Real-Time Control of Production Systems with Constrained Time Windows

by

Feifan Wang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2021 by the
Graduate Supervisory Committee:

Feng Ju, Chair
Ronald Askin
Pitu Mirchandani
Nital Patel

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

A production system is commonly restricted by time windows. For example, perishability is a major concern in food processing and requires products, such as yogurt, beer and meat, not to stay in buffer for long. Semiconductor manufacturing is faced with oxidation and moisture absorption issues, if a product in buffer is exposed to air for long. Machine reliability is a major source of uncertainty in production systems that causes residence time constraints to be unsatisfied, leading to potential product quality issues. Rapid advances in sensor technology and automation provide potentials to manage production in real time, but the system complexity, brought by residence time constraints, makes it difficult to optimize system performance while providing a guaranteed product quality.

To contribute to this end, this dissertation is dedicated to modeling, analysis and control of production systems with constrained time windows. This study starts with a small-scale serial production line with two machines and one buffer. Even the simplest serial lines could have too large state space due to the consideration of residence time constraints. A Markov chain model is developed to approximately analyze its transient behavior with a high accuracy. An iterative learning algorithm is proposed to perform real-time control.

The analysis of two-machine serial line contributes to the further analysis of more general and complex serial lines with multiple machines. Residence time constraints can be required in multiple stages. To deal with it, a two-machine-one-buffer subsystem isolated from a multi-stage serial production line is firstly analyzed and then acts as a building block to support the aggregation method for overall system performance. The proposed aggregation method substantially reduces the complexity of the problem while maintaining a high accuracy.

A decomposition-based control approach is proposed to control a multi-stage serial production line. A production system is decomposed into small-scale subsystems, and an iterative aggregation procedure is then used to generate a production control policy. The decomposition-based control approach outperforms general-purpose reinforcement learning method by delivering significant system performance improvement and substantial reduction on computation overhead.

Semiconductor assembly line is a typical production system, where products are restricted by time windows and production can be disrupted by machine failures. A production control problem of semiconductor assembly line is presented and studied, and thus total lot delay time and residence time constraint violation are minimized.

DEDICATION

To my parents,

who gave me the courage to pursue the Ph.D. degree

ACKNOWLEDGMENTS

I would like to thank Dr. Feng Ju for being my research advisor and dissertation committee chair. He is a wise researcher, easy-going professor, patient supervisor and good friend. I am grateful that I can learn so much from him. Without his support, this dissertation could not have been possible. I would also like to express my gratitude to my dissertation committee members, Dr. Ronald Askin, Dr. Pitu Mirchandani and Dr. Nital Patel, for helpful advice on my research.

I am grateful for my collaborators, Dr. Yan Lu from National Institute of Standards and Technology, Nils Hofmann and Dr. Kyle Rowe from Local Motors, Balakrishnan Ananthanarayanan and Dr. Husam Dauod from Intel, and Dr. Yu-Li Huang from Mayo Clinic. They have provided me with great opportunities to work on real-world problems, and their extensive experience allowed me to know production systems and health care delivery systems better.

I would also like to thank all my labmates in the lovely academic family. I appreciate the invaluable help from Dr. Yunyi Kang, as I was starting my journey in the Ph.D. program. Thank you to Sepehr Fathizadan, Yingjie Ren, Chih-Kai Chang and Yutong Su. It is enjoyable to work with them.

I want to say thank you to Penghui Zhang, Chao Wang, Dr. Guanqi Fang, Xiushuang Li, Xufeng Yao, Dr. Yuhan Sun, Dr. Carly Metcalfe, Brittany Fischer, Logan Mathesen, and many other ASU friends, who made my life at ASU memorable.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Background

A part in a production system spends a certain amount of time in buffer before entering its downstream machine for further processes. In many production systems, the amount of time that a part spends in buffer imposes a significant impact on the quality of its final product. This kind of production systems can be seen in food industry, battery production, automotive paint shop, semiconductor manufacturing, etc. The time that a part has been staying in a buffer is its residence time. To guarantee an acceptable quality of the final product, residence time constraints, including maximum allowable residence time and minimum required residence time, are always set to specify the upper bound and lower bound of residence time, respectively.

Constraints on residence time are widely studied in food industry due to its perishability. For instance, perishable nature is an important feature in dairy production (Wang *et al.*, 2010). Yogurt goes through several processes from raw milk to intermediate products and finally to final products, and each stage is under strict time limits (Amorim *et al.*, 2013). Agricultural products are often subject to perishability (Ahumada and Villalobos, 2011). One example is the kimchi, which occupies a large share of overall consumption of main agricultural products in Korea. To produce kimchi, cabbage is processed to be salted cabbage, and it is then further processed to be kimchi (Shin *et al.*, 2019). Both the cabbage and the salted cabbage are perishable, and a storage time window exists for each stage of the kimchi production (Shin *et al.*, 2019). Pork is highly consumed meat. One challenging problem for meat packing plant is

the planning and scheduling of operations for processing carcasses, and it is aimed at managing inventories of perishable products and balancing the benefits between demand and production (Albornoz *et al.*, 2015). Xie and Li (2012) introduce a case study from a meat production company that produces hot dog, bologna, bacon, etc., with perishability concerns. The perishability is also an issue in production systems of bread and bakery products. An entire automated production line is often used to produce bread and bakery products, and quality deterioration of bread and bakery products could occur during the stoppage due to machine failure (Liberopoulos and Tsarouhas, 2005).

Residence time constraints are considered in battery production (Ju, 2015; Ju *et al.*, 2017b). Chemical materials in cell assembly process are filled into cells to form electrodes. Cells are baked to a certain temperature, before each material can be added. Those processes should be done within a certain time limit. A cell will be scrapped, if such a time limit cannot be satisfied. Intermediate batteries often have to wait in buffer due to demand uncertainty, technical failures and other production disruptions (Ju, 2015).

In automotive paint shop, the car body goes into oven to solidify the coating. The car body needs to cool down after the heat treatment, and it takes a small amount of time. However, the longer the time that a car body is exposed to air, the more likely that the car body is contaminated with dirt. The analysis and control of the automotive paint shop should take residence time constraints into consideration.

In semiconductor manufacturing, residence time receives substantial attention as well. In semiconductor fabrication, cluster tools are adopted to process wafers. A general cluster tool contains several process modules, and the time that a wafer stays in a process module should be both lower and upper bounded (Yang *et al.*, 2016; Pan *et al.*, 2017; Wang *et al.*, 2018b). A wafer will be premature if it leaves the

process module earlier than the minimum required residence time, whereas a wafer will have quality problems if it stays in the process module for a longer time than the maximum allowable residence time. Different types of time limits are also observed in other process steps of wafer fabrication (Klemmt and Mönch, 2012; Kim and Lee, 2019). Products in a semiconductor assembly line are restricted by time windows due to the concerns of oxidation and moisture absorption (Han and Kim, 2017). Failures caused by moisture include popcorn cracking (Galloway and Miles, 1996), deformation (Yoon *et al.*, 2008) and adhesion degradation (Tee and Zhong, 2004).

Production systems with residence time constraints are difficult to monitor and control. Traditionally, the analysis of a such production system is based on their steady states. Long-term behaviors of a production system are analyzed to obtain heuristic approaches that keep the production system in a low defect rate. As a result, production capacity is often not fully utilized due to the quality concern. The rapid development in information and communication technologies (ICT) provides new opportunities to monitor and control production systems with residence time constraints (Lu and Ju, 2017; Wang *et al.*, 2017; Stephan *et al.*, 2010; Weyer *et al.*, 2015). The state of a production system can be monitored, and proper controls based on real-time data are able to be applied. However, as residence time and its constraints are required in the modeling, it imposes difficulty on analysis and control of a production system. In this dissertation, production systems with residence time constraints are studied. The study is aimed at providing production engineers and managers quantitative tools for system performance evaluation and real-time operation management.

1.2 Analysis and Control of Two-Machine Serial Line

My research starts with a two-machine serial line with residence time constraints. It is the simplest serial line that consists of two machines and one buffer. The study on a two-machine serial line will play as an building block for the study on a multi-stage serial line.

However, it is difficult to analyze and control a serial line with residence time constraints, even for those with only two machines and one buffer. The challenge lies in the large state space of the serial line. As all parts in the buffer is under residence time constraints, the buffer occupancy is not sufficient to capture the state of a buffer. The residence time of each part is also important information to describe system dynamics. The consideration of residence time of each part can result in a large state space that is too large to perform any analysis.

In order to optimize the production performance in a timely manner, the transient behavior of the two-machine serial line with residence time constraints and a real-time control strategy need to be investigated. In this study, a Markov chain model is developed to analyze the transient behavior of a two-machine geometric serial line with constraints on both maximum allowable residence time and minimum required residence time. Compared with the simulation, the proposed analytical method is shown to estimate the system's transient performance with high accuracy. Structural properties are investigated based on the model to provide insights into the effects of residence time constraints and buffer capacity on system performance. An iterative learning algorithm is proposed to perform real-time controls, which improves the system performance by balancing the trade-off between the production rate and scrap rate. Specifically, a control policy derived from Markov Decision Processes is

implemented as an initial control policy, and the Bayesian method is then applied to the run time data to improve the control policy.

1.3 Analysis of Multi-Stage Serial Lines

Multi-stage serial line refers to a serial line with more than two machines. Though a two-machine serial line with residence time constraints can be analyzed and controlled in real time, the method cannot be directly extended to the multi-stage line with residence time constraints. The challenge still lies in the large state space. The state space grows exponentially, as the number of machines in a multi-stage line increases.

In this study, a method to analyze a multi-stage serial line with residence time constraints is proposed. To analyze such a system, a two-machine-one-buffer subsystem, isolated from the multi-stage geometric serial production line, is taken as a building block, and develop a Markov chain model to analyze the subsystem first. Based on the analysis of two-machine-one-buffer subsystems, the aggregation method is applied to obtain both the steady-state and transient performance of a multi-stage serial production line. Through simulation experiments, the proposed aggregation method is shown to maintain a high accuracy in estimating both steady state and transient performance measures. The main contribution of this study is twofold. One is the approximate method to model a two-machine-one-buffer subsystem, which makes the analysis of a two-machine-one-buffer subsystem efficient and also provides a building block for the aggregation method. The other is the aggregation method, which evades the direct modeling for multi-stage serial production lines with residence time constraints.

1.4 Decomposition-Based Real-Time Control of Multi-Stage Serial Lines

Residence time constraints are often required in multiple stages of a production system, but limited research has been on real-time production control of such a large-scale production system. An extendable method that supports real-time control of a multi-stage serial line is worth studying.

In this study, a novel decomposition-based control approach is proposed by decomposing a production system into small-scale subsystems based on domain knowledge and their structural relationship. The subsystems are simple enough to derive a control solution using local information. An iterative aggregation procedure is then used to generate production control policy with convergence guarantee. Compared with a general-purpose reinforcement learning based method, the decomposition-based control can deliver significant improvements on system performance and substantial reduction on computation overhead, which makes it applicable for real-time production decision making.

1.5 Fast Response to Machine Failures in Semiconductor Assembly Lines

Semiconductor assembly line is a complex production system. Benefiting from its flexibility, a semiconductor assembly line can process different types of products from different orders at the same time, and a master schedule is created every shift to optimize production within the next three weeks. However, semiconductor assembly lines are susceptible to machine failures, causing the master schedule to be sub-optimal or even infeasible. Specifically, machine failures can result in due time not to be met and residence time constraints to be violated. In this study, machine failures in semiconductor assembly lines are classified into two categories, short machine failure and long machine failure. To handle short machine failure, extra time is added to

each operation of each lot to make the master schedule robust. The assembly line can recover on its own without intervention. When long machine failure occurs, a mixed integer programming model is formulated to adjust the master schedule. The original master schedule is taken as a warm start, and a short period schedule is obtained with CPLEX Optimizer for the semiconductor assembly line to follow immediately. In this way, the semiconductor assembly line can respond to long machine failure fast without replacing the whole master schedule, or it can give master scheduler enough time to remake a new master schedule. Thus, the negative impact of machine failure is minimized. Data from shop floor are collected. Using those data, a simulation model is developed with Python and SimPy package to simulate a real-world semiconductor assembly line and evaluate the proposed method. The experiment results show that the proposed method can achieve fast response to machine failures in semiconductor assembly lines.

1.6 Organization of the Document

The rest of the dissertation is organized as follows. Chapter 2 reviews the related literature. Chapter 3 presents a method to analyze and control a two-machine serial line with residence time constraints. Chapter 4 extends system analysis from a two-machine serial line to a multi-stage serial line. In Chapter 5, a novel method to control a multi-stage serial line is proposed. Chapter 6 introduces and solves a real-world production control problem in semiconductor assembly line. Finally, Chapter 7 concludes the dissertation.

Chapter 2

LITERATURE REVIEW

2.1 Residence Time Constraints Due to Perishability

Perishability is usually a main reason why residence time constraints should be considered, and it is a process that prevents an item from being used for its intended original use (Raafat, 1991). Perishability includes decay, damage, spoilage, evaporation, obsolescence, pilferage, loss of utility of loss of marginal value of a commodity that results in decreasing usefulness from the original one (Kärkkäinen, 2003). The terms, deterioration and perishability, are usually interchangeable (Pahl and Voß, 2014). Research on this kind of models was initially applied to blood banks and distribution of blood (Dumas and Rabinowitz, 1977; Brodheim and Prastacos, 1979; Beliën and Forcé, 2012; Pahl and Voß, 2014).

Different classifications are proposed to deal with perishability. For example, Nahmias (1982) classifies perishability into fixed lifetime perishability and random lifetime perishability. In fixed lifetime perishability, lifetime is a specified number of periods or a length of time independent of all other parameters in the system. In random lifetime perishability, lifetime is a random variable with a specified probability distribution. Raafat (1991) provides two categories of perishability. In the first category, items, such as style goods in fashion merchandising, become simultaneously obsolete at the end of the planning horizon. In the second category, items deteriorate throughout their planning horizon. The second category can be further divided into two classes: items with fixed shelf life such as blood and items with continuous decay and random lifetime such as radioactive material. Deteriorating items are classified

into constant-utility perishable goods, such as prescription drugs, decreasing-utility perishable goods, such as fresh produce, and increasing-utility perishable goods, such as some wines (Raafat, 1991). Amorim *et al.* (2013) provide a review of the literature, and the review suggests that different categories of perishability overlap among each other and are highly tailored for a specific propose. Amorim *et al.* (2013) propose a more comprehensive framework for classifying perishability. Perishability is studied from three dimensions: physical product deterioration, authority limits and customer value (Amorim *et al.*, 2013). Physical product deterioration reflects the actual physical state of an item, authority limits represent external regulations and conventions, and customer value stands for the willingness that a customer has to pay for an item (Amorim *et al.*, 2013).

In manufacturing environment, the dimension of authority limits is usually applied. A fixed threshold, obtained from experiments or domain knowledge, is set to represent residence time constraints. Perishability only refers to the upper bound of residence time, and in practice the lower bound of residence time is often required in a production system as well.

2.2 Study of Residence Time Constraints on Serial Production Line

Serial line, also called transfer line, refers to a category of production lines that are usually applied to mass production (Li and Meerkov, 2009; Gershwin, 1994; Papadopoulos *et al.*, 2019). A serial line consists of a predetermined sequence of machines. Practically, serial lines are highly automatic production lines with low manpower, low flexibility and high production throughput. In a serial line, the uncertainty is primarily from machine reliability, and it can result in a huge production loss. Thus, serial lines under the uncertainty of machine reliability are widely studied during the past decades.

Each machine in a serial line is subject to a reliability model. Bernoulli machine is the simplest reliability model (Li and Meerkov, 2009; Zhang *et al.*, 2013; Ju *et al.*, 2017a). A Bernoulli machine has two states. It can be either up or down with a fixed probability each cycle. It is a reliability model for a machine that has a short downtime compared with cycle time (Meerkov and Zhang, 2008). When the downtime is due to breakdown, it usually takes a longer time to repair. In this case, geometric machines are applicable (Gershwin, 1994; Meerkov *et al.*, 2010; Kang *et al.*, 2017b). Similar to Bernoulli machine, a geometric machine can be up or down. Two probabilities, failure probability and repair probability, are specified to model a machine. Multistage degradation machine reliability model is a natural extension of the geometric machine (Kang and Ju, 2016, 2018). There are several working states and one failure state. Machine state keeps transferring from a good state to a worse state, until the machine fails. Exponential machine is a continuous time reliability model (Li and Meerkov, 2009; Colledani and Gershwin, 2013; Levantesi *et al.*, 2003). In continuous model, the transitions of failure and repair are modeled by two rates, failure rate and repair rate. Brownian motion model, as a more generic reliability model, is introduced in serial line with machine degradation status monitored (Kang *et al.*, 2018, 2019).

Serial lines with residence time constraints are studied. One direction of those studies is to estimate the probability distribution of residence time (Shi and Gershwin, 2012; Angius *et al.*, 2016; Shi and Gershwin, 2016). The study on such a probability distribution helps design buffer capacity to reduce defective rate. However, serial lines in those studies can only identify a defective part at the end of the serial line. It wastes resources to process a defective part, before its defect is identified. Naebulharam and Zhang (2014) study a production system where the defect is able to be soon detected, and the quality buy rate is introduced to evaluate system performance. Another

direction is to take residence time into modeling. Ju *et al.* (2015) evaluate the two-machine Bernoulli line with perishable intermediate products, and Ju *et al.* (2017b) further study the production control of the two-machine Bernoulli line. Kang *et al.* (2017a) and Wang *et al.* (2019) extend the analysis from a Bernoulli machine to a geometric machine, which is a more general reliability model in practice. When residence time is modeled, the state space can become too large to perform analysis. Approximate methods are used to model residence time (Ju *et al.*, 2015, 2017b; Kang *et al.*, 2017a; Wang *et al.*, 2019).

The aggregation method and decomposition method provide frameworks to approximately evaluate multi-stage serial production lines. Gershwin (1987, 1994) proposes the decomposition method. Li and Meerkov (2009) propose the aggregation method to estimate the steady-state performance of multi-stage Bernoulli serial production lines. Zhang *et al.* (2013) extend the aggregation method to perform transient analysis for multi-stage Bernoulli serial production lines. Lee and Li (2017) and Lee *et al.* (2018) study Bernoulli serial production lines with waiting time limits through the aggregation method. Chen *et al.* (2016) apply the aggregation method to geometric serial production lines by defining virtual geometric machines.

