Master’s Thesis

LIME & Technische Universiteit Eindhoven

Queueing Analysis of Circular Production Systems with Applications to Circular Economy

Author:
Kevin ten Braak (0740805)

Supervisors:
ir. J.P.E. Buskens
dr.ir. R.J.J.P.H. Bierbooms
dr.ir. M.A.A. Boon

Eindhoven, April 2016
## Contents

1 Introduction  

2 Assembly Systems  
   2.1 The pure assembly system with two unbounded buffers  
   2.2 The pure assembly system with one unbounded buffer  
   2.3 Assembly system with two bounded buffers  
   2.4 Assembly system with one bounded buffer  
      2.4.1 A generating functions approach  
      2.4.2 Matrix Geometric approach for assembly Systems with one bounded buffer and one unbounded buffer  
      2.4.3 Matrix Geometric approach for Assembly Systems with two limited buffers  

3 An algorithm to analyze circular production systems  
   3.1 A non circular production system  
      3.1.1 Derivation of the algorithm  
      3.1.2 Algorithm A  
   3.2 A production line with tandem assembly queues and loops  
      3.2.1 Derivation of the algorithm  
      3.2.2 Algorithm B  
   3.3 A production line with tandem assembly queues, loops and sources  
      3.3.1 Derivation of the algorithm  
      3.3.2 Algorithm C  

4 Results  
   4.1 Results Case 1  
   4.2 Results Case 2  
   4.3 Results Case 3  
   4.4 Results Case 4  
   4.5 Results Case 5  
   4.6 Results Case 6  
   4.7 Results Case 7  
   4.8 Results Case 8  
   4.9 Results Case 9, Case 10 and Case 11  

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>12</td>
</tr>
<tr>
<td>2.4.1</td>
<td>14</td>
</tr>
<tr>
<td>2.4.2</td>
<td>20</td>
</tr>
<tr>
<td>2.4.3</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>27</td>
</tr>
<tr>
<td>3.1.1</td>
<td>29</td>
</tr>
<tr>
<td>3.1.2</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>33</td>
</tr>
<tr>
<td>3.2.1</td>
<td>35</td>
</tr>
<tr>
<td>3.2.2</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>38</td>
</tr>
<tr>
<td>3.3.1</td>
<td>39</td>
</tr>
<tr>
<td>3.3.2</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>48</td>
</tr>
<tr>
<td>4.4</td>
<td>49</td>
</tr>
<tr>
<td>4.5</td>
<td>50</td>
</tr>
<tr>
<td>4.6</td>
<td>51</td>
</tr>
<tr>
<td>4.7</td>
<td>52</td>
</tr>
<tr>
<td>4.8</td>
<td>53</td>
</tr>
<tr>
<td>4.9</td>
<td>55</td>
</tr>
</tbody>
</table>
5 Conclusion and discussion

5.1 Conclusion and recommendations regarding the algorithm .......................... 59
5.2 Conclusion and recommendations to use the model for the quantification of the circular economy ......................................................................................... 60
Chapter 1

Introduction

The circular economy
The concept of a circular economy, introduced by David Pearce in 1990, is an industrial system that emphasizes on closing resource loops. In an circular economy, waste is minimized by the restorative or regenerative intention and design of the economy. In the perfect case waste would not even exist and products would be designed and optimized for a cycle of disassembly and reuse. Such a concept should reduce frequently used linear 'take-make-dispose' approach, which is leading to scarcity, volatility, and pricing levels that are unaffordable for the manufacturing base of economies.

Much research about achieving a circular economy and its benefits has already been done. Many of these projects and studies focus mainly on the opportunities this circular economy has, and how this economy could be achieved. Most papers agree on three main goals that one strives for when trying to realize a circular economy:

1. To reduce dependencies on scarce natural resources.
2. To allow companies to generate revenue from 'waste'.
3. To create more innovative, responsive, and customer-centric products and services.

Andersen (2007) provides an introduction to some of the fundamental principles and approaches in environmental economics, mainly based on the system designed by Pearce (1990). The Ellen MacArthur Foundation (2013), abbreviated as EMF, sketches the shift of material in a circular economy in a broad and detailed manner, see Figure 1.1. Phillips (2014) describes the same circular shift of material only relevant to the company itself. Philips already provides some circular services as it sells light as a service to some companies, instead of selling the products that create the light. In this business strategy, Philips keeps the ownership of her products, which creates benefits in the circular resource productivity. In this project, we assume that the model presented by EMF is a realistic presentation of the circular economy, and we use it as a base for the mathematical model that is developed in the project.
CHAPTER 1. INTRODUCTION

The main focus of most papers about circular economy are rather theoretical and they do not describe the specific actions a company needs to take in order to ‘get circular’. An exception is the Circularity Assessment Tool (CAT) that the company Circle Economy is developing at the moment. This Tool gives a measure of the circularity of companies and investment portfolios. The idea is that investors and organizations using the tool gain insights on areas to improve order to reduce portfolio and business risks, identify new opportunities, enable long-term growth, and in the end be able to become more circular.

These tools are exactly the kind of business opportunities the current business environment is craving for. Many theoretical analyses of the circular economy have been made. However, to convince companies to change their linear policy to a more circular one, the benefits of such a circular policy should be quantified instead of qualified. By quantifying the profit and the environmental benefits that could be made and created by investing in circularity, the threshold for companies to adapt to a more circular environment can be lowered. In this thesis we introduce a novel approach to perform a quantitative analysis of systems in the circular economy, based on queueing theory.

**Queueing theory**
Queueing theory is the mathematical study of waiting lines, or queues. The study is considered to be a branch of operations research because of the many practical applications it has in environments where business related decisions need to be made. Multiple common examples of queueing models
are traffic lights, telecommunication, queues in the supermarket, and production chains.

A queueing model usually consists of one or more queues in which customers have to wait before they can be served by one or more servers. Many variations of these models exist and many researches about these queueing models in (circular) production lines have been done. For example, Papadopoulos and Heavey (1996) provide a bibliography of material concerned with modeling of production and transfer lines using queueing networks. An approach for estimating the throughput of closed cyclic queueing networks with blocking is derived by Tolio and Gershwin (1998).

In general, a model of the circular economy shares certain features with a typical production chain. In those production chains situations often occur in which two items are combined together. A suitable queueing model for combining items or pairing customers is the assembly queue. An assembly queue has, in contrast to an ordinary queueing model, two or more queues (one for every type of customer/item). Only when at least one customer of every type is present in the queues, the server can serve both of these customers and pair them together. Where in a standard queueing model a customer only has to wait for the customers in front of him to be served, in an assembly queue he also has to wait for the arrival of a customer of the other type. Because of this extra dependency, assembly systems are more difficult to analyze than a normal queueing model. Moreover, even with the existing literature of production environments, it is well-known that circular production systems with fork-join components are notoriously hard to analyse. Although queueing theory is a very common technique in many applications, such as telecommunication, road traffic, computer networks, and manufacturing environments, it has not been used in the context of circular economy, to the best of our knowledge.

**Circular economy and assembly queues combined**

As the quantitative analysis of circular models within the circular economy is the main goal of this project, a simplified version of the model of the Ellen MacArthur Foundation (Figure 1.1) is constructed (see Figure 1.2).

Raw materials are obtained from Sources. Thereafter, the raw materials are cultivated by the Raw Material Manufacturer and subsequently transported to the Parts Manufacturer who combines (different) materials into parts/components. The components are, at their turn, transported to the Product Manufacturer which combines the components into products that can be sold or leased by the Service Provider. Note that there are two nodes of User in the model. The User on the left is part of the circular economy of the product. Here the product is sold as a service, although the product is at the user, it is still property of the company. The green circle indicates that the company producing the product has control over the flow of the materials, parts, and products from the User to the other nodes. The green circles (including the circle around the Sources) thus stand for the places where the decisions are made that have the biggest influence on the total circular flow of the product.

The yellow orange rectangle represents the part of the production flow where a product is sold and the company loses control over the flow of the product. The User can decide whether to resell the product, sell back the product to the service provider (or store), or to dispose the product. Selling back and reselling the product are also part of the circular economy. Selling back however, does
CHAPTER 1. INTRODUCTION

Figure 1.2: The circular model of product

not happen with many different product. Services such as Ebay however, have stimulated the flow
of resold items. Unfortunately, there is still a huge part of products that is disposed by the user,
while a company could have recycled, refurbished, remanufactured, redistributed or maintained it.
For this reason, we are particularly interested in the big circular part of the sketch in Figure 1.2.

In this thesis we build a queueing model that can be used to approximate the performance mea-
sures of circular production chains. The algorithm is applied to and tested on a model that closely
resembles the circular model of Figure 1.2. Subsequently, the algorithm is evaluated and the results
are discussed. However, since every production chain is different for each product, it is the way of
constructing the approximation algorithm that gives this project its value.

The main two goals of this research are:

1. To analyze circular queueing models in which assembly queues are involved and to analyze
   the way of constructing algorithms that can approximate important performance measures.

2. To take a first step in linking queueing theory to the quantification of the possibilities and
   benefits that a circular production chain has to offer.
The methods and algorithms that are used in this research are based on the paper of Gerswhin (1987) in which a decomposition method is considered that approximates performance measures of tandem queues with finite storage space and blocking. The accuracy of the results of the method are determined by comparing the results of the approximation with the results of a simulation. In this research we also use a decomposition method of which the accuracy is evaluated by a simulation. However, the difference with the tandem queues that Gerswhin considered, is that we use queueing models that include circularity as well as assembly queues.

In Section 2 an assembly system is considered and multiple ways of determining the equilibrium probabilities (provided that they exist) are described. In Section 3 a model is built that resembles the model sketched in Figure 1.2. Step by step the model is expanded, and accordingly, algorithms are derived for approximating important performance measures. In Section 4 several cases are considered of which performance measures are computed by simulation and approximated by the algorithm of Section 3. In the last section, Section 5, the results are discussed and recommendations for further research are given.
Chapter 2

Assembly Systems

To be able to analyze circular production chains in which items are combined together, we first need to consider the assembly process itself as it is more complex than the average process in more general queueing systems. A queueing system that combines multiple items into one, is called an assembly queue. In an assembly queue multiple types of customers arrive and each type joins a different queue. The server only starts serving when specific combinations of types of customers are present in the queues. The server then serves all the customers of this specific combination simultaneously. In most cases found in the literature, the combination of exactly one customer of every type is the only combination that the server serves. Note that in the case of the model presented in Chapter 1 (where the customers are materials, components or products) it is also possible to allow combinations that do not consist of materials or components of each type.

Assembly queues have a prominent place in most (circular) production chains as many products are made from different types of materials, which preferably are recyclable. It is the process of combining these materials that can be analyzed by assembly queues.

In this chapter, we will restrict ourselves to models with two customer types only. Extending the analysis to multiple customers types is known to be very challenging and beyond the scope of this thesis. In the next chapter, we develop approximation methods, based on results from this chapter, that can be applied to any number of customer types.

An assembly system for which the time needed to transform a complete group of customers into an assembled group is zero, is called a pure assembly system (see Figure 2.1). The arrivals to the queueing models considered in this chapter are assumed to be Poisson distributed. Poisson arrivals imply exponential distributed arrival times. When a random variable $X$ is Poisson or exponentially distributed with parameter $\lambda$, this is denoted by $X \text{ Poi}(\lambda)$ or $X \text{ Exp}(\lambda)$ respectively.
CHAPTER 2. ASSEMBLY SYSTEMS

Ferschl (1991) states that it is always possible to separate an assembly system into a pure assembly system, followed by a queueing system with a single waiting line of complete sets that have to be processed/combined into one product. Given the arrival rate of complete sets at the queue, the latter is not hard to analyze.

The remainder of this chapter is organized as follows: To get more insight in the stability conditions of assembly queues, we consider pure assembly systems (without server). In Section 2.1, we analyze a pure assembly queue with unlimited buffer space for both of its queues and we prove the system to be unstable. In Section 2.2 we perform the same analysis for a pure assembly queue which has one buffer with an unlimited buffer space. A stability condition for this system is derived.

In Sections 2.3 and 2.4 two (not pure) assembly queues are considered. For each assembly queue a flow diagram is presented and the corresponding balance equations are derived. In Section 2.3 an assembly queue with one buffer with unlimited buffer space is considered, while in Section 2.4 an assembly queue is considered of which both buffers have limited buffer space. In the last two sections, approximation methods for the equilibrium probabilities of assembly queues with at least one buffer with limited space, are derived. In Section 2.4.1, the generating function approach, with which equilibrium probabilities for an assembly queue with one unbounded buffer can be computed, is described. In sections 2.4.2 and 2.4.3 the matrix geometric approach, for computing the equilibrium probabilities of assembly queues with either one bounded buffer or two bounded buffers, is described.

2.1 The pure assembly system with two unbounded buffers

In this section, a queueing system consisting of one assembly queue and two types of arriving customers is considered. The presence of a server is ignored. Two types ($K = 1, 2$) of customers arrive at the pure assembly system according to Poisson arrival streams with parameters $\lambda_1$ and $\lambda_2$ respectively. We take the service time to be equal to zero. For now, only the set $\{1, 2\}$ is a possible combination that can leave the system immediately. We model the system as a Continuous-time Markov chain described by $(n_1, n_2)$, where $n_1$ and $n_2$ denote the number of customers of type 1 and 2 respectively present in the system. Three cases are distinguished:
1. there are no customers in the system: \( n_1 = n_2 = 0 \),
2. there are only customers of type 1 in the system: \( n_1 > 0, n_2 = 0 \),
3. there are only customers of type 2 in the system: \( n_2 > 0, n_1 = 0 \).

Note that, since a combination of customers of type 1 and 2 can immediately leave the system without delay, the customers present in the system will all be of the same type. The state of the system is described by the number of customers of each type. When \( n_1 \) customers of type 1 are present in the system, the system is in state \((n_1,0)\). Upon arrival of a type 1 customer, the system goes to state \((n_1+1,0)\), but on arrival of a type 2 customer the system goes to state \((n_1-1,0)\). In Figure 2.2 the corresponding flow diagram is presented.

![Flow diagram for the pure assembly queue with two types of customers](image)

**Figure 2.2:** Flow diagram for the pure assembly queue with two types of customers

**Proposition 2.1.1: the Markov chain is not ergodic**

*Proof.* The Markov chain does not have an equilibrium distribution, meaning that the system is always unstable. We distinguish two cases. In case 1: \( \lambda_1 \neq \lambda_2 \). Without loss of generality, we assume \( \lambda_1 > \lambda_2 \). In this case the average incoming number of type 1 customers is bigger than the average incoming number of type 2 customers which causes the queue of type 1 customers to blow up as time elapses. Because of symmetry, \( \lambda_1 < \lambda_2 \) is not stable either. In case 2: \( \lambda_1 = \lambda_2 \); one would expect the assembly system to be stable, since the ratio between the arrival rates of the two customers is equal to the ratio of customers needed to complete a set that can leave the assembly system. However, the opposite is true. Even when the ratio between arrival rates is equal to the ratio of customers that is needed to complete a set, the system is unstable. The latter can be seen by making a link to a simple symmetric random walk.

If \( \{X_k\} \) are independent and identically distributed (i.i.d.) random variables with \( \mathbb{P}(X_k = 1) = 1 - \mathbb{P}(X_k = -1) = p \) and \( n \in \mathbb{N} \), then a *simple random walk* is a stochastic sequence \( \{S_n\} \), with \( S_0 = 0 \), defined by

\[
S_n = \sum_{k=1}^{n} X_k.
\]

The simple random walk is a walk that can only make steps of \( \pm 1 \). In a simple random walk that is *symmetric*, the probabilities of stepping left or right are equal, so \( p = \frac{1}{2} \).
Note that, when considering the assembly system (with equal arrival rates) at embedded epochs, immediately after a transition to a new state, we obtain a simple random walk, for which it has been proven (Alm 2002) that it is not a stable system as the expected time of returning to a starting point is infinite.

This brings us to the conclusion that whatever value the arrival rates have or what their relative ratio is, the pure assembly system with unlimited buffers will never be stable. A formal proof of instability of pure assembly systems with \( K \geq 2 \) types of customers and unlimited buffer is given by Harrison (1973).

