*Exact*	39.82	41.34	–	–	–	3.92	4.07	–	–	–	–	–	–	–	–
K = 8	*MVABLO*	38.56	40.35	–	–	–	3.91	4.09	–	–	–			–	–	–
	MVABLO-m	38.57	40.36	–	–	–	3.91	4.09	–	–	–	0.00	30.00	–	–	–
	Simulation	39.25	40.94	–	–	–	3.91	4.08	–	–	–	0.01	30.17	–	–	–
	δ(MVABLO-m)	3.1	2.4	–	–	–	0.3	0.5	–	–	–			–	–	–
	δ(Simulation)	1.4	1.0	–	–	–	0.1	0.3	–	–	–			–	–	–
Ex. B	*Exact*	9.80	13.27	–	–	–	6.37	8.63	–	–	–	–	–	–	–	–
K = 15	*MVABLO*	7.61	14.24	–	–	–	5.22	9.77	–	–	–			–	–	–
	MVABLO-m	7.61	14.25	–	–	–	5.22	9.78	–	–	–	1.50	0.00	–	–	–
	Simulation	9.77	13.40	–	–	–	6.32	8.68	–	–	–	2.15	0.26	–	–	–
	δ(MVABLO-m)	22.3	7.4	–	–	–	18.0	13.3	–	–	–			–	–	–
	δ(Simulation)	0.3	1.0	–	–	–	0.7	0.5	–	–	–			–	–	–
Ex. C	*Exact*	36.58	29.84	–	–	–	10.38	8.98	–	–	–	–	–	–	–	–
K = 20	*MVABLO*	34.27	31.76	–	–	–	11.02	9.62	–	–	–			–	–	–
	MVABLO-m	34.28	31.70	–	–	–	10.39	9.61	–	–	–	13.32	0.00	–	–	–
	Simulation	36.36	30.10	–	–	–	10.94	9.06	–	–	–	13.06	0.02	–	–	–
	δ(MVABLO-m)	6.3	6.2	–	–	–	0.1	7.0	–	–	–			–	–	–
	δ(Simulation)	0.6	0.9	–	–	–	5.4	0.9	–	–	–			–	–	–
Ex. D	*Exact*	13.85	6.38	–	–	–	17.12	7.88	–	–	–	–	–	–	–	–
K = 25	*MVABLO*	14.23	6.01	–	–	–	17.57	7.42	–	–	–			–	–	–
	MVABLO-m	14.23	6.01	–	–	–	17.58	7.42	–	–	–	0.00	2.40	–	–	–
	Simulation	13.90	6.40	–	–	–	17.12	7.88	–	–	–	0.08	2.21	–	–	–
	δ(MVABLO-m)	2.7	5.8	–	–	–	2.7	5.8	–	–	–			–	–	–
	δ(Simulation)	0.4	0.3	–	–	–	0.0	0.0	–	–	–			–	–	–
Ex. E	*Exact*	124.40	79.16	–	–	–	30.57	19.43	–	–	–	–	–	–	–	–
K = 50	*MVABLO*	122.37	78.80	–	–	–	30.41	19.58	–	–	–			–	–	–
	MVABLO-m	122.37	78.81	–	–	–	30.41	19.59	–	–	–	0.00	40.00	–	–	–
	Simulation	122.72	78.21	–	–	–	30.54	19.46	–	–	–	0.42	38.44	–	–	–
	δ(MVABLO-m)	1.6	0.4	–	–	–	0.5	0.8	–	–	–			–	–	–
	δ(Simulation)	1.3	1.2	–	–	–	0.1	0.1	–	–	–			–	–	–
Ex. 0	*Exact*	11.8370	1.6220	13.5480	–	–	11.8300	1.6210	13.5440	–	–	–	–	–	–	–
K = 27	*MVABLO*	11.804	1.664	14.400	–	–	11.436	1.612	13.952	–	–	0.00	0.67	10.00	–	–
	MVABLO-m	11.804	1.664	14.400	–	–	11.436	1.613	13.951	–	–	0.00	0.67	10.00	–	–
	Simulation	11.8356	1.6148	13.5461	–	–	11.8370	1.6149	13.5478	–	–	0.00	0.56	9.11	–	–
	δ(MVABLO-m)	0.3	3.1	6.3	–	–	3.4	0.1	3.0	–	–				–	–
	δ(Simulation)	0.012	0.443	0.014	–	–	0.059	0.374	0.028	–	–				–	–
Ex. 0	*Exact*	11.8420	2.6190	13.5560	–	–	–	–	–	–	–	–	–	–	–	–
K = 28	*MVABLO*	11.436	3.306	14.650	–	–	–	–	–	–	–	–	–	–	–	–
	MVABLO-m	11.436	3.306	14.650	–	–	10.894	3.150	13.956	–	–	0.00	2.00	10.00	–	–
	Simulation	11.8962	2.6221	13.6099	–	–	11.8419	2.6100	13.5481	–	–	0.01	1.07	9.21	–	–
	δ(MVABLO-m)	3.9	26.1	7.6	–	–	8.0	20.7	3.0	–	–				–	–
	δ(Simulation)	0.458	0.120	0.397	–	–	–	–	–	–	–				–	–
Ex. 0	*Exact*	11.8490	3.6170	13.5680	–	–	–	–	–	–	–	–	–	–	–	–
K = 29	*MVABLO*	10.894	5.075	14.652	–	–	–	–	–	–	–	–	–	–	–	–
	MVABLO-m	10.894	5.07	14.652	–	–	10.318	4.806	13.876	–	–	0.00	3.00	10.00	–	–
	Simulation	11.8965	3.64	13.6092	–	–	11.8356	3.6247	13.5397	–	–	0.01	1.57	9.14	–	–
	δ(MVABLO-m)	8.4	39.3	7.7	–	–	12.8	32.6	2.5	–	–				–	–
	δ(Simulation)	0.401	0.735	0.304	–	–	–	–	–	–	–				–	–
Ex. 0	*Exact*	11.8660	4.6130	13.5900	–	–	–	–	–	–	–	–	–	–	–	–
K = 30	*MVABLO*	10.317	6.569	14.625	–	–	–	–	–	–	–	–	–	–	–	–
	MVABLO-m	10.318	6.570	14.625	–	–	9.822	6.254	13.923	–	–	0.00	3.67	10.00	–	–
	Simulation	11.8791	4.6198	13.6058	–	–	11.8378	4.6036	13.5586	–	–	0.03	2.08	9.15	–	–
	δ(MVABLO-m)	13.1	42.2	7.5	–	–	17.0	35.9	2.7	–	–				–	–

Continued on next page

Table A.1 *Continued from previous page*

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{t}_5$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{k}_5$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$	$\bar{b}_5$
	δ(Simulation)	0.111	0.148	0.117	–	–	–	–	–	–	–				–	–
Ex. 0	*Exact*	11.9650	6.6160	13.7380	–	–	–	–	–	–	–	–	–	–	–	–
K = 32	*MVABLO*	9.391	9.638	14.666	–	–	–	–	–	–	–	–	–	–	–	–
	MVABLO-m	9.391	9.639	14.666	–	–	8.918	9.154	13.928	–	–	0.00	5.33	10.00	–	–
	Simulation	11.9734	6.6231	13.7502	–	–	11.8451	6.5519	13.6029	–	–	0.11	3.13	9.25	–	–
	δ(MVABLO-m)	21.6	45.5	6.7	–	–	24.7	39.7	2.4	–	–				–	–
	δ(Simulation)	0.070	0.107	0.089	–	–	–	–	–	–	–				–	–
Ex. 1	*MVABLO*	11.501	0.999	2.231	–	–	11.25	0.997	2.770	–	–	–	–	–	–	–
K = 15	*MVABLO-m*	11.501	1.000	2.833	–	–	11.251	0.978	2.772	–	–	0.00	0.00	2.00	–	–
	Simulation	11.7848	0.9899	2.3457	–	–	11.6909	0.9819	2.3269	–	–	0.00	0.00	1.59	–	–
	δ(MVABLO-m)	2.4	1.0	20.8	–	–	3.8	0.4	19.1	–	–				–	–
Ex. 1	*MVABLO*	11.980	1.010	7.332	–	–	11.793	0.994	7.212	–	–	–	–	–	–	–
K = 20	*MVABLO-m*	11.983	1.010	7.333	–	–	11.791	0.994	7.215	–	–	0.00	0.00	5.00	–	–
	Simulation	11.7533	1.0609	7.0727	–	–	11.8202	1.0669	7.1128	–	–	0.00	0.04	4.83	–	–
	δ(MVABLO-m)	2.0	4.8	3.7	–	–	0.2	6.9	1.4	–	–				–	–
Ex. 1	*MVABLO*	13.254	15.927	8.455	–	–	11.621	13.965	7.413	–	–	–	–	–	–	–
K = 33	*MVABLO-m*	13.243	15.942	8.458	–	–	11.610	13.975	7.415	–	–	2.50	9.00	6.00	–	–
	Simulation	13.0817	14.5301	8.5895	–	–	11.9245	13.2448	7.8295	–	–	1.05	7.81	5.99	–	–
	δ(MVABLO-m)	1.2	9.7	1.5	–	–	2.6	5.5	5.3	–	–				–	–
Ex. 2	*MVABLO*	6.925	8.061	4.185	–	–	3.683	4.204	2.183	–	–	–	–	–	–	–
K = 10	*MVABLO-m*	6.926	8.061	4.186	–	–	3.612	4.205	2.183	–	–	2.00	0.00	1.50	–	–
	Simulation	7.9260	8.4341	5.1538	–	–	3.6841	3.9204	2.3955	–	–	2.41	0.24	2.50	–	–
	δ(MVABLO-m)	12.6	4.4	18.8	–	–	1.9	7.2	8.9	–	–				–	–
Ex. 2	*MVABLO*	7.967	9.019	8.011	–	–	3.825	4.330	3.846	–	–	–	–	–	–	–
K = 12	*MVABLO-m*	7.968	9.020	8.012	–	–	3.825	4.330	3.846	–	–	2.00	0.00	4.50	–	–
	Simulation	8.6703	10.0284	8.7263	–	–	3.7937	4.3880	3.8181	–	–	2.96	1.00	4.78	–	–
	δ(MVABLO-m)	8.1	10.1	8.2	–	–	0.8	1.3	0.7	–	–				–	–
Ex. 3	*MVABLO*	1.674	5.007	7.639	–	–	2.093	3.130	4.776	–	–	–	–	–	–	–
K = 10	*MVABLO-m*	1.674	5.007	7.640	–	–	2.093	3.131	4.776	–	–	1.40	0.00	0.00	–	–
	Simulation	1.3505	5.4324	6.7495	–	–	1.8150	3.6571	4.5278	–	–	0.92	0.00	0.00	–	–
	δ(MVABLO-m)	24.0	7.8	13.2	–	–	15.3	14.4	5.5	–	–				–	–
Ex. 3	*MVABLO*	2.166	5.372	6.567	–	–	3.195	3.962	4.843	–	–	–	–	–	–	–
K = 12	*MVABLO-m*	2.166	5.372	6.567	–	–	3.195	3.962	4.843	–	–	1.40	0.00	0.00	–	–
	Simulation	2.4370	5.8034	7.2229	–	–	3.2677	3.8972	4.8351	–	–	1.77	0.00	0.00	–	–
	δ(MVABLO-m)	11.1	7.4	9.1	–	–	2.2	1.7	0.2	–	–				–	–
Ex. 4	*MVABLO*	13.252	3.545	3.037	–	–	5.905	1.128	0.967	–	–	–	–	–	–	–
K = 8	*MVABLO-m*	13.252	3.545	3.037	–	–	5.905	1.128	0.967	–	–	0.00	1.25	1.25	–	–
	Simulation	14.3231	4.2421	3.5955	–	–	5.7538	1.2166	1.0293	–	–	0.00	2.05	1.95	–	–
	δ(MVABLO-m)	7.5	16.4	15.5	–	–	2.6	7.3	6.1	–	–				–	–
Ex. 4	*MVABLO*	14.226	7.032	6.324	–	–	5.978	2.113	1.900	–	–	–	–	–	–	–
K = 10	*MVABLO-m*	14.226	7.033	6.324	–	–	5.986	2.114	1.901	–	–	0.00	3.75	3.75	–	–
	Simulation	14.6031	8.0292	6.4959	–	–	5.8470	2.2980	1.8550	–	–	0.01	4.70	4.18	–	–
	δ(MVABLO-m)	2.6	12.4	2.6	–	–	2.4	8.0	2.5	–	–				–	–
Ex. 4	*MVABLO*	14.964	9.986	9.151	–	–	5.749	2.740	2.511	–	–	–	–	–	–	–
K = 11	*MVABLO-m*	14.964	9.986	9.151	–	–	5.749	2.740	2.511	–	–	0.00	6.25	6.25	–	–
	Simulation	14.6690	9.8269	8.1232	–	–	5.8594	2.8071	2.3332	–	–	0.03	5.91	5.33	–	–
	δ(MVABLO-m)	2.0	1.6	12.6	–	–	1.9	2.4	7.6	–	–				–	–
Ex. 5	*MVABLO*	14.874	5.482	2.931	–	–	5.542	1.930	0.528	–	–	–	–	–	–	–
K = 8	*MVABLO-m*	14.874	5.482	2.931	–	–	5.542	1.930	0.528	–	–	0.00	3.31	1.69	–	–
	Simulation	14.1417	4.7042	2.9927	–	–	5.6449	1.7753	0.5798	–	–	0.01	1.93	1.66	–	–