2.3 Real-Time Production Control

Performance measures of a serial production line with residence time constraints include production rate and scrap rate. One way to improve the system is to stop several machines from producing each cycle according to real-time system state to reduce scrap rate without sacrificing too much production rate. Thus, a control problem arises and becomes worth studying. Real-time control of two-machine serial lines is studied (Ju *et al.*, 2017b; Wang *et al.*, 2019). After problem approximation,

the optimal control policy is derived. Such a method works well for a small-scale problem but is not directly extendable.

Production control is studied to achieve a desired system performance, and it has been investigated for decades. Due to a lack of access to real-time system state, early practice of production control mainly focuses on simple static system settings (Jaegler *et al.*, 2018) and event-driven/rule-based approaches (Thürer *et al.*, 2019), and those strategies are still widely used (Ju *et al.*, 2016; Cao and Xie, 2015). Supported by the Internet of Things technologies, real-time production control based on real-time system state becomes possible, and it provides potentials to further improve a production system (Lu *et al.*, 2016). Reinforcement learning is a way to perform real-time production control and enhance production performance (Stricker *et al.*, 2018; Waschneck *et al.*, 2018). Through training, reinforcement learning enables a complex production system to find a real-time action that can improve its performance. However, such a way to control production becomes difficult in many cases for two reasons. First, training a reinforcement learning model is computationally expensive. Second, learning methods, such as artificial neural networks, are black box models, and it is difficult to combine domain knowledge into them. Another direction to control a complex production system is through a decentralized way (Wang *et al.*, 2017; Lu and Ju, 2017; Wang *et al.*, 2018a). The introduction of multi-agent system (MAS) and holonic manufacturing system (HMS) attempts to address the production control problem in this way (Leitão, 2009; Barbosa *et al.*, 2015; Giret *et al.*, 2016). The decentralized control aims at achieving flexible control, significantly reducing computational efforts, and improving system performance globally. However, it is difficult to have all three objectives well achieved, and there is a lack of mathematical models supporting decentralized production control.

2.4 Semiconductor Assembly Lines

Semiconductor assembly lines are susceptible to uncertain disruptions, such as machine failures. For production systems that carry out production without using an explicit schedule, such as serial lines, real-time production control, reactive scheduling and online scheduling are common ways to address uncertainty (Qiao *et al.*, 2018; Gupta *et al.*, 2016). The control policy can be created beforehand, and in the run time one may control production by following the predetermined rule. When a production system is complex and flexible, a schedule is often required and one may make a deterministic but robust schedule so that it can deal with uncertainty on its own without intervention (Liu *et al.*, 2007). If real-time intervention has to be given, the original schedule can be totally replaced or partially changed. If a scheduling algorithm has computation time small enough, the same algorithm can also act as a rescheduling method and quickly create a new schedule in response to disruptions (Chung *et al.*, 2014; Park *et al.*, 2019). As an alternative way, one may use rescheduling methods that adjust but maintain the original schedule, including right shift rescheduling, single machine oriented match-up rescheduling, machine group oriented match-up rescheduling, affected operation rescheduling and fix-sequence rescheduling (Qiao *et al.*, 2012, 2018; Abumaizar and Svestka, 1997; Mason *et al.*, 2004). The semiconductor assembly line in this study is complex, and thus a schedule is always required. Long machine failure occurs, so one cannot merely rely on a robust schedule. Considering the long computation time to make a master schedule, total rescheduling is not applicable to this problem. A proper way to control production in response to machine failures in such a semiconductor assembly line has not been fully studied.

Another motivation of fast response to machine failures in semiconductor assembly lines is from residence time constraints. In semiconductor manufacturing, both

fabrication and assembly are restricted by residence time constraints. In wafer fabrication, a wafer is processed at a process module and required to be unloaded within a time limit (Zhu *et al.*, 2014; Pan *et al.*, 2017). Different types of time limits are also observed in other process steps of wafer fabrication (Klemmt and Mönch, 2012; Kim and Lee, 2019). Products in a semiconductor assembly line are restricted by time windows due to the concerns of oxidation and moisture absorption (Han and Kim, 2017). Failures caused by moisture include popcorn cracking (Galloway and Miles, 1996), deformation (Yoon *et al.*, 2008) and adhesion degradation (Tee and Zhong, 2004). However, not much attention has been paid to production control of semiconductor assembly lines considering residence time constraints.

A semiconductor assembly line is often formulated and solved as a flexible job shop problem, which is a job shop problem with parallel machines and thus a NP-hard problem (Mastrolilli and Gambardella, 2000). Practically, a semiconductor assembly line has more requirements, making the scheduling more complex not to mention production control in real time. Process steps of semiconductor assembly commonly include wafer mount, wafer sawing, die attach, wire bond and inspection (Chen and Lo, 2012). Re-entrance is involved, and setup on a machine is required depending on product type and may takes hours (Chung *et al.*, 2014; Huh *et al.*, 2018; Park *et al.*, 2019). The number of setup operators could also be limited (Chung *et al.*, 2014). In addition, internal and external variability, such as unscheduled machine downtime and rush orders, is commonly seen (Chung *et al.*, 2014). Meta-heuristic methods are often used to create production schedule (Lin and Chen, 2015; Hsieh and Cheng, 2018). However, given a schedule, how to respond to disruption is worth more research.

Chapter 3

ANALYSIS AND CONTROL OF TWO-MACHINE SERIAL LINE

3.1 Problem Formulation

The two-machine serial line with residence time constraints under study in this chapter is shown in Figure 3.1. Raw materials or parts enter machine m_1 to be processed and continue to flow into buffer B, where they wait to enter machine m_2 if available. Parts with residence time in the buffer exceeding a certain threshold need to be scrapped, while those who have not met the minimum required residence time cannot go downstream. The following assumptions define the machines, the buffer and their interactions.

(i) The production line consists of two machines (denoted as m_1 and m_2) and a buffer B between them.

(ii) Both machines are synchronized with a constant processing time (cycle time), which is the time to process a part.

(iii) Machines are subject to failures, and their reliability models are independent. The machine reliability models follow geometric distribution. Specifically, if machine m_i is up in cycle $(k-1)$, it will still be up with probability $(1-p_i)$ and down with probability p_i during the kth cycle, for $i = 1, 2$, and $k = 2, 3, \cdots$. If machine m_i is down in cycle $(k-1)$, it will be up with probability r_i and down with probability $(1-r_i)$ during the kth cycle, for $i = 1, 2$, and $k = 2, 3, \cdots$. Here p_i and r_i are defined as the failure and repair probability, respectively, for $i = 1, 2$.

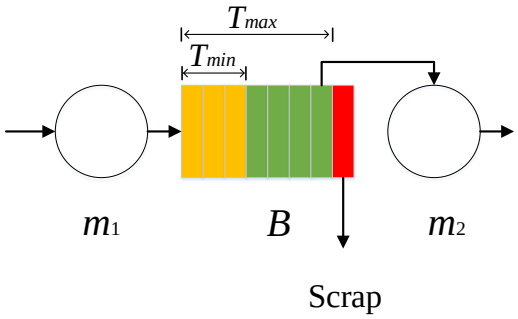


Figure 3.1: Geometric Line with Residence Time Constraints

(iv) Buffer B has finite capacity N ($1 \leq N < \infty$). First-in-first-out (FIFO) policy is assumed regarding the buffer outflow process.

(v) The maximum allowable residence time in buffer B is characterized by T_{max}, counted as the number of cycles. A part will be scrapped directly from the buffer immediately at the beginning of the cycle when its residence time reaches T_{max}. Let $T_{max} \geq N$, otherwise N has no effect on the system.

(vi) The minimum required residence time in buffer B is denoted as T_{min}, counted as the number of cycles. A part is allowed to be produced by machine m_2 only when the residence time of the part reaches or exceeds T_{min}.

(vii) Machine m_1 is blocked during a time slot if at the beginning of the cycle, a) machine m_1 is up, b) buffer B is full, c) machine m_2 does not take a part, and d) there will be no part scrapped from B at the beginning of the next cycle. Machine m_2 is never blocked. In addition, block-before-service policy is assumed.

(viii) Machine m_2 is starved during a time slot if machine m_2 is up, and no part in the buffer B has residence time larger than or equal to T_{min}. Machine m_1 is never starved.

The problem to be studied can be formulated as: *Under assumptions (i)-(viii) shown in this section, develop a method to evaluate the transient behavior of the two-machine geometric serial line with residence time restraints, investigate system properties, and optimize system performance in real time.* The solution to the problem is developed in the next section.

3.2 Modeling

3.2.1 System States and Transition Equations

In order to investigate the transient behavior of a production system under the assumptions (i)-(viii) in this chapter, a Markov chain model is introduced. First of all, the state of the Markov chain model needs to be specified. The buffer occupancy in buffer B, the residence time of each part, and the state (up and down) of machine m_1 and m_2 are required to describe a state of the production system. If one wants to accurately capture the state of the production system, the total number of states M is

$$M = 4 \sum_{i=0}^{N} \left(\begin{array}{c} T_{max} \\ i \end{array} \right). \tag{3.1}$$

The buffer occupancy is in the range of $\{0, 1, \cdots, N\}$. For each part, the residence time is in the range of $\{0, 1, \cdots, T_{max} - 1\}$. If one fixes the buffer occupancy to be $i \in \{0, 1, \cdots, N\}$, the total number of combinations of residence time for i parts in the buffer is $\binom{T_{max}}{i}$. The total number of combinations of residence time for all possible buffer occupancy is $\sum_{i=0}^{N} \binom{T_{max}}{i}$. There are two machines, and each machine can be up or down. Thus, the total number of system states can be calculated by Equation (3.1). The state space grows exponentially when T_{max} and N increase. For instance, if $T_{max} = N$, the number of total states is $M = 2^{N+2}$. Therefore, the

problem soon becomes intractable as T_{max} and N increase. In order to tackle this issue, an approximate model is considered where only the residence time of the first part is considered. To characterize the system dynamics, a discrete time Markov chain model is developed, and the state space for the approximate model is denoted by $\mathcal{S}$. $n \in \{0, 1, \cdots, N\}$ denotes the buffer occupancy. $\tau_1 \in \{0, 1, \cdots, T_{max} - 1\}$ denotes the residence time for which the first part has been staying in the buffer. s_i denotes the state of machine m_i, for $i = 1, 2$. Specifically, $s_i = 1$ means machine m_i is up during a cycle and $s_i = 0$ represents machine m_i is down, for $i = 1, 2$. Besides, when $n = 0$, which means that there is no part in B, the corresponding residence time τ_1 is recorded as 0. Furthermore, when $0 < n \leq N$, the residence time of the head part should be greater than or equal to $(n - 1)$, which means $n - 1 \leq \tau_1 \leq T_{max} - 1$, for $0 < n \leq N$. Thus, a specific system state can be represented as $(n, \tau_1, s_1, s_2) \in \mathcal{S}$. For a two-machine geometric serial line with the buffer capacity N, maximum allowable residence time T_{max} and minimum required residence time T_{min}, all feasible states (n, τ_1, s_1, s_2) at arbitrary time t are collected in Table 3.1. The total number of states M is

$$\begin{aligned} M &= 4\left[\sum_{i=1}^{N}(T_{max} - i + 1) + 1\right] \\ &= 4\left[NT_{max} - \frac{N(N-1)}{2} + 1\right]. \end{aligned} \tag{3.2}$$

The buffer occupancy is in the range of $\{0, 1, \cdots, N\}$. If one fixes the buffer occupancy to be $i \in \{1, \cdots, N\}$, then τ_1 is in the range of $\{i - 1, i, \cdots, T_{max} - 1\}$. There are $(T_{max} - i + 1)$ possible values of τ_1. If the buffer occupancy is 0, the only value of τ_1 is 0. Thus, the total number of combinations is $\left[\sum_{i=1}^{N}(T_{max} - i + 1) + 1\right]$. Considering the state of two machines, the total number of system states for the approximate model is shown in Equation (3.2). Compared to Equation (3.1), the size of the state space for the approximate model has been reduced significantly.

Table 3.1: System States at Arbitrary Time t

		$\tau_1 = 0$		$\tau_1 = 1$		$\cdots$	$\tau_1 = N-1$		$\cdots$	$\tau_1 = T_{max}-1$	
		$s_2 = 0$	$s_2 = 1$	$s_2 = 0$	$s_2 = 1$	$\cdots$	$s_2 = 0$	$s_2 = 1$	$\cdots$	$s_2 = 0$	$s_2 = 1$
$n = 0$	$s_1 = 0$	(0,0,0,0)	(0,0,0,1)	-	-	$\cdots$	-	-	$\cdots$	-	-
	$s_1 = 1$	(0,0,1,0)	(0,0,1,1)	-	-	$\cdots$	-	-	$\cdots$	-	-
$n = 1$	$s_1 = 0$	(1,0,0,0)	(1,0,0,1)	(1,1,0,0)	(1,1,0,1)	$\cdots$	(1,N-1,0,0)	(1,N-1,0,1)	$\cdots$	(1,T_{max}-1,0,0)	(1,T_{max}-1,0,1)
	$s_1 = 1$	(1,0,1,0)	(1,0,1,1)	(1,1,1,0)	(1,1,1,1)	$\cdots$	(1,N-1,1,0)	(1,N-1,1,1)	$\cdots$	(1,T_{max}-1,1,0)	(1,T_{max}-1,1,1)
$n = 2$	$s_1 = 0$	-	-	(2,1,0,0)	(2,1,0,1)	$\cdots$	(2,N-1,0,0)	(2,N-1,0,1)	$\cdots$	(2,T_{max}-1,0,0)	(2,T_{max}-1,0,1)
	$s_1 = 1$	-	-	(2,1,1,0)	(2,1,1,1)	$\cdots$	(2,N-1,1,0)	(2,N-1,1,1)	$\cdots$	(2,T_{max}-1,1,0)	(2,T_{max}-1,1,1)
$n = 3$	$s_1 = 0$	-	-	-	-	$\cdots$	(3,N-1,0,0)	(3,N-1,0,1)	$\cdots$	(3,T_{max}-1,0,0)	(3,T_{max}-1,0,1)
	$s_1 = 1$	-	-	-	-	$\cdots$	(3,N-1,1,0)	(3,N-1,1,1)	$\cdots$	(3,T_{max}-1,1,0)	(3,T_{max}-1,1,1)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n = N$	$s_1 = 0$	-	-	-	-	$\cdots$	(N,N-1,0,0)	(N,N-1,0,1)	$\cdots$	(N,T_{max}-1,0,0)	(N,T_{max}-1,0,1)
	$s_1 = 1$	-	-	-	-	$\cdots$	(N,N-1,1,0)	(N,N-1,1,1)	$\cdots$	(N,T_{max}-1,1,0)	(N,T_{max}-1,1,1)

Define $x(n, \tau_1, s_1, s_2, t)$ as the probability for state (n, τ_1, s_1, s_2) at the beginning of the tth cycle, given that the initial state is known. The transition equations can formulate the state evolution with time. Start with the state (j, i, s_1, s_2) in the $(t+1)$th cycle, where $2 \leq j \leq N-1$, $j \leq i \leq T_{max}-1$, and $s_1, s_2 = 0, 1$. The state means that there are j parts in the buffer, the residence time of the first part in the buffer is i, and the states for both machines are s_1 and s_2, respectively. All possible transitions to state (j, i, s_1, s_2) are illustrated as follows.

(a) $(j-1, i-1, 1, 0)$. In this case, machine m_1 is up, and machine m_2 is down. Buffer B is not full in cycle t, and the residence time of the first part is less than $(T_{max}-1)$. After this cycle, both the buffer occupancy and the residence time of the first part will increase by 1.

(b) $(j-1, i-1, 1, 1)$. In this case, both machines are up. A new part will enter the buffer after this cycle, but no part will leave the buffer. Since machine m_2 is up, this transition occurs when $i - 1 < T_{min}$.

(c) $(j, i-1, 0, 0)$. Both machines are down, and the residence time of the first part is less than $(T_{max} - 1)$. After this cycle, no part will enter or leave the buffer, and the residence time of the first part will increase by 1.

(d) $(j, i-1, 0, 1)$. Machine m_1 is down, and machine m_2 is up. After this cycle, no part will enter or leave the buffer. Since machine m_2 is up, this transition occurs when $i - 1 < T_{min}$.

(e) $(j, k, 1, 1)$, for $\max(i, T_{min}) \leq k \leq T_{max} - 1$. In this case, a new part will enter the buffer, and the first part in the buffer will leave the buffer. In the next cycle, the first part will have residence time i.

(f) $(j, T_{max}-1, 1, 0)$. In this case, a new part will enter the buffer and the first part in the buffer will be scrapped from the buffer. In the next cycle, the new head part will have residence time i.

(g) $(j+1, k, 0, 1)$, for $\max(i, T_{min}) \leq k \leq T_{max}-1$. Machine m_1 is down, and machine m_2 is up. Thus, no part will enter the buffer, and the first part in the buffer will leave the buffer. In the next cycle, the first part will have residence time i.

(h) $(j+1, T_{max}-1, 0, 0)$. Both machines are down, and the residence time of the first part is $(T_{max}-1)$. Thus, no part will enter the buffer, and the first part will be scrapped from the buffer. In the next cycle, the first part will have residence time i.