### 2.2 The pure assembly system with one unbounded buffer

In this section we again consider a pure assembly system with two types of arriving customers of which only one has an unlimited buffer, assume this is type 1. The maximum allowed number of customers of type 2 is \( K \). In the rest of this report we assume that when a queue has a buffer of size \( K \), it includes the space of the customer that is currently in process by the server. It is also conventional to say that the ‘real’ buffer in this case should be \( K - 1 \), but for sake of simplicity the customer in process also uses one place of the buffer. This model corresponds to the following flow diagram (see Figure 2.3).

![Flow diagram for the pure assembly queue with two types of customers with one bounded buffer](image)

**Figure 2.3: Flow diagram for the pure assembly queue with two types of customers with one bounded buffer**

**Proposition 2.2.1:** The stability condition for this model is \( \tilde{\rho} = \frac{\lambda_1}{\lambda_2} < 1 \).

**Proof.** Note that, it is clear that this assembly system is equivalent to an \( M/M/1 \) model where the service rate \( \mu \) is equal to the arrival rate of customers of type 2. Consequently, the stability condition for this model is

\[
\hat{\rho} = \frac{\lambda_1}{\lambda_2} < 1.
\]

Here \( \hat{\rho} \) is the occupation rate \( \frac{\lambda_1}{\lambda_2} \) in the corresponding \( M/M/1 \) queue, but in the original model it can be considered as a sort of relative arrival rate. Simple calculations by balance equations, recursion or generating function approach give the equilibrium distribution.
2.3 Assembly system with two bounded buffers

In this section we consider the flow diagram and balance equations of an assembly system with two bounded buffers, as they provide useful insights for derivation of the algorithm of Section 2.4.3. In this model we assume a server is present and that it has $\text{Exp}(\mu)$ distributed assembly times. There are still only two customers types that enter the system with Poisson arrivals (with $\text{Poi}(\lambda_1)$ and $\text{Poi}(\lambda_2)$ distributions). In this type of system both the queue lengths of customers of type 1 and 2 are bounded by $K_1$ and $K_2$ respectively. This ensures the system to be stable since it is not possible for either one of the queues to grow infinitely large.

The balance equations for this model are given by:

$$p_{a,b} = \tilde{\rho}^{a+K-b}(1 - \tilde{\rho}),$$

where $a$ and $b$ are the number of customers of type 1 and 2 respectively.
(a) \( (\lambda_1 + \lambda_2)p_{0,0} = \mu p_{1,1} \)

(b) \( \lambda_2 p_{K_1,0} = \lambda_1 p_{K_1-1,0} \)

(c) \( \lambda_1 p_{0,K_2} = \lambda_2 p_{0,K_2-1} \)

(d) \( \mu p_{K_1, K_2} = \lambda_2 p_{K_1, K_2-1} + \lambda_1 p_{K_1-1, K_2} \)

(e) \( (\lambda_1 + \lambda_2)p_{i,0} = \mu p_{i+1,0} + \lambda_1 p_{i-1,0} \) for \( i = 1, ..., K_1 - 1 \)

(f) \( (\lambda_1 + \lambda_2)p_{0,j} = \mu p_{1,j+1} + \lambda_2 p_{0,j-1} \) for \( j = 1, ..., K_2 - 1 \)

(g) \( (\lambda_2 + \mu)p_{K_1,j} = \lambda_2 p_{K_1,j-1} + \lambda_1 p_{K_1-1,j} \) for \( j = 1, ..., K_2 - 1 \)

(h) \( (\lambda_1 + \mu)p_{i,K_2} = \lambda_1 p_{i-1,K_2} + \lambda_2 p_{i,K_2-1} \) for \( i = 1, ..., K_1 - 1 \)

(i) \( (\lambda_1 + \lambda_2 + \mu)p_{i,j} = \mu p_{i+1,j+1} + \lambda_1 p_{i-1,j} + \lambda_2 p_{i,j-1} \) for \( i = 1, ..., K_1 - 1 \) and \( j = 1, ..., K_2 - 1 \)

Equations (2.3.1.a), (2.3.1.b), (2.3.1.c), and (2.3.1.d) correspond to the four corners in the flow diagram, equations (2.3.1.e), (2.3.1.f), (2.3.1.g), and (2.3.1.h) correspond to the four edges in the flow diagram, and equation (2.3.1.i) corresponds to the nodes that are not in the boundary of the flow diagram. Since we want to solve the balance equations for all \( p_{i,j} \) (\( i = 0, ..., K_1, j = 0, ..., K_2 \)), at least \( (K_1 + 1) \cdot (K_2 + 1) \) independent equations are needed. Since the balance equations above only provide \( (K_1 + 1) \cdot (K_2 + 1) - 1 \) independent equations, we add the normalization condition:

\[
\sum_{i=0}^{K_1} \sum_{j=0}^{K_2} p_{i,j} = 1. \tag{2.3.2}
\]

Solving the equations (2.3.1) and (2.3.2) for \( p_{i,j} \) (\( i = 0, ..., K_1, j = 0, ..., K_2 \)), leads to the equilibrium probabilities \( p_{i,j} \) which can be found numerically very efficient. However, since the set of \( K_1 \cdot K_2 \) of equations can get really big for increasing \( K_1 \) and \( K_2 \), another way of solving the equilibrium probabilities for this system is presented in Section 2.4.3.

2.4 Assembly system with one bounded buffer

In this section we consider the flow diagram and balance equations of an assembly system with one bounded buffers, as they provide useful insights for derivation of the algorithm of Section 2.4.2. Again assume \( \text{Exp}(\mu) \) distributed service times and Poisson arrival processes. However, now only the customers of type 1 have a bounded buffer, the number of customers of type 2 can grow unlimited.
2.4. ASSEMBLY SYSTEM WITH ONE BOUNDED BUFFER

Figure 2.5: Flow diagram for the pure assembly queue with two types of customers, one bounded buffer and one unbounded buffer

The balance equations are almost identical to the balance equations of the system in Section 2.3 except that there are three equations missing. Two of these equations vanished due to the missing corners of the flow diagram and one equation vanished due to the missing edge at the now unbounded buffer of the customers of type 2. Also note that the range of the index $j$ is now unlimited.

(a) $(\lambda_1 + \lambda_2)p_{0,0} = \mu p_{1,1}$

(b) $\lambda_2 p_{1,0} = \lambda_1 p_{K_1-1,0}$

(c) $(\lambda_1 + \lambda_2)p_{i,0} = \mu p_{i+1,1} + \lambda_1 p_{i-1,0}$ for $i = 1, \ldots, K_1 - 1$

(d) $(\lambda_1 + \lambda_2)p_{0,j} = \mu p_{1,j+1} + \lambda_1 p_{0,j-1}$ for $j = 1, 2, \ldots$

(e) $(\lambda_2 + \mu)p_{K_1,j} = \lambda_2 p_{K_1,j-1} + \lambda_1 p_{K_1-1,j}$ for $j = 1, 2, \ldots$

(f) $(\lambda_1 + \lambda_2 + \mu)p_{i,j} = \mu p_{i+1,j+1} + \lambda_1 p_{i-1,j} + \lambda_2 p_{i,j-1}$ for $i = 1, \ldots, K_1 - 1$ and $j = 1, 2, \ldots$
Equations (2.4.1.a) and (2.4.1.b) correspond to the only two corners in the flow diagram, equations (2.4.1.c), (2.4.1.d), and (2.4.1.e) correspond to the three edges in the flow diagram, and equation (2.4.1.f) corresponds to the nodes that are not in the boundary of the flow diagram. Again, the normalization condition is needed:

\[
\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} p_{i,j} = 1.
\]  

Contradictory to the system of equations in Section 2.3, this system consists of infinitely many equations whereby it is not possible to merely solve the balance equations ‘by hand’. In the two following sections two methods that are able to solve the equilibrium probabilities are presented. In Section 2.4.1 an approach that uses probability generating functions is described. The approach in Section 2.4.2 uses the Matrix Geometric approach. In the algorithms derived in Section 3 the Matrix Geometric approach is used as the generating functions approach is only applicable to assembly queues with one buffer with unlimited space. Thereby, the Matrix Geometric approach makes use of matrices, which are practical for the derived algorithms. The advantage of the generating functions approach is that generating functions are very useful for computing (high-order) moments of a distribution. Thereby, the equilibrium probabilities of a system can also quickly be obtained by computing the coefficients of the Taylor expansion of the corresponding generating function.

2.4.1 A generating functions approach

In this section, the system of Section 2.4 is considered in terms of generating functions in order to solve the equilibrium probabilities of an assembly system. The generating function approach is a well-established technique that results in a symbolic expression for the transforms of random variables which, to the experts, can be more insightful and provide additional information. However, since the generating functions method is only applicable to assembly queues with one buffer that has an infinity capacity, it is not used in the remainder of my research.

This section is arranged as follows: First we define the generating functions belonging to the assembly system. Next, the difference equations are solved in order to find a closed expression for the generating functions. Subsequently, an explicit formula for \( G_0(z) \) is given and thereafter we describe a manner to calculate the unknown terms \( G_n(0) \). Finally, in the end of this section it is explained how the equilibrium probabilities can be extracted from the computed generating functions.

The generating function of the equilibrium probabilities of an assembly queue

For the ease of notation instead of \( K_1 \), \( K \) is used as the maximum number of customers of type 1 allowed in the assembly system. The stability condition for the assembly queue with one bounded buffer is not derived in this section, since it is derived in Section 2.4.2.
We define
\[ p_{i,j} = P(N_1 = i, N_2 = j) \quad \text{for } i = 0, 1, ..., K, \text{ and } j = 0, 1, 2, ... \]
where \( N_1 \) and \( N_2 \) represent the number of customers of type 1 and type 2 respectively. Then the probability generating function \( G_i \) is given by:
\[ G_i(z) = \sum_{j=0}^{\infty} p_{i,j} z^j. \]

By multiplying the balance equations (2.4.1) by \( z^j \) and summing over \( j \) from 0 to infinity, the following relations between the generating functions \( G_i(z) \) are obtained;
\[ G_0(z) = \frac{\mu G_1(z) - G_1(0)}{z(\lambda_1 + \lambda_2(1 - z))}, \quad (2.4.3) \]
derived from balance equations (2.4.1.a) and (2.4.1.d),
\[ G_K(z) = \frac{\mu G_K(0) + \lambda_1 G_{K-1}(z)}{\mu + \lambda_2(1 - z)}, \quad (2.4.4) \]
derived from balance equations (2.4.1.b) and (2.4.1.e), and
\[ G_i(z) = \frac{\mu (G_{i+1}(z) - G_{i+1}(0)) + \lambda_1 G_{i-1}(z) + \mu G_i(0)}{\lambda_1 + \lambda_2(1 - z) + \mu} \quad \text{for } i = 1, ..., K - 1, \quad (2.4.5) \]
derived from balance equations (2.4.1.c) and (2.4.1.f).

Finding a closed form expression for \( G_i(z) \)

Next, the balance equations of Section 2.4.1 are solved in terms of \( G_i(z) \) by using standard methods for solving difference equations.

Lemma 2.4.1.1: Closed form expression for \( G_i(z) \)

Given equations (2.4.3), (2.4.4), and (2.4.5), the closed form expression of \( G_i(z) \) is given by
\[ G_i(z) = (d_1[z](x_1[z])^i + d_2[z](x_2[z])^i)G_0(z) + \sum_{n=1}^{i} ((c_1[z](x_1[z])^{i-n} + c_2[z](x_2[z])^{i-n})G_n(0)). \quad (2.4.6) \]
with
\[ x_1[z] = \frac{1}{2}(m[z] - \sqrt{m^2[z] - 4q[z]}), \quad x_2[z] = \frac{1}{2}(m[z] + \sqrt{m^2[z] - 4q[z]}), \quad (2.4.7) \]
\[ d_1[z] = \frac{m[z] + \sqrt{m^2[z] - 4q[z]} - 2w[z]}{2\sqrt{m^2[z] - 4q[z]}}, \quad d_2[z] = \frac{-m[z] + \sqrt{m^2[z] - 4q[z]} + 2w[z]}{2\sqrt{m^2[z] - 4q[z]}}, \quad (2.4.8) \]
Proof. First we rewrite (2.4.3) into
\[ G_1(z) = \frac{z}{\mu} (\lambda_1 + \lambda_2(1-z))G_0(z) + G_1(0), \] \tag{2.4.9}
and (2.4.5) into
\[ G_i(z) = \frac{z}{\mu}((\lambda_1 + \lambda_2(1-z) + \mu)G_{i-1}(z) - \lambda_1 G_{i-2}(z) - \mu G_{i-1}(0)) + G_i(0) \quad \text{for } i = 2, ..., K. \tag{2.4.10} \]

From now on, for the purpose of abbreviation, the following functions are used in this section:
\[ q[z] = \frac{z\lambda_1}{\mu}, \quad \tag{2.4.11} \]
\[ m[z] = \frac{z(\lambda_1 + (1-z)\lambda_2 + \mu)}{\mu}, \quad \tag{2.4.12} \]
\[ w[z] = \frac{z(\lambda_1 + (1-z)\lambda_2)}{\mu}. \tag{2.4.13} \]

Now (2.4.9) and (2.4.10) can be written as
\[ G_1(z) = w[z]G_0(z) + G_1(0), \tag{2.4.14} \]
and
\[ G_i(z) = m[z]G_{i-1}(z) - q[z]G_{i-2}(z) - zG_{i-1}(0) + G_i(0) \quad \text{for } i = 2, ..., K. \tag{2.4.15} \]

It is now evident, that by using the equations (2.4.14) and (2.4.15) every \( G_i(z) \) can be expressed in terms of \( G_0(z) \) and \( G_n(0) \) \((1 \leq n \leq i)\). Subsequently, the equations are split into equations in terms of \( G_i(z) \) and \( G_n(0) \). First consider the equations in terms of \( G_i(z) \), which are named \( \hat{G}_i(z) \) in order to avoid confusion with the actual \( G_i(z) \).

\[ \begin{align*}
(a) \quad \hat{G}_0(z) &= G_0(z), \\
(b) \quad \hat{G}_1(z) &= w[z]G_0(z), \\
(c) \quad \hat{G}_i(z) &= m[z]\hat{G}_{i-1}(z) - q[z]\hat{G}_{i-2}(z).
\end{align*} \tag{2.4.16} \]

Clearly this is a recursive sequence (or difference equation), for which an explicit formula can be found by finding the roots of the characteristic polynomial
\[ = (x[z])^2 - m[z]x[z] - q[z] = 0. \tag{2.4.17} \]

The solutions to this characteristic polynomial are
\[ x_1[z] = \frac{1}{2}(m[z] - \sqrt{m^2[z] - 4q[z]}), \quad x_2[z] = \frac{1}{2}(m[z] + \sqrt{m^2[z] - 4q[z]}), \tag{2.4.18} \]
2.4. ASSEMBLY SYSTEM WITH ONE BOUNDED BUFFER

which implies the general solution

\[ \hat{G}_i(z) = (d_1[z](x_1[z])^i + d_2[z](x_2[z])^i)G_0(z). \]  
\[ (2.4.19) \]

The functions \( d_1[z] \) and \( d_2[z] \) can be found by using the equations (2.4.16.a) and (2.4.16.b). Solving the system of equations

\[
\begin{align*}
d_1[z] + d_2[z] &= 1, \quad (2.4.20) \\
d_1[z]x_1[z] + d_2[z]x_2[z] &= w[z], \quad (2.4.21)
\end{align*}
\]

leads to

\[
\begin{align*}
d_1[z] &= \frac{m[z] + \sqrt{m^2[z] - 4q[z]} - 2w[z]}{2\sqrt{m^2[z] - 4q[z]}}, \quad (2.4.22) \\
d_2[z] &= \frac{-m[z] + \sqrt{m^2[z] - 4q[z]} + 2w[z]}{2\sqrt{m^2[z] - 4q[z]}}, \quad (2.4.23)
\end{align*}
\]

To find an explicit formula for the coefficients of the terms of \( G_n(0) \) in \( G_i(z) \) is a bit more tricky, but follows the same process. Assume the pgf (probability generating function) \( G_i(z) \) is the one that needs to be computed. From (2.4.15) it is clear that \( G_i(z) \) contains one term \( G_i(0) \) with coefficient 1. One iteration shows that \( G_i(z) \) contains \( G_{i-1}(0) \) with coefficient \(-z + m[z]\). Define \( F(n) \) (with \( 0 \leq n \leq i \)) to be the coefficient of \( G_n(0) \) in the expression of \( G_i(z) \). Then \( F(n) \) satisfies the following recurrence equations:

\[
\begin{align*}
(a) & \quad F(0) = 1, \\
(b) & \quad F(1) = -z + m[z], \\
(c) & \quad F(n) = -q[z]F(n-2) + m[z]F[n-1]. \quad (2.4.25)
\end{align*}
\]

The explicit formula for \( F(n) \), is given by

\[ F(n) = c_1[z](x_1[z])^{i-n} + c_2[z](x_2[z])^{i-n}, \quad \text{for } n = 1, \ldots, i. \]  
\[ (2.4.26) \]

Note that (2.4.25.c) is exactly the same difference equation as (2.4.16.c), which results in the same solutions \( x_1[z] \) and \( x_2[z] \) as given in (2.4.7). However, since the initial terms of (2.4.25) differ from (2.4.16), the coefficients \( c_1[z] \) and \( c_2[z] \) differ from \( d_1[z] \) and \( d_2[z] \):

\[
\begin{align*}
c_1[z] &= \frac{2z + \sqrt{m^2[z] - 4q[z]} - m[z]}{2\sqrt{m^2[z] - 4q[z]}}, \quad c_2[z] = \frac{-2z + \sqrt{m^2[z] - 4q[z]} + m[z]}{2\sqrt{m^2[z] - 4q[z]}}. \quad (2.4.27)
\end{align*}
\]

Combining the results of (2.4.19) and (2.4.26) leads to the closed form expression of \( G_i(z) \).
CHAPTER 2. ASSEMBLY SYSTEMS

Finding the explicit formula of $G_0(z)$

In this section we present the explicit formula of $G_0(z)$.