Continued on next page

Table A.1 *Continued from previous page*

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{t}_5$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{k}_5$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$	$\bar{b}_5$
	δ(MVABLO-m)	5.2	16.5	2.1	–	–	1.8	8.7	8.9	–	–				–	–
Ex. 5	*MVABLO*	14.260	9.348	4.180	–	–	5.678	3.518	0.805	–	–	–	–	–	–	–
K = 10	*MVABLO-m*	14.260	9.348	4.180	–	–	5.678	3.517	0.805	–	–	0.00	4.96	2.54	–	–
	Simulation	14.4788	9.1221	4.3569	–	–	5.7447	3.4254	0.8298	–	–	0.14	4.58	2.79	–	–
	δ(MVABLO-m)	1.5	2.5	4.1	–	–	1.2	2.7	3.0	–	–				–	–
Ex. 5	*MVABLO*	14.195	10.383	4.343	–	–	5.981	4.134	0.885	–	–	–	–	–	–	–
K = 11	*MVABLO-m*	14.195	10.383	4.343	–	–	5.981	4.134	0.885	–	–	0.00	4.96	2.54	–	–
	Simulation	14.9346	11.6980	4.7203	–	–	5.8050	4.3113	0.8836	–	–	0.39	6.17	3.10	–	–
	δ(MVABLO-m)	5.0	11.2	8.0	–	–	3.0	4.1	0.1	–	–				–	–
Ex. 6	*MVABLO*	3.792	9.339	11.784	7.426	–	1.172	2.886	3.644	2.296	–	–	–	–	–	–
K = 10	*MVABLO-m*	3.793	9.334	11.785	7.426	–	1.173	2.886	3.644	2.296	–	0.00	0.00	0.00	0.00	–
	Simulation	7.3241	11.2817	9.4141	6.4685	–	2.1236	3.2713	2.7295	1.8755	–	2.97	2.54	0.82	0.61	–
	δ(MVABLO-m)	48.2	17.3	25.2	14.8	–	44.8	11.8	33.5	22.4	–					–
Ex. 6	*MVABLO*	8.0907	11.236	8.997	8.379	–	2.645	3.673	2.941	2.739	–	–	–	–	–	–
K = 12	*MVABLO-m*	8.0907	11.237	8.997	8.379	–	2.645	3.674	2.942	2.740	–	2.60	2.80	0.00	0.00	–
	Simulation	10.5343	12.4819	10.0542	8.7964	–	3.0194	3.5779	2.8814	2.5210	–	4.98	3.17	1.43	1.98	–
	δ(MVABLO-m)	23.2	10.0	10.5	4.7	–	12.4	2.7	2.1	8.7	–					–
Ex. 6	*MVABLO*	14.459	12.902	9.655	13.976	–	3.969	3.542	2.651	3.837	–	–	–	–	–	–
K = 14	*MVABLO-m*	14.460	12.903	9.6550	13.976	–	3.970	3.542	2.651	3.837	–	7.80	2.80	2.40	7.20	–
	Simulation	13.4899	14.3718	12.9673	12.5171	–	3.5402	3.7717	3.4031	3.2850	–	7.21	4.61	3.17	4.48	–
	δ(MVABLO-m)	7.2	10.2	25.5	11.7	–	12.1	6.1	22.1	16.8	–					–
Ex. 7	*MVABLO*	4.273	13.150	18.310	9.671	–	1.411	4.344	6.049	3.195	–	–	–	–	–	–
K = 15	*MVABLO-m*	4.272	13.151	18.304	9.675	–	1.412	4.345	6.047	3.197	–	0.00	0.00	0.00	0.00	–
	Simulation	5.9437	16.1317	17.4521	9.9185	–	1.8031	4.8936	5.2945	3.0088	–	2.13	3.28	2.09	1.91	–
	δ(MVABLO-m)	28.1	18.5	4.9	2.5	–	21.7	11.2	14.2	6.2	–					–
Ex. 7	*MVABLO*	7.857	18.002	17.280	12.612	–	2.5369	5.812	5.579	4.072	–	–	–	–	–	–
K = 18	*MVABLO-m*	7.858	18.002	17.280	12.612	–	2.5369	5.812	5.579	4.072	–	5.20	2.80	0.00	1.80	–
	Simulation	8.4615	20.3880	22.0951	14.1130	–	2.3410	5.6407	6.1135	3.9046	–	4.10	5.71	4.48	4.39	–
	δ(MVABLO-m)	7.1	11.7	21.8	10.6	–	8.4	3.0	8.7	4.3	–					–
Ex. 8	*MVABLO*	1.6853	7.9283	3.6759	5.8070	4.2781	0.7210	3.3918	1.5726	2.4843	1.8302	–	–	–	–	–
K = 10	*MVABLO-m*	1.6853	7.9283	3.6759	5.8070	4.2781	0.7210	3.3918	1.5726	2.4843	1.8302	0.00	0.00	0.00	0.00	0.00
	Simulation	2.5520	7.4972	4.6939	7.3762	3.3324	1.0027	2.9456	1.8442	2.8981	1.3093	1.08	1.22	1.43	1.85	0.69
	δ(MVABLO-m)	34.0	5.8	21.7	21.3	28.4	28.1	15.1	14.7	14.3	39.8					
Ex. 8	*MVABLO*	3.7523	7.6793	4.0303	8.3349	3.1672	1.6699	3.4176	1.7936	3.7093	1.4095	–	–	–	–	–
K = 12	*MVABLO-m*	3.7523	7.6793	4.0303	8.3349	3.1672	1.6699	3.4176	1.7936	3.7093	1.4095	2.00	0.00	0.00	1.60	0.00
	Simulation	3.4684	9.7033	5.5234	9.2742	3.8705	1.3072	3.6570	2.0817	3.4953	1.4587	1.91	2.32	2.06	2.98	1.15
	δ(MVABLO-m)	8.2	20.9	27.0	10.1	18.2	27.7	6.5	13.8	6.1	3.4					
Ex. 8	*MVABLO*	5.271	10.018	7.286	10.111	5.448	1.935	3.677	2.674	3.712	2	–	–	–	–	–
K = 14	*MVABLO-m*	1039.7	1975.8	1437.0	1994.1	1074.4	1.935	3.678	2.675	3.712	1.99997	1038.00	1969.50	1432.80	1987.20	1072.00
	Simulation	5.5161	12.0114	8.1650	11.6981	5.3298	1.8077	3.9363	2.6758	3.8336	1.7466	3.75	4.13	4.09	4.77	2.49
	δ(MVABLO-m)	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	7.1	6.6	0.0	3.2	14.5					
Ex. 9	*MVABLO*	1895.23	2948.00	99.60	–	–	19.17	29.82	1.01	–	–	–	–	–	–	–
K = 50	*MVABLO-m*	1895.236	2948.427	99.605	–	–	19.170	29.823	1.007	–	–	1700.00	0.00	0.00	–	–
	Simulation	1895.81	2989.92	100.24	–	–	19.007	29.989	1.004	–	–	1715.13	0.00	0.00	–	–
	δ(MVABLO-m)	0.0	1.4	0.6	–	–	0.9	0.6	0.3	–	–				–	–
Ex. 9	*MVABLO*	3549.64	2601.73	1460.92	–	–	34.970	25.630	14.390	–	–	–	–	–	–	–
K = 75	*MVABLO-m*	3549.64	2601.73	1460.92	–	–	34.97	25.63	14.394	–	–	3200.00	0.00	760.00	–	–
	Simulation	3484.07	3029.01	1062.21	–	–	34.505	29.989	10.505	–	–	3147.95	0.00	508.62	–	–
	δ(MVABLO-m)	1.9	14.1	37.5	–	–	1.4	14.5	37.0	–	–				–	–