To get the transition equations to be presented in a concise way, notations $P_{1,1}^{(k)}$, $P_{1,0}^{(k)}$, $P_{0,1}^{(k)}$, and $P_{0,0}^{(k)}$, for $k = 1, 2$, are introduced. Specifically, $P_{i,j}^{(k)}$ denotes the probability that machine m_k that is in state i transitions to state j. They are expressed as follows.

$$\begin{aligned} P_{1,1}^{(k)} &= 1 - p_k, \\ P_{1,0}^{(k)} &= p_k, \\ P_{0,1}^{(k)} &= r_k, \\ P_{0,0}^{(k)} &= 1 - r_k, \quad k = 1, 2. \end{aligned} \tag{3.3}$$

Then the transitions regarding state (j, i, s_1, s_2) in cycle $(t+1)$ can be obtained using the following equation

$$
\begin{aligned}
x(j,i,s_1,s_2,t+1) =& x(j-1,i-1,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2} \\
&+ x(j-1,i-1,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&+ x(j,i-1,0,0,t)P^{(1)}_{0,s_1}P^{(2)}_{0,s_2} \\
&+ x(j,i-1,0,1,t)P^{(1)}_{0,s_1}P^{(2)}_{1,s_2}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&+ \sum_{k=\max(i,T_{min})}^{T_{max}-1} x(j,k,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2}\Phi(j,k,i-1) \\
&+ x(j,T_{max}-1,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2}\Phi(j,T_{max}-1,i-1) \\
&+ \sum_{k=\max(i,T_{min})}^{T_{max}-1} x(j+1,k,0,1,t)P^{(1)}_{0,s_1}P^{(2)}_{1,s_2}\Phi(j+1,k,i-1) \\
&+ x(j+1,T_{max}-1,0,0,t)P^{(1)}_{0,s_1}P^{(2)}_{0,s_2}\cdot\Phi(j+1,T_{max}-1,i-1),
\end{aligned}
\tag{3.4}
$$

where $2 \leq j \leq N-1$, $j \leq i \leq T_{max}-1$, and $s_1, s_2 = 0, 1$. $\mathbf{1}_{\mathbb{N}^+}(x)$ is an indicator function. $\mathbf{1}_{\mathbb{N}^+}(x) = 1$ if x is an positive integer ($x \in \mathbb{N}^+$), otherwise $\mathbf{1}_{\mathbb{N}^+}(x) = 0$. When the first part in the buffer is scrapped or produced by machine m_2, one needs to identify the residence time of the second part in buffer B. Since only the residence time of the first part in the buffer is recorded, it is impossible to exactly determine how long the second part has stayed. One can use an operator $\Phi(n, \tau_1, \tau_2)$, proposed by Kang *et al.* (2017a), to estimate the conditional probability that the second part in buffer B has residence time τ_2 given that there are n parts in the buffer and the first one has stayed for τ_1 cycles.

The transitions for the rest of the states can be obtained in a similar way, shown in Appendix A. Using these equations, one can derive the probability distribution of states at any time and conduct transient analysis on the system performance.

3.2.2 State Transition Matrix

To analyze the system performance, the state transition matrix by transforming the four-dimension state space into a one-dimension vector is constructed. Specifically, (n, τ_1, s_1, s_2) is ranked based on buffer occupancy n first and then the residence time τ_1 for the first part in the buffer, followed by the state of machine m_1 and m_2 (s_1 and s_2) in the third and fourth. A sorting method is utilized by introducing an order operator $I(n, \tau_1, s_1, s_2)$ which is defined as

$$I(n,\tau_1,s_1,s_2) = \begin{cases} 2s_1 + s_2 + 1, & \text{if } n = 0, \\ 4\left[\dfrac{1}{2}(n-1)(2T_{max}+2-n) \right. & \\ \left. +\tau_1 - n + 2\right] + 2s_1 + s_2 + 1, & \text{if } n > 0. \end{cases} \tag{3.5}$$

Denote by $X(t)$ an M-dimension row vector for system state probability at time t. $X(t)$ is be expressed as $X(t) = [x_1(t), x_2(t), \cdots, x_{I(n,\tau_1,s_1,s_2)}(t), \cdots, x_M(t)]$. In addition, $x_{I(n,\tau_1,s_1,s_2)}(t) = x(n, \tau_1, s_1, s_2, t)$. Furthermore, define Q as the $M \times M$ state transition matrix of the Markov chain described in the transition equations above. Specially,

$$Q = [Q_{ij}]_{M\times M}\,, \tag{3.6}$$

where Q_{ij} represents the probability that the system is in state $(n', \tau_1', s_1', s_2')$ given that the state was $(n'', \tau_1'', s_1'', s_2'')$ in the previous cycle, where $i = I(n'', \tau_1'', s_1'', s_2'')$ and $j = I(n', \tau_1', s_1', s_2')$. Using the state sorting method and matrix notation, the state evolution could be formulated as

$$X(t+1) = X(t)Q,\ t = 1, 2, \cdots\,. \tag{3.7}$$

3.3 System Performance and Properties

3.3.1 Performance Measures

To investigate the system dynamics, the performance measures are defined as below.

- *Production Rate,* $PR(t)$: the expected number of parts produced by machine m_2 in the tth cycle;
- *Consumption Rate,* $CR(t)$: the expected number of parts consumed by machine m_1 in the tth cycle;
- *Scrap Rate,* $SR(t)$: the expected number of scrapped parts in the tth cycle;
- *Work-In-Process,* $WIP(t)$: the expected buffer occupancy in buffer B at the beginning of the tth cycle.

As the state and transition matrix are defined, the above performance measures can be evaluated theoretically. Given a production system defined by assumptions (i)-(viii), the performance measures can be estimated in the following way:

$$\widehat{PR}(t) = \sum_{n=1}^{N} \sum_{\tau_1=\max(n-1,T_{min})}^{T_{max}-1} \sum_{s_1=0}^{1} x_{I(n,\tau_1,s_1,1)}(t), \tag{3.8}$$

$$\begin{aligned}\widehat{CR}(t) = &\sum_{s_2=0}^{1} x_{I(0,0,1,s_2)}(t) + \sum_{n=0}^{N-1} \sum_{\tau_1=n-1}^{T_{max}-1} \sum_{s_2=0}^{1} x_{I(n,\tau_1,1,s_2)}(t) \\ &+ \sum_{\tau_1=\max(N-1,T_{min})}^{T_{max}-1} x_{I(N,\tau_1,1,1)}(t) + x_{I(N,T_{max}-1,1,0)}(t),\end{aligned} \tag{3.9}$$

$$\widehat{SR}(t) = \sum_{n=1}^{N} \sum_{s_1=0}^{1} x_{I(n,T_{max}-1,s_1,0)}(t), \tag{3.10}$$

$$\widehat{WIP}(t) = \sum_{n=1}^{N} \sum_{\tau_1=n-1}^{T_{max}-1} \sum_{s_1=0}^{1} \sum_{s_2=0}^{1} n x_{I(n,\tau_1,s_1,s_2)}(t), \tag{3.11}$$

where $\widehat{PR}(t)$, $\widehat{CR}(t)$, $\widehat{SR}(t)$, and $\widehat{WIP}(t)$ are the estimates from the analytical method for $PR(t)$, $CR(t)$, $SR(t)$, and $WIP(t)$, respectively. The above equations provide analytical formulas to evaluate system performance measures. Specifically, $\widehat{PR}(t)$ is the probability that machine m_2 is up, the buffer is not empty during the tth cycle and the residence time of the first part in the buffer has reached or exceeded T_{min}. $\widehat{CR}(t)$ is the probability that machine m_1 is up and there is at least one spot available in the buffer. Similarly, $\widehat{SR}(t)$ can be calculated by using the instances that machine m_2 is down and the first part in the buffer has residence time $(T_{max} - 1)$, representing that the first part is about to be scrapped. Finally, $\widehat{WIP}(t)$ estimates the expected buffer occupancy in the tth cycle.

3.3.2 Accuracy Evaluation

To evaluate the accuracy of the proposed method, the analytical results are compared with the results obtained from simulation experiments. A MATLAB program is constructed to perform the simulation and compare it with the analytical method. Parameter settings are generated randomly from a predefined range. For each parameter setting, $10,000$ replications are carried out for simulation. The experiment for each setting follows the procedures below.

1) A parameter setting is randomly generated from the following parameter set:

$$\begin{aligned} p_1, p_2 &\in [0.4, 0.8]\,, \\ r_1, r_2 &\in [0.3, 0.8]\,, \\ N &\in \{3, 4, 5, 6, 7\}\,, \\ T_{max} &\in \{N+1, N+2, N+3\}\,, \\ T_{min} &\in \{1, 2\}\,. \end{aligned} \tag{3.12}$$

Residence time constraints are usually concerned about in practice when the first machine has higher efficiency than the second machine. Thus, we let the setting satisfy $p_1 < p_2$ and $r_1 > r_2$.

2) Set the buffer to be empty initially, and set the initial state of both machines to be up.

3) Run the simulation for a total of $T = 200$ time slots.

4) $PR(t)$, $CR(t)$, $SR(t)$ and $WIP(t)$ are unknown. The performance measures estimated through the simulation are unbiased, so the performance measures obtained from the simulation are used to represent $PR(t)$, $CR(t)$, $SR(t)$ and $WIP(t)$.

5) Take the average of performance measures of the last 100 time slots as the simulated performance in the steady state. The steady state performance measures $PR(\infty)$, $CR(\infty)$, $SR(\infty)$ and $WIP(\infty)$ are estimated through the simulation as follows:

$$PR(\infty) = \frac{1}{100} \sum_{t=T-99}^{T} PR(t), \tag{3.13}$$

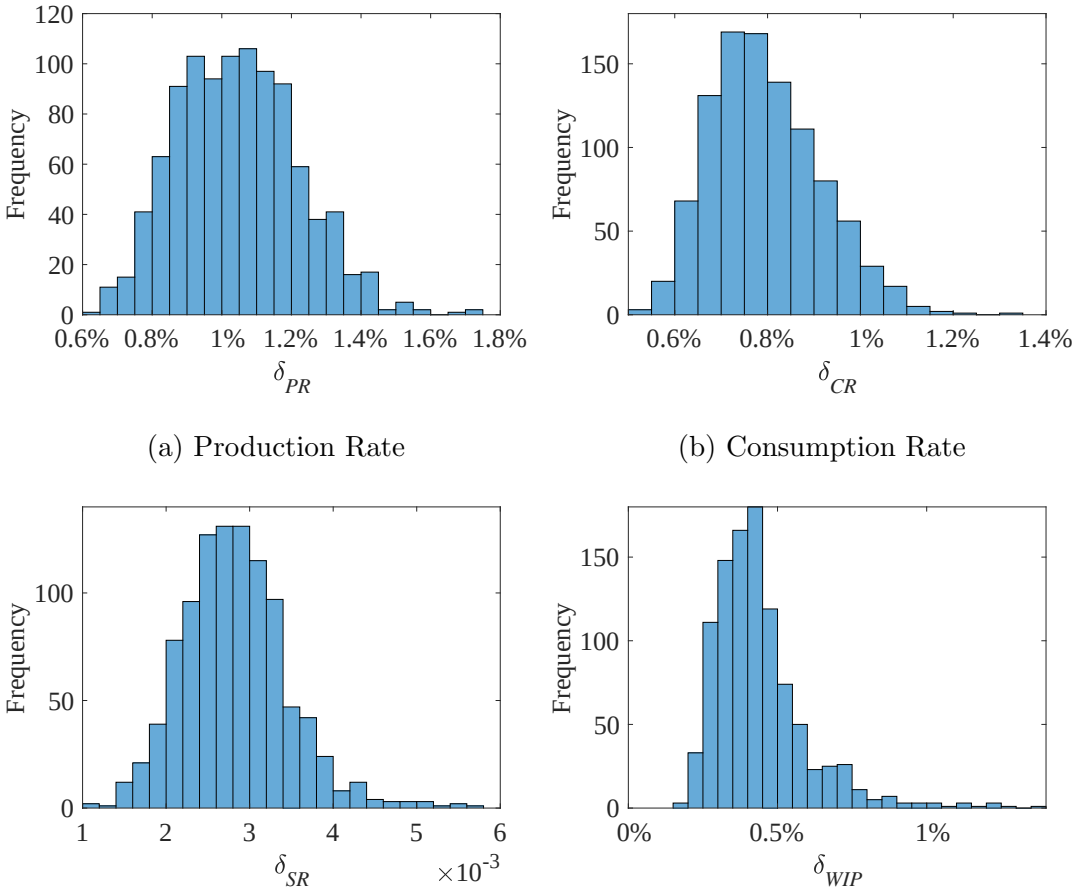
$$CR(\infty) = \frac{1}{100} \sum_{t=T-99}^{T} CR(t), \tag{3.14}$$

$$SR(\infty) = \frac{1}{100} \sum_{t=T-99}^{T} SR(t), \tag{3.15}$$

$$WIP(\infty) = \frac{1}{100} \sum_{t=T-99}^{T} WIP(t). \tag{3.16}$$

The following error metrics are introduced to evaluate the accuracy:

$$\delta_{PR} = \frac{1}{T} \sum_{t=1}^{T} \frac{\left| PR(t) - \widehat{PR}(t) \right|}{PR(\infty)} \times 100\%, \tag{3.17}$$



(a) Production Rate

(b) Consumption Rate

(c) Scrap Rate

(d) Work-In-Progress

Figure 3.2: Accuracy of Performance Measures Estimated by the Analytical Method

$$\delta_{CR} = \frac{1}{T}\sum_{t=1}^{T}\frac{\left|CR(t) - \widehat{CR}(t)\right|}{CR(\infty)} \times 100\%, \tag{3.18}$$

$$\delta_{SR} = \frac{1}{T}\sum_{t=1}^{T}\left|SR(t) - \widehat{SR}(t)\right|, \tag{3.19}$$

$$\delta_{WIP} = \frac{1}{T}\sum_{t=1}^{T}\frac{\left|WIP(t) - \widehat{WIP}(t)\right|}{WIP(\infty)} \times 100\%. \tag{3.20}$$

It is worth mentioning that since $SR(\infty)$ is typically very small (less than 0.1), the absolute error is used to evaluate the accuracy of $\widehat{SR}(t)$.

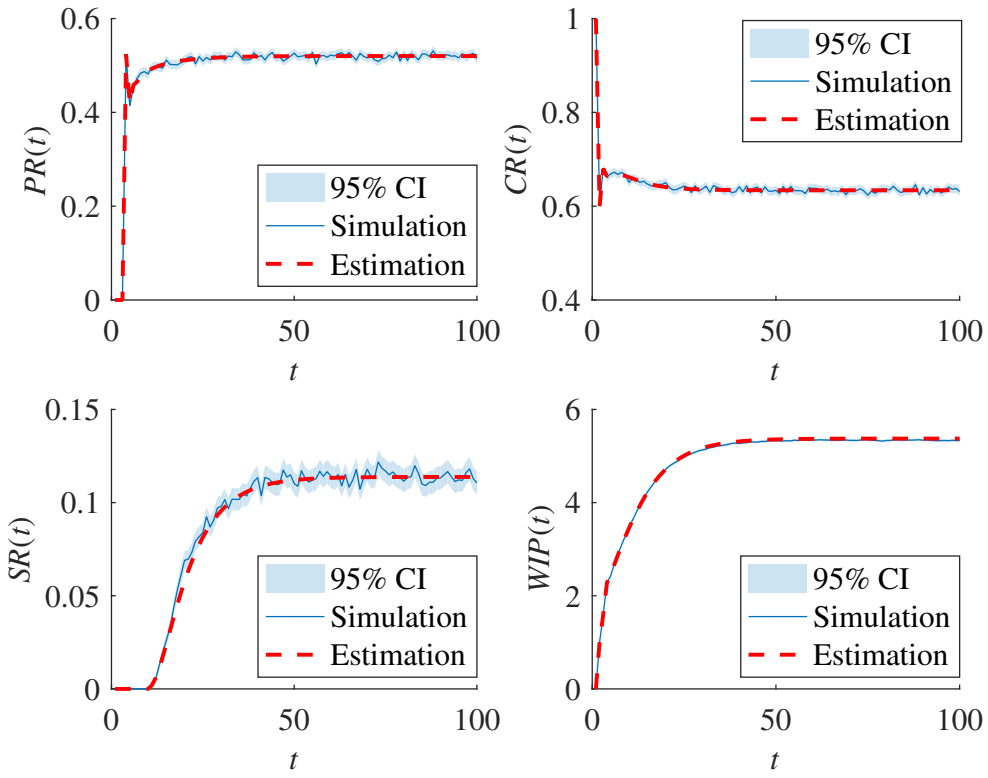


Figure 3.3: Comparison of the Performance Measures between the Simulation and Estimation

1,000 randomly selected settings are tested using both the analytical model and simulation. The resulting accuracy is summarized in Figure 3.2. As one can see from the figure, the performance measures obtained from the analytical method have small errors. Specifically, the mean values of δ_{PR}, δ_{CR}, δ_{SR} and δ_{WIP} are 1.05%, 0.8%, 0.0028 and 0.44%, respectively.

To illustrate how the analytical model captures the system behavior in both the transient and steady state, the setting below is selected and the performance measures obtained from the analytical method are compared with the simulation.

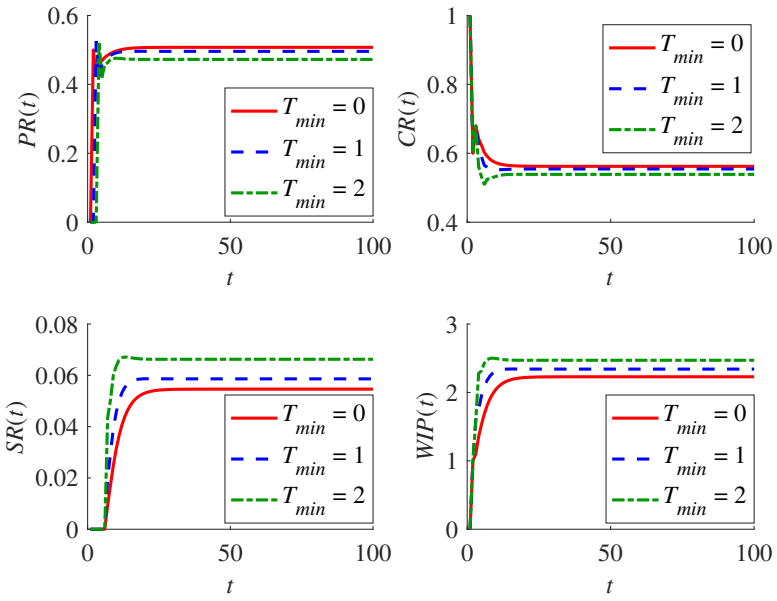
$$p_1 = 0.4, p_2 = 0.5, r_1 = 0.8, r_2 = 0.55, N = 7, T_{max} = 10, T_{min} = 2. \qquad (3.21)$$

The result is shown in Figure 3.3. Simulated performance measures are plotted in solid lines, with the shaded area indicating the 95% confidence interval. The dashed lines represent estimation using the proposed analytical model. As one can see, estimation and simulation results are close in the whole period. Specifically, during the transient period, the estimated performance measures can clearly capture the system dynamics, which ensures that the estimation can provide high accuracy for transient analysis in two-machine geometric lines with residence time constraints.