Theorem 2.4.1.1: Explicit formula of $G_0(z)$

The explicit formula of $G_0(z)$ is given by

$$G_0(z) = \left( \frac{\mu}{\mu + \lambda_2(1-z)} G_K(0) + \frac{\lambda_1}{\mu + \lambda_2(1-z)} \sum_{n=1}^{K-1} ((c_1[z](x_1[z])^{k-1-n} + c_2[z](x_2[z])^{k-1-n})G_n(0)) 
- \sum_{n=1}^{K} ((c_1[z](x_1[z])^{k-n} + c_2[z](x_2[z])^{k-n})G_n(0)) \right) / \left( (d_1[z](x_1[z])^K + d_2[z](x_2[z])^K) - \frac{\lambda_1}{\mu + \lambda_2(1-z)} (d_1[z](x_1[z])^{K-1} + d_2[z](x_2[z])^{K-1}) \right)$$

(2.4.28)

Proof. In Lemma 2.4.1.1, a closed formula for $G_i(z)$ was derived in terms of the still unknown $G_0(z)$ and $G_n(0)$'s (for $1 \leq n \leq i$). Note that (2.4.4) is not used yet in the derivation of $G_i(z)$, and therefore can be used to find a formula for $G_0(z)$, independent of $G_i(z)$ (where $0 \leq i \leq K$). Substituting the formulas for $G_K(z)$ and $G_{K-1}(z)$, which follow from (2.4.6), into (2.4.4) gives the desired result.

Solving the $G_n(0)$

Formula (2.4.28) gives an explicit formula of $G_0(z)$. However, there are still some unknown terms in the formula: the $G_n(0)$ for $n = 1, ..., K$. Winands et al. (2009) describes a way to calculate these terms. Assume that

$$G(z) = \sum_{i=0}^{K} G_i(z). \quad (2.4.29)$$

From (2.4.6) and (2.4.28) it is clear that $G(z)$ has a denominator of at least degree $K$. Now, since $G(z)$ is bounded in $|z| \leq 1$, the zeros within the unit circle of the numerator of $G(z)$ must be canceled by corresponding zeros of the denominator. One of the zeros equals 1 and leads to a trivial equation. However, the normalization condition

$$\sum_{i=0}^{K} G_i(z) \bigg|_{z=1} = 1 \quad (2.4.30)$$

provides the extra equation that is needed in order to get a set of $K$ linear equations. Unfortunately, it is possible for the numerator of (2.4.28) to have roots within the circle that coincide, so that they lead to the same linear equation. However, for $G(z)$ to be bounded in $|z| \leq 1$, the numerator should not only have the same roots as the denominator but also with the same multiplicity. Therefore, additional equations can be obtained by requiring that the derivative(s) of the numerator should also vanish where the denominator has a zero of higher multiplicity.
In the case of this assembly system, the denominator of (2.4.28) has \( \left\lceil \frac{k}{2} \right\rceil \) distinct roots (including zero) within the unit circle and the root zero has multiplicity \( \left\lfloor \frac{k}{2} \right\rfloor \). Taking \( \left\lfloor \frac{k}{2} \right\rfloor - 1 \) times the derivative of the enumerator, the additional equations can be determined. Together with the normalization equation, \( k \) equations are now obtained by which the \( G_n(0) \) can be determined.

The equilibrium probabilities

Since the \( G_0(z) \) and \( G_n(0) \) in formula (2.4.28) are known, the probability generating functions \( G_i(z) \) can be computed. After retrieving the expressions for \( G_i(z) \) one could derive the equilibrium probabilities by inverting the probability generating function. Several numerical inversion algorithms have been developed in the literature, see for example Abate and Whitt (1991). In the implementations used in this research we have simply relied on the Mathematica functions to obtain Taylor Series from the probability generating functions. The equilibrium probability \( p_{i,j} \) is the coefficient of \( z^j \) of \( G_i(z) \) where \( i = 0, \ldots, k \) and \( j \in \mathbb{N} \).
2.4.2 Matrix Geometric approach for assembly Systems with one bounded buffer and one unbounded buffer

In this section, the Matrix Geometric approach is explained and used to find the equilibrium probabilities for the assembly queue with one unlimited buffer (of Section 2.4), as was done before by the generating function approach. In the section hereafter, the Matrix Geometric approach is also applied to the assembly queue with two limited buffers.

The Matrix Geometric approach is a method for the analysis of continuous-time Markov chains whose transition rate matrices have a repetitive block structure. The Markov process has to be a quasi birth-death process, in which the process can switch to only neighboring levels within the Markov chain. The method requires a transition matrix with a tridiagonal block structure as in matrix $Q$, see (2.4.31).

The queue of Section 2.4 is considered. The buffer for the customer of type 1 has size $K_1$, which includes the place used by the customer that is served by the server. The Markov process is sketched in Figure 2.5. Each set of states \{$(0, i), (1, i), ..., (K_1 - 1, i), (K_1, i)$\} is called level $i$. The generator generator $Q$ describes the transition between these levels:

$$Q = \begin{bmatrix}
B_{00} & A_0 & A_2 & A_1 & A_0 \\
A_2 & A_1 & A_0 & A_2 & A_1 \\
A_2 & A_1 & A_0 & A_2 & A_1 \\
& & & &
\end{bmatrix},$$

(2.4.31)

where all the matrices $B_{00}, A_0, A_1, A_2$ are square matrices of dimension $(K_1 + 1) \times (K_1 + 1)$. Since level 0 is the only level that consists of states that behave differently than the states of other levels, the upper left matrix $B_{00}$ in $Q$ differs from the other matrices on the diagonal of $Q$. The matrix $B_{00}$ is constructed as follows:

$$B_{00} = \begin{bmatrix}
-(\lambda_1 + \lambda_2) & \lambda_1 & & & \\
& -(\lambda_1 + \lambda_2) & \lambda_1 & & \\
& & & \ddots & \ddots \\
& & & & -(\lambda_1 + \lambda_2) & \lambda_1 \\
& & & & & -(\lambda_1 + \lambda_2) \\
& & & & & & \lambda_1 \\
& & & & & & -(\lambda_1 + \lambda_2) \\
& & & & & & & -\lambda_2
\end{bmatrix}.$$  

(2.4.32)
The only difference between matrix $B_{00}$ and matrix $A_1$ is caused by the property that in level 0 a state can not be left due to the finish of a product while this is, of course, possible for $K_1$ states in all other levels. So $A_1$ is:

$$A_1 = \begin{bmatrix} -(\lambda_1 + \lambda_2) & \lambda_1 & & & \\ -(\lambda_1 + \lambda_2 + \mu) & -(\lambda_1 + \lambda_2 + \mu) & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \lambda_1 \\ & & & -(\lambda_1 + \lambda_2 + \mu) & -(\lambda_2 + \mu) \end{bmatrix}. \quad (2.4.33)$$

Matrix $A_0$ contains the \textit{forward flow} from level $i$ to level $i+1$ while matrix $A_2$ contains the \textit{backward flow} from level $i$ to level $i-1$. Since there are only a few possibilities to flow forward or backward, both matrices $A_2$ and $A_0$ are sparse:

$$A_2 = \begin{bmatrix} 0 & \mu & & \\ \mu & 0 & \ddots & \\ & \ddots & \ddots & \mu \\ & & \mu & 0 \end{bmatrix}, \quad A_0 = \begin{bmatrix} \lambda_2 & & & \\ & \lambda_2 & & \\ & & \ddots & \ddots \\ & & & \lambda_2 \end{bmatrix}. \quad (2.4.34)$$

Since the queue for customers of type 1 is limited, the system will always be stable with respect to the number of customers of type 1. The only way for the system to be unstable, would be when the number of waiting customers of type 2 grows to infinity. This means that the process is stable, or in other words $Q$ is ergodic, if and only if

$$\pi A_0 e < \pi A_2 e, \quad (2.4.35)$$

where $e$ is the column vector of ones and $\pi = (\pi_0, \pi_1, ..., \pi_{K_1})$ the equilibrium distribution of the Markov process with generator $A_0 + A_1 + A_2$. The latter means that $\pi$ should meet the constraints...
\[ \pi(A_0 + A_1 + A_2) = 0 \text{ and } \pi e = 1. \]

Since \( A_0 \) represented the forward flow, \( \pi A_0 e \) is the mean forward flow. Likewise, \( \pi A_2 e \) is the mean backward flow. Because of this, condition (2.4.35) is also known as Neuts’ mean drift condition. Neuts gives a formal proof of this condition in his work about Matrix Geometric solutions in stochastic models (see Neuts (1981)).

Assume \( A^* = A_0 + A_1 + A_2 \), then

\[
A^* = \begin{bmatrix}
-\lambda_1 & \lambda_1 \\
\mu & - (\lambda_1 + \mu) & \lambda_1 \\
\ddots & \ddots & \ddots \\
\ddots & \ddots & \ddots & \ddots \\
\mu & - (\lambda_1 + \mu) & \lambda_1 \\
\mu & - \mu & - \mu 
\end{bmatrix}.
\]

(2.4.36)

**Proposition 2.4.1:**

The stability condition for an assembly system with one bounded buffer is given by:

\[
\rho_2 < 1 - \frac{1 - \rho_1}{1 - \rho_1^{K_1+1}},
\]

(2.4.37)

with \( \rho_1 = \frac{\lambda_1}{\mu} \) and \( \rho_2 = \frac{\lambda_2}{\mu} \).

**Proof.** From solving \( \pi A^* = 0 \) leads towards the desired solution. \( \square \)

**The algorithm**

Provided that the Markov process Q is ergodic, the equilibrium probability vectors \( p_i \) are given by

\[
p_i = (p(0, i), p(1, i), ..., p(K_1, i)) = p_i R^{i-1}, \quad \text{for } i = 1, 2, ..., (2.4.38)
\]

where the rate matrix \( R \) is the minimal nonnegative solution of the equation

\[
A_0 + RA_1 + R^2 A_2 = 0, \quad (2.4.39)
\]

see also the report of van Leeuwaarden (2014). The equilibrium vectors \( p_0 \) and \( p_1 \) can be determined by solving the matrix equations
2.4. ASSEMBLY SYSTEM WITH ONE BOUNDED BUFFER

\[
p_0 B_{00} + p_1 A_2 = 0,
\]
\[
p_0 A_0 + p_1 A_1 + p_1 R A_2 = 0, \quad (2.4.40)
\]
\[
p_0 e + p_1 (I - R)^{-1} e = 1.
\]

In his report Van Leeuwaarden (2014) derives a simple scheme to determine \( R \). He solves a fixed point equation by successive substitutions:

\[
R_{k+1} = -(A_0 + R^2_k A_2)A_1^{-1}, \quad \text{for } k = 0, 1, 2, \ldots, \quad (2.4.41)
\]

starting with the zero matrix \( R_0 = 0 \). As \( k \) tends to infinity, the matrix \( R_k \) converges to \( R \). This algorithm is very simple but not efficient. More efficient algorithms have been developed by, for example, Latouche and Ramaswami (1993) and (1999).

In short, the algorithm works as follows:

- **Step 1: Initialize algorithm**
  Determine \( B_{00}, A_0, A_1, \) and \( A_2 \).
  Set matrix \( R_0 = 0, k = 1, \) and choose a small \( \epsilon \).

- **Step 2: Compute \( R_k \)**
  Use formula (2.4.41) to compute \( R_k \).

- **Step 3: Check convergence**
  If \( ||R_k - R_{k-1}|| > \epsilon \), do \( k = k + 1 \) and go back to **Step 2**.
  Else proceed to **Step 4**.

- **Step 4: Solve \( p_0 \) and \( p_1 \)**
  Solve \( p_0 \) and \( p_1 \) by solving the matrix equations in (2.4.40).

- **Step 5: Calculate the equilibrium vectors**
  The equilibrium vector corresponding to the \( i \)-th level is equal to \( p_i = p_1 R^{i-1} \).

2.4.3 Matrix Geometric approach for Assembly Systems with two limited buffers

In this section the same Matrix Geometric Approach as in Section 2.4.2 is considered but this time applied to an assembly system with two bounded buffers, and thus with a finite state space. Of course it is also possible to compute the equilibrium probabilities of such a system by solving the balance equations. However, since we already have the algorithm of Section 2.4.2 only a few adjustments are necessary in order to create a working method.

Elhafsi and Molle (2007) not only derived an algorithm for solving a general finite state space QBD process, but the algorithm is also compared to other algorithms that serve the same purpose.
In this report, the algorithm constructed by Elhafsi is applied to a general assembly system with two finite buffers. For a more general approach, the reader is referred to (see Elhafsi and Molle (2007)).

Since both the buffers of the assembly system are finite, the generator matrix $Q$ now has finite dimensions $(K_1 + 1)(K_2 + 1) \times (K_1 + 1)(K_2 + 1)$ and differs only from (2.4.31) in the last column of matrices:

$$Q = \begin{bmatrix}
B_{00} & A_0 \\
A_2 & A_1 & A_0 \\
A_2 & A_1 & A_0 & & \\
& & & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots \\
& & & & \ddots & \ddots \\
& & & & & \ddots \\
& & & & & & A_2 & A_1 & A_0 \\
& & & & & & A_2 & A_1 & A_k
\end{bmatrix}, \quad (2.4.42)$$

where matrix $A_k$ contains the boundary balance equations and looks like

$$A_k = \begin{bmatrix}
-\lambda_1 & \lambda_1 \\
-(\lambda_1 + \mu) & \lambda_1 & & \ddots \\
& & \ddots & \ddots & \ddots \\
& & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots \\
& & & & -(\lambda_1 + \mu) & \lambda_1 \\
& & & & & -\mu
\end{bmatrix}, \quad (2.4.43)$$

The matrices $A_0$, $A_1$, and $A_2$ are the same as in Section 2.4.2. However, the addition of the extra boundary causes in general the stationary distribution not to have a matrix geometric structure of the form of equation (2.4.38). The rate matrix $R$ is, in contrast to the rate matrix of Section 2.4.2, dependent on the different levels in the Markov process. By assuming that the considered system is an assembly system with nonzero arrival and process rates, the generator matrix $Q$ is irreducible. This yields that the matrices $B_{00}$, $A_1$, and $A_k$ are nonsingular, which is needed for the algorithm.
2.4. ASSEMBLY SYSTEM WITH ONE BOUNDED BUFFER

The algorithm
This algorithm described below is mainly used in the following chapters, in order to determine the equilibrium probabilities of assembly systems. A more general algorithm can be found in Elhafsi and Molle (2007).