Continued on next page

Table A.1 *Continued from previous page*

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{t}_5$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{k}_5$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$	$\bar{b}_5$
Ex. 9	*MVABLO*	3547.60	2577.41	4978.13	–	–	34.769	25.261	39.696	–	–	–	–	–	–	–
$\mathbf{K} = 100$	*MVABLO-m*	3547.63	2577.42	4078.14	–	–	34.770	25.261	39.969	–	–	3200.00	150.00	2080.00	–	–
	Simulation	3538.75	3072.04	3631.48	–	–	34.547	29.984	35.451	–	–	3197.81	14.84	1845.91	–	–
	δ(MVABLO-m)	0.3	16.1	12.3	–	–	0.6	15.8	12.7	–	–				–	–
Ex. 10	*MVABLO*	20.035	131.570	793.460	60.136	–	0.996	6.544	39.467	2.991	–	–	–	–	–	–
$\mathbf{K} = 50$	*MVABLO-m*	20.035	131.571	793.465	60.136	–	0.997	6.544	39.468	2.991	–	0.00	100.00	0.00	0.00	–
	Simulation	21.239	120.981	783.537	63.212	–	1.069	6.105	39.641	3.181	–	0.00	92.88	2.44	0.00	–
	δ(MVABLO-m)	5.7	8.8	1.3	4.9	–	6.8	7.2	0.4	6.0	–					–
Ex. 10	*MVABLO*	72.28	609.69	787.36	59.45	–	3.54	29.91	38.62	2.92	–	–	–	–	–	–
$\mathbf{K} = 75$	*MVABLO-m*	72.29	609.70	787.37	59.45	–	3.55	29.91	38.63	2.92	–	45.00	460.00	0.00	0.00	–
	Simulation	68.003	573.026	794.704	58.451	–	3.401	28.752	39.924	2.922	–	32.37	434.97	1.64	0.00	–
	δ(MVABLO-m)	6.3	6.4	0.9	1.7	–	4.3	4.0	3.2	0.2	–					–
Ex. 10	*MVABLO*	405.68	705.54	931.45	312.50	–	17.22	29.95	39.55	13.26	–	–	–	–	–	–
$\mathbf{K} = 100$	*MVABLO-m*	405.69	705.54	931.46	312.50	–	17.23	29.96	39.55	13.27	–	240.00	560.00	135.00	110.00	–
	Simulation	394.289	621.583	835.080	243.489	–	18.798	29.675	39.921	11.602	–	214.24	472.36	39.00	62.08	–
	δ(MVABLO-m)	2.9	13.5	11.5	28.3	–	8.4	0.9	0.9	14.4	–					–

Table A.2: Results for queueing networks with exponentially distributed service times, FCFS disciplined single server nodes and multiple job classes.

Network:		$\bar{k}_{11}$	$\bar{k}_{12}$	$\bar{k}_1$	$\bar{k}_{21}$	$\bar{k}_{22}$	$\bar{k}_2$	$\bar{k}_{31}$	$\bar{k}_{32}$	$\bar{k}_3$	$\bar{k}_{41}$	$\bar{k}_{42}$	$\bar{k}_4$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$
Ex. 0	*MVABLO-m*	–	–	11.436	–	–	1.613	–	–	13.951	–	–	–	0.00	0.67	10.00	
$\mathbf{K} = (27)$	*Simulation*	–	–	11.8370	–	–	1.6149	–	–	13.5478	–	–	–	0.00	0.56	9.11	
	δ(MVABLO-m)	–	–	3.4	–	–	0.1	–	–	3.0	–	–	–				
Ex. 0	*MVABLO-m, R = 1*	–	–	11.436	–	–	1.613	–	–	13.951	–	–	–	0.00	0.67	10.00	–
$\mathbf{K} = (4, 23)$	*MVABLO-m, R = 2*	1.694	9.742	11.436	0.239	1.374	1.613	2.067	11.884	13.951	–	–	–	0.00	0.67	10.00	–
	Simulation	6.426	5.407	11.833	0.875	0.758	1.633	7.236	6.299	13.535	–	–	–	0.00	0.56	9.10	–
	δ(MVABLO-m, R = 2)	73.6	80.2	3.4	72.7	81.3	1.2	71.4	88.7	3.1	–	–	–				–
Ex. 0	*MVABLO-m, R = 1*	–	–	11.436	–	–	1.613	–	–	13.951	–	–	–	0.00	0.67	10.00	–
$\mathbf{K} = (6, 21)$	*MVABLO-m, R = 2*	2.541	8.895	11.436	0.358	1.254	1.613	3.100	10.851	13.951	–	–	–	0.00	0.67	10.00	–
	Simulation	6.403	5.432	11.835	0.868	0.745	1.613	7.297	6.255	13.552	–	–	–	0.00	0.56	9.10	–
	δ(MVABLO-m, R = 2)	60.3	63.7	3.4	58.7	68.4	0.0	57.5	73.5	2.9	–	–	–				–
Ex. 0	*MVABLO-m, R = 1*	–	–	11.436	–	–	1.613	–	–	13.951	–	–	–	0.00	0.67	10.00	–
$\mathbf{K} = (10, 17)$	*MVABLO-m, R = 2*	4.236	7.201	11.436	0.597	1.015	1.613	5.167	8.784	13.951	–	–	–	0.00	0.67	10.00	–
	Simulation	6.375	5.457	11.832	0.868	0.744	1.612	7.292	6.264	13.556	–	–	–	0.00	0.57	9.10	–
	δ(MVABLO-m, R = 2)	33.6	31.9	3.3	31.2	36.5	0.0	29.1	40.2	2.9	–	–	–				–
Ex. 0	*MVABLO-m, R = 1*	–	–	11.436	–	–	1.613	–	–	13.951	–	–	–	0.00	0.67	10.00	–
$\mathbf{K} = (14, 13)$	*MVABLO-m, R = 2*	5.930	5.506	11.436	0.836	0.776	1.613	7.234	6.717	13.951	–	–	–	0.00	0.67	10.00	–
	Simulation	6.412	5.418	11.830	0.875	0.754	1.629	7.275	6.266	13.541	–	–	–	0.00	0.56	9.07	–
	δ(MVABLO-m, R = 2)	7.5	1.6	3.3	4.5	3.0	1.0	0.6	7.2	3.0	–	–	–				–
Ex. 0	*MVABLO-m, R = 1*	–	–	11.436	–	–	1.613	–	–	13.951	–	–	–	0.00	0.67	10.00	–
$\mathbf{K} = (21, 6)$	*MVABLO-m, R = 2*	8.895	2.541	11.436	1.254	0.358	1.613	10.851	3.100	13.951	–	–	–	0.00	0.67	10.00	–
	Simulation	6.355	5.482	11.837	0.867	0.747	1.614	7.281	6.268	13.549	–	–	–	0.00	0.57	9.16	–
	δ(MVABLO-m, R = 2)	40.0	53.6	3.4	44.7	52.1	0.1	49.0	50.5	3.0	–	–	–				–
Ex. 3	*MVABLO-m*	–	–	3.195	–	–	3.962	–	–	4.843	–	–	–	1.40	0.00	0.00	–
$\mathbf{K} = (12)$	*Simulation*	–	–	3.2677	–	–	3.8972	–	–	4.8351	–	–	–	1.77	0.00	0.00	–
	δ(MVABLO-m)	–	–	2.2	–	–	1.7	–	–	0.2	–	–	–			–	
Ex. 3	*MVABLO-m, R = 1*	–	–	3.195	–	–	3.962	–	–	4.843	–	–	–	1.40	0.00	0.00	–
$\mathbf{K} = (2, 10)$	*MVABLO-m, R = 2*	0.530	2.637	3.167	0.627	2.998	3.625	0.842	4.365	5.207	–	–	–	1.40	0.00	0.00	–