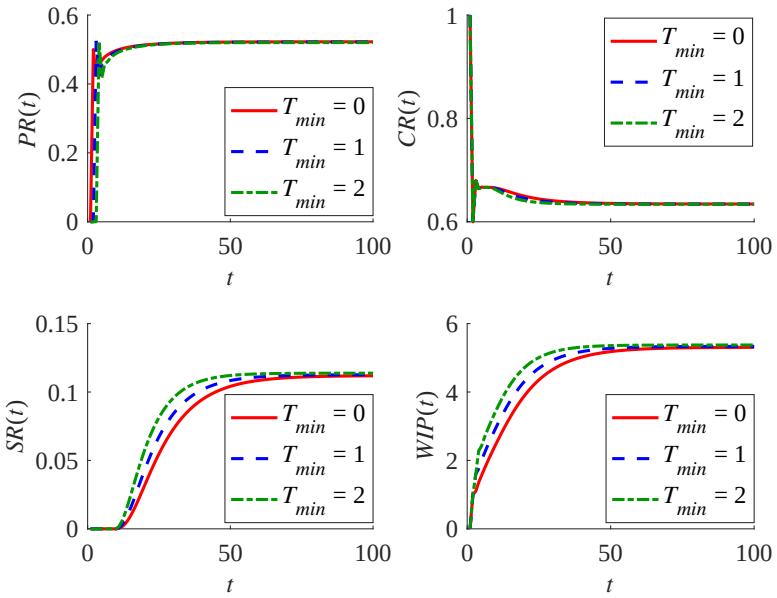
3.3.3 Effect of Residence Time Constraints on System Performance

When residence time constraints are applied to a production system, the performance of the production system changes. A production system without residence time constraint can be viewed as a special case of production systems with residence time constraints $T_{min} = 0$ and $T_{max} = \infty$. As T_{min} increases and T_{max} decreases, the performance is influenced gradually. Some performance measures may be sensitive to residence time constraints, while some others may not. There is no closed-form equation to express performance measures, so it is not obvious to see how residence time constraints influence the performance measures. In this subsection, numerical experiments are conducted to study how T_{min} and T_{max} influence production rate, consumption rate, scrap rate, and work-in-process.

Here starts the analysis of residence time constraints from T_{min}. Two settings are selected based on the parameter setting in Equation (3.21). In the first setting, set $N = 3$ and $T_{max} = 6$ to create a case with small N and small T_{max}. In the second setting, set $N = 7$ and $T_{max} = 10$ to create a case with large N and large T_{max}. To test the influence of T_{min} on performance measures, let T_{min} be equal to 0, 1, and 2, and use the analytical model to evaluate the performance measures. The performance measures of the setting with small N and small T_{max} are shown in Figure 3.4a, while



(a) $N = 3$, $T_{max} = 6$



(b) $N = 7$, $T_{max} = 10$

Figure 3.4: Effect of T_{min} on System Performance

the performance measures of the setting with large N and large T_{max} are shown in Figure 3.4b. It suggests that neither production rate nor consumption rate is sensitive to T_{min}, especially when N and T_{max} are large. The scrap rate does not have a large change due to the conservation of flow, but this small change looks obvious when N and T_{max} are small. The reason is that the scrap rate is relatively small. The work-in-process is easily influenced by T_{min} when N and T_{max} are small. When N and T_{max} become large, scrap rate and work-in-process have obvious difference for different T_{min} in transient stage, but converge to some values close with each other in steady state. The system starts with an empty buffer occupancy, and a different T_{min} causes a different production rate and work-in-process in transient state, which then cause a different scrap rate in transient state. Machine m_1 has a larger repair probability and smaller failure probability than machine m_2. Thus, buffer occupancy is kept in a high level in the long term when N and T_{max} are large. It results in the overlap of performance measures in the steady state with different T_{min}, which is shown in Figure 3.4b.

To compare the impact that T_{min} has on performance measures in different settings, parameter settings are randomly generated from the parameter set given in Equation (3.12). For each setting, T_{min} is set to be 2 and 0, respectively. Use the performance measures for $T_{min} = 2$ minus the performance measures for $T_{min} = 0$, and plot the difference and relative difference for production rate, consumption rate, scrap rate and work-in-process in Figure 3.5. It suggests that a larger T_{min} results in a smaller production rate, a smaller consumption rate, a larger scrap rate, and a larger work-in-process. It also suggests that the scrap rate and work-in-process are sensitive to T_{min}. The difference of scrap rate is small, but the relative difference of the scrap rate is large due the small denominator. Significant difference of production

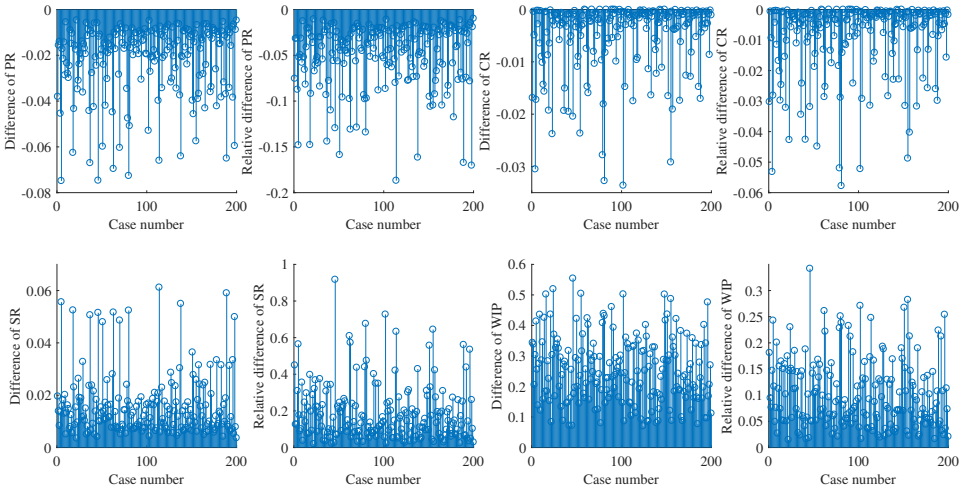
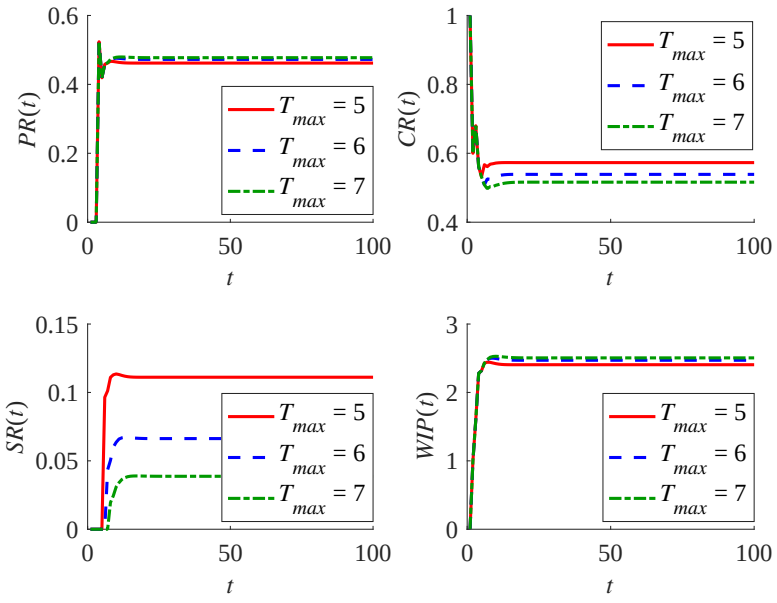


Figure 3.5: Difference and Relative Difference of Performance Measures with $T_{min} = 2$ and $T_{min} = 0$

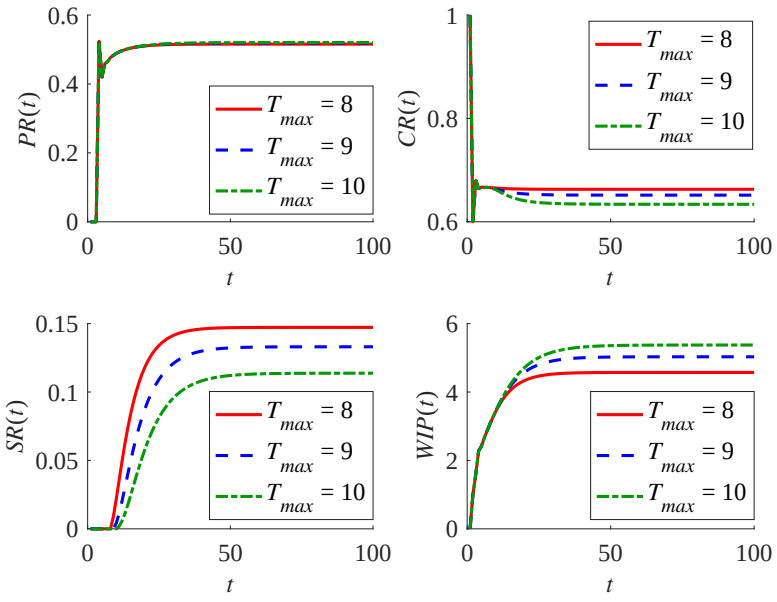
rate can be observed, but there is no significant difference of consumption rate for most cases.

The analysis is performed on T_{max} in a similar way. Two settings are selected based on the parameter setting in Equation (3.21). In the first setting, set $N = 3$ and $T_{max} = 5, 6, 7$ to create a case with small N and small T_{max}. In the second setting, set $N = 7$ and $T_{max} = 8, 9, 10$ to create a case with large N and large T_{max}. The performance measures of the setting with small N and small T_{max} are shown in Figure 3.6a, while the performance measures of the setting with large N and large T_{max} are shown in Figure 3.6b.It suggests that the production rate is not sensitive to T_{max}, while the consumption rate and scrap rate are sensitive to T_{max}. The work-in-process is sensitive to T_{max} in steady state when N and T_{max} are large.

Parameter settings are randomly generated from the parameter set given in Equation (3.12). For each setting, T_{max} is set to be 10 and 8, respectively. Use the performance measures for $T_{max} = 10$ minus the performance measures for $T_{max} = 8$, and



(a) $N = 3$



(b) $N = 7$

Figure 3.6: Effect of T_{max} on System Performance

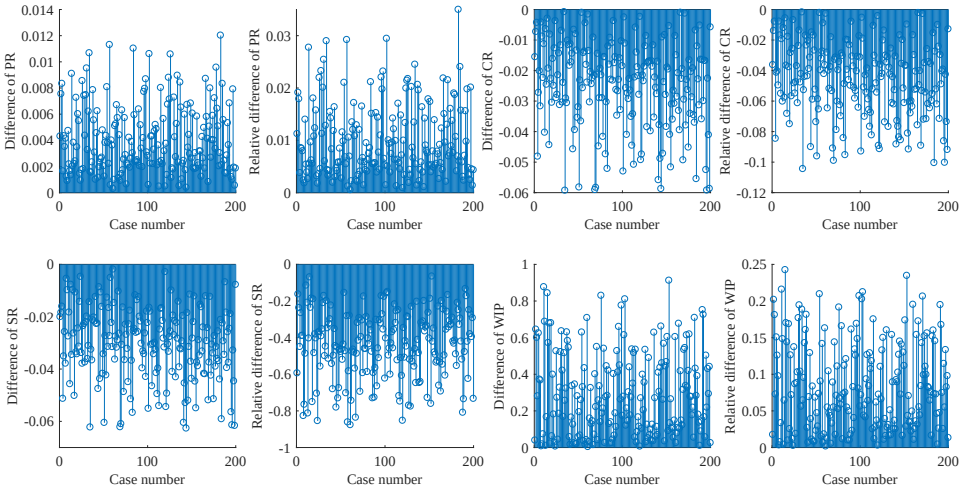


Figure 3.7: Difference and Relative Difference of Performance Measures with $T_{max} = 10$ and $T_{max} = 8$

plot the difference and relative difference for performance measures in Figure 3.7. A larger T_{max} leads to a larger production rate, a smaller consumption rate, a smaller scrap rate, and a larger work-in-process. The relative difference of the scrap rate is large, and it reaches 40% for most cases. The relative difference of the work-in-process varies a lot. It is as small as 0 in some cases, but it also reaches 25% in some other cases. Significant difference of consumption rate can be observed, but there is no significant difference of production rate for most cases.

3.3.4 Effect of Buffer Capacity on System Performance

Buffer capacity is a parameter that can be easily modified by the designer of a production system. Buffer is aimed at providing smooth production under uncertainties. A proper buffer capacity makes facilities in a production system be fully utilized. However, unnecessary buffer capacity can cause a production system not to be lean, which can further cause a series of problems. Thus, the design of lean

production lines is important. Monotonicity properties of $PR(t)$, $SR(t)$ and $WIP(t)$ with respect to N are discussed in this subsection, and a case study is provided to illustrate the effect of buffer capacity on system performance.

Consider two two-machine geometric serial lines defined by assumptions (i)-(viii), and denote them as system ρ and system ζ, respectively. To complete the proof for monotonicity properties, a new way is used to represent the state of the two-machine geometric serial line. For system ρ in cycle $t \in \mathbb{N}^+$, the state is defined as $X_t^\rho = (\varphi^\rho, s_1, s_2)$, where the set $\varphi^\rho = \{\varphi_1, \varphi_2, \cdots\}$ represents arrival time to the buffer for all the parts waiting in the buffer. Let $\varphi^\rho_{[1]}, \varphi^\rho_{[2]}, \cdots$ be a sequence of the components in set φ^ρ in increasing order, and it satisfies that $\varphi^\rho_{[i]} < \varphi^\rho_{[j]}$ for all i and j that $i < j$ and $\varphi^\rho_{[i]}, \varphi^\rho_{[j]} \in \varphi^\rho$. For $i = 1, 2$, $s_i = 1$ means machine m_i is up during cycle t, and $s_i = 0$ represents down. Similarly, the definition and notation are applied to system ζ. Define $\phi = (\phi_1^1, \phi_1^2, \phi_2^1, \phi_2^2, \cdots, \phi_t^1, \phi_t^2, \cdots)$ to be a sample path for system ρ and system ζ. For $t = 1, 2, \cdots$, and $i = 1, 2$, $\phi_t^i \in \{0, 1\}$ represents the state of machine m_i in cycle t. Sample path ϕ is a realization of the stochastic process.

Lemma 1. *System ρ and system ζ have the same parameters except N. Denote the buffer capacities for system ρ and system ζ as N^ρ and N^ζ, respectively. Assume that both systems have the same initial state and sample path. $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ for any cycle t, if $N^\rho + 1 = N^\zeta$.*

Proof. See Appendix B. □

Theorem 1. *$WIP(t)$ is monotonically increasing in N.*

Proof. From Lemma 1, $\varphi^\rho \subseteq \varphi^\zeta$ at any time, so $|\varphi^\rho| \leq \left|\varphi^\zeta\right|$ at any time. $WIP(t)$ is monotonically increasing in N. □

Theorem 2. *$PR(t)$ is monotonically increasing in N.*

Proof. From Lemma 1, $\varphi^{\rho} \subseteq \varphi^{\zeta}$ at any time. Assume that production on the machine m_2 happens in system ρ at the end of cycle t. From the assumption, $t - \varphi^{\rho}_{[1]} \geq T_{min}$ and $\phi^2_t = 1$ in cycle t. Since $\varphi^{\rho} \subseteq \varphi^{\zeta}$ in cycle t, $\varphi^{\rho}_{[1]} \in \varphi^{\zeta}$ and $\varphi^{\zeta}_{[1]} \leq \varphi^{\rho}_{[1]} \leq t - T_{min}$. Thus, $t - \varphi^{\zeta}_{[1]} \geq T_{min}$ in cycle t, and production also happens in system ζ at the end of cycle t. Therefore, $PR(t)$ is monotonically increasing in N. □

Theorem 3. *$SR(t)$ is monotonically increasing in N.*

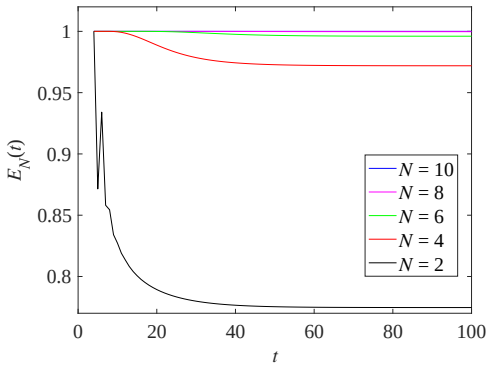
Proof. From Lemma 1, $\varphi^{\rho} \subseteq \varphi^{\zeta}$ at any time. Assume that the scrap happens in system ρ at the end of cycle t. From the assumption, $t - \varphi^{\rho}_{[1]} = T_{max} - 1$ and $\phi^2_t = 0$ in cycle t. Since $\varphi^{\rho} \subseteq \varphi^{\zeta}$, then $\varphi^{\rho}_{[1]} \in \varphi^{\zeta}$ and the scrap also happens in system ζ at the end of cycle t. Therefore, $SR(t)$ is monotonically increasing in N. □

It has been proved that production rate, work-in-process, and scrap rate are monotonically increasing in N. A large production rate is preferred, but one may want to prevent work-in-process and scrap rate from being large. Thus, the design of lean production lines is aimed at minimizing work-in-process and scrap rate by selecting a buffer capacity under which the production rate can meet the customer demand. A case study is provided to give insights into the effect of buffer capacity on system performance, and it is helpful for the designer to pick a proper buffer capacity for a lean production line.