• **Step 1: Initialize algorithm**
  Determine $B_{00}$, $A_0$, $A_1$, $A_2$ and $A_k$.

• **Step 2: Compute the $R_i$’s**
  Set matrix $R_{K_2} = -A_0 A_k^{-1}$.

  Compute $R_{K_2-1}$ to $R_1$ by using the relation
  $R_i = -A_0 (A_1 + R_{i+1} A_2)^{-1}$

  Set $R_1 = I$, where $I$ is an identity matrix of dimension $(K_1 + 1) \times (K_1 + 1)$
  Set $R_0 = -A_2 B_{00}^{-1}$.

• **Step 3: Compute the $R_i^{*}$’s**
  Set $R_1^{*} = R_1$.
  Compute $R_2^{*}$ to $R_{K_2}^{*}$ by using the relation
  $R_i^{*} = R_{i-1}^{*} \cdot R_i$.

• **Step 4: Solve $p_0$ and $p_1$**
  Solve $p_0$ and $p_1$ by solving the matrix equations
  $p_1 \cdot (R_0 \cdot A_0 + A_1 + R_2 \cdot A_2) = 0$, and
  $p_1 (R_0 \cdot e + \sum_{m=1}^{K_2} R_m^{*}) = 1$

• **Step 5: Calculate the equilibrium vectors**
  The equilibrium vector corresponding to the $i$-th level is equal to $p_i = p_1 R_i^{*}$. 
Chapter 3

An algorithm to analyze circular production systems

The main result of this section is an algorithm to analyze circular production systems in order to give useful insights in important performance measures in such systems. Being able to predict the results of certain decisions in a circular production model can save a lot of costs for the companies and users involved in this production model.

Starting with a model of a production system with three combined assembly queues, we keep increasing the model until it almost fully resembles the circular model of Figure 1.2. This final model is an example of a circular production system. Since this system can be different for other products, the emphasis of the section lies in the relations between assembly queues and non assembly queues in a circular production system. These relations can easily be used to analyze other circular production systems containing assembly queues and non assembly queues.

The section is organized as follows. Three queueing models in which assembly queues and other type of queues are chained, are considered. In every following section, the queueing model is expanded in order to resemble better the circular model created in Chapter 1. In Section 3.1, a queueing model is considered in which three assembly systems produce a product that consists of 4 different items in total. The products leave the system after being ‘used’ by a machine that represents the user. In Section 3.2 the same queueing model is considered, only this time the items, components or products can be recycled, refurbished or reused. In Section 3.3 sources are added to the model in the form of $M/M/1/c$ queues.

3.1 A non circular production system

In this section the knowledge of the preceding sections about assembly queues, their stability and methods to derive their equilibrium probabilities, is combined. A simple system is created that displays a situation in a pure linear economy. Products are produced, consumed and thrown away with no possibility to recycle. The system consists of two parallel scheduled assembly queueing systems $C1$ and $C2$ that combine (different) items into (different) components which, in the next
stage are combined into complete \textit{products} by a third assembly queueing system \(P1\). The finished products then stay for an exponential distributed time in machine \(U1\) before leaving the system. A sketch of the system is presented in Figure 3.1.

In comparison with the model presented in Figure 1.2, machines \(C1\) and \(C2\) represent the process of the components manufacturer, machine \(P1\) resembles the product manufacturer and machine \(U1\) resembles the user who consumes or uses the product, but then throws it away so that products leave the system after a certain amount of time.

Assume four different items of type 1, 2, 3, and 4 arrive with exponential distributed inter arrival times. Machine \(C1\) combines two items of type 1 and 2 into components of type 12, \(C2\) combines two items of type 3 and 4 into components of type 34, and \(P1\) combines two components of type 12 and 34 into products of type 1234, which represent the product that is build from the four initial items. The last machine \(U1\) is, in contrast to \(C1\), \(C2\), and \(P1\), not an assembly queueing system, but a 'simple' \(M/M/\infty\) queue. For now it is assumed that there is an infinite number of customers willing to buy the produced products. The processing time of \(U1\) represents the time that a customer uses or consumes the product until it is thrown into the dustbin.

As an assembly queue has to process two different items, each assembly system contains two different queues. Since the capacity to store items, components and products is typically finite, all assembly queues have limited buffers. The maximum (buffer)size of these queues is equal to \(K_{C1}^{C1}\) and \(K_{C1}^{C2}\), \(K_{C2}^{C1}\) and \(K_{C2}^{C2}\), and \(K_{P1}^{P1}\) and \(K_{P1}^{P2}\) for \(C1\), \(C2\), and \(P1\) respectively. Although assembly systems with one unlimited buffer can be analyzed with the use of techniques described in earlier sections, it is the stability that plays an important role now. Since the assembly systems are de-
3.1. A NON CIRCULAR PRODUCTION SYSTEM

Pendent, and in following models their dependency will become stronger, it is hard to derive the conditions under which the total system of combined assembly queues would be stable. The service times of $C_1$, $C_2$, $P_1$ and $U_1$, described by $B^{C_1}$, $B^{C_2}$, $B^{P_1}$ and $B^{U_1}$, are exponentially distributed with rates $\mu^{C_1}$, $\mu^{C_2}$, $\mu^{P_1}$ and $\mu^{U_1}$. Items of type 1, 2, 3, and 4 arrive with respective arrival rates $\lambda^{C_1}$, $\lambda^{C_2}$, $\lambda^{P_1}$, $\lambda^{U_1}$.

3.1.1 Derivation of the algorithm

The approximation algorithm is an iterative algorithm that considers every queue in the model as a separate queueing system which is, as a consequence, stand-alone from the other queueing systems in the model. The dependency on other queueing systems in the model, will be contained in updated arrival rates and process rates of the stand-alone queue. We approximate the distribution of the inter departure times by an exponential distribution. This means that we only need the first moment of the inter departure times in order to determine the corresponding distribution. Since the inter arrival times of a machine are the inter departure times of the preceding machine in the system, we also have a exponential arrivals to every separate considered queue in the system. More accuracy can be obtained by fitting distributions on more moments. For example, when fitting on the first two moments one can use Erlang-2 or Cox distributions instead of exponential distributions.

The arrival rates to $P_1$

The idea is that first machines $C_1$ and $C_2$ are considered. Since the arrival rates and the processing rates are known, it is possible to calculate the equilibrium probabilities with the use of the approaches described in Sections 2.4.1, 2.4.3 or by solving the balance equations given in Section 2.3. Whenever equilibrium probabilities of assembly systems are computed in the following sections, this is done by the Matrix Geometric Approach because it is a very simple and easy implementable method.

After computing the equilibrium probabilities of $C_1$ and $C_2$ (presented by the probability matrices $P^{C_1}$ and $P^{C_2}$) under $\lambda^{C_1}$, $\lambda^{C_2}$, $\lambda^{P_1}$, $\lambda^{U_1}$, and $\mu^{C_1}$, $\mu^{C_2}$ these probabilities can be used to compute the first moment of the inter departure times of components of type 12 and 34.

The formula for the first moment of the inter departure time of $C_1$ is derived below. The formula of the inter departure times of $C_2$ is derived in the same manner. Just after a departure, we always have to wait a service time of the following product. So the average inter departure time is at least $E[B^{C_1}]$. However, when the departing product leaves behind at least one empty queue, we also have to wait for a new arrival to the empty queue(s). Three cases can be distinguished:

1. The produced component leaves behind an empty queue 1 and a nonempty queue 2. In that case the next component can only be manufactured when a new item of type 1 has arrived. So on average we need to wait an additional time equal to the mean inter arrival time of items of type 1 (to machine $C_1$), which is denoted as $E[A^{C_1}_1]$.

2. The produced component leaves behind an empty queue 2 and a nonempty queue 1. The next component can only be manufactured when a new item of type 2 has arrived. So on
average we need to wait an additional time equal to the mean inter arrival time of items of type 2 (to machine C1), which is denoted as $\mathbb{E}[A_{2}^{C1}]$.

3. The produced component leaves behind two empty queues. Since the machine can only manufacture a component when items of both types 1 and 2 are present in the queues, we have to wait the maximum of the two inter arrival times. The expected value of the maximum of two exponential distributed random variables with parameters $\gamma_1$ and $\gamma_2$, say $X \sim \text{Exp}(\gamma_1), Y \sim \text{Exp}(\gamma_2)$ is equal to $\mathbb{E}[\max\{X,Y\}] = \frac{1}{\gamma_1} + \frac{1}{\gamma_2} - \frac{1}{\gamma_1 + \gamma_2}$. Thus in this case $\mathbb{E}[\max\{A_{1}^{C1}, A_{2}^{C1}\}] = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2}$, where $A_{1}^{C1}$ and $A_{2}^{C1}$ represent the inter arrival times of items of type 1 and 2 to C1 respectively.

Note that it is only possible to have a departure, when there is at least 1 item of each type present in the machine. So, when computing the mean departure rate of C1, we need to condition on the states in which it was possible for machine C1 to produce a new component. These are only the states $p_{i,j}^{C1}$ with $i,j > 0$. By conditioning on these states, we take the starvation of the queues of machine P1 into account. Starvation is the event in which P1 is not processing items because there are not enough components delivered from machines C1 and C2.

Combining the knowledge of above, we get the following formula for the mean inter departure times of components of type 12 leaving machine C1:

$$
\mathbb{E}[D^{C1}] = \mathbb{E}[B^{C1}] + \frac{\sum_{j=2}^{K_{C1}^{I}} p_{i,j}^{C1} \mathbb{E}[A_{1}^{C1}]}{\sum_{i=1}^{K_{C1}^{I}} \sum_{j=1}^{K_{C1}^{J}} p_{i,j}^{C1}} + \sum_{i=2}^{K_{C1}^{I}} p_{i,1}^{C1} \mathbb{E}[A_{2}^{C1}] + p_{1,1}^{C1} \mathbb{E}[\max\{A_{1}^{C1}, A_{2}^{C1}\}],
$$

(3.1.1)

where $\mathbb{E}[A_{1}^{C1}] = \frac{1}{\lambda_1}$ and $\mathbb{E}[A_{2}^{C1}] = \frac{1}{\lambda_2}$.

In other words:

$$
\mathbb{E}[D^{C1}] = \frac{1}{\mu_{C1}} + \frac{\sum_{j=2}^{K_{C1}^{I}} p_{i,j}^{C1} \frac{1}{\lambda_1}}{\sum_{i=2}^{K_{C1}^{I}} p_{i,1}^{C1} \frac{1}{\lambda_1} + \sum_{i=2}^{K_{C1}^{I}} p_{i,1}^{C1} \frac{1}{\lambda_1} + p_{1,1}^{C1} \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2})}{\sum_{i=1}^{K_{C1}^{I}} \sum_{j=1}^{K_{C1}^{J}} p_{i,j}^{C1}}.
$$

(3.1.2)

In the same manner $\mathbb{E}[D^{C2}]$ can be computed.

We approximate the mean inter departure distribution by assuming that the departure distributions of C1 and C2 are exponential. Since we know the first moment of these inter departure times $\mathbb{E}[D^{C1}]$, we approximate the departure distribution of components of type 34 (consisting of items of type 3 and 4) by an $\text{Exp}(\frac{1}{\mathbb{E}[D^{C1}]})$ distribution and the departure distribution of type 34 by an $\text{Exp}(\frac{1}{\mathbb{E}[D^{C2}]})$ distribution. Since the departure distribution of machine C1 is equal to the arrival distribution of components of type 12 to queue 1 of machine P1, and likewise for the components of type 34, we can now consider machine P1 as an stand-alone queueing system with arrival rates
3.1. A NON CIRCULAR PRODUCTION SYSTEM

\[ \lambda_1^{P_1} = \frac{1}{E[D^{C_1}]} \quad \text{and} \quad \lambda_2^{P_1} = \frac{1}{E[D^{C_2}]} \]
and process rate \( \mu^{P_1} \).

**Equilibrium probabilities of \( P_1 \) and the updated process rates of \( C_1 \) and \( C_2 \)**

Next, the equilibrium probabilities of \( P_1 \) (in probability matrix form \( P^{P_1} \)) are computed. If both of the buffers of machine \( P_1 \) were unlimited, the approximation algorithm would not need to iterate. Machine \( P_1 \) would still be dependent of machines \( C_1 \) and \( C_2 \) because of starvation, but \( P_1 \) would have no influence on \( C_1 \) and \( C_2 \). However, since in this model both buffers of \( P_1 \) are finite, it is possible that \( P_1 \) has to block machines \( C_1 \) and/or \( C_2 \). For example, when machine \( P_1 \) contains \( K^{C_1}_{P_1} \) components of type 12, it cannot receive any more components of type 12.

The form of blocking is **blocking before service (BBS)**, which means that, in the example above, machine \( C_1 \) directly stops working after finishing the component of type 12 that fills up the \( K^{C_1}_{P_1} \)-th place in \( P_1 \). Another form of blocking that is typically considered is **blocking after service (BAS)**, in which \( C_1 \) would still produce a component although the buffer of \( P_1 \) is full. If \( P_1 \) is still full at the moment that \( C_1 \) has finished processing the next component, machine \( C_1 \) is blocked. Machine \( C_1 \) then stores an already processed component until there is space available in \( P_1 \).

The mean blocking time for a machine is added to the mean process time as a sort of penalty. These updated processing times are marked with an asterisk. We get

\[
\mathbb{E}[B^{C_1*}] = \mathbb{E}[B^{C_1}] + \sum_{j=0}^{K^{P_1}_{P_1}} p_{K^{P_1}_{P_1},j}^{P_1} \mathbb{E}[RB^{P_1}] + \sum_{j=0}^{K^{P_1}_{P_1}} p_{0,K^{P_1}_{P_1}}^{P_1} \mathbb{E}[A^{P_1}],
\] (3.1.3)

and

\[
\mathbb{E}[B^{C_2*}] = \mathbb{E}[B^{C_2}] + \sum_{i=0}^{K^{P_1}_{P_1}} p_{K^{P_1}_{P_1},i}^{P_1} \mathbb{E}[RB^{P_1}] + \sum_{i=0}^{K^{P_1}_{P_1}} p_{0,K^{P_1}_{P_1}}^{P_1} \mathbb{E}[A^{P_1}],
\] (3.1.4)

The first term in formulas (3.1.3) and (3.1.4) is the normal process time of the corresponding machine. The second term comes from the event in which a machine is blocked due to a full buffer in machine \( P_1 \). In this case, Machine \( C_1 \) and/or \( C_2 \) have to wait for a residual service time of machine \( P_1 \) (\( \mathbb{E}[RB^{P_1}] \)) to have available space in its buffers again. Since the machine \( P_1 \) has exponential distributed service times, we can use that \( \mathbb{E}[RB^{P_1}] = \mathbb{E}[B^{P_1}] \) by the memoryless property of the exponential distribution.

In the special case that one of the buffers of \( P_1 \) is full but the other buffer is empty, extra blocking occurs. To see this, assume that buffer 1 of \( P_1 \) is full (\( K^{P_1}_{P_1} \) places are taken) while the other buffer is empty (there are no components of type 34 present in \( P_1 \)). In that case, \( P_1 \) can only start processing if there arrives a component of type 34, so on average \( P_1 \) blocks machine \( C_1 \) with \( \mathbb{E}[A^{P_1}] \) extra time units. In terms of arrival rates and process rates formula (3.1.3) and (3.1.4) become
\[ \mu^{C1*} = \frac{1}{\mu^{C1}} + \sum_{j=0}^{K^{P1}} p^{P1}_{K^{P1},j}/\mu^{P1} + p^{P1}_{K^{P1},0}/\lambda^{P1}_2, \]  

(3.1.5)

and

\[ \mu^{C2*} = \frac{1}{\mu^{C2}} + \sum_{i=0}^{K^{P1}} p^{P1}_{i,K^{P1}}/\mu^{P1} + p^{P1}_{0,K^{P1}}/\lambda^{P1}_1. \]  

(3.1.6)

Now, we are back at the beginning of the production line. Again the equilibrium probabilities of the isolated assembly queues \( C1 \) and \( C2 \) are computed, but this time with the use of \( \mu^{C1*} \) and \( \mu^{C2*} \) instead of \( \mu^{C1} \) and \( \mu^{C2} \).