Continued on next page

Table A.2 *Continued from previous page*

Network:		$\bar{k}_{11}$	$\bar{k}_{12}$	$\bar{k}_1$	$\bar{k}_{21}$	$\bar{k}_{22}$	$\bar{k}_2$	$\bar{k}_{31}$	$\bar{k}_{32}$	$\bar{k}_3$	$\bar{k}_{41}$	$\bar{k}_{42}$	$\bar{k}_4$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$
	Simulation	1.488	1.821	3.309	1.775	2.054	3.829	2.120	2.742	4.862	–	–	–	1.83	0.00	0.00	–
	δ(MVABLO-m, $R = 2$)	64.4	44.8	4.3	64.7	46.0	5.3	60.3	59.2	7.1	–	–	–				–
Ex. 3	*MVABLO-m*, $R = 1$	–	–	3.195	–	–	3.962	–	–	4.843	–	–	–	1.40	0.00	0.00	–
$\mathbf{K} = (4, 8)$	*MVABLO-m*, $R = 2$	1.064	2.121	3.185	1.311	2.510	3.822	1.624	3.369	4.993	–	–	–	1.40	0.00	0.00	–
	Simulation	1.472	1.804	3.277	1.819	2.073	3.892	2.106	2.725	4.831	–	–	–	1.79	0.00	0.00	–
	δ(MVABLO-m), $R = 2$	27.7	17.5	2.8	27.9	21.1	1.8	22.9	23.6	3.4	–	–	–				–
Ex. 3	*MVABLO-m*, $R = 1$	–	–	3.195	–	–	3.962	–	–	4.843	–	–	–	1.40	0.00	0.00	–
$\mathbf{K} = (6, 6)$	*MVABLO-m*, $R = 2$	1.602	1.599	3.201	2.055	1.970	4.025	2.343	2.431	4.774	–	–	–	1.40	0.00	0.00	–
	Simulation	1.487	1.821	3.308	1.787	2.051	3.838	2.136	2.718	4.854	–	–	–	1.83	0.00	0.00	–
	δ(MVABLO-m, $R = 2$)	7.7	12.2	3.2	15.0	4.0	4.9	9.7	10.6	1.7	–	–	–				–
Ex. 3	*MVABLO-m*, $R = 1$	–	–	3.195	–	–	3.962	–	–	4.843	–	–	–	1.40	0.00	0.00	–
$\mathbf{K} = (8, 4)$	*MVABLO-m*, $R = 2$	2.059	0.956	3.015	2.794	1.285	4.079	3.147	1.759	4.906	–	–	–	1.40	0.00	0.00	–
	Simulation	1.502	1.817	3.319	1.761	1.998	3.759	2.144	2.778	4.922	–	–	–	1.81	0.00	0.00	–
	δ(MVABLO-m, $R = 2$)	37.1	47.4	9.2	58.7	35.7	8.5	46.7	36.7	0.3	–	–	–				–
Ex. 3	*MVABLO-m*, $R = 1$	–	–	3.195	–	–	3.962	–	–	4.843	–	–	–	1.40	0.00	0.00	–
$\mathbf{K} = (10, 2)$	*MVABLO-m*, $R = 2$	2.407	0.480	2.887	3.511	0.673	4.183	4.082	0.847	4.929	–	–	–	1.40	0.00	0.00	–
	Simulation	1.474	1.805	3.279	1.825	2.086	3.911	2.081	2.729	4.810	–	–	–	1.78	0.00	0.00	–
	δ(MVABLO-m, $R = 2$)	63.3	73.4	11.9	92.3	67.7	7.0	96.2	69.0	2.5	–	–	–				–
Ex. 5	*MVABLO-m*	–	–	5.981	–	–	4.134	–	–	0.885	–	–	–	0.00	4.96	2.54	–
$\mathbf{K} = (11)$	*Simulation*	–	–	5.8050	–	–	4.3113	–	–	0.8836	–	–	–	0.39	6.17	3.10	–
	δ(MVABLO-m)	–	–	3.0	–	–	4.1	–	–	0.1	–	–	–				
Ex. 5	*MVABLO-m*, $R = 1$	–	–	5.981	–	–	4.134	–	–	0.885	–	–	–	0.00	4.96	2.54	–
$\mathbf{K} = (1, 10)$	*MVABLO-m*, $R = 2$	0.544	5.437	5.981	0.376	3.758	4.134	0.080	0.804	0.885	–	–	–	0.00	4.96	2.54	–
	Simulation	2.906	2.912	5.818	2.136	2.152	4.288	0.444	0.449	0.894	–	–	–	0.39	6.24	3.25	–
	δ(MVABLO-m, $R = 2$)	81.3	86.7	2.8	82.4	74.7	3.6	81.9	79.1	1.0	–	–	–				–
Ex. 5	*MVABLO-m*, $R = 1$	–	–	5.981	–	–	4.134	–	–	0.885	–	–	–	0.00	4.96	2.54	–
$\mathbf{K} = (3, 8)$	*MVABLO-m*, $R = 2$	1.631	4.350	5.981	1.128	3.007	4.134	0.241	0.644	0.885	–	–	–	0.00	4.96	2.54	–
	Simulation	2.916	2.897	5.813	2.165	2.152	4.316	0.438	0.432	0.870	–	–	–	0.38	6.25	3.13	–
	δ(MVABLO-m), $R = 2$	44.1	50.1	2.9	47.9	39.7	4.2	44.9	48.9	1.7	–	–	–				–
Ex. 5	*MVABLO-m*, $R = 1$	–	–	5.981	–	–	4.134	–	–	0.885	–	–	–	0.00	4.96	2.54	–
$\mathbf{K} = (5, 6)$	*MVABLO-m*, $R = 2$	2.719	3.262	5.981	1.879	2.255	4.134	0.402	0.483	0.885	–	–	–	0.00	4.96	2.54	–
	Simulation	2.929	2.882	5.812	2.172	2.134	4.306	0.442	0.440	0.882	–	–	–	0.38	6.20	3.11	–
	δ(MVABLO-m, $R = 2$)	7.2	13.2	2.9	13.5	5.7	4.0	9.0	9.8	0.4	–	–	–				–
Ex. 5	*MVABLO-m*, $R = 1$	–	–	5.981	–	–	4.134	–	–	0.885	–	–	–	0.00	4.96	2.54	–
$\mathbf{K} = (8, 3)$	*MVABLO-m*, $R = 2$	4.350	1.631	5.981	3.007	1.128	4.134	0.644	0.241	0.885	–	–	–	0.00	4.96	2.54	–
	Simulation	2.912	2.903	5.815	2.127	2.139	4.266	0.467	0.452	0.919	–	–	–	0.38	6.19	3.23	–
	δ(MVABLO-m), $R = 2$	49.4	43.8	2.9	41.3	47.3	3.1	37.9	46.6	3.7	–	–	–				–
Ex. 5	*MVABLO-m*, $R = 1$	–	–	5.981	–	–	4.134	–	–	0.885	–	–	–	0.00	4.96	2.54	–
$\mathbf{K} = (10, 1)$	*MVABLO-m*, $R = 2$	5.437	0.544	5.981	3.758	0.376	4.134	0.804	0.080	0.885	–	–	–	0.00	4.96	2.54	–
	Simulation	2.904	2.913	5.817	2.158	2.141	4.300	0.436	0.446	0.883	–	–	–	0.36	6.29	3.09	–
	δ(MVABLO-m, $R = 2$)	87.3	81.3	2.8	74.1	82.4	3.8	84.3	82.0	0.2	–	–	–				–
Ex. 7	*MVABLO-m*	–	–	2.5369	–	–	5.812	–	–	5.579	–	–	4.072	5.20	2.80	0.00	1.80
$\mathbf{K} = (18)$	*Simulation*	–	–	2.3410	–	–	5.6407	–	–	6.1135	–	–	3.9046	4.10	5.71	4.48	4.39
	δ(MVABLO-m)	–	–	8.4	–	–	3.0	–	–	8.7	–	–	4.3				
Ex. 7	*MVABLO-m*, $R = 1$	–	–	2.537	–	–	5.812	–	–	5.579	–	–	4.072	5.20	2.80	0.00	1.80
$\mathbf{K} = (2, 16)$	*MVABLO-m*, $R = 2$	0.282	2.255	2.537	0.646	5.166	5.812	0.620	4.959	5.579	0.452	3.619	4.072	5.20	2.80	0.00	1.80
	Simulation	1.258	1.083	2.342	3.045	2.578	5.622	3.299	2.791	6.090	2.109	1.837	3.946	4.08	5.66	4.69	4.54
	δ(MVABLO-m, $R = 2$)	77.6	108.1	8.3	78.8	100.4	3.4	81.2	77.7	8.4	78.5	97.0	3.2				
Ex. 7	*MVABLO-m*, $R = 1$	–	–	2.537	–	–	5.812	–	–	5.579	–	–	4.072	5.20	2.80	0.00	1.80
$\mathbf{K} = (6, 12)$	*MVABLO-m*, $R = 2$	0.846	1.691	2.537	1.937	3.875	5.812	1.860	3.719	5.579	1.357	2.715	4.072	5.20	2.80	0.00	1.80

Continued on next page

Table A.2 *Continued from previous page*

Network:		$\bar{k}_{11}$	$\bar{k}_{12}$	$\bar{k}_1$	$\bar{k}_{21}$	$\bar{k}_{22}$	$\bar{k}_2$	$\bar{k}_{31}$	$\bar{k}_{32}$	$\bar{k}_3$	$\bar{k}_{41}$	$\bar{k}_{42}$	$\bar{k}_4$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$
	Simulation	1.270	1.067	2.338	3.018	2.613	5.631	3.273	2.824	6.097	2.135	1.800	3.934	4.04	5.60	4.64	4.46
	δ(MVABLO-m, R = 2)	33.4	58.5	8.5	35.8	48.3	3.2	43.2	31.7	8.5	36.4	50.9	3.5				
Ex. 7	*MVABLO-m, R = 1*	–	–	2.537	–	–	5.812	–	–	5.579	–	–	4.072	5.20	2.80	0.00	1.80
$\mathbf{K} = (9, 9)$	*MVABLO-m, R = 2*	1.268	1.268	2.537	2.906	2.906	5.812	2.790	2.790	5.579	2.036	2.036	4.072	5.20	2.80	0.00	1.80
	Simulation	1.276	1.083	2.360	3.042	2.602	5.644	3.283	2.817	6.100	2.113	1.784	3.897	4.22	5.81	4.56	4.49
	δ(MVABLO-m, R = 2)	0.6	17.1	7.5	4.5	11.7	3.0	15.0	1.0	8.5	3.7	14.1	4.5				
Ex. 7	*MVABLO-m, R = 1*	–	–	2.537	–	–	5.812	–	–	5.579	–	–	4.072	5.20	2.80	0.00	1.80
$\mathbf{K} = (12, 6)$	*MVABLO-m, R = 2*	1.691	0.846	2.537	3.875	1.937	5.812	3.719	1.860	5.579	2.715	1.357	4.072	5.20	2.80	0.00	1.80
	Simulation	1.261	1.086	2.347	3.035	2.604	5.638	3.280	2.828	6.108	2.106	1.801	3.907	4.08	5.73	4.55	4.34
	δ(MVABLO-m, R = 2)	34.2	22.2	8.1	27.7	25.6	3.1	13.4	34.2	8.7	28.9	24.6	4.2				
Ex. 7	*MVABLO-m, R = 1*	–	–	2.537	–	–	5.812	–	–	5.579	–	–	4.072	5.20	2.80	0.00	1.80
$\mathbf{K} = (16, 2)$	*MVABLO-m, R = 2*	2.255	0.282	2.537	5.166	0.646	5.812	4.959	0.620	5.579	3.619	0.452	4.072	5.20	2.80	0.00	1.80
	Simulation	1.269	1.088	2.357	3.039	2.590	5.629	3.294	2.800	6.095	2.115	1.804	3.919	4.09	5.69	4.59	4.51
	δ(MVABLO-m, R = 2)	77.6	74.1	7.6	70.0	75.1	3.3	50.5	77.9	8.5	71.1	74.9	3.9				

Table A.3: Results for queueing networks with exponentially distributed service times, FCFS disciplined multiple server nodes and single job classes.