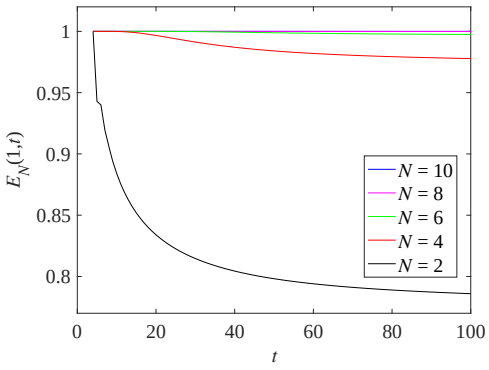
The range of buffer capacity that can be selected is $\{1, \cdots, T_{max}\}$. When buffer capacity is set to be 1, the production system is a Just-in-Time (JIT) system. Machine m_1 is block after producing one part, until this part is consumed by machine m_2 or scrapped. The production rate, work-in-process and scrap rate are the smallest in this setting. Given a finite T_{max} and infinite N, the largest buffer occupancy that can occur is equal to T_{max}. Thus, the largest buffer capacity is equal to T_{max}, and it leads to the largest production rate, work-in-process and scrap rate. Define the line

efficiency $E_N(t)$ to be the ratio of the production rate in cycle t for buffer capacity N to the production rate in cycle t for the largest buffer capacity given the initial state. The line efficiency in steady state can be represented as $E_N(\infty)$. The line efficiency $E_N(t)$ does not exist in cycle t if $PR(t)$ for the largest buffer capacity is equal to 0 in cycle t. The accumulative line efficiency $E_N(t, t')$ from cycle t to t' is defined to be the radio of accumulative production rate from t to t', $\sum_{i=t}^{t'} PR(i)$, for the buffer capacity N to the accumulative production rate for the largest buffer capacity.

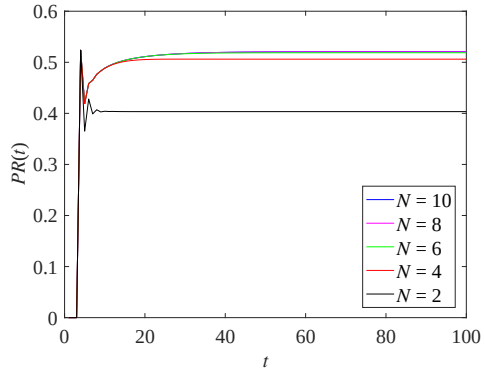
A case study is given to illustrate the effect of buffer capacity on system performance. The parameter setting for the case study is the same as the parameter setting in Equation (3.21) except buffer capacity N. The range of buffer capacity that can be selected is $\{1, \cdots, 10\}$. Select $N = 2$, $N = 4$, $N = 6$, $N = 8$, and $N = 10$ as the buffer capacity. The buffer is empty and both machines are up in the first cycle. Use the analytical model to obtain the system performance from the first cycle to the 100th cycle, shown in Figure 3.8. Figure 3.8a and Figure 3.8b shows that the line efficiency and the accumulative line efficiency are close to 1 when buffer capacity is selected as $N = 6$, $N = 8$, or $N = 10$. When the buffer capacity $N = 4$, the line efficiency and the accumulative line efficiency in the 100th cycle are 97.19% and 97.78%, respectively. It means that the production loss within first 100 cycles is only 2.22% due to the reduction of 6 units of buffer capacity. When the customer demand is small and estimated to be only 75% of the maximum production capacity, the buffer capacity $N = 2$ is sufficient to satisfy the customer demand. When Figure 3.8c is compared with Figure 3.8d and Figure 3.8e, it is observed that the production rate is not sensitive to N but scrap rate and work-in-process are sensitive to N. When the buffer capacity is set to be $N = 4$, the 2.22% production loss results in 86.02% reduction of scrap rate and 43.05% reduction of work-in-process. If the customer demand is as small as 75% of the maximum production capacity and the buffer capacity is set



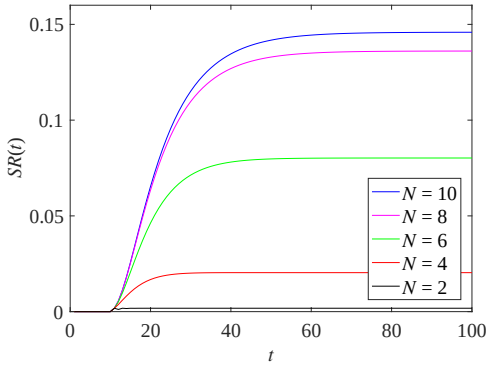
(a) Line Efficiency



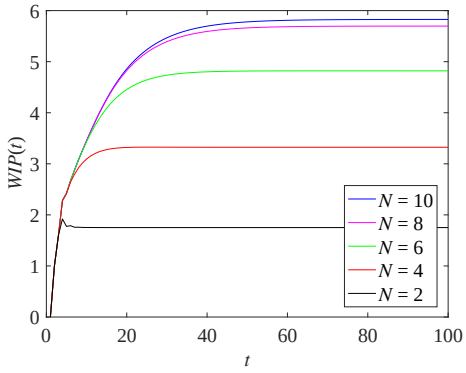
(b) Accumulative Line Efficiency



(c) Production Rate



(d) Scrap Rate



(e) Work-In-Process

Figure 3.8: Effect of Buffer Capacity on System Performance

to be $N = 2$, the scrap rate can be reduced by 98.77% and the work-in-process can be reduced by 69.98%.

The analysis of the effect of buffer capacity on system performance supports the design of lean production lines, which are beneficial to production systems with residence time constraints. By setting a proper buffer capacity based on the customer demand, a lean production line can be designed and scrap rate and work-in-process can be significantly reduced.

3.4 Real-Time Production Control

The analytical method has been applied to capture the dynamics of the two-machine geometric serial line. To achieve better performance, the control based on real-time production information can be implemented. In this section, a control policy to optimize the system performance is proposed based on manipulation of m_1. Specifically, a control policy actively turns the state of m_1 down to prevent more parts from entering the buffer and suffering long residence time. With such a control policy, the residence time for parts in the buffer, as well as the scrap rate due to long residence time, can be reduced. The rest of this section provides the detail about how the initial control policy can be obtained from an MDP model and how the control policy could be improved by Bayesian methods and run time data.

3.4.1 Modeling of Discounted Markov Decision Processes

First of all, the optimization objective should be specified. In practice, it is desired to reduce the scrap rate and increase the production rate, which means to maximize $PR(t) - \omega SR(t)$. ω $(0 < \omega < 1)$ is the weight, and it is determined by the cost of parts and the price of products. To find the trade-off of production rate and scrap rate,

an MDP model is built to analyze the control problem of the system. The relative parameters of this MDP problem are listed as follows.

- Decision epochs: $t \in \mathbb{N}^+$.
- System states at each decision epoch: $(n, \tau_1, s_1, s_2) \in \mathcal{S}$.
- Actions at each decision epoch:

$$A_{(n,\tau_1,s_1,s_2)} = \{1, 0\}, \forall (n, \tau_1, s_1, s_2) \in \mathcal{S}, \tag{3.22}$$

  which means in an arbitrary cycle t when the system state is (n, τ_1, s_1, s_2), the action $a(n, \tau_1, s_1, s_2)$ satisfies $a(n, \tau_1, s_1, s_2) \in A_{(n,\tau_1,s_1,s_2)}$. $a(n, \tau_1, s_1, s_2) = 1$ represents a control that does not manually intervene the reliability model of m_1, and $a(n, \tau_1, s_1, s_2) = 0$ means a control that turns m_1 down intentionally at the end of the current cycle.

- Reward function: Denote the reward function by $r(n, \tau_1, s_1, s_2)$ for $(n, \tau_1, s_1, s_2) \in \mathcal{S}$. Specifically,

$$r(n, \tau_1, s_1, s_2) = PR(t) - \omega SR(t), \tag{3.23}$$

  for any t such that the system in state (n, τ_1, s_1, s_2).

- Transition probability matrix:

$$Q_{a(n,\tau_1,s_1,s_2)} = \begin{cases} Q, & \text{if } a(n, \tau_1, s_1, s_2) = 1, \\ Q(p_1 = 1, r_1 = 0), & \text{if } a(n, \tau_1, s_1, s_2) = 0, \end{cases} \tag{3.24}$$

  where $Q(p_1 = 1, r_1 = 0)$ is obtained by setting p_1 as 1 and r_1 as 0 in the original matrix Q presented in Equation (3.6), representing that m_1 is turned down with probability 100% after the current cycle.

- The expected total discounted reward of policy π:

$$v_\lambda^\pi(y_1) = E_{y_1}^\pi \left\{ \sum_{t=1}^{\infty} \lambda^{t-1} r(y_t) \right\}. \tag{3.25}$$

  where $\lambda \in [0, 1)$ is the discount, $y_1 \in \mathcal{S}$ represents the initial state of the system, and $y_t \in \mathcal{S}$ represents the state in cycle t.

The optimal expected discounted reward and optimal policy are given as follows.

$$v_\lambda^*(y_1) = v_\lambda^{\pi^*}(y_1) = \max_\pi E_{y_1}^\pi \left\{ \sum_{t=1}^{\infty} \lambda^{t-1} r(y_t) \right\}, \tag{3.26}$$

$$\pi^* \in \arg\max_\pi E_{y_1}^\pi \left\{ \sum_{t=1}^{\infty} \lambda^{t-1} r(y_t) \right\}. \tag{3.27}$$

The MDP model for two-machine geometric serial lines has been constructed. The control policy can be obtained by implementing the value iteration algorithm to the MDP model. The value iteration algorithm is an iterative algorithm that is guaranteed to converge to the optimal solution. To briefly illustrate the algorithm, $v^k(n, \tau_1, s_1, s_2)$, or $v^k(i)$ where $i = (n, \tau_1, s_1, s_2)$, is introduced as the total discounted reward with the initial state (n, τ_1, s_1, s_2) estimated in the kth iteration of the value iteration algorithm, for $k = 0, 1, 2, \cdots$. Set the initial total discounted reward $v^0(i) = 0$ for all $i \in \mathcal{S}$. In the $(k+1)$th iteration,

$$v^{k+1}(i) = \max_{a(i) \in A_i} \left\{ r(i) + \sum_{j \in \mathcal{S}} \lambda P(j \mid i, a(i)) v^k(j) \right\}, \tag{3.28}$$

for all $i \in \mathcal{S}$. $P(j \mid i, a(i))$ denotes the transition probability from state i to state j given the action $a(i)$, and the transition probability $P(j \mid i, a(i))$ can be obtained from the transition matrix presented in Equation (3.24). For any $i \in \mathcal{S}$, $v^{k+1}(i)$ increases monotonically in k and converges to $v_\lambda^*(i)$ (Bertsekas, 2012). With $v_\lambda^*(i)$ known for all $i \in \mathcal{S}$, the optimal action in state i, $a^*(i)$, is obtained as follows.

$$a^*(i) \in \arg\max_{a(i) \in A_i} \left\{ r(i) + \sum_{j \in \mathcal{S}} \lambda P(j \mid i, a(i)) v_\lambda^*(j) \right\}. \tag{3.29}$$

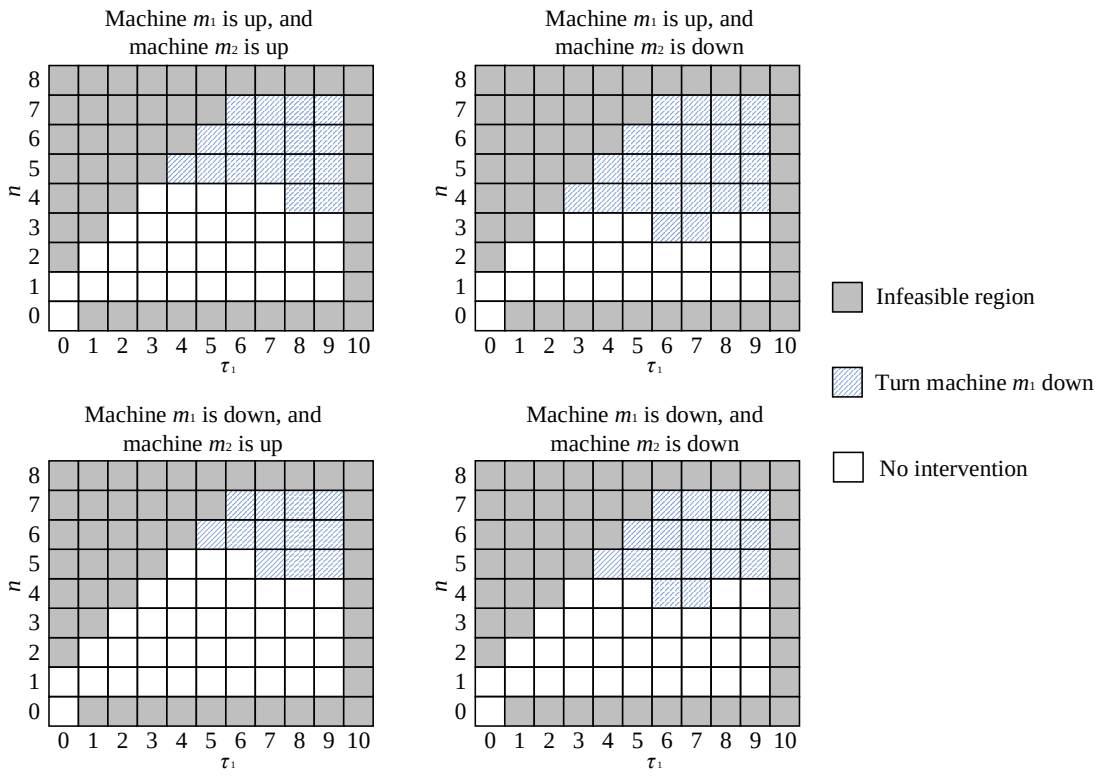


Figure 3.9: The Control Policy Obtained from the MDP Model

Since the optimal solution may not be unique, the right-hand side of the expression can be a set of the optimal actions. With the optimal action for each system state obtained, the optimal control policy π^* can be derived.

3.4.2 Control Policy Evaluation

Further numerical experiments are conducted to illustrate the effectiveness of the control policy obtained from the proposed MDP model. The parameters for the numerical experiment are set to be the same as Equation (3.21). In addition, set $\omega = 0.7$ and $\lambda = 0.99$. With the help of MATLAB, the control policy is obtained and shown in Figure 3.9. Four charts, based on four combinations of machine states, are provided. In each chart, the horizontal axis represents the residence time of the first

part in the buffer and the vertical axis represents the buffer occupancy. A block in a chart represents a state of the system and the action to the state is represented by the color and pattern. The grey blocks in the charts represent the infeasible region. The blocks in each chart surrounded by the grey blocks are the feasible region. Among those blocks, the action that keeps machine m_1 unchanged is taken when the system state is in a while block, and the action that turns machine m_1 down is taken when the system state is in a block with a blue pattern. Several features of the control policy can be observed from Figure 3.9. There are mainly three features that can be observed from the control policy.

1) The buffer occupancy is an important factor to consider when applying an action. In the feasible region of each chart in Figure 3.9, there is a boundary line that separates the white blocks from the blocks with a blue pattern. The boundary lines characterize the control policy. By observing the boundary lines, one can realize that no intervention is needed on machine m_1 when the buffer occupancy is low. Machine m_1 is required to be turned down, when the buffer occupancy is high. Buffer is aimed at providing smooth production under uncertainties, and thus production on machine m_1 is encouraged when the buffer occupancy is low. However, it takes a long time to consume all parts in the buffer by machine m_2 when the buffer occupancy is high. A high buffer occupancy means a high risk of the scrap due to residence time constraints. Thus, machine m_1 is turned down to prevent the buffer occupancy from being too high.

2) The action depends on machine states. The control policy shown in Figure 3.9 also suggests that more parts can be accepted to wait in the buffer when machine m_1 is down and machine m_2 is up, while a smaller number of parts are allowed to stay in the buffer when machine m_1 is up and machine m_2 is down. The reason is that

the working of machine m_2 promotes the consumption of parts in the buffer and then reduces the risk of scrap, while the the working of machine m_1 encourages the accumulation of parts in the buffer and then increases the risk of scrap. Therefore, the control policy is more likely to turn machine m_1 down when machine m_1 is up and machine m_2 is down.

3) The control policy is not sensitive to the residence time of the first part in the buffer. Besides, the monotonic boundary can be observed when machine m_2 is up, while a u-shaped boundary is for the case where machine m_2 is down. The reasons for such phenomena are two-fold. The first focus is on why the boundary line declines when τ_1 is small. The reason is that τ_2 is estimated based on the values of τ_1 and n. It is more likely to obtain a large τ_2 if τ_1 is large. Similarly, it is more likely to obtain a large τ_2 if n is large. Thus, when τ_1 increases, the control policy tends to prevent n from increasing. Under such control policy, when the first part is produced by machine m_2 or scrapped, the new head part tends to have a small τ_1. This can reduce the risk of scrap for the new head part. This explains why the dividing line goes down when τ_1 is small. However, the manipulation on machine m_1 cannot change τ_2 in reality. When a part enters the buffer, its residence time increases by 1 each cycle no matter how machine m_1 works. All the parts that already exist in the buffer will not influenced by machine m_1. The reason that the control policy tends to believe that τ_2 can be reduced by preventing n from increasing is that the control policy assumes that $\Phi(n, \tau_1, \tau_2)$ keeps unchanged after the control policy is applied. This assumption can cause errors and also result in the room for further improvement for the control policy obtained from MDP model, which motivates the further study in Section 6.3. Second, the reason why the boundary line goes up when τ_1 is large is provided as follows. As is

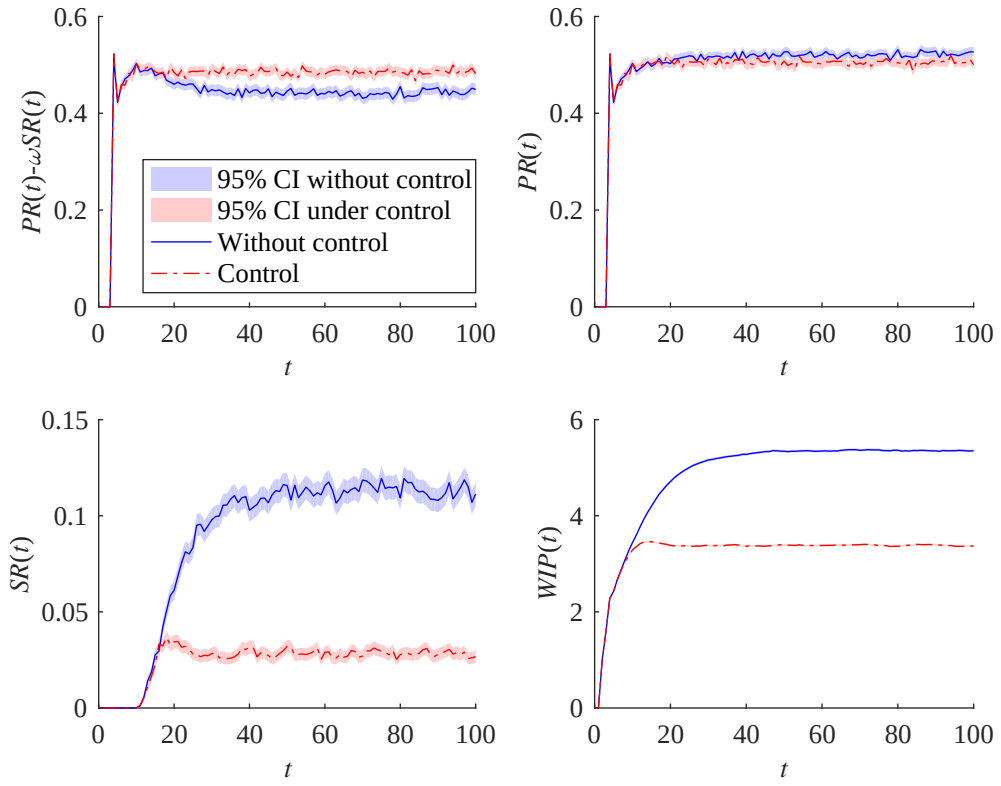


Figure 3.10: Comparison of Performance Measures with and without Control

mentioned, the manipulation on machine m_1 actually makes no difference in parts that have been in the buffer. The scrap of parts that have been in the buffer can decrease the length of the queue in the buffer, and it increases the probability that a new-release part from machine m_1 is finally produced by machine m_2. When τ_1 is large, the probability that the first part in the buffer will be scrapped is high. It encourages machine m_1 to keep producing more products.