The algorithm tries to find a balance between the starvation of machines \( C1 \) and machines \( C2 \) and the blocking of machine \( P1 \) by iterating over the three assembly systems.

The equilibrium probabilities of \( U1 \)

Although the algorithm iterates over the entire line of machines, machine \( U1 \) needs not to be considered in these iterations. This last machine does not give feedback of any form to the other queueing systems in the model and thus has no influence on the production line. Therefore this system can be discarded until the algorithm has converged in terms of probability matrices of \( C1 \), \( C2 \) and \( P1 \). After the \( P^{P1} \) converged, the equilibrium probabilities of \( U1 \) can be easily computed. Machine \( U1 \) is an \( M/M/\infty \) queue with process rate \( \mu^{U1} \) and has complete products arriving to its system with mean inter arrival time equal to the mean inter departure time of products of \( P1 \).

By our assumption that the departures of the machines are exponentially distributed, the arrivals to \( U1 \) are poisson with parameter \( \frac{1}{E[D^{P1}]} \). Using the formula for the equilibrium probabilities of an \( M/M/\infty \) queue derived by Adan and Resing (2002), the equilibrium probabilities of \( U1 \) can be computed as follows:

\[ p_i = \left( \frac{1}{E[D^{P1}]\mu^{U1}} \right)^i e^{-\frac{1}{E[D^{P1}]\mu^{U1}}} = (E[D^{P1}]\mu^{U1})^i e^{-(E[D^{P1}]\mu^{U1})} \]  

(3.1.7)

3.1.2 Algorithm A

In this section the algorithm for solving the equilibrium probabilities of the model described in Section 3.1 is presented, with the use of the derived formulas from Section 3.1.1.

- **Step 0, startup the algorithm**
  Set \( P^{P1} \in [0,1]^{K^{P1}} \times [0,1]^{K^{P1}} \) to be a zero matrix and choose a small \( \epsilon \) that is used to determine whether the algorithm should stop or not. The smaller the \( \epsilon \), the more iterations the algorithm runs and the more accurate the equilibrium matrices are.
  Set \( \mu^{C1*} = \mu^{C1} \) and \( \mu^{C2*} = \mu^{C2} \).

- **Step 1, calculate the equilibrium probabilities of \( C1 \) and \( C2 \)**
  Given \( \mu^{C1*} \) and \( \mu^{C2*} \), compute \( P^{C1}(\in [0,1]^{K^{C1}} \times [0,1]^{K^{C1}}) \) and \( P^{C2}(\in [0,1]^{K^{C2}} \times [0,1]^{K^{C2}}) \).
3.2. A PRODUCTION LINE WITH TANDEM ASSEMBLY QUEUES AND LOOPS

- **Step 2**, calculate the mean departure times from $C1$ and $C2$
  Using $\mu^{C1}$ and $\mu^{C2}$, calculate $E[D^{C1}]$ and $E[D^{C2}]$ (with the use of formula (3.1.2)).

- **Step 3**, calculate the equilibrium probabilities of $P1$
  Calculate $D^{P1}$ by considering it to be a single assembly queue with arrival rates $\lambda^{P1}_1 = \frac{1}{E[A^{P1}_1]} = E[D^{P1}_1]$ and $\lambda^{P1}_2 = \frac{1}{E[A^{P1}_2]} = E[D^{P1}_2]$.

- **Step 4**, compute the updated service rates of $C1$ and $C2$
  Compute the new $\mu^{C1*}$ and $\mu^{C2*}$ by formulas (3.1.5) and (3.1.6). Note that $\mu^{C1}$ and $\mu^{C2}$ itself never change during the algorithm.

- **Step 5**, convergence check
  If the FrobeniusNorm between two consecutive updated matrices $P^{P1}$ is bigger than $\epsilon$ go to **step 2**, else proceed to **step 6**.

- **Step 6**, compute the equilibrium probabilities of $U1$
  Compute the equilibrium probabilities of $U1$ by using formula (3.1.7).

Note that in step 5 the FrobeniusNorm is chosen to determine the distances between two updated matrices. There are other possibilities for the stop criterion. One could use other norms and/or apply these norms on other probabilities, matrices or performance measures. After running the algorithm above on several different instances, it became clear that the algorithm converged in all the test cases within 8 iterations using an $\epsilon < 0.0001$.

3.2 A production line with tandem assembly queues and loops

In this section we consider the same system of queues as in Section 3.1, but as we want to investigate models in the circular economy, loops are added to the linear model of Section 3.1. The purpose of this section is to analyze the circular production model. Figure 3.2 shows the tandem queueing system that is the topic of this section.
Again, $C1$, $C2$ and $P1$ are the assembly systems and $U1$ is an $M/M/\infty$ queue representing the user. This time it is possible to recycle items (materials), components or even complete products after use by the customer. Since our model starts to look more and more like the circular model of Chapter 1, we will from now on call the incoming items materials. Combined materials form components and combined components form a product.

After the user has consumed the product, he no longer throws it away, but recycles it with given probabilities, defined by

$$r_i = \mathbb{P}(\text{Material } i \text{ is reusable/recyclable}) \quad (3.2.1)$$

for every material (thus $i = 1, 2, 3, 4$). So with probability $1 - r_i$, a material of type $i$ is broken and can not be repaired. Assume that the two materials of a component can be recycled. Then it is unnecessary to decompose this component into its materials in order to recycle the latter so that they can form a component again. In the circular economy, we want to reuse materials, components and products while decreasing their value as little as possible. If a component contains two reusable materials, there is no need to split it up into materials again. This component is immediately sent to machine $P1$ where it can form a product again, together with a component of the other type. If all four materials of a product are recyclable/not broken, the product can be directly used again. In ‘real life’ this would for example be in the case when a user resells his product to another user.

An example: assume that the materials of type 1, 3, 4 are recyclable and the material of type 2 is thrown away. Now the following events occur: The material of type 2 leaves the model, the
material of type 1 goes back to queue 1 of machine C1, and the component of type 34 is send to queue 2 of machine P1.

Unfortunately, the situation can occur in which a material or component is not broken and therefore could be recycled, while the buffer this material or component has to join is full. In that case the material or component is, although possible, not recycled and thrown away. Since the machines in the model work with blocking before service, it is also not possible for a recyclable material or component to join a queue that has only 1 space left in the buffer. Otherwise the situation could occur in which a machine is producing a material or component for the next machine in the production line, while the buffer of that machine gets filled by recycled materials or components meanwhile. Consequently, the first machine gets blocked while it was already processing a material or component. Since this event conflicts with the blocking before service policy, it is prevented from happening by the constraint which only allows recycled materials and components to join a queue when it has at least two empty spots in its buffer.

3.2.1 Derivation of the algorithm

Since loops have been added to the queueing model of the previous sections, the dependency between the machines in the model differs from this previous model. However, the idea behind the algorithm remains the same. We still iterate over the queueing systems in the model, determining the equilibrium probabilities for every queue while considering it as an isolated queue.

Updated arrival rates to C1 and C2

Since the loops of machine U1 influence the rate of incoming materials, the 'updated' arrival rates \( \lambda_{C1}^{C1*}, \lambda_{C2}^{C1*}, \lambda_{C1}^{C2*}, \) and \( \lambda_{C2}^{C2*}, \) are introduced:

\[
\begin{align*}
\lambda_{C1}^{C1*} &= \lambda_{C1}^{C1} + r_1 (1 - r_2) \frac{1}{\mathbb{E}[D_{U1}]} \\
\lambda_{C2}^{C1*} &= \lambda_{C2}^{C1} + r_2 (1 - r_1) \frac{1}{\mathbb{E}[D_{U1}]} \\
\lambda_{C1}^{C2*} &= \lambda_{C1}^{C2} + r_3 (1 - r_4) \frac{1}{\mathbb{E}[D_{U1}]} \\
\lambda_{C2}^{C2*} &= \lambda_{C2}^{C2} + r_4 (1 - r_3) \frac{1}{\mathbb{E}[D_{U1}]},
\end{align*}
\]  

(3.2.2)

where \( \mathbb{E}[D_{U1}] \) is the mean inter departure time from used products leaving machine U1. The departure rate is equal to \( \frac{1}{\mathbb{E}[D_{U1}]} \). A material only goes back to machines C1 and C2 when it is recyclable and the material with which it could form a component is thrown away. Otherwise the two materials would stay in the form of a component and go to P1. The updated arrival rate of materials of type 1 for example, is thus the original arrival rate added with the departure rate from U1 multiplied with the probability that a material of type 1 goes back to C1 and the material of type 2 does not go back to C1.
CHAPTER 3. AN ALGORITHM TO ANALYZE CIRCULAR PRODUCTION SYSTEMS

Updated arrival rates to $P_1$
The loop from machine $U_1$ to $P_1$ influences the arrival rates $\lambda_1^{P_1}$ and $\lambda_2^{P_1}$ as follows:

$$\lambda_1^{P_1} = \frac{1}{\mathbb{E}[D_{C_1}]} + r_1 r_2 (1 - r_3 r_4) \frac{1}{\mathbb{E}[D_{U_1}]},$$

$$\lambda_2^{P_1} = \frac{1}{\mathbb{E}[D_{C_2}]} + r_3 r_4 (1 - r_1 r_2) \frac{1}{\mathbb{E}[D_{U_1}]}. \tag{3.2.3}$$

The arrival rates of the components are increased by the departure rate of products from machine $U_1$ multiplied by the probability that a whole component is recycled. This can only happen when both materials of the concerning component are not thrown away and not both other materials are recycled, since in that last case one would reuse the whole product instead of splitting it into two components.

Updated arrival rates to $U_1$
The last loop that is not yet considered, is the loop that sends used products from machine $U_1$ back to machine $U_1$. In order to analyze this loop, we look at a very simple queueing model containing only 1 machine at which customers arrive with rate $\lambda$, the server works with rate $\mu$, and with probability $p$ a customer is not satisfied with the service. These customers return to the queue so that they can be served again.

$$\lambda + \gamma p = \gamma, \tag{3.2.4}$$

which immediately gives us $\gamma = \frac{\lambda}{1 - p}$. Note that the process rate of the machine does not influence the departure rate at all.
Since the arrival rate of products from $P1$ to $U1$ is equal to $\frac{1}{E[D_{r1}]}$, the mean inter-departure time of products from $U1$ can thus be computed by

$$E[D_{U1}] = (1 - r_1 r_2 r_3 r_4)E[D_{P1}] = (1 - r_{1234})E[D_{P1}], \quad (3.2.5)$$

where $r_1 r_2 r_3 r_4$, to which we will from now on refer to as $r_{1234}$, is the probability that none of the materials of a component has to be thrown away.

For a general $M/G/\infty$ queue, the equilibrium probabilities are given by

$$p_i = \frac{\rho^n}{n!} e^{-\rho}, \quad (3.2.6)$$

where $\rho$ denotes the mean amount of work that arrives per unit time. So in this case $\rho = \frac{1}{E[D_{P1}]} \mu_{U1} (1 - r_{1234})$, which yields

$$p_i = \frac{(\frac{1}{E[D_{P1}]} \mu_{U1} (1 - r_{1234}))^i}{i!} e^{-\left(\frac{1}{E[D_{P1}]} \mu_{U1} (1 - r_{1234})\right)} = ((E[D_{P1}] \mu_{U1} (1 - r_{1234}))^i i!)^{-1} e^{-\left(E[D_{P1}] \mu_{U1} (1 - r_{1234})\right)} \quad (3.2.7)$$

### 3.2.2 Algorithm B

In this section the algorithm for solving the equilibrium probabilities of the model described in Section 3.2 is presented, with the use of the derived formulas from sections 3.1.1 and 3.2.1.

- **Step 0**, startup the algorithm
  Set $P_{P1} \in [0, 1]^{K_{P1}} \times [0, 1]^{K_{P1}}$ to be a zero matrix and choose a small $\epsilon$ that is used to determine whether the algorithm should stop or not. The smaller the $\epsilon$ the more iterations the algorithm runs and the more accurate the equilibrium matrices are.

  Set $\lambda_1^{C1*} = \lambda_1^{C1}, \lambda_2^{C1*} = \lambda_2^{C1}, \lambda_1^{C2*} = \lambda_1^{C2}$, and $\lambda_2^{C2*} = \lambda_2^{C2}$. Also set $\mu^{C1*} = \mu^{C1}$ and $\mu^{C2*} = \mu^{C2}$.

- **Step 1**, calculate the equilibrium probabilities of $C1$ and $C2$
  Given $\mu^{C1*}$ and $\mu^{C2*}$, compute $P_{C1} (\in [0, 1]^{K_{C1}} \times [0, 1]^{K_{C1}})$ and $P_{C2} (\in [0, 1]^{K_{C2}} \times [0, 1]^{K_{C2}})$.

- **Step 2**, calculate the arrival rates to $P1$
  Using (the original) $\mu^{C1}$ and $\mu^{C2}$, calculate $E[C_{C1}]$ and $E[C_{C2}]$ (with the use of formula (3.1.2) and $\lambda_1^{C1*}, \lambda_2^{C1*}, \lambda_1^{C2*}$, and $\lambda_2^{C2*}$).

  If $E[D_{U1}]$ has been calculated in a previous loop, use formula’s (3.2.3) to add the extra amount of arrival rates coming from the recycled components. Otherwise, don’t change $E[C_{C1}]$ and $E[C_{C2}]$.

- **Step 3**, calculate the equilibrium probabilities of $P1$
  Calculate $P_{P1}$ by considering it to be a single assembly queue with arrival rates $\lambda_1^{P1} = \frac{1}{E[C_{C1}]} = \frac{1}{E[A_{C1}]}$ and $\lambda_2^{P1} = \frac{1}{E[C_{C2}]} = \frac{1}{E[A_{C2}]}$. 


• **Step 4, compute the updated service rates of C1 and C2**
  Compute \( \mu_{C1}^* \) and \( \mu_{C2}^* \) by formulas (3.1.5) and (3.1.6). Note that \( \mu_{C1} \) and \( \mu_{C2} \) itself never change during the algorithm.

• **Step 5, compute the mean departure time from P1**
  Compute the mean inter departure time of items leaving \( P1 \), \( \mathbb{E}[D_{P1}] \), by applying (3.1.2) with \( \mu_{P1}, \lambda_{P11}, \lambda_{P12}, \) and \( P_{P1} \).

• **Step 6, compute the mean inter departure time of U1**
  Use formula (3.2.5) to compute \( \mathbb{E}[D_{U1}] \).

• **Step 7, update the arrival rates with the extra recycle arrival rates**
  Use formula’s (3.2.2) and (3.2.3) to update the arrival rates with the extra arrivals that come from the recycle loops.

• **Step 8, convergence check**
  If the FrobeniusNorm between two consecutive updated matrices \( P_{P1} \) is bigger than \( \epsilon \) go to **step 2**, else proceed to **step 9**.

• **Step 9, compute the equilibrium probabilities of U1**
  Compute the equilibrium probabilities of \( U1 \) by using formula (3.2.7).

Note that in step 8 the FrobeniusNorm is chosen to determine the distances between two updated matrices. There are other possibilities for the stop criterion. One could use other norms and/or apply these norms on other probabilities, matrices or performance measures. The algorithm converged for all the test cases within 40 iterations using an \( \epsilon < 0.0001 \).

### 3.3 A production line with tandem assembly queues, loops and sources

In this section, the production line is expanded once more. Four sources are added to the model of Section 3.2. The purpose of this section is not only to expand the production line that will be analyzed in following sections, but also to show the ease of extending the algorithm making use of previously derived relations in sections 3.1.1 and 3.2.1.

The arrival rates to the sources represent the rates at which new material is mined from the source. The source has a given process rate at which it makes the raw materials ready for the component manufacturer. An overview of the complete model is given in Figure 3.4.
3.3.1 Derivation of the algorithm

The addition of the sources only influences the arrival rates to $C_1$ and $C_2$. On the other side, the sources are only influenced by blocking from $C_1$ and $C_2$. In order to derive an algorithm for this extended production line, only these two relations have to be analyzed.