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$
Ex. 0	*MVABLO-m*	11.804	1.664	14.400	–	11.436	1.613	13.951	–	0.00	0.67	10.00	–
$\mathbf{K} = 27$	*Simulation*	11.8356	1.6148	13.5461	–	11.8370	1.6149	13.5478	–	0.00	0.56	9.11	–
$m = (1, 1, 1)$	*δ(MVABLO-m)*	0.3	3.1	6.3	–	3.4	0.1	3.0	–				–
Ex. 0	*MVABLO-m*	6.445	3.669	4.601	–	11.826	6.732	8.442	–	0.50	0.00	2.00	–
$\mathbf{K} = 27$	*Simulation*	6.130	3.345	5.341	–	11.171	6.095	9.734	–	0.28	0.00	0.00	–
$m = (2, 1, 1)$	*δ(MVABLO-m)*	5.2	9.7	13.9	–	5.9	10.4	13.3	–				–
Ex. 0	*MVABLO-m*	6.272	4.316	3.679	–	11.870	8.168	6.963	–	3.00	0.00	1.50	–
$\mathbf{K} = 27$	*Simulation*	5.450	4.947	3.167	–	10.848	9.848	6.305	–	0.76	0.00	0.00	–
$m = (4, 1, 1)$	*δ(MVABLO-m)*	15.1	12.8	16.2	–	9.4	17.1	10.4	–				–
Ex. 0	*MVABLO-m*	6.645	3.990	4.352	–	11.971	7.189	7.840	–	4.50	0.00	2.00	–
$\mathbf{K} = 27$	*Simulation*	5.384	5.002	3.122	–	10.761	9.998	6.241	–	0.75	0.00	0.00	–
$m = (8, 1, 1)$	*δ(MVABLO-m)*	23.4	20.2	39.4	–	11.2	28.1	25.6	–				–
Ex. 0	*MVABLO-m*	6.254	4.267	3.584	–	11.971	8.168	6.861	–	4.50	0.00	1.42	–
$\mathbf{K} = 27$	*Simulation*	5.394	5.011	3.126	–	10.763	9.999	6.239	–	0.48	0.00	0.00	–
$m = (12, 1, 1)$	*δ(MVABLO-m)*	16.0	14.8	14.6	–	11.2	18.3	10.0	–				–
Ex. 0	*MVABLO-m*	5.968	0.668	6.859	–	11.941	1.336	13.723	–	0.00	0.00	4.50	–
$\mathbf{K} = 27$	*Simulation*	5.954	0.848	6.691	–	11.915	1.697	13.389	–	0.00	0.01	3.41	–
$m = (2, 2, 2)$	*δ(MVABLO-m)*	0.2	21.3	2.5	–	0.2	21.3	2.5	–				–
Ex. 0	*MVABLO-m*	3.208	0.541	3.478	–	11.984	2.023	12.993	–	0.00	0.00	2.25	–
$\mathbf{K} = 27$	*Simulation*	2.989	0.576	3.175	–	11.973	2.308	12.719	–	0.00	0.00	0.87	–
$m = (4, 4, 4)$	*δ(MVABLO-m)*	7.3	6.0	9.6	–	0.1	12.4	2.2	–				–
Ex. 0	*MVABLO-m*	1.489	0.508	1.388	–	11.875	4.051	11.073	–	0.00	0.00	0.75	–
$\mathbf{K} = 27$	*Simulation*	1.499	0.507	1.368	–	11.995	4.058	10.947	–	0.00	0.00	0.10	–
$m = (8, 8, 8)$	*δ(MVABLO-m)*	0.7	0.1	1.4	–	1.0	0.2	1.2	–				–
Ex. 0	*MVABLO-m*	0.978	0.510	0.758	–	11.754	6.131	9.115	–	0.00	0.00	0.33	–
$\mathbf{K} = 27$	*Simulation*	1.006	0.500	0.767	–	11.954	5.937	9.110	–	0.00	0.00	0.00	–
m=(12, 10, 14)	*δ(MVABLO-m)*	2.9	2.0	1.1	–	1.7	3.3	0.1	–				–
Ex. 3	*MVABLO-m*	2.166	5.372	6.567	–	3.195	3.962	4.843	–	1.40	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	2.4370	5.8034	7.2229	–	3.2677	3.8972	4.8351	–	1.77	0.00	0.00	–

Continued on next page

Table A.3 *Continued from previous page*

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$
$m = (1, 1, 1)$	*δ(MVABLO-m)*	11.1	7.4	9.1	–	2.2	1.7	0.2	–				–
Ex. 3	*MVABLO-m*	1.830	5.496	6.811	–	2.751	4.131	5.119	–	1.40	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	2.208	5.848	7.392	–	3.001	3.974	5.025	–	1.59	0.00	0.00	–
$m = (2, 1, 1)$	*δ(MVABLO-m)*	17.1	6.0	7.9	–	8.3	3.9	1.9	–				–
Ex. 3	*MVABLO-m*	1.708	5.547	6.895	–	2.585	4.198	5.217	–	1.40	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	2.217	5.543	7.481	–	3.047	3.802	5.152	–	1.15	0.00	0.00	–
$m = (4, 1, 1)$	*δ(MVABLO-m)*	22.9	0.1	7.8	–	15.2	10.4	1.3	–				–
Ex. 3	*MVABLO-m*	1.653	5.573	6.935	–	2.508	4.229	5.263	–	1.40	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	2.348	5.013	7.517	–	3.271	3.493	5.236	–	0.00	0.00	0.00	–
$m = (8, 1, 1)$	*δ(MVABLO-m)*	29.6	11.2	7.7	–	23.3	21.1	0.5	–				–
Ex. 3	*MVABLO-m*	3.983	1.204	7.395	–	5.771	0.872	5.357	–	2.80	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	3.625	1.251	8.184	–	5.222	0.904	5.874	–	0.00	0.00	0.00	–
$m = (1, 4, 1)$	*δ(MVABLO-m)*	9.9	3.7	9.6	–	10.5	3.5	8.8	–				–
Ex. 3	*MVABLO-m*	3.983	1.200	7.399	–	5.771	0.869	5.360	–	2.80	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	3.698	1.204	8.268	–	5.258	0.855	5.887	–	0.00	0.00	0.00	–
$m = (1, 7, 1)$	*δ(MVABLO-m)*, $R = 2$	7.7	0.3	10.5	–	9.7	1.7	9.0	–				–
Ex. 3	*MVABLO-m*	1.091	2.819	3.685	–	3.014	3.895	5.091	–	0.70	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	1.134	2.978	3.709	–	3.038	3.988	4.974	–	0.61	0.00	0.00	–
$m = (2, 2, 2)$	*δ(MVABLO-m)*	3.8	5.3	0.6	–	0.8	2.3	2.4	–				–
Ex. 3	*MVABLO-m*	0.551	1.669	2.074	–	2.731	4.132	5.137	–	0.35	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	0.503	1.582	1.927	–	2.675	4.201	5.125	–	0.08	0.00	0.00	–
$m = (4, 4, 4)$	*δ(MVABLO-m)*	9.6	5.5	7.7	–	2.1	1.6	0.2	–				–
Ex. 3	*MVABLO-m*	0.200	1.224	1.611	–	1.484	4.540	5.976	–	0.00	0.00	0.00	–
$\mathbf{K} = 12$	*Simulation*	0.318	1.200	1.399	–	2.357	4.450	5.193	–	0.00	0.00	0.00	–
$m = (8, 7, 6)$	*δ(MVABLO-m)*	37.0	2.0	15.2	–	37.1	2.0	15.1	–				–
Ex. 5	*MVABLO-m*	14.195	10.383	4.343	–	5.981	4.134	0.885	–	0.00	4.96	2.54	–
$\mathbf{K} = 11$	*Simulation*	14.9346	11.6980	4.7203	–	5.8050	4.3113	0.8836	–	0.39	6.17	3.10	–
$m = (1, 1, 1)$	*δ(MVABLO-m)*	5.0	11.2	8.0	–	3.0	4.1	0.1	–				–
Ex. 5	*MVABLO-m*	7.230	6.450	1.984	–	5.568	4.694	0.739	–	0.00	0.83	0.42	–
$\mathbf{K} = 11$	*Simulation*	7.595	7.750	2.713	–	5.145	4.961	0.894	–	0.45	0.00	0.00	–
$m = (2, 1, 1)$	*δ(MVABLO-m)*	4.8	16.8	26.9	–	8.2	5.4	17.4	–				–
Ex. 5	*MVABLO-m*	5.376	7.045	2.175	–	4.519	5.597	0.884	–	2.03	0.00	0.37	–
$\mathbf{K} = 11$	*Simulation*	4.610	6.995	3.207	–	3.972	5.693	1.335	–	0.26	0.00	0.00	–
$m = (4, 1, 1)$	*δ(MVABLO-m)*	16.6	0.7	32.2	–	13.8	1.7	33.8	–				–
Ex. 5	*MVABLO-m*	5.093	6.514	2.272	–	4.537	5.484	0.979	–	2.03	0.00	0.37	–
$\mathbf{K} = 11$	*Simulation*	4.339	6.950	3.338	–	3.814	5.771	1.415	–	0.00	0.00	0.00	–
$m = (6, 1, 1)$	*δ(MVABLO-m)*	17.4	6.3	31.9	–	19.0	5.0	30.8	–				–
Ex. 5	*MVABLO-m*	7.275	5.235	2.519	–	5.954	4.049	0.997	–	0.00	2.48	1.27	–
$\mathbf{K} = 11$	*Simulation*	7.461	5.276	3.098	–	5.879	3.934	1.187	–	0.02	1.83	1.02	–
$m = (2, 2, 2)$	*δ(MVABLO-m)*	2.5	0.8	18.7	–	1.3	2.9	16.0	–				–
Ex. 5	*MVABLO-m*	3.556	2.392	1.478	–	5.989	3.807	1.204	–	0.00	0.83	0.42	–
$\mathbf{K} = 11$	*Simulation*	3.694	2.349	2.039	–	5.890	3.540	1.569	–	0.00	0.13	0.03	–
$m = (4, 4, 4)$	*δ(MVABLO-m)*	3.7	1.8	27.5	–	1.7	7.5	23.3	–				–
Ex. 5	*MVABLO-m*	2.375	1.600	1.167	–	5.869	3.737	1.394	–	0.00	0.28	0.14	–
$\mathbf{K} = 11$	*Simulation*	2.510	1.668	1.473	–	5.755	3.615	1.629	–	0.00	0.00	0.00	–
$m = (6, 6, 6)$	*δ(MVABLO-m)*	5.4	4.0	20.8	–	2.0	3.4	14.5	–				–
Ex. 7	*MVABLO-m*	7.858	18.002	17.280	12.612	2.5369	5.812	5.579	4.072	5.20	2.80	0.00	1.80