To compare the transient behaviors of systems with and without control, simulation experiments are conducted. Simulation runs 100 cycles and repeats 10,000 times. The results are plotted in Figure 3.10. The solid blue line represents the original case where no control is employed, while the dashed red line characterizes the controlled

case. The shaded areas stand for the 95% confidence interval. The result suggests that by applying the real-time control policy generated by the MDP model, considerable reduction on scrap rate can be achieved without sacrificing too much on the production gain.

3.4.3 Policy Improvement by Bayesian Methods

The MDP model provides an approach to obtain a control policy to the two-machine geometric serial line, and the simulation study has suggested that this control policy can balance the trade-off between production rate and scrap rate. This control policy derived from MDP model can be further improved when the system starts to work and real-time data become available. Approximations are used in the modeling of the two-machine geometric serial line. Specifically, $\Phi(n, \tau_1, \tau_2)$ is used in the Markov chain model under the assumption that consumption on machine m_1 follows an independent geometric reliability model. This assumption may not be followed when control policies are implemented in the system. Therefore, the control policy from MDP model may not perform well in some cases. In this subsection, Bayesian methods and run time data are applied to improve the original control policy. Specifically, take the originally estimated $\Phi(n, \tau_1, \tau_2)$ as a prior belief and make use of run time data to update $\Phi(n, \tau_1, \tau_2)$. A new control policy can be derived from the updated $\Phi(n, \tau_1, \tau_2)$, and in the meantime, the new control policy brings new errors to $\Phi(n, \tau_1, \tau_2)$. Run time data are collected to continue the iteration, until no better control policy could be found.

First fix n and τ_1. Let $\boldsymbol{p}^{n,\tau_1} = \left(p_0^{n,\tau_1}, p_1^{n,\tau_1}, \cdots, p_{T_{max-2}}^{n,\tau_1}\right)^T$ denote a T_{max-1}-dimension vector to represent the probabilities that different values of τ_2 occur. Specifically, $p_i^{n,\tau_1} = P(\tau_2 = i \mid n, \tau_1)$, for $i = 0, 1, \cdots, T_{max-2}$. Each time a part in buffer is scrapped or produced by machine m_2, an observation of τ_2 is got. As-

sume during a period of time there are m observations of τ_2. Given n and τ_1, m observations of τ_2 follow the multinomial distribution with parameters m and $\boldsymbol{p}^{n,\tau_1}$, denoted by $\boldsymbol{c}|\boldsymbol{p}^{n,\tau_1}\sim Multinomial\left(m,\boldsymbol{p}^{n,\tau_1}\right)$. The problem is to estimate $\boldsymbol{p}^{n,\tau_1}$ based on $\Phi(n,\tau_1,\tau_2)$ and m observations. The estimated $\boldsymbol{p}^{n,\tau_1}$ results in a new $\Phi(n,\tau_1,\tau_2)$, which could help generate a better control policy.

In the Bayesian method, assume that $\boldsymbol{p}^{n,\tau_1}$ follows the Dirichlet distribution with vector parameters $\boldsymbol{\alpha}^{n,\tau_1}=\left(\alpha_0^{n,\tau_1},\alpha_1^{n,\tau_1},\cdots,\alpha_{T_{max}-2}^{n,\tau_1}\right)^T$, denoted by $\boldsymbol{p}^{n,\tau_1}\sim Dir\left(\boldsymbol{\alpha}^{n,\tau_1}\right)$. Given n and τ_1, the observations of τ_2 are modeled as the multinomial distribution with a Dirichlet prior. The probability density function is given by

$$f\left(\boldsymbol{p}^{n,\tau_1};\boldsymbol{\alpha}^{n,\tau_1}\right)=\frac{1}{B\left(\boldsymbol{\alpha}^{n,\tau_1}\right)}\prod_{i=0}^{T_{max}-2}\left(p_i^{n,\tau_1}\right)^{\alpha_i^{n,\tau_1}-1}, \tag{3.30}$$

where

$$B\left(\boldsymbol{\alpha}^{n,\tau_1}\right)=\frac{\Pi_{i=0}^{T_{max}-2}\Gamma\left(\alpha_i^{n,\tau_1}\right)}{\Gamma\left(\sum_{i=0}^{T_{max}-2}\alpha_i^{n,\tau_1}\right)}, \tag{3.31}$$

and

$$\sum_{i=0}^{T_{max}-2}p_i^{n,\tau_1}=1, \tag{3.32}$$

$$p_i^{n,\tau_1}\in[0,1]\ \text{ for } i=0,1,\cdots,T_{max}-2. \tag{3.33}$$

Let $\boldsymbol{c}=\left(c_0,c_1,\cdots,c_{T_{max}-2}\right)^T$ be the number of occurrences of each τ_2 among m observations. Then,

$$\begin{aligned}P\left(\boldsymbol{p}^{n,\tau_1}|\boldsymbol{c}\right)&\propto P\left(\boldsymbol{c}|\boldsymbol{p}^{n,\tau_1}\right)P\left(\boldsymbol{p}^{n,\tau_1}\right)\\&\propto\Pi_{i=0}^{T_{max}-2}\left(p_i^{n,\tau_1}\right)^{c_i}\Pi_{i=0}^{T_{max}-2}\left(p_i^{n,\tau_1}\right)^{\alpha_i^{n,\tau_1}-1}\\&=\Pi_{i=0}^{T_{max}-2}\left(p_i^{n,\tau_1}\right)^{\alpha_i^{n,\tau_1}+c_i-1}.\end{aligned} \tag{3.34}$$

The posterior follows Dirichlet distribution:

$$\boldsymbol{p}^{n,\tau_1}|\boldsymbol{c}\sim Dir\left(\boldsymbol{\alpha}^{n,\tau_1}+\boldsymbol{c}\right). \tag{3.35}$$

The parameters of the prior is obtained from the estimated $\Phi(n, \tau_1, \tau_2)$. Define μ as a weight for prior belief. The parameter α_i^{n,τ_1} can be obtained

$$\alpha_i^{n,\tau_1} = \mu\Phi(n, \tau_1, \tau_2 = i) = \mu P(\tau_2 = i|n, \tau_1), \quad \text{for } i = 0, 1, \cdots, T_{max} - 2. \tag{3.36}$$

During the run time, $\boldsymbol{c}$ for different n and τ_1 is collected, and then it can be used to update $\Phi(n, \tau_1, \tau_2)$:

$$P(\tau_2 = i|n, \tau_1) = \frac{\alpha_i^{n,\tau_1} + c_i}{\sum_{j=0}^{T_{max}-2}\left(\alpha_j^{n,\tau_1} + c_j\right)}, \quad \text{for } i = 0, 1, \cdots, T_{max} - 2. \tag{3.37}$$

After the observations are modeled, run time data could be used to update the control policy based on the Bayesian methods. An algorithm is developed to make use of Bayesian methods to improve the control policy as shown in Figure 3.11. The algorithm starts with a control policy derived from the MDP model. The update cycle number θ is determined. Each time the number of observations reaches θ, the prior information and run time data are combined to calculate a new $\Phi(n, \tau_1, \tau_2)$ and its corresponding control policy. If the control policy is different from the original one, the new control policy is implemented and Equation (3.36) is then used to obtain a new prior. Each time a new control policy is obtained, the current control policy and the updated $\Phi(n, \tau_1, \tau_2)$ are applied to the analytical model. If the average reward does not improve any more, stop the process and use the control policy obtained from the previous iteration permanently. The iteration continues until it reaches the total production cycles Σ or it cannot improve the system.

To illustrate how the algorithm improves the control policy and system performance, simulation experiments are conducted to compare the control with and without the integration with Bayesian methods. The Bayesian method works when the estimated $\Phi(n, \tau_1, \tau_2)$ is not accurate due to the implementation of a control policy. Therefore, the Bayesian method is more likely to work, when machine m_1 does not

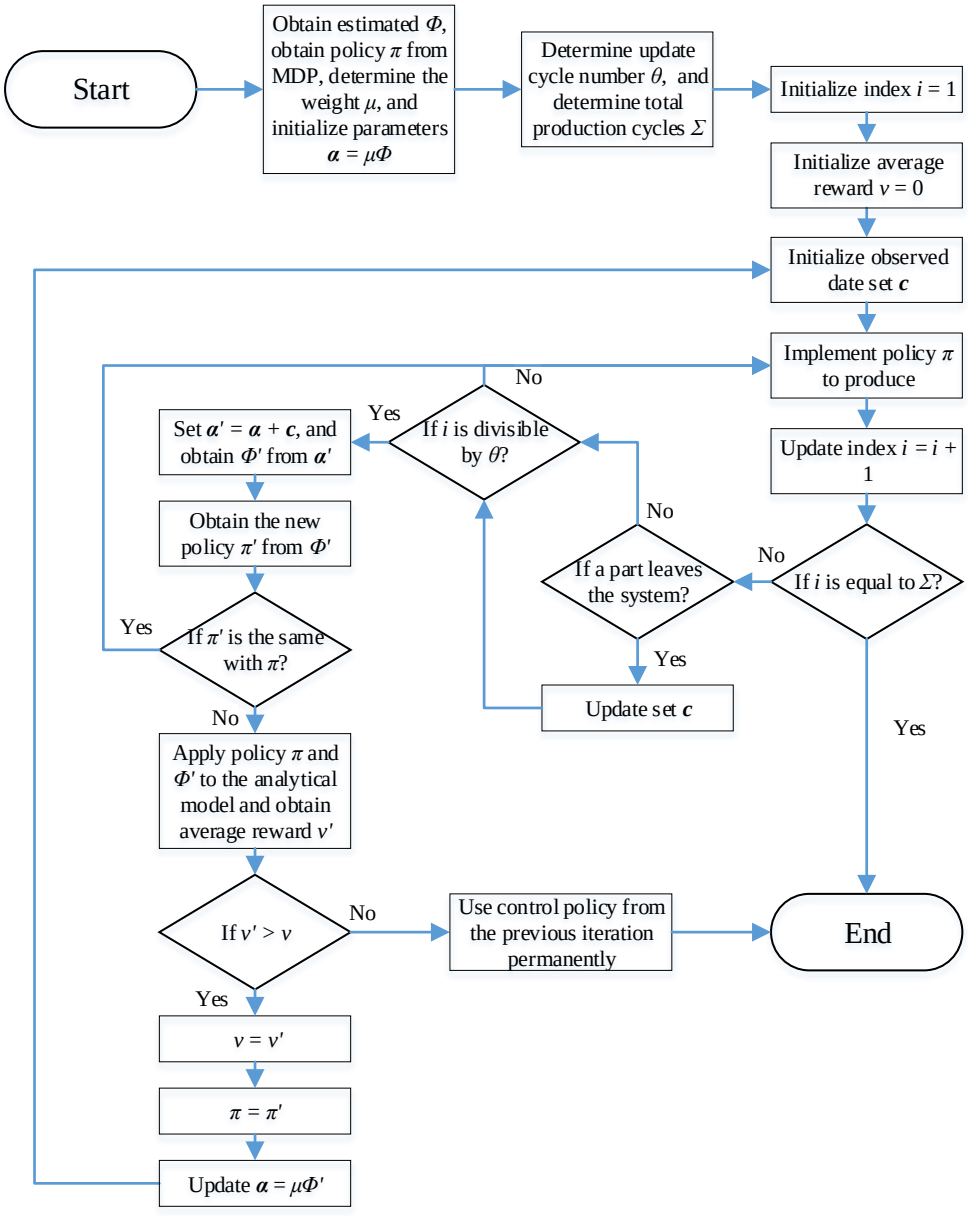
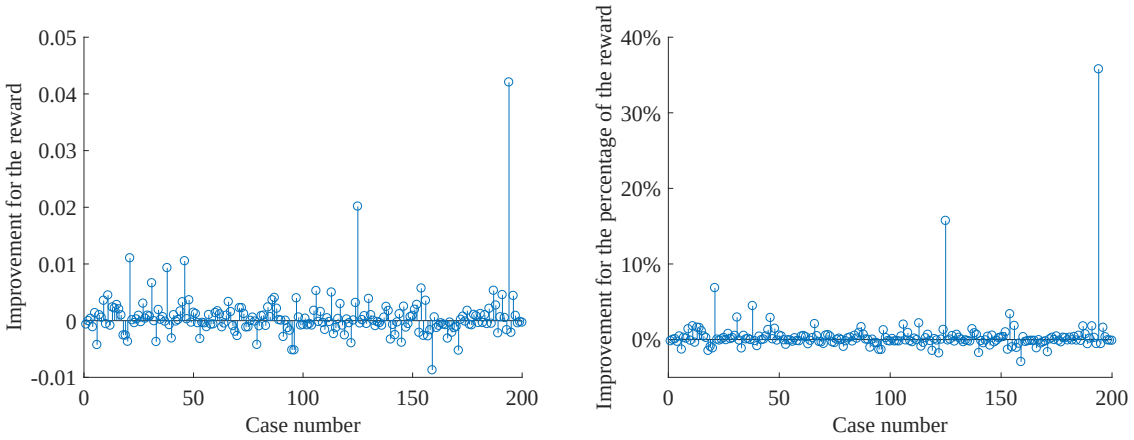


Figure 3.11: Flow Chart of Algorithm Based on Bayesian Methods



(a) (b)

Figure 3.12: Improvement of Reward for Bayesian Methods

fail often but is frequently turned down according to the control policy applied. Thus, select parameters randomly from the parameter set in Equation (3.12), and also set $p_1 = 0.1$. In addition, set $\omega \in (0.65, 0.75)$, $\mu = 50$, and $\lambda = 0.99$. The simulation runs for 1,800 cycles, and 2,000 replications are carried out in each parameter setting. The original control policy is taken as a prior, and $\Phi(n, \tau_1, \tau_2)$ is updated each 300 cycles. It means $\Sigma = 1800$ and $\theta = 300$. 200 settings are chosen randomly from the predefined range. For each case, use the average reward of the control with Bayesian methods during the last 300 cycles minus the average reward without Bayesian methods. Positive difference means that the algorithm leads to larger rewards. The results for the 200 settings are given in Figure 3.12a and 3.12b. Figure 3.12a shows the difference of rewards of the 200 cases, while 3.12b demonstrates the percentage of improvement of these cases. These figures suggest substantial performance improvement can be achieved by using the Bayesian method for policy improvement. Among the 200 illustrated cases, most of them do not deliver much improvement, typically about 1-5%, because the approximation in the MDP model does not often lead to a large error and the control policy obtained from the MDP model is good enough for

most of these cases. However, there are cases where the production performance can benefit substantial improvement from the Bayesian method combining with the original MDP model. As one could see, there is one setting that the reward is increased by as much as 0.4, which is about 35% improvement. There are also other several cases that show significant improvement. Typically, noticeable improvement could be obtained when the first machine has a relatively small failure probability, in which case the approximation method tends to have large deviation.