The departure rates of the sources $S_1$, $S_2$, $S_3$, and $S_4$

The derivation of the departure rates of the sources follows the same reasoning as the derivation of (3.1.1). However, an $M/M/1/c$ queue is considered instead of an assembly queue. If a material leaves the source empty after its departure, the source only has to wait for 1 new arrival for it to be able to process again. In the case of the assembly system, we have to wait for two arrivals, one of each type. This yields:

$$E[D_{S1}] = E[B_{S1}] + \frac{p_{S1}^{S1} E[A_{S1}]}{\sum_{i=1}^{c} p_{S1}^{S1}},$$  \hspace{1cm} (3.3.1)$$

where $E[A_{S1}] = \frac{1}{\lambda S1}$.

Specifically:
\[ E[D_{S1}] = \frac{1}{\mu_{S1}} + \frac{p_{D_{S1}}^{S1}}{\lambda_{S1}(1 - p_{D_{S1}}^{0})}, \]  

(3.3.2)

\[ E[D_{S2}], E[D_{S3}] \text{ and } E[D_{S4}] \] can be computed in the same way.

The updated process rates of the sources

The sources are possibly blocked by \( C_1 \) or \( C_2 \) if their buffers are full. Since the blocking comes from an assembly queue, we can use formulas (3.1.3) and (3.1.4) to write

\[ E[B_{S1}^{*}] = E[B_{S1}] + \sum_{j=0}^{K_{S1}^{C1}} p_{K_{S1}^{C1}} p_{S_{j},C_{1}}^{*} E[R_{S_{j},C_{1}}] + p_{0,C_{1}}^{S_{j}} E[A_{j,C_{1}}], \]  

(3.3.3)

and

\[ E[B_{S2}^{*}] = E[B_{S2}] + \sum_{i=0}^{K_{S1}^{C1}} p_{K_{S1}^{C1}} p_{S_{i},C_{2}}^{*} E[R_{S_{i},C_{2}}] + p_{0,C_{2}}^{S_{i}} E[A_{i,C_{2}}]. \]  

(3.3.4)

The same formulas hold for \( S_3 \) and \( S_4 \).

The equilibrium probabilities of the sources

The sources \( S_1, S_2, S_3, \) and \( S_4 \) are all \( M/M/1/c \) queues, where \( c \) is the maximum possible items present in the source (including the item that is under service). For a general \( M/M/1/c \) queue with arrival rate \( \lambda \) and process rate \( \mu \), the equilibrium probabilities are computed by

\[ p_{i} = \frac{1 - \rho}{1 - \rho^{c+1}} \rho^{i}, \quad \text{for } i = 0, ..., c, \]  

(3.3.5)

where \( \rho = \frac{\lambda}{\mu} \). Note that in our case \( c = K_{S1}^{C1} \).

3.3.2 Algorithm C

The algorithm for the model including the sources and the recycle loops is presented below.

- **Step 0, startup the algorithm**

  Set \( P^{P1} \in [0,1]^{K_{S1}} \times [0,1]^{K_{S1}} \) to be a zero matrix and choose a small \( \epsilon \) that is used to determine whether the algorithm should stop or not. The smaller the \( \epsilon \), the more iterations the algorithm runs and the more accurate the equilibrium matrices are.

  Set \( \mu^{S1*} = \mu^{S1}, \mu^{S2*} = \mu^{S2}, \mu^{S3*} = \mu^{S3}, \mu^{S4*} = \mu^{S4}, \mu^{C1*} = \mu^{C1}, \) and \( \mu^{C2*} = \mu^{C2} \).

- **Step 1, calculate the equilibrium probabilities of \( S_1, S_2, S_3, \) and \( S_4 \)**

  Given \( \mu^{S1*}, \mu^{S2*}, \mu^{S3*}, \) and \( \mu^{S4*} \) compute \( P^{S1}, P^{S2}, P^{S3}, \) and \( P^{S4} \) with formula (3.3.5).
3.3. A PRODUCTION LINE WITH TANDEM ASSEMBLY QUEUES, LOOPS AND SOURCES

- **Step 2**, calculate the arrival rates to \( P_1 \)
  Using (the original) \( \mu^{S_1}, \mu^{S_2}, \mu^{S_3}, \) and \( \mu^{S_4} \), calculate \( \lambda_{C1} = \frac{1}{\mathbb{E}[D_{S1}]} \) and the inter departure rates of the other sources (with the use of formula (3.3.2)).

- **Step 3**, calculate the equilibrium probabilities of \( C_1 \) and \( C_2 \)
  Given \( \mu^{C_1*}, \mu^{C_2*}, \lambda^{C_1}, \lambda^{C_2}, \) and \( \lambda^{C_2} \), compute \( P_{C1}(\in [0, 1]^{K_{C1} \times [0, 1]^{K_{C1}}}) \) and \( P_{C2}(P_{C2} \in [0, 1]^{K_{C2} \times [0, 1]^{K_{C2}}}) \).

- **Step 4**, calculate the arrival rates to \( P_1 \)
  Using (the original) \( \mu^{C_1} \) and \( \mu^{C_2} \), calculate \( \mathbb{E}[D_{C1}] \) and \( \mathbb{E}[D_{C2}] \) (with the use of formula (3.1.2) and \( \lambda^{C1}, \lambda^{C2}, \) and \( \lambda^{C2} \)).

  If \( \mathbb{E}[D_{U1}] \) has been calculated in a previous loop, use formula’s (3.2.3) to add the extra amount of arrival rates coming from the recycled components. Otherwise, don’t change \( \mathbb{E}[D_{C1}] \) and \( \mathbb{E}[D_{C2}] \).

- **Step 5**, compute the updated service rates of \( S_1, S_2, S_3 \) and \( S_4 \)
  Compute \( \mu^{S_1*}, \mu^{S_2*}, \mu^{S_3*}, \) and \( \mu^{S_4*} \) by formulas (3.3.3) and (3.3.4). Note that \( \mu^{S_1} \) and the other (original) process rates never change during the algorithm.

- **Step 6**, calculate the equilibrium probabilities of \( P_1 \)
  Calculate \( P_{P1} \) by considering it to be a single assembly queue with arrival rates \( \lambda_{P1} = \frac{1}{\mathbb{E}[D_{P1}]} = \frac{1}{\mathbb{E}[A_{P1}]} \) and \( \lambda_{P1} = \frac{1}{\mathbb{E}[D_{P1}]} = \frac{1}{\mathbb{E}[A_{P1}]} \).

- **Step 7**, compute the updated service rates of \( C_1 \) and \( C_2 \)
  Compute \( \mu^{C1*} \) and \( \mu^{C2*} \) by formulas (3.1.5) and (3.1.6). Note that \( \mu^{C1} \) and \( \mu^{C2} \) itself never change during the algorithm.

- **Step 8**, compute the mean departure time from \( P_1 \)
  Compute the mean inter departure time of items leaving \( P_1, \mathbb{E}[D_{P1}] \), by applying (3.1.2) with \( \mu^{P1}, \lambda^{P1}, \lambda^{P1}, \) and \( P_{P1} \).

- **Step 9**, compute the mean inter departure time of \( U_1 \)
  Use formula (3.2.5) to compute \( \mathbb{E}[D_{U1}] \).

- **Step 10**, update the arrival rates with the extra recycle arrival rates
  Use formula’s (3.2.2) and (3.2.3) to update the arrival rates with the extra arrivals that come from the recycle loops.

- **Step 11**, convergence check
  If the FrobeniusNorm between two consecutive updated matrices \( P_{P1} \) is bigger than \( \epsilon \) go to **step 2**, else proceed to **step 9**.

- **Step 12**, compute the equilibrium probabilities of \( U_1 \)
  Compute the equilibrium probabilities of \( U_1 \) by using formula (3.2.7).
Note that in step 11 the *FrobeniusNorm* is chosen to determine the distances between two updated matrices. There are other possibilities for the stop criterion. One could use other norms and/or apply these norms on other probabilities, matrices or performance measures. The algorithm converged for all the test cases within 40 iterations using an $\epsilon < 0.0001$. 

In order to evaluate the algorithm of Section 3.3.2, this section discusses the results of the algorithm tested on 11 cases. The cases are defined by the arrival rates of the four different materials, their recycle probability, the process rates of the sources and buffers, and their buffer sizes. In Table 4.1 the 11 cases are presented. Every case is evaluated and discussed in a different section.
### Table 4.1: All Cases evaluated in Section 4

<table>
<thead>
<tr>
<th>Case</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$\lambda_3$</th>
<th>$\lambda_4$</th>
<th>$\mu_{S1}$</th>
<th>$\mu_{S2}$</th>
<th>$\mu_{S3}$</th>
<th>$\mu_{S4}$</th>
<th>$\mu_{C1}$</th>
<th>$\mu_{C2}$</th>
<th>$\mu_{P1}$</th>
<th>$\mu_{U1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.8</td>
<td>1.4</td>
<td>1</td>
<td>1.2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>(10,10)</td>
<td>(10,10)</td>
<td>(20,20)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>0.8</td>
<td>0.8</td>
<td>0.9</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td>0.005</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td>6</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td>7</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>(10,10)</td>
<td>(10,10)</td>
<td>(20,20)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.4</td>
<td>0.4</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>30</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.6</td>
<td>2.8</td>
<td>2</td>
<td>2.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.6</td>
<td>2.8</td>
<td>2</td>
<td>2.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.85</td>
<td>0.8</td>
<td>0.9</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.6</td>
<td>2.8</td>
<td>2</td>
<td>2.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>(5,5)</td>
<td>(5,5)</td>
<td>(10,10)</td>
<td></td>
</tr>
</tbody>
</table>
Case 1 is a case with no extreme valued parameters. The recycle probabilities of every material are equal to 0.5 and all the machines have a process rate of 1. The results of both the simulation and algorithm applied to this case are presented in Table 4.2.

For every machine the average throughput, the average sojourn times of each queue, and the blocking probabilities are computed. Sojourn time queue 1 is the average sojourn time of items in the first queue of a queueing model, and Sojourn time queue 2 denotes the average sojourn time of items in the second queue of a queueing model (which is only present in assembly queues). For machine \( P_1 \) and \( U_1 \) the blocking probability is not computed, since those machines can not be blocked. Instead, for \( U_1 \) the average queue length is computed. This is an important performance measure as it represents the average number of products on the market.

In the first rows the results from the simulation are shown, in the rows thereafter the computations from the approximation are presented, and in the last rows the accuracy of the approximation is computed for each performance measure. The accuracy is expressed as the relative difference between the performance measures computed by the simulation and approximation. However, especially for the blocking probabilities (or other performance measures with values close to zero) the relative difference can get very big while the absolute difference is actually very small. Therefore, when a big relative difference occurs with very small values, the accuracy can still be quite good. The format of the table of results is the same for all the cases in Section 4.

In Case 1 the material with the lowest arrival rate is material 1, which means that this material is the bottleneck as all the process rates of the machines (except \( U_1 \)) are equal to 1. The mean sojourn time of material 1 in the source \( S_1 \) is therefore also shorter than the mean sojourn time of the other materials in the other sources.

The blocking probability of source \( S_1 \) is, as one would expect since the arrival rate to \( S_1 \) is much lower than the arrival rate to \( S_2 \), much bigger than the blocking probability of source \( S_2 \). Also the throughput of \( U_1 \) is higher than the throughput of any source because of the recycling of materials, components, and products. Since machine \( U_1 \) is a \( M/M/\infty \) queue, the expected sojourn time is equal to the mean service time of the machine, which is \( \frac{1}{\mu_1} \).

The relative difference between the simulation and the approximation of the throughput of this case is at most 2.8%, which is an acceptable error. The sojourn time has a maximum relative difference of 9.7%, and although the relative difference of some of the estimated blocking probabilities is high (34.8%), the absolute difference is small. On average, the estimated performance measures are quite accurate.
Table 4.2: Example Case

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.393</td>
<td>0.503</td>
<td>0.45</td>
<td>0.5</td>
<td>0.559</td>
<td>0.586</td>
<td>0.689</td>
<td>0.735</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.685</td>
<td>6.308</td>
<td>1.986</td>
<td>3.604</td>
<td>2.282</td>
<td>3.639</td>
<td>4.096</td>
<td>199.806</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6.93</td>
<td>5.577</td>
<td>8.287</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.032</td>
<td>0.427</td>
<td>0.114</td>
<td>0.287</td>
<td>0.023</td>
<td>0.135</td>
<td>-</td>
<td>146.803</td>
</tr>
</tbody>
</table>

Approximation

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.393</td>
<td>0.489</td>
<td>0.456</td>
<td>0.509</td>
<td>0.55</td>
<td>0.579</td>
<td>0.671</td>
<td>0.716</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.653</td>
<td>6.719</td>
<td>1.871</td>
<td>3.423</td>
<td>2.304</td>
<td>3.47</td>
<td>4.493</td>
<td>200</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6.977</td>
<td>5.419</td>
<td>7.83</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.037</td>
<td>0.433</td>
<td>0.105</td>
<td>0.273</td>
<td>0.031</td>
<td>0.116</td>
<td>-</td>
<td>143.21</td>
</tr>
</tbody>
</table>

Relative difference

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0%</td>
<td>-2.8%</td>
<td>1.3%</td>
<td>1.8%</td>
<td>-1.6%</td>
<td>-1.2%</td>
<td>-2.6%</td>
<td>-2.6%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-1.9%</td>
<td>6.5%</td>
<td>-5.8%</td>
<td>-5.0%</td>
<td>1.0%</td>
<td>-4.6%</td>
<td>9.7%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>0.7%</td>
<td>-2.8%</td>
<td>-5.5%</td>
<td>-5.5%</td>
<td>-5.5%</td>
<td>-5.5%</td>
<td>-5.5%</td>
<td>-5.5%</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>15.6%</td>
<td>1.4%</td>
<td>-7.9%</td>
<td>-4.9%</td>
<td>34.8%</td>
<td>-14.1%</td>
<td>-2.4%</td>
<td>-2.4%</td>
</tr>
</tbody>
</table>

4.2 Results Case 2

Case 2 has parameters, except for the recycle probabilities, with doubled values of the parameters of Case 1. One would expect the throughput to be approximately doubled with regard to the throughput of Case 1, which clearly is the case. Machine S1, which produces the materials with the lowest arrival rates, therefore forms the bottleneck of the total model. For this reason, it is approximately never blocked and consequently behaves like an M/M/1 queue. From the standard formula for the queue length and the sojourn time of an M/M/1 queue (with arrival rate $\lambda$ and process rate $\mu$), which are given by

$$E[L] = \frac{\rho}{1-\rho}$$ and $$E[S] = \frac{\frac{\rho}{\mu}}{1-\rho},$$ (4.2.1)

where $\rho = \frac{\lambda}{\mu}$, it is easy to see that after doubling $\lambda$ and $\mu$ the mean queue length stays the same but the mean sojourn time is halved. This is exactly what happens to the performance measures of S1. The 'halved sojourn time property' also seems to hold for queue 1 of C1 and queue 1 of P1, which are the queues that follow up on machine S1. The reverse happens to the performance measures of the other (queues of the) machines. Their mean sojourn time increases only very little, while the average queue length more than doubles. Note that this is still consistent with Little's law on the mean queue length, the mean sojourn time and the throughput:

$$E[L] = E[S]\theta,$$ (4.2.2)
where \( \theta \) is the throughput of a machine. Note that, except for machine \( S_1 \) as it is almost never blocked, we can not use Little’s law with arrival rates \( \lambda \) since there is blocking involved.

The mean sojourn time of \( U_1 \) is completely independent of the performance measures of the other machines in the model, as it only depends on the process rate of the machine.

Except for the high relative difference for low values of blocking probabilities, the highest relative difference of 20.1\% is found in the mean sojourn time of queue 1 of machine \( S_4 \). The origin of this considerably high error is elaborated on in case 8 in Section 4.8.