Continued on next page

Table A.3 *Continued from previous page*

Network:		$\bar{t}_1$	$\bar{t}_2$	$\bar{t}_3$	$\bar{t}_4$	$\bar{k}_1$	$\bar{k}_2$	$\bar{k}_3$	$\bar{k}_4$	$\bar{b}_1$	$\bar{b}_2$	$\bar{b}_3$	$\bar{b}_4$
$\mathbf{K} = 18$	*Simulation*	8.4615	20.3880	22.0951	14.1130	2.3410	5.6407	6.1135	3.9046	4.10	5.71	4.48	4.39
$m = (1, 1, 1, 1)$	*δ(MVABLO-m)*	7.1	11.7	21.8	10.6	8.4	3.0	8.7	4.3				
Ex. 7	*MVABLO-m*	6.970	12.418	20.332	15.170	2.286	4.072	6.668	4.975	2.60	1.40	12.00	3.60
$\mathbf{K} = 18$	*Simulation*	6.462	17.260	18.197	15.575	2.023	5.404	5.697	4.876	0.00	0.00	6.78	0.00
$m = (1, 1, 2, 1)$	*δ(MVABLO-m)*	7.9	28.1	11.7	2.6	13.0	24.6	17.0	2.0				
Ex. 7	*MVABLO-m*	7.670	13.037	17.033	13.738	2.682	4.559	5.956	4.804	2.60	0.00	12.00	1.80
$\mathbf{K} = 18$	*Simulation*	6.290	17.069	17.799	15.824	1.987	5.392	5.623	4.999	0.00	0.00	7.16	0.00
$m = (1, 1, 4, 1)$	*δ(MVABLO-m)*	21.9	23.6	4.3	13.2	35.0	15.5	5.9	3.9				
Ex. 7	*MVABLO-m*	6.681	15.205	23.625	15.407	1.974	4.493	6.981	4.553	2.60	5.20	19.20	3.60
$\mathbf{K} = 18$	*Simulation*	6.272	16.937	17.823	15.782	1.987	5.366	5.647	5.000	0.00	0.00	2.73	0.00
$m = (1, 1, 7, 1)$	*δ(MVABLO-m)*	6.5	10.2	32.5	2.4	0.7	16.3	23.6	8.9				
Ex. 7	*MVABLO-m*	4.040	7.831	9.374	5.745	2.694	5.223	6.251	3.831	1.30	1.40	0.00	0.00
$\mathbf{K} = 18$	*Simulation*	4.163	9.578	10.353	6.636	2.438	5.610	6.064	3.887	1.05	1.16	0.85	0.98
$m = (2, 2, 2, 2)$	*δ(MVABLO-m)*	3.0	18.2	9.5	13.4	10.5	6.9	3.1	1.4				
Ex. 7	*MVABLO-m*	2.239	5.206	6.797	4.098	2.197	5.110	6.671	4.022	0.00	0.00	0.00	0.00
$\mathbf{K} = 18$	*Simulation*	2.749	6.124	6.832	4.360	2.466	5.493	6.129	3.911	0.29	0.33	0.20	0.19
$m = (3, 3, 3, 3)$	*δ(MVABLO-m)*	18.6	15.0	0.5	6.0	10.9	7.0	8.8	2.8				
Ex. 7	*MVABLO-m*	1.115	3.206	4.766	3.355	1.613	4.639	6.895	4.853	0.00	0.40	1.44	0.60
$\mathbf{K} = 18$	*Simulation*	1.892	2.971	4.337	3.375	2.708	4.253	6.208	4.831	0.00	0.00	0.03	0.00
$m = (3, 6, 7, 5)$	*δ(MVABLO-m)*	41.1	7.9	9.9	0.6	40.5	9.1	11.1	0.5				

APPENDIX *B*

Data

Table B.1: Data service times diverters.

		Measured times				**Times per carrier**		
Period	Div	decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*	# containers	decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*
1	2	28.8	26.0	89.0	3	9.6	8.7	29.7
1	2	16.3	10.6	23.5	1	16.3	10.6	23.5
1	2			52.8	1			52.8
1	2	42.4	19.8		2	21.2	9.9	
1	2			39.3	1			39.3
1	2			45.8	1			45.8
1	2			46.6	1			46.6
1	2	14.4	15.2	52.2	1	14.4	15.2	52.2
1	2	15.2	12.5	59.0	1	15.2	12.5	59.0
1	2	8.5			1	8.5		
1	2		22.8		2		11.4	
1	2		47.2		1		47.2	
1	2		66.3		1		66.3	
1	2		6.2		1		6.2	
1	2	15.0	11.2	15.5	1	15.0	11.2	15.5
1	2			34.4	1			34.4
1	2			69.2	2			34.6
1	2	15.4	14.0	58.1	1	15.4	14.0	58.1
1	2			57.1	2			28.6
1	2	117.1			9	13.0		
1	2		43.9		5		8.8	
1	2			17.4	1			17.4
1	2			27.8	1			27.8
1	2			56.4	2			28.2
1	2			30.5	1			30.5
1	2	20.8			1	20.8		
1	2		42.9		5		8.6	
1	2			46.6	2			23.3
1	2			79.3	2			39.7
1	2	11.8			1	11.8		
1	2	14.7			2	7.4		
1	2		31.2		3		10.4	
1	2			56.0	2			28.0
1	2			38.7	1			38.7
1	2	8.1	9.9	29.9	1	8.1	9.9	29.9

Continued on next page

Table B.1 *Continued from previous page*

Period	Div	**Measured times** decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*	# con-tai-ners	**Times per carrier** decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*
1	2	20.6	14.5		2	10.3	7.3	
1	2			28.5	1			28.5
1	2			30.1	1			30.1
1	2	12.0	32.0	149.7	1	12.0	32.0	149.7
1	2			45.1	1			45.1
1	2	10.3	11.3	42.4	1	10.3	11.3	42.4
1	2	31.3			1	31.3		
1	2		25.4	57.9	2		12.7	29.0
1	2			55.9	2			28.0
1	2		44.3		6		7.4	
1	2	11.8			1	11.8		
1	2		36.3		4		9.1	
1	2	27.9	14.0	20.4	1	27.9	14.0	20.4
1	2	11.9	11.5	12.5	1	11.9	11.5	12.5
1	2	120.3	23.9		8	15.0	3.0	
1	2	32.3		18.7	3	10.8		6.2
1	2			11.0	1			11.0
1	2	14.2	14.5	40.4	1	14.2	14.5	40.4
1	2	6.0			1	6.0		
1	2		43.9		4		11.0	
1	2			15.5	1			15.5
1	2			24.5	1			24.5
1	2			30.7	1			30.7
1	2	23.3	25.4		2	11.7	12.7	
1	2			45.1	1			45.1
1	2			25.9	1			25.9
1	2	30.5	30.5		3	10.2	10.2	
1	2			48.8	2			24.4
1	2			33.8	1			33.8
1	2	19.1			2	9.6		
1	2	20.6			1	20.6		
1	2	10.3			1	10.3		
1	2		16.7		4		4.2	
1	2	13.0	13.0	42.9	1	13.0	13.0	42.9
1	2	14.3	10.8	38.8	1	14.3	10.8	38.8
1	2			42.1	2			21.1
1	2		19.1	41.4	1		19.1	41.4
1	2			54.3	2			27.2
1	2			59.8	2			29.9
1	2	15.8	35.9		3	5.3	12.0	
1	2			67.1	2			33.6
2	2	33.7	29.2		3	11.2	9.7	
2	2			41.3	2			20.7
2	2			28.8	1			28.8
2	2			64.0	1			64.0
2	1		41.6		4		10.4	
2	1			39.6	2			19.8
2	1			72.6	2			36.3
2	1			27.9	1			27.9
2	1			26.4	1			26.4
2	1			40.5	1			40.5
2	1		36.4		3		12.1	
2	1			77.9	2			39.0
2	1			27.5	1			27.5
2	1			41.8	1			41.8

Continued on next page

Table B.1 *Continued from previous page*

Period	Div	**Measured times** decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*	# con-tai-ners	**Times per carrier** decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*
2	2			61.2	2			30.6
2	2			60.9	1			60.9
2	2			69.1	2			34.6
2	2			24.8	1			24.8
2	2			58.8	2			29.4
2	2	54.1			3	18.0		
2	2	31.0			2	15.5		
2	2	31.5			2	15.8		
2	2	14.4		32.1	1	14.4		32.1
2	2			65.1	2			32.6
2	2			109.5	2			54.8
2	2			37.6	1			37.6
2	2	68.4			6	11.4		
2	2	22.1			2	11.1		
2	2		12.6	16.3	1		12.6	16.3
2	2			34.9	1			34.9
2	2			15.6	1			15.6
2	2			62.7	2			31.4
2	2		113.9		10		11.4	
2	2			25.2	1			25.2
2	2			10.5	1			10.5
2	2			16.1	1			16.1
2	2			35.7	2			17.9
2	2			21.1	1			21.1
2	2			47.3	2			23.7
2	2	35.4			2	17.7		
2	2	26.0			2	13.0		
2	2		57.2		7		8.2	
2	2			54.2	2			27.1
2	2	29.5			2	14.8		
2	2	27.3			2	13.7		
2	2		60.2		4		15.1	
2	2			57.8	1			57.8
2	2			23.8	1			23.8
2	2		56.2		6		9.4	
2	2	14.7			1	14.7		
2	2			55.0	2			27.5
2	2			46.4	2			23.2
2	2			30.2	1			30.2
2	2			29.6	1			29.6
2	2		55.2		6		9.2	
2	2			33.0	1			33.0
2	2	22.8			2	11.4		
2	2	18.4			1	18.4		
2	2		47.7		4		11.9	
2	2			41.8	1			41.8
2	2			68.6	2			34.3
2	2			30.7	1			30.7
2	2			51.1	1			51.1
2	2	24.3			1	24.3		
2	2		24.7		2		12.4	
2	2			14.4	1			14.4
2	2			35.6	1			35.6
2	2			41.4	1			41.4
2	2	14.2			1	14.2		

Continued on next page

Table B.1 *Continued from previous page*

Period	Div	**Measured times** decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*	# con-tai-ners	**Times per carrier** decouple and move to printer *(sec)*	scan and label *(sec)*	to storage and back *(sec)*
2	2		40.8		2		20.4	
2	2			27.4	1			27.4
2	2			59.3	1			59.3
2	2		52.9		4		13.2	
2	2			4.3	1			4.3
2	2			43.8	1			43.8
2	2			35.3	1			35.3
2	2		39.8		2		19.9	
2	2			63.1	1			63.1
2	2			107.4	2			53.7
2	2	15.3			3	5.1		
2	2		72.4		7		10.3	
2	2			36.2	1			36.2
2	2			90.6	2			45.3
2	2			38.4	1			38.4
2	2			56.2	1			56.2
2	2			54.0	1			54.0
2	2			41.7	1			41.7
2	2			53.8	1			53.8
2	2			55.6	1			55.6
2	2	12.7			1	12.7		
Mean						13.8	13.6	34.8

Table B.2: Some examples of picdocs.