A case study is used to illustrate in detail how Bayesian methods make use of run time data to improve the control policy. The parameters for the case study are

$$p_1 = 0.1, p_2 = 0.65, r_1 = 0.45, r_2 = 0.35, N = 3,$$
$$T_{max} = 5, T_{min} = 1, \omega = 0.7, \lambda = 0.99, \mu = 50. \tag{3.38}$$

Set $\Sigma = 2400$ and $\theta = 300$, and simulation repeats for 20,000 times under such parameter setting. In this case study, three control policies are compared. The first control method is the original control from the approximation based MDP model. The second model is the original control integrated with Bayesian methods. The third control method is the optimal control obtained through the exact MDP model without any approximation. The comparison of the performance measures for the three control methods is provided in Figure 3.13. It is obvious that the optimal control policy without approximation delivers the best performance across all four measures. There exists a clear gap in terms of each performance measure between the original control and the optimal control as approximation is involved. For the original control and the control with Bayesian methods, their performance overlaps in the first 300 cycles since no run time data is collected initially. After the first 300 cycles, control policy is updated by the Bayesian method which significantly improve the approximation accuracy thus leading to better system performance. The

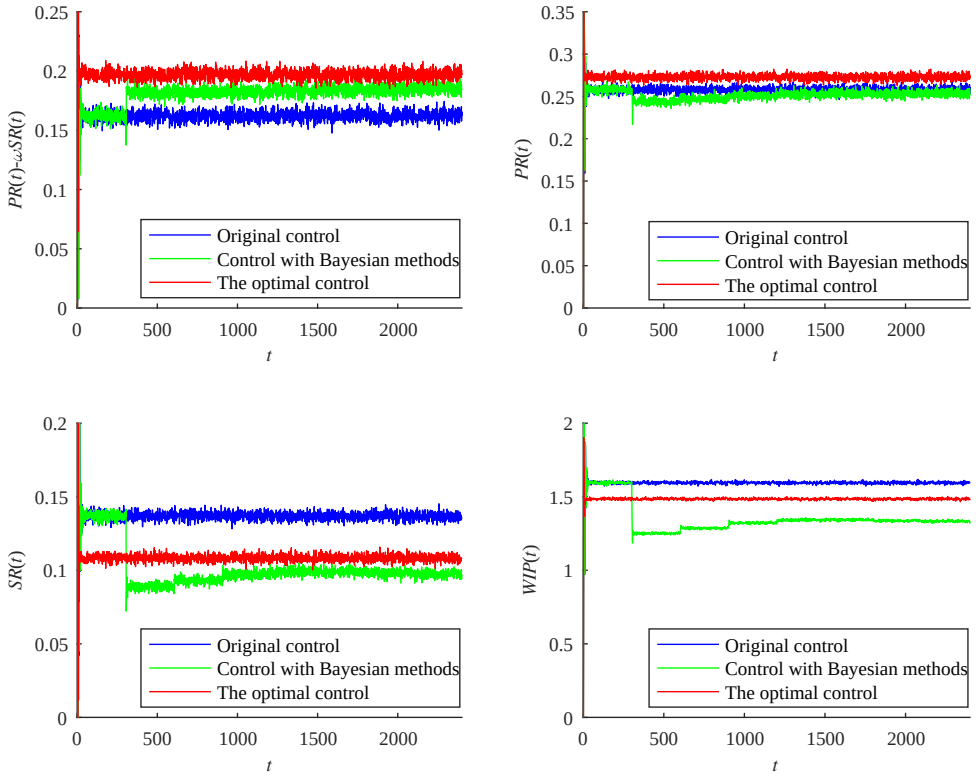
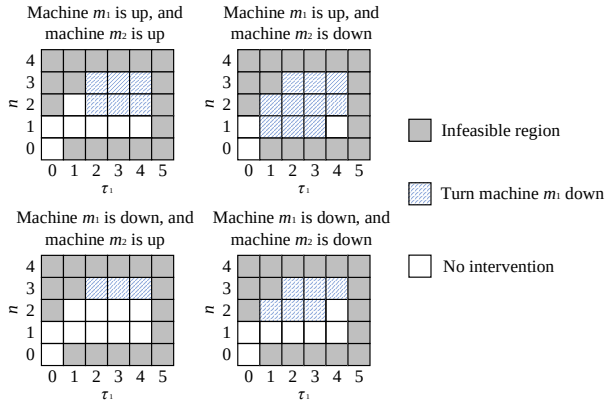
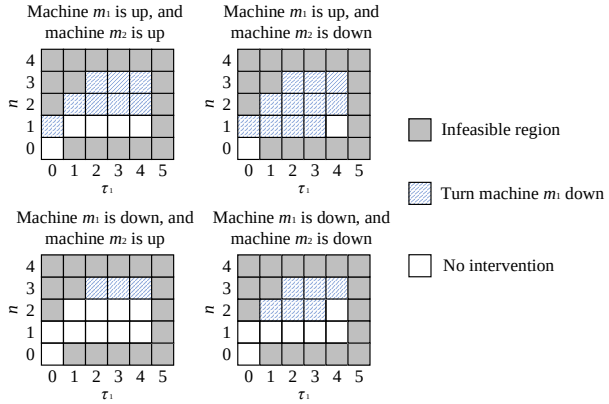


Figure 3.13: Comparison of Performance Measure with and without Bayesian Methods

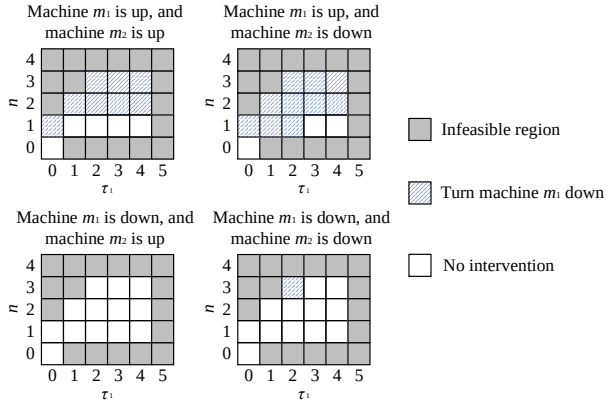
difference between control policies is illustrated in Figure 3.14. As more data is collected and applied, the improvement from the Bayesian method diminishes and the system performance under such control becomes eventually close to the optimal control.



(a) Control Policy from MDP



(b) Control Policy Improved by Bayesian Methods



(c) The Optimal Control Policy

Figure 3.14: Comparison of Control Policies without and with Bayesian Methods

Chapter 4

ANALYSIS OF MULTI-STAGE SERIAL LINE

4.1 List of Symbols for this Chapter

- D: Number of machines in multi-stage line.
- m_i: The ith machine of multi-stage line.
- m_1^{sub}: The first machine of subsystem.
- m_2^{sub}: The second machine of subsystem.
- s_i^{sub}: The state of machine m_i^{sub} in subsystem.
- p_i: Failure probability of machine m_i in multi-stage line.
- p_i^{sub}: Failure probability of machine m_i^{sub} in subsystem.
- r_i: Repair probability of machine m_i in multi-stage line.
- r_i^{sub}: Repair probability of machine m_i^{sub} in subsystem.
- B_i: The ith buffer in multi-stage line.
- B: The buffer in subsystem.
- N_i: Capacity of buffer B_i.
- N: Capacity of buffer B.
- n: Buffer occupancy for buffer B.

- $T_{i,min}$: Minimum required residence time for buffer B_i.
- T_{min}: Minimum required residence time for buffer B.
- $T_{i,max}$: Maximum allowable residence time for buffer B_i.
- T_{max}: Maximum allowable residence time for buffer B.
- τ_i: Residence time of the ith part in buffer B.
- $p_i^s(t)$: Starvation probability of machine m_i in cycle t.
- p^s: Starvation probability of machine m_1^{sub}.
- $p_i^b(t)$: Blockage probability of machine m_i in cycle t.
- p^b: Blockage probability of machine m_2^{sub}.
- $P_{1,1}^{(k)}$: Probability that machine m_k^{sub} that is up in one cycle is still up in the next cycle.
- $P_{1,0}^{(k)}$: Probability that machine m_k^{sub} that is up in one cycle is down in the next cycle.
- $P_{0,1}^{(k)}$: Probability that machine m_k^{sub} that is down in one cycle is up in the next cycle.
- $P_{0,0}^{(k)}$: Probability that machine m_k^{sub} that is down in one cycle is still down in the next cycle.
- $x(n, \tau_1, s_1^{sub}, s_2^{sub}, t)$: Probability for state $(n, \tau_1, s_1^{sub}, s_2^{sub})$ in cycle t.
- $\Phi(n, \tau_1, \tau_2)$: Conditional probability that the second part in buffer B has residence time τ_2 given that there are n parts in buffer B and the first part in buffer B has residence time τ_1.

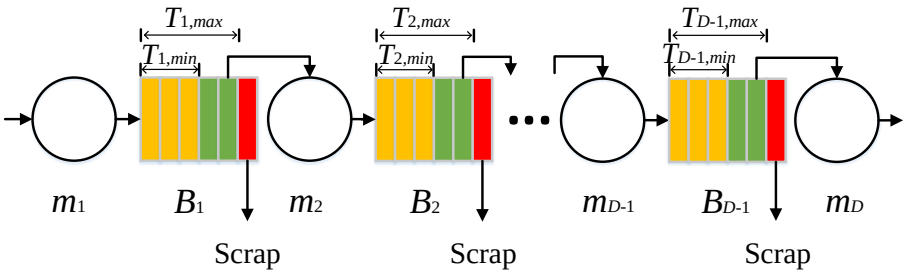


Figure 4.1: A Multi-Stage Geometric Serial Line with Residence Time Limits

4.2 Problem Formulation

In this chapter, a method that analyzes a multi-stage line with residence time constraints is proposed. For simplicity purpose, the term "multi-stage line" is used to represent a multi-stage geometric serial production line with residence time limits for the rest of the chapter. The multi-stage line under study is shown in Figure 4.1. Raw materials enter machine m_1 to be processed and continue to flow downstream until they finish the process in machine m_D or get scrapped from a buffer. The following assumptions define the machines, the buffers, and their interactions.

(i) The multi-stage line consists of D machines (denoted by $m_1, m_2, \cdots, m_D$) and $(D-1)$ buffers (denoted by $B_1, B_2, \cdots, B_{D-1}$), where $D > 2$.

(ii) All machines are synchronized with a constant processing time (cycle time), which is the time to process a part.

(iii) Machines are subject to failures, and their reliability models are independent. The reliability model for each machine follows a geometric distribution. Specifically, if machine m_i is up in cycle $(k-1)$, it will still be up with probability $(1-p_i)$ and down with probability p_i during the k-th cycle, for $i = 1, 2, \cdots, D$, and $k = 2, 3, \cdots$. If machine m_i is down in cycle $(k-1)$, it will be up with

probability r_i and down with probability $(1 - r_i)$ during the k-th cycle, for $i = 1, 2, \cdots, D$, and $k = 2, 3, \cdots$. Here p_i and r_i are defined as the failure probability and repair probability, respectively, for $i = 1, 2, \cdots, D$. The machine efficiency of machine m_i, denoted by e_i, is represented by $e_i = \frac{r_i}{r_i + p_i}$.

(iv) Buffer B_i has a finite capacity N_i $(1 \leq N_i < \infty)$, for $i = 1, 2, \cdots, D - 1$. First-in-first-out (FIFO) policy is assumed regarding the buffer outflow process.

(v) The maximum allowable residence time for parts in buffer B_i is characterized by $T_{i,max}$, for $i = 1, 2, \cdots, D - 1$, counted as the number of cycles. A part in buffer B_i will be scrapped immediately at the beginning of the cycle when its residence time reaches $T_{i,max}$. Let $T_{i,max} \geq N_i$, otherwise N_i has no effect on the system.

(vi) The minimum required residence time for parts in buffer B_i is denoted by $T_{i,min}$, for $i = 1, 2, \cdots, D - 1$, counted as the number of cycles. A part is allowed to leave buffer B_i and enter machine m_{i+1} only when its residence time reaches or exceeds $T_{i,min}$.

(vii) Machine m_i, for $i = 1, 2, \cdots, D - 1$, is blocked during a time slot, if at the beginning of the cycle, (a) machine m_i is up, (b) buffer B_i is full, (c) machine m_{i+1} does not produce a part in this cycle due to machine failure or blockage, and (d) there will be no part scrapped from buffer B_i at the beginning of the next cycle. Machine m_D is never blocked. In addition, block-before-service policy is assumed.

(viii) Machine m_i, for $i = 2, \cdots, D$, is starved during a time slot, if machine m_i is up and no part in buffer B_{i-1} has residence time greater than or equal to $T_{i-1,min}$. Machine m_1 is never starved.

The problem to be studied is to develop a method under assumptions (i)-(viii) to evaluate both the steady-state and transient behaviors of the multi-stage line. Specifically, the system behavior of a multi-stage line is described by performance measures defined as follows.

- *Production Rate,* $PR_i(t)$*, for* $i = 1, \cdots, D$: the expected number of parts produced by machine m_i in cycle t;
- *Overall Production Rate,* $PR(t)$: the expected number of parts produced by the multi-stage line in cycle t;
- *Overall Consumption Rate,* $CR(t)$: the expected number of parts that enter the multi-stage line in cycle t;
- *Scrap Rate,* $SR_i(t)$*, for* $i = 1, \cdots, D-1$: the expected number of scrapped parts from buffer B_i in cycle t;
- *Overall Scrap Rate,* $SR(t)$: the expected number of scrapped parts from the multi-stage line in cycle t;
- *Work-in-process,* $WIP_i(t)$*, for* $i = 1, \cdots, D-1$: the expected number of parts in buffer B_i in cycle t;
- *Overall Work-in-process,* $WIP(t)$: the expected number of parts in the multi-stage line in cycle t;
- *Starvation Probability,* $p_i^s(t)$*, for* $i = 1, \cdots, D$: the probability that machine m_i is starved in cycle t, when machine m_i is up;
- *Blockage Probability,* $p_i^b(t)$*, for* $i = 1, \cdots, D$: the probability that machine m_i is blocked in cycle t, when machine m_i is up.

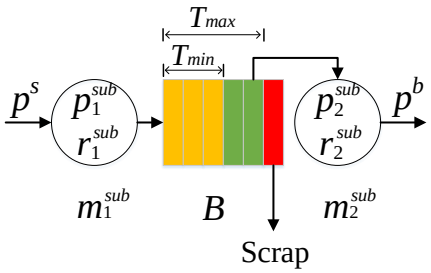


Figure 4.2: A Two-Machine-One-Buffer Subsystem

For a multi-stage line, the overall production rate is equal to the production rate of the last machine, and thus $PR(t) = PR_D(t)$ for all t. The overall consumption rate is equal to the production rate of the first machine, so $CR(t) = PR_1(t)$ for all t. Scrap occurs in each buffer in the multi-stage line. Thus, the overall scrap rate is the summation of scrap rates of all buffers. Similarly, the overall work-in-process is the summation of work-in-processes of all buffers. Thus, $SR(t) = \sum_{i=1}^{D-1} SR_i(t)$ and $WIP(t) = \sum_{i=1}^{D-1} WIP_i(t)$ for all t. By assumptions (vii) and (viii), $p_1^s(t) = 0$ and $p_D^b(t) = 0$ for all t.

4.3 Two-Machine-One-Buffer Subsystems

4.3.1 Model Formulation

The multi-stage line cannot be modeled directly using a single Markov chain due to its large state space. For instance, for a multi-stage line with 10 machines, 9 buffers, buffer capacity being $N_i = 6$ and maximum allowable residence time $T_{i,max} = 8$ for $i = 1, 2, \cdots, 9$, the total number of system states is as large as 3.5×10^{24}. Alternatively, one can start with a simple two-machine-one-buffer subsystem with a much smaller state space as shown in Figure 4.2, which plays a role as a building block for the

performance evaluation of multi-stage line latter. In the rest of the chapter, the term "subsystem" is used to represent the two-machine-one-buffer subsystem.

A subsystem consists of two machines (denoted by m_1^{sub} and m_2^{sub}) and a buffer (denoted by B). Similar to a machine in a multi-stage line, machine m_i^{sub} in a subsystem is characterized by failure probability p_i^{sub} and repair probability r_i^{sub}, for $i = 1, 2$. Buffer B is described by its maximum allowable residence time T_{max}, minimum required residence time T_{min}, and buffer capacity N. A subsystem, isolated from a multi-stage line, is influenced by its upstream buffer through starvation and its downstream buffer through blockage. In order to cope with such effects, two probabilities, starvation probability p^s and blockage probability p^b, are used to model the starvation from the upstream buffer and the blockage from the downstream buffer, respectively. Specifically, p^s presents the probability that starvation occurs to machine m_1^{sub} when machine m_1^{sub} is up. p^b denotes the probability that blockage occurs to machine m_2^{sub} when machine m_2^{sub} is up.

To analyze a subsystem, only residence time of the first part in buffer B is included in the state, instead of recording residence times of all the parts. The rest of the residence time is then estimated using approximation. The detail of the approximate method is discussed in Section 4.3.3. Let $n \in \{0, 1, \cdots, N\}$ denote the buffer occupancy in buffer B. $\tau_1 \in \{0, 1, \cdots, T_{max} - 1\}$ denotes the residence time of the first part in buffer B. Denote the states of machines m_1^{sub} and m_2^{sub}'s states by s_1^{sub} and s_2^{sub}, respectively. Specifically, $s_i^{sub} = 1$ means that machine m_i^{sub} is up, and $s_i^{sub} = 0$ means that machine m_i^{sub} is down. Then, the system state of a subsystem is represented by $\left(n, \tau_1, s_1^{sub}, s_2^{sub}\right)$. For the same example mentioned above with 3.5×10^{24} states, the number of states of the approximate model for each subsystem is 136. The size of state space for a single model to be analyzed is significantly reduced.

4.3.2 Transition Equations

Based on the states of subsystems, the transition equations can be constructed. Let $x(n, \tau_1, s_1^{sub}, s_2^{sub}, t)$ denote the probability of state $(n, \tau_1, s_1^{sub}, s_2^{sub})$ in cycle t. Let $P_{i,j}^{(k)}$ denote the conditional probability that the state of machine m_k^{sub} is j given that its state is i in the previous cycle, for $i, j = 0, 1$ and $k = 1, 2$. Specifically, $P_{1,1}^{(k)} = 1 - p_k^{sub}$, $P_{1,0}^{(k)} = p_k^{sub}$, $P_{0,1}^{(k)} = r_k^{sub}$, and $P_{0,0}^{(k)} = 1 - r_k^{sub}$, for $k = 1, 2$. One can use the operator $\Phi(n, \tau_1, \tau_2)$, which is defined as the conditional probability that the second part in buffer B has residence time τ_2 given that there are n parts in buffer B and the first part in buffer B has residence time τ_1. The method to estimate $\Phi(n, \tau_1, \tau_2)$ will be introduced in Section 4.3.3 in detail. Let us consider the state $(1, i, s_1^{sub}, s_2^{sub})$ in cycle $(t+1)$, for $1 \leq i \leq T_{max} - 1$ and $s_1^{sub}, s_2^{sub} = 0, 1$. There is one part in the buffer and its residence time could be any feasible value except 0. The transition equation for state $(1, i, s_1^{sub}, s_2^{sub})$ can be expressed as

$$
\begin{aligned}
&x(1,i,s_1^{sub},s_2^{sub},t+1)\\
&=x(1,i-1,0,0,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{0,s_2^{sub}}\\
&+x(1,i-1,1,0,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{0,s_2^{sub}}\\
&+x(1,i-1,0,1,t)p^bP^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\mathbf{1}_{\mathbb{N}^+}(i-T_{min})\\
&+x(1,i-1,1,1,t)p^sp^bP^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\mathbf{1}_{\mathbb{N}^+}(i-T_{min})\\
&+x(1,i-1,0,1,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i)\\
&+x(1,i-1,1,1,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i)\\
&+\sum_{j=\max(i,T_{min})}^{T_{max}-2}x(2,j,0,1,t)(1-p^b)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\Phi(2,j,i-1)\\
&+\sum_{j=\max(i,T_{min})}^{T_{max}-2}x(2,j,1,1,t)p^s(1-p^b)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\Phi(2,j,i-1)\\
&+x(2,T_{max}-1,0,0,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{0,s_2^{sub}}\Phi(2,T_{max}-1,i-1)\\
&+x(2,T_{max}-1,1,0,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{0,s_2^{sub}}\Phi(2,T_{max}-1,i-1)\\
&+x(2,T_{max}-1,0,1,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\Phi(2,T_{max}-1,i-1)\\
&+x(2,T_{max}-1,1,1,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}}\Phi(2,T_{max}-1,i-1),
\end{aligned}
\tag{4.1}
$$

where $\mathbf{1}_{\mathbb{N}^+}(x)$ is an indicator function. $\mathbf{1}_{\mathbb{N}^+}(x)=1$ if x is an positive integer ($x\in\mathbb{N}^+$), and 0 otherwise. Similar to Equation (4.1), transition equations for all the other states can be formulated as shown in Appendix C.