Table 4.3: Example case with doubled parameters

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.8</td>
<td>1.079</td>
<td>0.977</td>
<td>1.105</td>
<td>1.195</td>
<td>1.304</td>
<td>1.493</td>
<td>1.592</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>0.837</td>
<td>6.65</td>
<td>1.291</td>
<td>3.03</td>
<td>1.229</td>
<td>3.592</td>
<td>2.074</td>
<td>99.854</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7.379</td>
<td>5.891</td>
<td>10.589</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.002</td>
<td>0.439</td>
<td>0.09</td>
<td>0.282</td>
<td>0.002</td>
<td>0.207</td>
<td>-</td>
<td>158.972</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approximation</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.8</td>
<td>1.038</td>
<td>0.989</td>
<td>1.146</td>
<td>1.194</td>
<td>1.33</td>
<td>1.492</td>
<td>1.591</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>0.834</td>
<td>7.185</td>
<td>1.061</td>
<td>2.42</td>
<td>1.224</td>
<td>2.959</td>
<td>2.004</td>
<td>100</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7.351</td>
<td>5.514</td>
<td>10.665</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.002</td>
<td>0.448</td>
<td>0.049</td>
<td>0.243</td>
<td>0.001</td>
<td>0.192</td>
<td>-</td>
<td>159.113</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relative difference</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.0%</td>
<td>-3.8%</td>
<td>1.2%</td>
<td>3.7%</td>
<td>-0.1%</td>
<td>2.0%</td>
<td>-0.1%</td>
<td>-0.1%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-0.4%</td>
<td>8.0%</td>
<td>-17.8%</td>
<td>-20.1%</td>
<td>-0.4%</td>
<td>-17.6%</td>
<td>-3.4%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-0.4%</td>
<td>-6.4%</td>
<td>0.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.0%</td>
<td>2.1%</td>
<td>-45.6%</td>
<td>-13.8%</td>
<td>0.0%</td>
<td>-7.2%</td>
<td></td>
<td>0.1%</td>
</tr>
</tbody>
</table>
4.3 Results Case 3

In this third case, all the recycle probabilities are equal to 0, which leads to a total average throughput that can not be bigger than the throughput of the source that forms the bottleneck of the model. As the material of type 1 has the lowest arrival rate of value 0.4, the throughput of all the machines is almost equal to this value.

Another observation is the halved mean queue length of machine \( U_1 \), which represents the average number of products used by customers at the same time. Clearly, the recycling has a lot of impact on the average queue length of \( U_1 \), which is not very surprising. In Section 4.4 the same case is treated, but instead with a lot of recycling.

Table 4.4: Example case without recycling

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.598</td>
<td>9.75</td>
<td>3.294</td>
<td>6.509</td>
<td>1.703</td>
<td>7.555</td>
<td>2.111</td>
<td>200.131</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.896</td>
<td>10.332</td>
<td>20.787</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.009</td>
<td>0.581</td>
<td>0.31</td>
<td>0.519</td>
<td>0.005</td>
<td>0.447</td>
<td>-</td>
<td>78.713</td>
</tr>
<tr>
<td>Approximation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>0.394</td>
<td>0.394</td>
<td>0.402</td>
<td>0.402</td>
<td>0.392</td>
<td>0.392</td>
<td>0.392</td>
<td>0.392</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.576</td>
<td>9.809</td>
<td>3.265</td>
<td>6.39</td>
<td>1.641</td>
<td>8.457</td>
<td>1.647</td>
<td>200</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10.954</td>
<td>10.492</td>
<td>23.216</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.006</td>
<td>0.58</td>
<td>0.306</td>
<td>0.508</td>
<td>0</td>
<td>0.527</td>
<td>-</td>
<td>78.432</td>
</tr>
<tr>
<td>Relative difference</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>0.3%</td>
<td>0.3%</td>
<td>2.3%</td>
<td>2.3%</td>
<td>-0.3%</td>
<td>-0.3%</td>
<td>-0.3%</td>
<td>-0.3%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-1.4%</td>
<td>0.6%</td>
<td>-0.9%</td>
<td>-1.8%</td>
<td>-3.6%</td>
<td>11.9%</td>
<td>-22.0%</td>
<td>-0.1%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>0.5%</td>
<td>1.5%</td>
<td>11.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>-33.3%</td>
<td>-0.2%</td>
<td>-1.3%</td>
<td>-2.1%</td>
<td>0.0%</td>
<td>17.9%</td>
<td>-0.4%</td>
<td></td>
</tr>
</tbody>
</table>
4.4 Results Case 4

The case considered in this section has very high recycle probabilities for every material. Theoretically, it is not possible to choose all recycle probabilities equal to 1. If none of the materials can get broken, only complete products are recycled and immediately return to machine $U_1$. The products can not leave the system anymore while materials keep being added to the system. This causes the model to never reach an equilibrium.

The mean queue length of $U_1$ has more than doubled in comparison with the queue length of $U_1$ in case 1 of Section 4.1, and approximately quadrupled in comparison with the case without recycling (Section 4.4). The recycle rates of Case 4 are much too high for real materials, but the case does show the great impact of these probabilities on the average queue length of $U_1$.

The accuracy of the approximation is quite good as the biggest relative difference is 4.6% (ignoring the high differences for the small blocking probabilities).

Table 4.5: Example case with high amount of recycling

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.393</td>
<td>0.495</td>
<td>0.459</td>
<td>0.471</td>
<td>0.57</td>
<td>0.569</td>
<td>0.845</td>
<td>1.656</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.711</td>
<td>6.585</td>
<td>1.788</td>
<td>4.397</td>
<td>2.405</td>
<td>2.958</td>
<td>5.188</td>
<td>200.064</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7.014</td>
<td>6.637</td>
<td>7.365</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.038</td>
<td>0.447</td>
<td>0.066</td>
<td>0.379</td>
<td>0.051</td>
<td>0.122</td>
<td>-</td>
<td>331.342</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approximation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.393</td>
<td>0.484</td>
<td>0.46</td>
<td>0.478</td>
<td>0.557</td>
<td>0.557</td>
<td>0.806</td>
<td>1.579</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.667</td>
<td>6.871</td>
<td>1.742</td>
<td>4.218</td>
<td>2.408</td>
<td>2.913</td>
<td>5.342</td>
<td>200</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7.037</td>
<td>6.468</td>
<td>7.074</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.042</td>
<td>0.454</td>
<td>0.065</td>
<td>0.372</td>
<td>0.064</td>
<td>0.129</td>
<td>-</td>
<td>315.854</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relative difference</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.0%</td>
<td>-2.2%</td>
<td>0.2%</td>
<td>1.5%</td>
<td>-2.3%</td>
<td>-2.1%</td>
<td>-4.6%</td>
<td>-4.6%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-2.6%</td>
<td>4.3%</td>
<td>-2.6%</td>
<td>-4.1%</td>
<td>0.1%</td>
<td>-1.5%</td>
<td>3.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-0.3%</td>
<td>-2.5%</td>
<td>-4.0%</td>
<td>-0.0%</td>
<td>0.0%</td>
<td>-1.5%</td>
<td>3.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>10.5%</td>
<td>1.6%</td>
<td>-1.5%</td>
<td>-1.8%</td>
<td>25.5%</td>
<td>5.7%</td>
<td>-4.7%</td>
<td>-4.7%</td>
</tr>
</tbody>
</table>
4.5 Results Case 5

Case 5 has the same parameters as Case 1 of Section 4.1, except for the recycle probabilities. In order to examine the impact of the amount of possible recycling of one material, only material 1 can be recycled (with a high probability).

Material 1 is, as it was in all the previous cases, the material which causes the bottleneck. In Case 6 (see Section 4.6) it is material 2 that is the only material that can be recycled while it is the material with the highest arrival rate.

As can be seen from the results in Table 4.6, it is no longer material 1 that forms the bottleneck. The throughput of machine $U_1$ is close to the arrival rate of the material with the second lowest arrival rate, which is material 3. In comparison with Case 3, the mean queue length of $U_1$ has increased with 17%. This implies that when a company can locate the material that forms the bottleneck, huge differences can be made in the total throughput of the production line.

Table 4.6: Example case with bottleneck recycle

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.346</td>
<td>0.461</td>
<td>0.461</td>
<td>0.461</td>
<td>0.461</td>
<td>0.461</td>
<td>0.461</td>
<td>0.46</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>4.607</td>
<td>7.482</td>
<td>1.703</td>
<td>4.49</td>
<td>8.574</td>
<td>2.388</td>
<td>19.68</td>
<td>199.976</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8.718</td>
<td>7.371</td>
<td>1.737</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.42</td>
<td>0.49</td>
<td>0.037</td>
<td>0.364</td>
<td>0.509</td>
<td>0</td>
<td>-</td>
<td>92.087</td>
</tr>
<tr>
<td>Approximation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>0.362</td>
<td>0.469</td>
<td>0.465</td>
<td>0.464</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>3.851</td>
<td>7.327</td>
<td>1.627</td>
<td>4.576</td>
<td>8.078</td>
<td>2.111</td>
<td>19.448</td>
<td>200</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8.737</td>
<td>7.682</td>
<td>1.854</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.397</td>
<td>0.48</td>
<td>0.021</td>
<td>0.366</td>
<td>0.487</td>
<td>0</td>
<td>-</td>
<td>92.0522</td>
</tr>
<tr>
<td>Relative difference</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>4.6%</td>
<td>1.5%</td>
<td>0.9%</td>
<td>0.7%</td>
<td>-0.2%</td>
<td>-0.2%</td>
<td>-0.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-16.4%</td>
<td>-2.1%</td>
<td>-4.5%</td>
<td>1.9%</td>
<td>-5.8%</td>
<td>-11.6%</td>
<td>-1.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>0.2%</td>
<td>4.2%</td>
<td>6.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>-5.5%</td>
<td>-2.0%</td>
<td>-43.2%</td>
<td>0.5%</td>
<td>-4.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
4.6 Results Case 6

Case 6 considers the same situation as Case 1 does, but this time only material 2 can be recycled. However, as this material is already the material with the highest arrival rate, the recycling has almost no influence on the rest of the system. In comparison with Case 3 in Section 4.3 only the performance measures of machines $S_2$ and $C_1$ are slightly changed. The relative difference in the simulated and approximated performance measures of machine $S_2$ however, are bigger than in Case 3, which suggest that a part of the approximation error finds its cause in the presence of blocking. In Case 8 in Section 4.8, this error is analyzed.

Table 4.7: Example case with recycling of material that is not the bottleneck

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.393</td>
<td>0.355</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
<td>0.393</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11.429</td>
<td>10.361</td>
<td>20.882</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.007</td>
<td>0.636</td>
<td>0.314</td>
<td>0.521</td>
<td>0.004</td>
<td>0.45</td>
<td>-</td>
<td>78.666</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approximation</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.394</td>
<td>0.375</td>
<td>0.402</td>
<td>0.402</td>
<td>0.392</td>
<td>0.392</td>
<td>0.392</td>
<td>0.392</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.575</td>
<td>10.554</td>
<td>3.263</td>
<td>6.388</td>
<td>1.62</td>
<td>8.453</td>
<td>1.647</td>
<td>200</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11.302</td>
<td>10.489</td>
<td>23.211</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.006</td>
<td>0.634</td>
<td>0.306</td>
<td>0.508</td>
<td>0</td>
<td>0.527</td>
<td>-</td>
<td>78.4446</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relative difference</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.3%</td>
<td>5.6%</td>
<td>2.3%</td>
<td>2.3%</td>
<td>-0.3%</td>
<td>-0.3%</td>
<td>-0.3%</td>
<td>-0.3%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-0.6%</td>
<td>-8.2%</td>
<td>-1.7%</td>
<td>-2.4%</td>
<td>-1.0%</td>
<td>10.9%</td>
<td>-21.2%</td>
<td>-0.2%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-1.1%</td>
<td>1.2%</td>
<td>11.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>-14.3%</td>
<td>-0.3%</td>
<td>-2.5%</td>
<td>-2.5%</td>
<td>-100.0%</td>
<td>0.0%</td>
<td>-0.3%</td>
<td></td>
</tr>
</tbody>
</table>
4.7 Results Case 7

This section considers Case 7, in which the sizes of the buffers of all the machines are doubled with respect to the parameters of Case 1. Comparing Table 4.2 with Table 4.8 implies that the doubled buffer sizes almost have no influence on the throughput of the system and on the performance measures of \( U_1 \). Moreover, some of the relative differences between the simulation and the approximation have become quite large (around the 20%). The latter has everything to do with the error discussed in Section 4.8.

Table 4.8: Example case with doubled buffer sizes

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.4</td>
<td>0.539</td>
<td>0.489</td>
<td>0.552</td>
<td>0.598</td>
<td>0.652</td>
<td>0.746</td>
<td>0.796</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>1.672</td>
<td>13.277</td>
<td>2.594</td>
<td>6.036</td>
<td>2.456</td>
<td>7.205</td>
<td>4.132</td>
<td>199.798</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>14.763</td>
<td>11.768</td>
<td>21.231</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.002</td>
<td>0.439</td>
<td>0.09</td>
<td>0.281</td>
<td>0.001</td>
<td>0.208</td>
<td>-</td>
<td>159.026</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relative difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
</tr>
</tbody>
</table>
4.8 Results Case 8

Case 8 is designed to amplify the cause of an error that occurred in some of the previous cases. The arrival rates $\lambda_1$ and $\lambda_2$ are equal and small in comparison with arrival rates $\lambda_3$ and $\lambda_4$. The process rates of all the machines are chosen big in comparison with the arrival rates, so that the dependency of the total model on the value of the arrival rates is bigger. Also note that the sizes of the buffers of sources $S_1$ and $S_2$ are very large and that there is no recycling possible.

Table 4.9 shows big relative differences for the mean sojourn time of materials processed in sources $S_1$ and $S_2$. Since the approximated mean sojourn time is smaller than the simulated mean sojourn time, the (absolute) relative difference can never get bigger than 100%. However, the real difference between the approximated mean sojourn time and the simulated mean sojourn time can get arbitrary large.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.394</td>
<td>0.394</td>
<td>0.394</td>
<td>0.394</td>
<td>0.394</td>
<td>0.394</td>
<td>0.394</td>
<td>0.394</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>15.506</td>
<td>18.102</td>
<td>7.336</td>
<td>9.874</td>
<td>5.55</td>
<td>12.584</td>
<td>0.104</td>
<td>9.999</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6.258</td>
<td>12.584</td>
<td>25.272</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.393</td>
<td>0.446</td>
<td>0.961</td>
<td>0.961</td>
<td>0</td>
<td>0.961</td>
<td>-</td>
<td>3.94</td>
</tr>
</tbody>
</table>

**Approximation**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.4</td>
<td>0.4</td>
<td>0.395</td>
<td>0.395</td>
<td>0.395</td>
<td>0.395</td>
<td>0.395</td>
<td>0.395</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>0.393</td>
<td>0.393</td>
<td>7.324</td>
<td>9.857</td>
<td>4.638</td>
<td>12.561</td>
<td>0.104</td>
<td>10</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.638</td>
<td>12.563</td>
<td>25.232</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.108</td>
<td>0.108</td>
<td>0.96</td>
<td>0.96</td>
<td>0</td>
<td>0.961</td>
<td>-</td>
<td>3.94701</td>
</tr>
</tbody>
</table>

**Relative difference**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>1.5%</td>
<td>1.5%</td>
<td>0.3%</td>
<td>0.3%</td>
<td>0.3%</td>
<td>0.3%</td>
<td>0.3%</td>
<td>0.3%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-97.5%</td>
<td>-97.8%</td>
<td>-0.2%</td>
<td>-0.2%</td>
<td>-16.4%</td>
<td>-0.2%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-25.9%</td>
<td>-0.2%</td>
<td>-0.2%</td>
<td>-0.2%</td>
<td>-25.9%</td>
<td>-0.2%</td>
<td>-0.2%</td>
<td>-0.2%</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>-72.5%</td>
<td>-75.8%</td>
<td>-0.1%</td>
<td>-0.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

To see why this phenomenon occurs, consider a model consisting of two $M/M/1/c$ sources followed by an assembly system. The arrival rates to the sources have the same values and the process rates of all the machines in the model are so high that the process times are negligible. The buffer sizes of the sources are $K^{S_1}$ and $K^{S_2}$ and are assumed to be equal. The buffer sizes of the two queues of the assembly machine are denoted by $K_1$ and $K_2$. Assume that the arrival rates to the assembly machine (the departure rates out of sources $S_1$ and $S_2$) are $\lambda_1$ and $\lambda_2$. Since the arrival rates to the two sources are equal and the two sources themselves are identical, the *average* throughput is also the same.
The approximation algorithm therefore treats the assembly system as a separate assembly machine with equal arrival rates and high process rate. The latter implies that the assembly system will be approximately a pure assembly machine that is most of the time in a state in which one of the queues is empty (see the red states in Figure 4.1).