Document ID	Employee number	Pick rule	Carrier type	Section	Carrier ID	Volume	Weight	Work area	# picks	Relation number	Relation name	Scheduled departure date	Departure time	Name employee
4185333	790	7	691	AP6	11875	0.28	81.6	VRS	7	66DE350	Denekamp	20/4/2012	18:14	Macko
4002161	668	57	691	ZUO	25013	0.72	322.5	VRS	69	66PU300	Dee Putten	4/4/2012	14;13	Girzycki
4166899	455	4	910	AP2	21828	0.79	231.8	AGF	13	66LO700	Lodder	19/4/2012	14:38	Tyczka
4196309	767	22	910	A3S	10986	0.46	322.1	BZ	28	66VE630	Verbeeten	21/4/2012	13:11	Karol
4003662	149	8	11	AP1	24373	0.81	166.8	AP	11	66HE300	Eland	4/4/2012	17:48	Frasiak
4119170	678	57	126	KAO	14952	0.84	211.4	KAAS	55	66VI600	Vijgen	14/4/2012	16:40	Luka
4219719	241	9	28	AP1	23849	0.52	109.9	AP	10	66ST130	Stationsplein	24/4/2012	14:15	Ijsselmuide
3831352	143	11	910	A3	22478	0.77	253.3	AGF	12	66KO310	Kolkman	21/3/2012	15:40	Rudolf
3831237	609	16	910	A3S	26301	0.55	379.6	BZ	19	66EE100	van Ee	21/3/2012	11;56	Kolacai
3844711	376	5	123	MAO	22984	0.08	6.9	MLT	5	66PA290	Martin Panis	22/3/2012	15:19	Baczewska
4195996	797	59	691	AP6	16283	0.7	340.8	VRS	70	66NI180	Nijenhuis Terborg	21/4/2012	13:42	Malecki
3844055	377	2	135	KAO	15708	0.01	4.2	KAAS	2	66TI080	van Tilborgh	22/3/2012	12:49	Kujda
3989022	491	16	910	A3S	15328	0.44	300.9	BZ	22	66SC920	Schalkoord	3/4/2012	16:54	Piotrowski
3904713	179	16	910	A3S	27081	0.22	140.9	BZ	22	66CR600	Crombach	27/3/2012	16:41	Labocha

Table B.3: Some examples from Qlikview.

Number of carriers	Work area	Date first pick	Time first pick	First pick lane nr	First pick column nr	First pick position nr	Date last pick	Time last pick	Last pick lane nr	Last pick column nr	Last pick position nr	mutation moment date	Relation number	Sales order number
1	KAAS	19/3/2012	0:06:28	AP8	2	1	19/3/2012	0:06:28	AP8	2	1	19/3/2012	66KL600	2786112077
1	MLT	19/3/2012	0:06:37	V10	12	2	19/3/2012	0:06:49	V11	12	1	19/3/2012	66HI150	2850112077
1	MLT	19/3/2012	0:06:39	V05	1	1	19/3/2012	0:09:50	V08	10	1	19/3/2012	66WI610	2894112077
1	VRS	19/3/2012	0:06:46	V63	7	1	19/3/2012	0:09:19	V64	4	1	19/3/2012	66WI680	2357612077
1	KAAS	19/3/2012	0:06:52	V65	13	1	19/3/2012	0:06:52	V65	13	1	19/3/2012	66TH600	2356812077
1	MLT	19/3/2012	0:07:50	AP3	1	1	19/3/2012	0:09:26	V02	1	2	19/3/2012	66HO610	2317412077
1	KAAS	19/3/2012	0:08:03	V65	25	1	19/3/2012	0:08:37	V65	31	1	19/3/2012	66KL600	2786612077
1	MLT	19/3/2012	0:08:05	V09	10	2	19/3/2012	0:08:26	V09	9	2	19/3/2012	66WI680	2773712077
1	MLT	19/3/2012	0:08:28	V09	12	2	19/3/2012	0:08:28	V09	12	2	19/3/2012	66VE610	2901912077
1	AGF	19/3/2012	0:08:53	P12	10	2	19/3/2012	0:08:53	P12	10	2	19/3/2012	66AR600	2937812077
1	MLT	19/3/2012	0:09:02	V11	14	2	19/3/2012	0:09:52	V12	2	1	19/3/2012	66WI680	2799712077
1	AGF	19/3/2012	0:09:19	P12	10	2	19/3/2012	0:09:19	P12	10	2	19/3/2012	66WI680	2775212077
2	BZ	19/3/2012	0:10:26	P02	7	1	19/3/2012	0:11:03	P02	9	1	19/3/2012	66VR110	2776212077
2	KAAS	19/3/2012	0:10:26	V15	39	1	19/3/2012	0:10:08	V15	8	1	19/3/2012	66LO210	1504612077

Table B.4: Possible service times for jobs that arrive from a load node at a conveyor node.

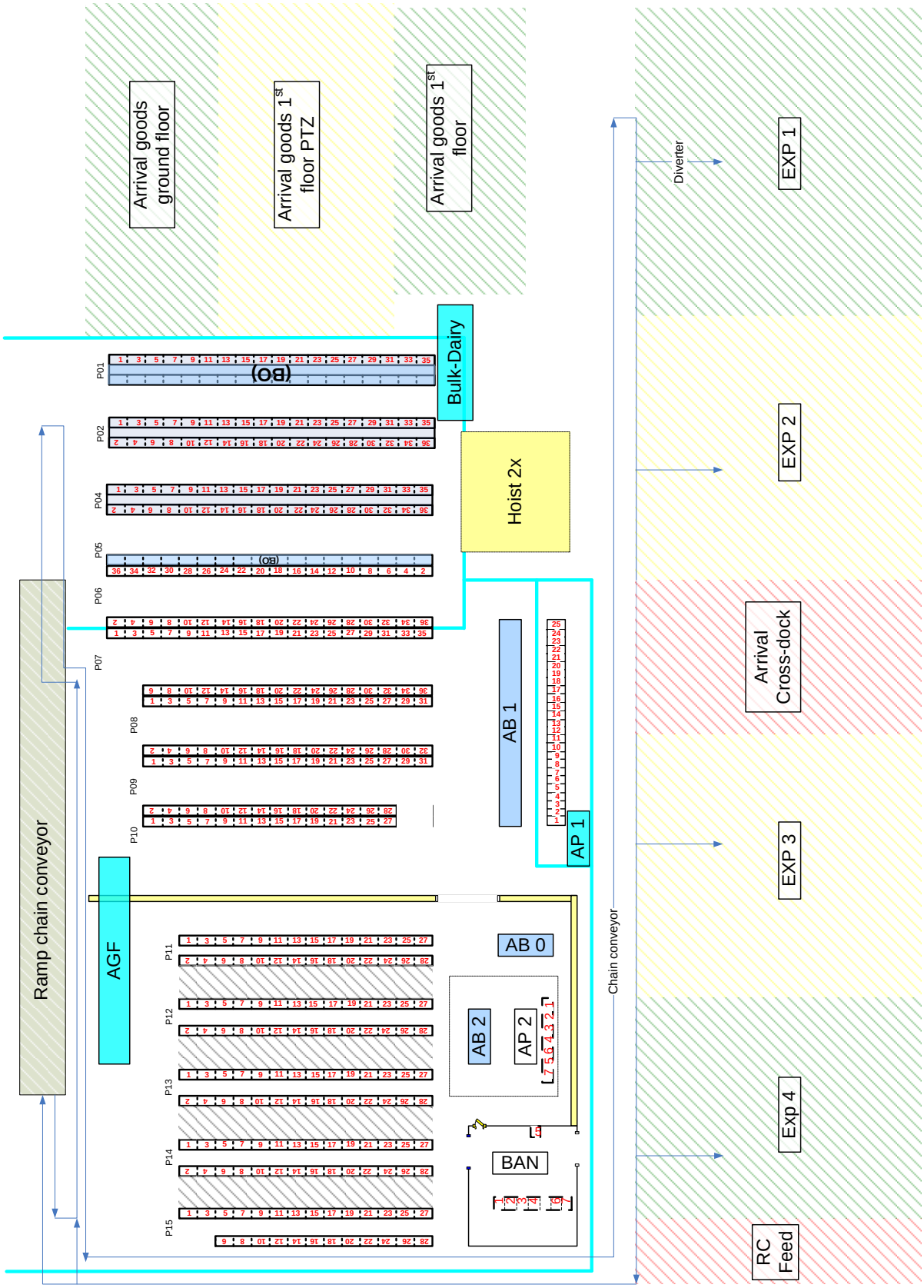
Node 12 corresponding to conveyor node after load node MLT		
Lane	Time (min)	Probability
AP3	3.5	0.010
V01	3.2	0.007
V02	2.9	0.062
V03	2.7	0.008
AP5	2.4	0.071
V04	2.4	0.058
V05	2.1	0.038
V06	1.8	0.022
V07	1.5	0.023
V08	1.2	0.080
V09	0.9	0.120
V10	0.6	0.161
V11	0.3	0.228
V12	0.0	0.112

Node 21 corresponding to conveyor node after load node VRS		
Lane	Time (min)	Probability
V64	9.1	0.127
V63	8.8	0.144
V62	8.6	0.137
V61	8.3	0.042
V60	8.0	0.009
AP7	7.8	0.106
V59	7.8	0.019
V58	7.5	0.041
V57	7.3	0.013
V56	7.0	0.009
V55	6.7	0.058
V54	6.5	0.254
V53	6.2	0.005
V52	5.9	0.001
V51	5.7	0.001
V50	5.4	0.001
AP5	6.2	0.000
AP4	5.2	0.002
AQ2	4.9	0.008
AQ1	4.9	0.023

Node 15 corresponding to conveyor node after load node BZ		
Lane	Time (min)	Probability
P01	1.4	0.060
P02	1.2	0.939
P03	1.0	0.000
P04	0.7	0.000
P05	0.5	0.000
P06	0.2	0.000
P07	0.0	0.000
P08	0.0	0.000

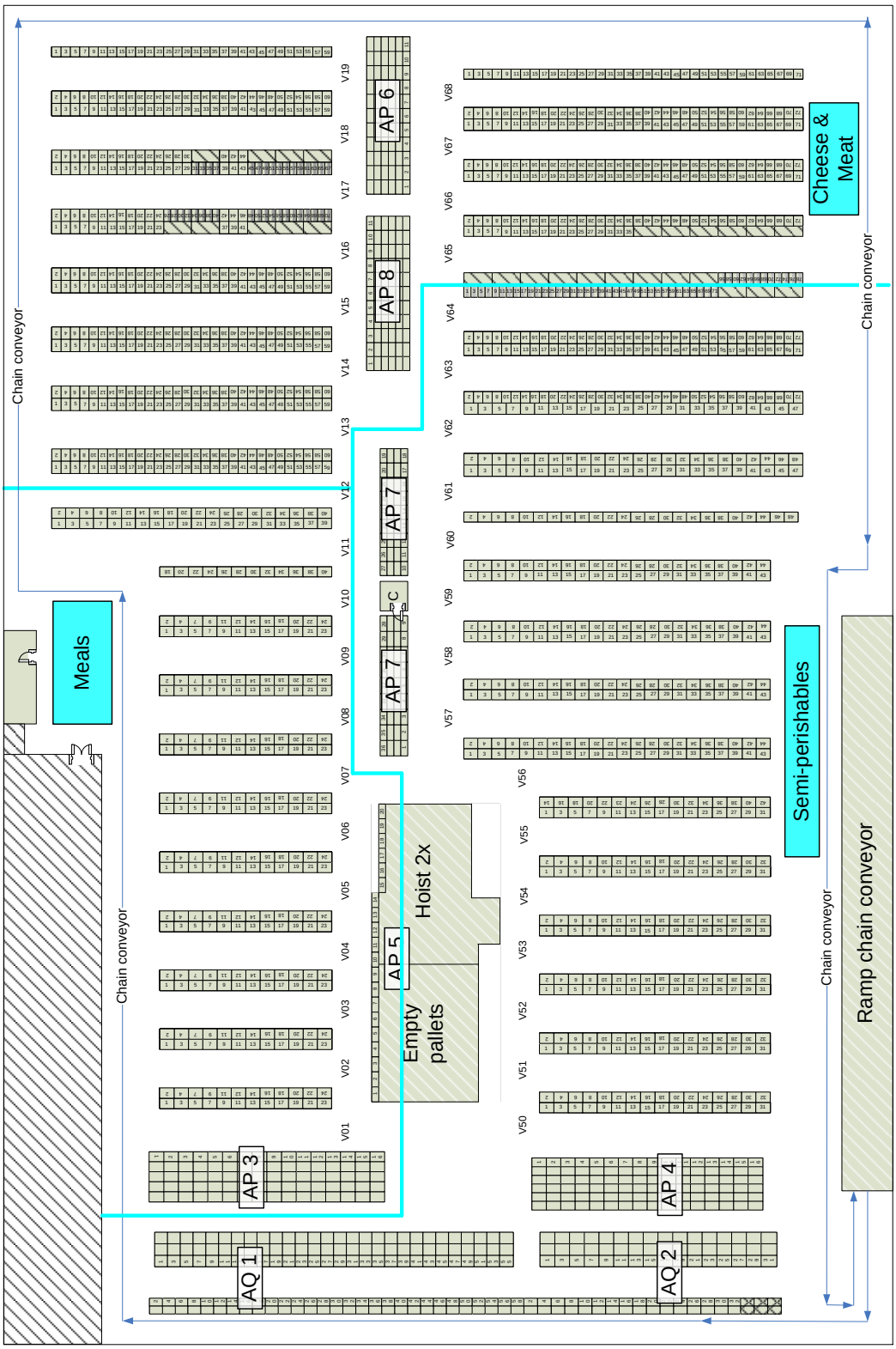
Node 16 corresponding to conveyor node after load node AGF		
Lane	Time (min)	Probability
P11	8.9	0.016
P12	8.5	0.177
P13	8.1	0.073
P14	7.7	0.043
P15	7.3	0.034
BAN	4.8	0.020
AB2	4.2	0.001
AP2	4.2	0.076
AB0	3.9	0.002
AP1	3.6	0.480
AB1	2.8	0.001
AP2	2.0	0.076

Node 13 corresponding to conveyor node after load node KAAS		
Lane	Time (min)	Probability
V12	6.8	0.011
V13	6.6	0.140
V14	6.3	0.007
V15	6.0	0.013
AP8	6.0	0.182
V16	5.7	0.037
V17	5.4	0.162
V18	5.2	0.014
V19	4.9	0.029
AP6	2.8	0.037
V68	0.9	0.018
V67	0.6	0.045
V66	0.3	0.024
V65	0.0	0.281



(a) Floor plan locations first floor.

Figure B.1: Floor plans.



(b) Floor plan locations second floor.

Figure B.1: Floor plans.

APPENDIX C

Pseudo code simulation of closed queueing networks allowing transfer and recirculation blocking and head-of-line priorities

The simulation is programmed in MATLAB. A graphical representation is given at page XXIV.

```
----------------------------------Input-----------------------------------
The total number of jobs
The total number of nodes
The total number of job classes
For all nodes N
  B(N)    Type of blocking at node N: equals 0 for transfer blocking or
          M if M is the replacement node
  X(N,R)  The service time distribution of class-R jobs at node N
  mu(N,R) The service time distribution parameter of class-R jobs at node N
  m(N)    The number of servers at node N
  c(N)    The node capacity at node N
For all nodes N, M and job classes R, S
  p(N,R,M,S) The routing probability: A class-R job in node N jumps
             towards node M in class S
  pr(N,R)    The priority of class R-jobs at node N

During the simulation, T equals the current time. We keep track on the
following properties of all jobs J at all simulated times T
  ST(J) service time until job J finished service. If job J is not in
        service, ST(J) is infinite
  PN(J) is the previous node of job J
  PC(J) is the previous class of job J
  PS(J) is the server of node PN(J) where J was served
  CN(J) is the current node of job J
  CC(J) is the current class of job J
  CS(J) is the current server of node CN(J) in which job J is served
  NN(J) is the next node job J will jump to
  NC(J) is the next class job J will be in
  NS(J) is the next server of node NN(J) where job J will be served
  BS(J) boolean; equals 1 if job J is blocking server CS(J) of node CN(J);
        equals 0 otherwise
```

```
--------------------------Performance measures-----------------------------
MNJ(N,R) Mean number class-R jobs in node N
MST(N,R) Mean sojourn time of class-R jobs in node N
MBT(N,R) Mean block time of class-R jobs in node N
MNV(N,R) Mean number of visits of class-R jobs to node N

-------------------------------Initialization--------------------------------
Place all jobs in a random node, taking the capacities into account, and
assign them a random class. For the jobs in a server, determine the
service time using the service time distributions.

--------------------------------Simulation-----------------------------------
WHILE *stopping criteria not satisfied*
  % Jump in time
  J  := next job that will finish its service
  dT := time until job J finishes its service
  T  := T + dT
  FOR all jobs J'
    ST(J') := ST(J') - dT
  END

  UPDATE for all nodes N and job classes R:
    MNJ(N,R), MST(N,R), MNV(N,R) and MBT(N,R)

  % determine next destination of job J
  NN(J) := next node of job J, determined using routing probabilities
  NC(J) := next class of job J, determined using routing probabilities
           and NN(J)

  % determine if recirculation blocking occurs
  IF number of job in node NN(J) equals c(NN(J)) and B(NN(J)) > 0
    % node NN(J) is full and recirculation blocking occurs
    NN(J) := B(NN(J))
    NC(J) := CC(J)
  END%if

  % determine if job J can jump to node NN(J)
  IF number of job in node NN(J) equals c(NN(J))
    % node NN(J) is full
    BS(J) := 1
  ELSE
    % job J jumps to node NN(J)
    PN(J) := CN(J)
    PC(J) := CS(J)
    PS(J) := CS(J)
    CN(J) := NN(J)
    CC(J) := NC(J)
```

```
    % we repeat this whileloop as long as there are blocked servers,
    % eventhough the blocking job is able to jump
    BOOL := 1
    WHILE (BOOL = 1)
      % job J either gets served at node CN(J) or joins the queue
      IF there is a free server at node CN(J)
        CS(J) := free server at node CN(J)
        ST(J) := random time using distribution X(CN(J),CC(J)) with
                 parameters mu(CN(J),CC(J))
      ELSE
        job J joins the queue of node CN(J)
      END%else

      % job J left node PN(J), so the server PS(J) becomes free
      IF there is a job in the queue of PN(J)
        J'     := job first in line of the queue at node PN(J)
        CS(J') := PS(J)
        ST(J') := random time using distribution X(CN(J),CC(J)) with
                  parameters mu(CN(J),CC(J))
      END%if

      % It might be true that a server is blocked due to the finite
      % capacity of node PN(J)
      BOOL := 0
      IF there is a blocked server due to finite capacity of node PN(J)
        BOOL  := 1
        J     := next blocking job due to finiteness in node PN(J), taking
                 pr(J) into account
        BS(J) := 0
        PN(J) :  CN(J)
        PC(J) := CC(J)
        PS(J) := CS(J)
        CN(J) := NN(J)
        CC(J) := NC(J)
      END%if
    END%while
  END%else
END%while
-------------------------------------------------------------------------------
```

START
T ← 0;
Determine the next job that finishes its service.
J ← next job that will finish its service;
dT ← time until job J finishes its service;
Time jump
T ← T + dT;
for all jobs J'
ST(J') ← ST(J') – dT;
Update the performance measures.
if T > 0.1RT
update MNJ(N,R), MST(N,R), MNV(N,R) and MBT(N,R) for all nodes N and job classes R;
Determine the next destination of job J using the routing probabilities. Take recirculation blocking into account if necessary
NN() ← next destination of job J;
NC() ← next job class of jobJ;
Is T > H?
yes
no
Is node NN() full?
yes
no
Job J is blocking node CN().
BS() ← 1;
END
Jump of job J.
PN() ← CN();
PC() ← CS();
PS() ← CS();
CN() ← NN();
CC() ← NC();
Is there a free server at node CN() at time T?
yes
no
Determine the service time of job J using the service time distribution X(CN()) of node CN().
ST() ← Service time of job J at node CN();
Job J joins the queue of node CN()
Is there a job in the queue of node PN() at time T?
yes
no
Is there a blocked server due to the finite capacity of node PN() at time T?
yes
no
The first job in the queue of PN() gets service. Determine the service time of that job using the service time distribution X(PN()) of node PN().
J'← the job first in the queue of node PN();
Put job J' in free server of node PN();
ST(J') ← Service time of job J' at node PN();
The next blocked job becomes unblocked (taking priorities into account) and the blocking job jumps to its next node.
J ← the first blocking job due to finite capacity of node PN() with hightes available priority;
BS() ← 0;
PN() ← CN();
PC() ← CS();
PS() ← CS();
CN() ← NN();
CC() ← NC();