Neglecting the process time, the red strip of states is actually a finite version of the strip of states presented in Figure 2.2, with length $K_1 + K_2 + 1$. Since the arrival rates to the assembly system are equal, the probability of going to either 'left' or 'right' are both $\frac{1}{2}$. Eventually, this means that the system has equal equilibrium for every state in the 'red strip', specifically:

$$p_{i,j} \approx \frac{1}{K_1 + K_2 + 1} \quad \text{if } i = 0 \text{ and/or } j = 0,$$

$$p_{i,j} \approx 0 \quad \text{otherwise.}$$

This is exactly how the approximation algorithm approximates the equilibrium probabilities of the assembly system in isolation. Unfortunately, here the assumption of the algorithm to view every
machine as a separate system causes the error. In reality, the strip of states does not have a length of \( K_1 + K_2 + 1 \) but is has a 'hidden' extension. Since the process rates of the sources are chosen very large, the sources only form an extra buffer for the assembly machine. The length of the strip of states is therefore extended with the buffer sizes of the two sources (see Figure 4.2, where \( K_1^* = K_1 + K_{S1} \) and \( K_2^* = K_2 + K_{S2} \)).

![Figure 4.2: The actual flow diagram](image)

This effect causes the assembly machine to spend much more time in states \((K_1, 0)\) and \((0, K_2)\) than predicted by the equilibrium probabilities computed in the approximation algorithm. Consequently, the sources are much more blocked in the simulation than in the approximation, which also causes the remarkable difference in mean sojourn times. The absolute difference in mean sojourn times computed in the simulation and the approximation algorithm can get arbitrary big when the buffer sizes of the sources are increased.

### 4.9 Results Case 9, Case 10 and Case 11

In order to get more insights in the benefits of recycling Cases 9, 10 and 11 consider situations in which ratio of arrival rates and process rates to and of the sources, is lower than in the previous cases. In Case 9 all the recycling probabilities are equal to \( \frac{1}{2} \), in Case 10 (see Table 4.11) the recycling probabilities are high, and in the last Case 11 (see Table 4.12) there is no possibility for recycling.
### Table 4.10: Case with high arrival and low process rate

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.44</td>
<td>0.44</td>
<td>0.439</td>
<td>0.441</td>
<td>0.574</td>
<td>0.574</td>
<td>0.698</td>
<td>0.744</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>8.254</td>
<td>10.933</td>
<td>6.21</td>
<td>8.575</td>
<td>3.936</td>
<td>3.909</td>
<td>5.664</td>
<td>199.738</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.957</td>
<td>3.962</td>
<td>5.711</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.117</td>
<td>0.119</td>
<td>0.115</td>
<td>0.119</td>
<td>0.052</td>
<td>0.054</td>
<td>-</td>
<td>148.633</td>
</tr>
</tbody>
</table>

**Approximation**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.445</td>
<td>0.446</td>
<td>0.444</td>
<td>0.446</td>
<td>0.555</td>
<td>0.555</td>
<td>0.663</td>
<td>0.707</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.149</td>
<td>4.156</td>
<td>5.954</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.144</td>
<td>0.147</td>
<td>0.142</td>
<td>0.147</td>
<td>0.064</td>
<td>0.062</td>
<td>-</td>
<td>141.489</td>
</tr>
</tbody>
</table>

**Relative difference**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>1.1%</td>
<td>1.4%</td>
<td>1.1%</td>
<td>1.1%</td>
<td>-3.3%</td>
<td>-3.3%</td>
<td>-5.0%</td>
<td>-5.0%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-1.5%</td>
<td>-1.4%</td>
<td>-1.3%</td>
<td>-1.2%</td>
<td>4.5%</td>
<td>4.7%</td>
<td>6.6%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>4.9%</td>
<td>4.9%</td>
<td>4.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>23.1%</td>
<td>23.5%</td>
<td>23.5%</td>
<td>23.5%</td>
<td>23.1%</td>
<td>14.8%</td>
<td>-</td>
<td>-4.8%</td>
</tr>
</tbody>
</table>

### Table 4.11: Case with high arrival and low process rate, more recycling

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.451</td>
<td>0.423</td>
<td>0.461</td>
<td>0.492</td>
<td>0.607</td>
<td>0.572</td>
<td>0.877</td>
<td>1.717</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>8.032</td>
<td>11.412</td>
<td>5.888</td>
<td>9.467</td>
<td>3.489</td>
<td>3.203</td>
<td>5.868</td>
<td>200.059</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.417</td>
<td>5.234</td>
<td>6.628</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.095</td>
<td>0.154</td>
<td>0.073</td>
<td>0.196</td>
<td>0.073</td>
<td>0.096</td>
<td>-</td>
<td>343.415</td>
</tr>
</tbody>
</table>

**Approximation**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.455</td>
<td>0.433</td>
<td>0.463</td>
<td>0.415</td>
<td>0.583</td>
<td>0.553</td>
<td>0.822</td>
<td>1.61</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.522</td>
<td>5.227</td>
<td>6.557</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.125</td>
<td>0.192</td>
<td>0.095</td>
<td>0.233</td>
<td>0.084</td>
<td>0.111</td>
<td>-</td>
<td>321.931</td>
</tr>
</tbody>
</table>

**Relative difference**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.9%</td>
<td>2.4%</td>
<td>0.4%</td>
<td>3.2%</td>
<td>-4.0%</td>
<td>-3.3%</td>
<td>-6.3%</td>
<td>-6.2%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-1.3%</td>
<td>-2.4%</td>
<td>-0.7%</td>
<td>-3.5%</td>
<td>6.0%</td>
<td>6.9%</td>
<td>-0.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>2.4%</td>
<td>-0.1%</td>
<td>-1.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>31.6%</td>
<td>24.7%</td>
<td>30.1%</td>
<td>18.9%</td>
<td>15.1%</td>
<td>15.6%</td>
<td>-</td>
<td>-6.3%</td>
</tr>
</tbody>
</table>
### 4.9. RESULTS CASE 9, CASE 10 AND CASE 11

Table 4.12: Case with high arrival and low process rate, without recycle

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.432</td>
<td>0.432</td>
<td>0.432</td>
<td>0.432</td>
<td>0.432</td>
<td>0.432</td>
<td>0.432</td>
<td>0.432</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.943</td>
<td>4.914</td>
<td>7.777</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.133</td>
<td>0.137</td>
<td>0.13</td>
<td>0.136</td>
<td>0.095</td>
<td>0.086</td>
<td>-</td>
<td>86.339</td>
</tr>
</tbody>
</table>

**Approximation**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.446</td>
<td>0.446</td>
<td>0.446</td>
<td>0.446</td>
<td>0.438</td>
<td>0.438</td>
<td>0.437</td>
<td>0.437</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>8.114</td>
<td>10.784</td>
<td>6.103</td>
<td>8.464</td>
<td>4.588</td>
<td>4.475</td>
<td>7.948</td>
<td>200</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.669</td>
<td>4.624</td>
<td>6.704</td>
<td>-</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>0.119</td>
<td>0.123</td>
<td>0.115</td>
<td>0.122</td>
<td>0.06</td>
<td>0.044</td>
<td>-</td>
<td>87.4183</td>
</tr>
</tbody>
</table>

**Relative difference**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>C1</th>
<th>C2</th>
<th>P1</th>
<th>U1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>3.2%</td>
<td>3.2%</td>
<td>3.2%</td>
<td>3.2%</td>
<td>1.4%</td>
<td>1.4%</td>
<td>1.2%</td>
<td>1.2%</td>
</tr>
<tr>
<td>Sojourn time queue 1</td>
<td>-3.7%</td>
<td>-3.3%</td>
<td>-3.5%</td>
<td>-3.3%</td>
<td>-5.9%</td>
<td>-6.7%</td>
<td>-3.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Sojourn time queue 2</td>
<td>-5.5%</td>
<td>-5.9%</td>
<td>-13.8%</td>
<td>-5.9%</td>
<td>-13.8%</td>
<td>-13.8%</td>
<td>-5.9%</td>
<td>-13.8%</td>
</tr>
<tr>
<td>Block prob / Queue length</td>
<td>-10.5%</td>
<td>-10.2%</td>
<td>-11.5%</td>
<td>-10.3%</td>
<td>-36.8%</td>
<td>-48.8%</td>
<td>-13.8%</td>
<td>1.3%</td>
</tr>
</tbody>
</table>

Comparing the results presented in Tables 4.10, 4.11, and 4.12, it becomes clear that in all three cases the low process rates of the sources form the bottleneck. However, in the case with mediate or much recycling, the throughput (from source to user) gets larger because of the recycling. The mean number of products at the 'market' and at the user (the queue length of \( U_1 \)) from Case 9, is twice as big as in Case 11. The mean number of products at the market and at the user is even four times bigger in Case 10, which shows the promising effects of being able to recycle.
Chapter 5

Conclusion and discussion

In this concluding section, a variety of topics is discussed. In Section 5.1 the results of Section 4 are discussed and recommendations are given regarding the use of the algorithm. Section 5.2 considers the implications of this research for the circular economy. In both sections, the possibilities for further research about assembly queues, the approximation algorithm, and their place in the circular economy is considered.

5.1 Conclusion and recommendations regarding the algorithm

In Section 4 eleven cases were considered in which performance measures were computed by simulation and approximated by the algorithm discussed in Section 3.3.2. In almost all of the cases, the relative differences between the results of the simulation and the results of the approximation were small, despite of the assumptions that were made by applying the algorithm which are heavily simplifying the model.

First of all the assumption is made that the departure process of every machine can be approximated by a Poisson process, which is obviously not the case with most of the machines in queueing models.

Secondly, the algorithm treats every machine as a stand alone queueing system. This is exactly the cause of the error discussed in Section 4.8. As stated before, this error can be made arbitrarily big by creating an environment in which the arrival rates to an assembly machine are equal and by increasing the buffer size(s) of the machines scheduled before this assembly machine. Fortunately, it is not realistic to have very large buffers as the space consumption (temporary storage of materials, components, or products) costs money.

The algorithm approximates values by using only first moments and fitting distributions on only this first moments. Before the implementation of the algorithm, it was not clear how much accuracy this simplifying assumption would cost. Testing the algorithm suggests that the accuracy does not suffer much from this assumption.

It might be possible to achieve more accuracy by approximating not only on the first moments of the departure and arrival distribution, but also on the second moment. Instead of fitting on exponential distributions, one could try fitting the distribution on an Erlang-2 distribution or a
Cox distribution depending on the squared coefficient of variation. However, it might be more interesting to firstly focus on the error that occurs because of the separate consideration of two consecutive machines in the production chain, as this error can get arbitrarily big. An option to avoid the error, is by expanding the Markov chain by conditioning on whether the departures of materials, components or products trigger a blocking event or not. The reasoning behind this is very simple, as it is more likely for a machine to get blocked again when it was blocked an instance ago than when the machine has not been blocked for a while. In his research of performance analysis of production chains, Bierbooms (2012) uses Markov chains that include this blocking dependency. It might be interesting to use similar methods to improve the accuracy. However, by conditioning on blocking, the Markov chain can grow in size very fast which could make an approximation algorithm slow.

It can also be interesting to see how well the algorithm approximates the performance measures of larger production chains. How large does the total model have to be in order for the algorithm to become unable to approximate accurate performance measures?

Whenever the algorithm tries to recycle a material, component or product of which there are already enough (the buffer of the corresponding machine is full), it throws it away and it leaves the system. In real life it is more realistic to lower the production rate or the arrival rates whenever recycling is sufficient enough. In further research, this option could be included in the algorithm. Also, since the approximation algorithm approximates the performance measures in the equilibrium of the queueing model, it is not capable of approximating the expected amount of time needed to reach this equilibrium. The simulation could be used compute this time.

All of the cases that were tested in Section 4 are tested on the same production chain. However, in reality there are a lot of different production chains in which all kinds of queueing systems are combined. Of course the algorithms constructed in Section 3 are only applicable to production chains with the exact same structure as the models described in the same section. However, the cases are just designed to be an example of how such an algorithm can be constructed and applied. What might be the most important to the reader, are the derived ways to calculate the equilibrium probabilities, the with blocking updated process rates, and the with starvation and recycling updated arrival rates to the different kinds of queueing machines in the model. By using those derived relations, the algorithm can easily be adapted to models with other structures. Since the algorithm considers every machine as a separate machine in the same order as the flow of materials, components, and products in the model, the general setup of the algorithm will not change drastically for other production systems.

5.2 Conclusion and recommendations to use the model for the quantification of the circular economy

It is hard to compare the specific cases considered in Section 4 with a general production chain, as every product has its own specific (circular) production chain. As the whole concept of circular
5.2. RECOMMENDATIONS FOR THE CIRCULAR ECONOMY

business is still quite new to producers, there are no real and available business cases to which the algorithms developed in this research can be tested. However, a company can still use the content of this research to develop their own algorithm (corresponding to their production chain) that is able to approximate and predict performance measures.

By testing and ‘playing’ with the settings of the parameters of their production chain, such a model can help the company in improving their business in multiple ways. Firstly, it is possible to locate the bottleneck in the production chain and find the best possible changes in the production chain that bring forth the maximal desired output. Secondly, the algorithm can help quantifying and predicting the extra profit that can be made by extending a linear production chain to a circular production chain.

As far as possible, a few practical conclusions can be drawn from the results of Section 4.

- Increasing buffer sizes is in most cases not the best solution. When a buffer is full, it is very likely caused by either the lack of other materials, components or products to combine with or by too slow processing rates.

- In Cases 1 and 9 it was possible to more than quadruple the average amount of products in use by increasing the recycling rate (see Cases 4 and 10), which shows the impact a circular policy can have.

- Increasing the recycling of materials, components or products that form the bottleneck in a production chain gives more result than increasing the recycling of materials, components or products that do not form the bottleneck. The bottlenecks can be easily traced with the use of the results of the algorithm.

A few more concrete examples in which an algorithm would provide the user with useful insights are:

- The whole production chain of a product (or service) is owned by a company. By scenario analysis the company can decide which (circular) strategy delivers the most benefits and profit. A model could be used within an optimization setting for computing the needed amount of raw materials when the recycling streams of materials, components and products are increased. Example:
  Consider a production chain in which light is sold as a service. Company A expects the demand of the light to be equal to X lighting hours next year. Questions that could be answered with a queueing model: How many raw materials need to be mined in order to satisfy the demand of light during the year (given the life time of the lamps)? And how much would recycling on any level (recycling/refurbishing/redistributing) reduce the number of mined materials?

Another example can be found in the results, comparing Case 1 and 3. In Case 3, no recycling is possible, while in Case 1 the recycling probabilities are equal to 0.5 for each material. This results in a doubled average queue length at $U1$. The algorithm not only shows the big impact of recycling, but also quantifies it.
• It is very likely that a production chain contains multiple factories owned by different companies (factory A, factory B, and factory C). A queueing algorithm can help to determine the added value of the implementation of the circular economy for each company separately, while also providing insights in the total added value of the complete production chain. Example: Factory B wants to lower their production rate while focusing on the recycling possibilities of their product. How does this influence the total stream of materials, components, and products? How much do the performance measures of factory B depend on the other two factories?

• As circularity can be found and applied in a wide variety of domains, the following example considers a circular model within the production line of a factory that fills boxes with a predetermined combination of products (of possibly the same type). Thereby, scenario analysis can be used to find optimal strategies within a given setting. Example: Company C is specialized in filling boxes with a predetermined combination of products (of possibly the same type). It is possible for the products to fall off the assembly line which causes the products not to be packed in a box. The event can also occur in which a product is not packed at the end of the assembly line when there was no box available for it to be packed in. The product has to rejoin the assembly line and is packed (if no failure occurs) in another box. Here the circularity takes its form in the products rejoining the assembly line. How does the speed of the (possibly) different production lines affect the number of failures (due to blocking or products falling off the assembly line? How can be dealt with failures in an optimal way?

By creating more insights in the possibilities and advantages a circular production chain and circular economy has to offer, the threshold for companies to actually switch to a more circular business policy can be made lower. In further research this application of queueing theory should be tested to real business cases, since it is the link between the application of queueing theory and the circular economy that creates the value of this report.