4.3.3 Approximation of Residence Time

The operator $\Phi(n,\tau_1,\tau_2)$ is used in Equation (4.1) for approximation of residence time. The operator $\Phi(n,\tau_1,\tau_2)$ is first proposed in Ju *et al.* (2015) and first applied to geometric serial line in Kang *et al.* (2017a). However, the first machine of a subsystem studied in this chapter can be starved, and the operator $\Phi(n,\tau_1,\tau_2)$ is influenced by starvation from the upstream buffer. Thus, the method that derives

$\Phi(n, \tau_1, \tau_2)$ in literature cannot be directly used. A method that derives $\Phi(n, \tau_1, \tau_2)$ by taking starvation into consideration is provided in what follows.

Given current time t and residence time of the first part τ_1, denote the state sequence of machine m_1^{sub} from cycle $(t - \tau_1 - 1)$ to $(t - 1)$ by a $(\tau_1 + 1)$-dimension vector V. Specifically,

$$V = \left(s_1^{sub}(t - \tau_1 - 1), s_1^{sub}(t - \tau_1), \cdots, s_1^{sub}(t - 1)\right), \tag{4.2}$$

where $s_1^{sub}(i)$ is the state of machine m_1^{sub} in cycle i. Denote by $\gamma(V)$ the probability that a sequence V occurs.

$$\gamma(V) = \prod_{i=1}^{\tau_1} P^{(1)}_{s_1^{sub}(t-\tau_1-2+i), s_1^{sub}(t-\tau_1-1+i)}. \tag{4.3}$$

Since starvation may occur to machine m_1^{sub}, it is possible that no part is produced by machine m_1^{sub} during one cycle even though machine m_1^{sub} is up. Therefore, the $(\tau_1 + 1)$-dimension vector W is defined as a sequence for production of machine m_1^{sub}. Specifically,

$$W = (w_1(t - \tau_1 - 1), w_1(t - \tau_1), \cdots, w_1(t - 1)), \tag{4.4}$$

where $w_1(i) = 1$ represents that m_1^{sub} produces a part in cycle i, and 0 otherwise. Then, $w_1(i) \leq s_1^{sub}(i)$ for any i. The probability that a sequence W occurs can be derived from the sequence V. Given W defined by Equation (4.4), define $\mathcal{C}$ to be a collection of all V that can result in W. Specifically,

$$\mathcal{C} = \{V \mid w_1(i) \leq s_1^{sub}(i), i = t - \tau_1 - 1, \cdots, t - 1\}. \tag{4.5}$$

We denote by $\Gamma(W)$ the probability that a sequence W occurs. Then, $\Gamma(W)$ can be expressed as

$$\Gamma(W) = \sum_{V \in \mathcal{C}} \gamma(V)(p^s)^k (1 - p^s)^{\tau_1 - k}, \tag{4.6}$$

where $k = \sum_{i=t-\tau_1}^{t-1}(s_1^{sub}(i) - w_1(i))$.

We define a set, denoted by $\mathcal{A}$, that consists of all possible W that satisfies $w_1(t-\tau_1-1) = 1$ and $\sum_{i=1}^{\tau_1+1} w_1(t-i) = n$. Similarly, let $\mathcal{B}$ be a set that contains all possible W that satisfies $w_1(t-\tau_1-1) = 1$, $w_1(t-\tau_2-1) = 1$, and $\sum_{i=1}^{\tau_2} w_1(t-i) = n-2$. Specifically,

$$\mathcal{A} = \left\{ W \middle| w_1(t-\tau_1-1) = 1, \sum_{i=1}^{\tau_1+1} w_1(t-i) = n \right\},$$

$$\mathcal{B} = \left\{ W \middle| w_1(t-\tau_1-1) = 1, w_1(t-\tau_2-1) = 1, \sum_{i=1}^{\tau_2} w_1(t-i) = n-2 \right\}.$$

Then, the operator $\Phi(n, \tau_1, \tau_2)$ can be estimated as follows.

$$\Phi(n, \tau_1, \tau_2) = \frac{\sum_{W\in\mathcal{B}} \Gamma(W)}{\sum_{W\in\mathcal{A}} \Gamma(W)}, \tag{4.7}$$

where the denominator is the probability that buffer occupancy is n and residence time of the first part is τ_1, while the numerator represents the probability that buffer occupancy is n, residence time of the first part is τ_1, and residence time of the second part is τ_2.

4.3.4 Performance Measures of Subsystems

The estimated performance measures of a subsystem, for $t \in \mathbb{N}^+ \cup \{\infty\}$, include production rate $\widehat{PR}^{sub}(t)$, consumption rate $\widehat{CR}^{sub}(t)$, Scrap Rate $\widehat{SR}^{sub}(t)$, work-in-process, $\widehat{WIP}^{sub}(t)$, starvation probability $\widehat{ST}^{sub}(t)$ and blockage probability $\widehat{BL}^{sub}(t)$. Given p^s, p^b, and $x(n, \tau_1, s_1^{sub}, s_2^{sub}, t)$, the performance measures are estimated as follows.

$$\widehat{PR}^{sub}(t) = (1-p^b)\sum_{n=1}^{N}\sum_{\tau_1=\max(n-1,T_{min})}^{T_{max}-1}\sum_{s_1^{sub}=0}^{1} x(n,\tau_1,s_1^{sub},1,t), \tag{4.8}$$

$$\begin{aligned}\widehat{CR}^{sub}(t) = (1-p^s)(&\sum_{s_2^{sub}=0}^{1} x(0,0,1,s_2^{sub},t) + \sum_{n=1}^{N-1}\sum_{\tau_1=n-1}^{T_{max}-1}\sum_{s_2^{sub}=0}^{1} x(n,\tau_1,1,s_2^{sub},t) \\ &+(1-p^b)\sum_{\tau_1=\max(N-1,T_{min})}^{T_{max}-2} x(N,\tau_1,1,1,t) + \sum_{s_2^{sub}=0}^{1} x(N,T_{max}-1,1,s_2^{sub},t)),\end{aligned} \tag{4.9}$$

$$\widehat{SR}^{sub}(t) = \sum_{n=1}^{N}\sum_{s_1^{sub}=0}^{1} x(n,T_{max}-1,s_1^{sub},0,t) + p^b\sum_{n=1}^{N}\sum_{s_1^{sub}=0}^{1} x(n,T_{max}-1,s_1^{sub},1,t), \tag{4.10}$$

$$\widehat{WIP}^{sub}(t) = \sum_{n=1}^{N}\sum_{\tau_1=n-1}^{T_{max}-1}\sum_{s_1^{sub}=0}^{1}\sum_{s_2^{sub}=0}^{1} nx(n,\tau_1,s_1^{sub},s_2^{sub},t), \tag{4.11}$$

$$\widehat{ST}^{sub}(t) = \begin{cases} \frac{\sum_{s_1^{sub}=0}^{1} x(0,0,s_1^{sub},1,t)+\sum_{n=1}^{\max(N,T_{min})}\sum_{\tau_1=n-1}^{T_{min}-1}\sum_{s_1^{sub}=0}^{1} x(n,\tau_1,s_1^{sub},1,t)}{\sum_{s_1^{sub}=0}^{1} x(0,0,s_1^{sub},1,t)+\sum_{n=1}^{N}\sum_{\tau_1=n-1}^{T_{max}-1}\sum_{s_1=0}^{1} x(n,\tau_1,s_1^{sub},1,t)}, \text{if } T_{min}>0, \\ \frac{\sum_{s_1^{sub}=0}^{1} x(0,0,s_1^{sub},1,t)}{\sum_{s_1^{sub}=0}^{1} x(0,0,s_1^{sub},1,t)+\sum_{n=1}^{N}\sum_{\tau_1=n-1}^{T_{max}-1}\sum_{s_1^{sub}=0}^{1} x(n,\tau_1,s_1^{sub},1,t)}, \text{if } T_{min}=0, \end{cases} \tag{4.12}$$

$$\widehat{BL}^{sub}(t) = \frac{\sum_{\tau_1=N-1}^{T_{max}-2} x(N,\tau_1,1,0,t) + p^b\sum_{\tau_1=N-1}^{T_{max}-2} x(N,\tau_1,1,1,t)}{\sum_{s_2=0}^{1} x(0,0,1,s_2,t) + \sum_{n=1}^{N}\sum_{\tau_1=n-1}^{T_{max}-1}\sum_{s_2=0}^{1} x(n,\tau_1,1,s_2,t)}. \tag{4.13}$$

The estimated production rate $\widehat{PR}^{sub}(t)$ is the expected number of parts the subsystem produces in cycle t. It is equal to probability that machine m_2^{sub} is up, there is at least one part in the buffer with residence time equal to or greater than T_{min}, and machine m_2^{sub} is not blocked. The estimated consumption rate $\widehat{CR}^{sub}(t)$ represents the expected number of parts that enter the subsystem in cycle t. It is equivalent to the probability that buffer is not full and machine m_1^{sub} produces a part. $\widehat{SR}^{sub}(t)$ denotes the estimated number of scrapped parts from the subsystem in cycle t. It can be calculated as the probability that residence time of the first part in the buffer reaches $(T_{max}-1)$ but machine m_2^{sub} is not able to produce due to machine failure or blockage. $\widehat{WIP}^{sub}(t)$ denotes the estimated number of parts in buffer B in the subsystem in cycle t. $\widehat{ST}^{sub}(t)$ and $\widehat{BL}^{sub}(t)$ are starvation probability of machine m_2^{sub} and blockage probability of machine m_1^{sub}, respectively. The denominator of Equation (4.12) presents the probability that machine m_2^{sub} is up, and the numerator

is the probability that machine m_2^{sub} is up and the buffer is empty. Similarly, the denominator of Equation (4.13) is the probability that machine m_1^{sub} is up, and the numerator is the probability that machine m_1^{sub} is up and the buffer is full.

4.4 Modeling Multi-Stage Line Using Aggregation Method

For a production line with multiple stages, the state space is typically too large to directly perform analysis. Alternatively, the aggregation method is typically pursued to estimate the performance measures of a multi-stage line based on the analysis of all its subsystems. The aggregation method for the multi-stage line consists of the steady-state analysis and transient analysis, which are to be introduced in this section.

4.4.1 Steady-State Analysis

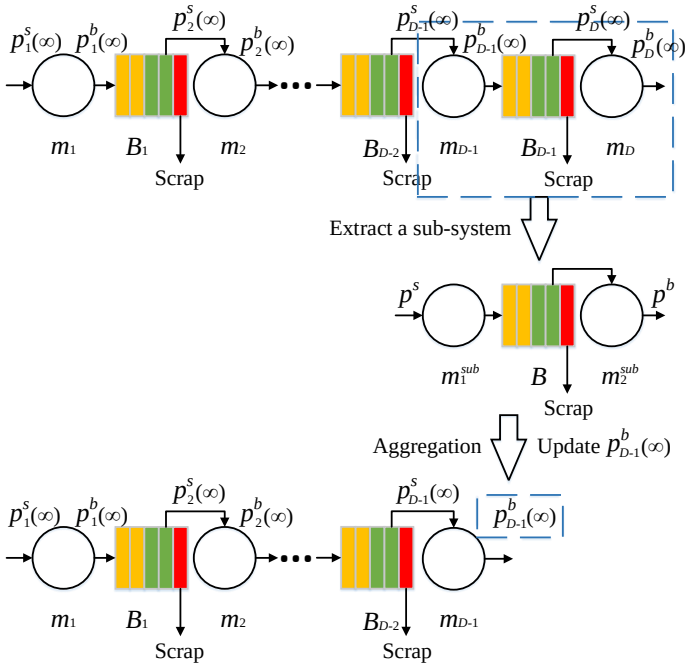
For a multi-stage line in the steady state, the state probability of each state keeps unchanged, the starvation probability and blockage probability of each machine become constant, and the expected performance measures do not vary with time. Given system parameters, the steady-state analysis is aimed at obtaining performance measures introduced in Section 4.2 for $t = \infty$.

When a multi-stage line is in the steady state, each subsystem, isolated from the multi-stage line, is also in the steady state. It means that the starvation probability p^s and the blockage probability p^b for any subsystem do not change with time. The steady-state probability $x(n, \tau_1, s_1^{sub}, s_2^{sub}, \infty)$ for all $(n, \tau_1, s_1^{sub}, s_2^{sub})$ can be obtained via transition equations such as Equation (4.1). Thus, the performance measures $\widehat{PR}^{sub}(\infty)$, $\widehat{CR}^{sub}(\infty)$, $\widehat{SR}^{sub}(\infty)$, $\widehat{WIP}^{sub}(\infty)$, $\widehat{ST}^{sub}(\infty)$, and $\widehat{BL}^{sub}(\infty)$ can be calculated by Equations (4.8)-(4.13). By the analysis of subsystems with $p_i^s(\infty)$ and $p_i^b(\infty)$ known, for $i = 1, \cdots, D$, $\widehat{PR}^{sub}(\infty)$ of the $(i-1)$-th subsystem can be the

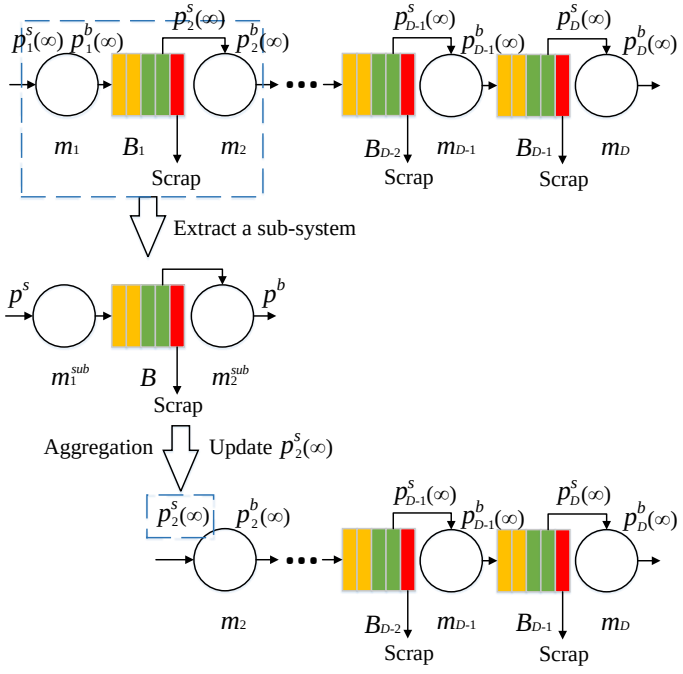
estimate of $PR_i(\infty)$ for $i = 2, \cdots, D$. $\widehat{CR}^{sub}(\infty)$ of the first subsystem can be the estimate of $PR_1(\infty)$. Let $SR_i(\infty)$ and $WIP_i(\infty)$ be $\widehat{SR}^{sub}(\infty)$ and $\widehat{WIP}^{sub}(\infty)$ of the i-th subsystem, respectively, for $i = 1, \cdots, D-1$. Then, $PR(\infty)$, $CR(\infty)$, $SR(\infty)$, and $WIP(\infty)$ can be derived.

The aggregation method provides iterative procedures to estimate $p_i^s(\infty)$ and $p_i^b(\infty)$ for $i = 1, \cdots, D$, shown in Figure 4.3. In each iteration, a backward aggregation and a forward aggregation are performed. Start with the first iteration. By the assumptions (vii) and (viii), $p_1^s(\infty) = 0$ and $p_D^b(\infty) = 0$. Set the initial $p_i^s(\infty)$ to be 0 for $i = 2, \cdots, D$ and initial $p_i^b(\infty)$ to be 0 for $i = 1, \cdots, D-1$.

(a) *Backward aggregation.* The first iteration starts from the backward aggregation, shown in Figure 4.3a. First take machine m_{D-1}, machine m_D, and buffer B_{D-1} to form a subsystem. In the subsystem, the parameters of machine m_1^{sub}, machine m_1^{sub} and buffer B are the same as the parameters of machine m_{D-1}, machine m_D and buffer B_{D-1}, respectively. The values of $p_{D-1}^s(\infty)$ and $p_D^b(\infty)$ of the multi-stage line are assigned to p^s and p^b of the subsystem, respectively. With all the parameters for a subsystem ready, the steady-state performance measures of the subsystem can be obtained. The steady-state blockage probability $\widehat{BL}^{sub}(\infty)$ of the subsystem is used to update the value of $p_{D-1}^b(\infty)$ in the multi-stage line. After this step, a new multi-stage line is created with the number of machines reduced by one, the number of buffers reduced by 1, and the blockage probability $p_{D-1}^b(\infty)$ updated. Then, the process continues by taking machine m_{D-2}, machine m_{D-1} and buffer B_{D-2} from the new multi-stage line to form a subsystem. Continue the process until the number of machines is reduced to be one and all the blockage probabilities $p_i^b(\infty)$, for $i = 1, \cdots, D-1$, are updated.



(a) Backward Aggregation



(b) Forward Aggregation

Figure 4.3: Steady-State Analysis of the Aggregation Method

(b) *Forward aggregation.* Similar to the backward aggregation, the forward aggregation takes two machines and one buffer to form a subsystem but starts from the left side of the multi-stage line, shown in Figure 4.3b. First take machine m_1, machine m_2 and buffer B_1 to form a subsystem. The parameters of machine m_1, machine m_2 and buffer B_1 of the multi-stage line are assigned to machine m_1^{sub}, machine m_2^{sub} and buffer B of the subsystem, respectively. p^s and p^b of the subsystem are assigned the values of $p_1^s(\infty)$ and $p_2^b(\infty)$ of the multi-stage line, respectively. By performing analysis on the subsystem, one can obtain the steady-state starvation probability $\widehat{ST}^{sub}(\infty)$, which is then used to replace $p_2^s(\infty)$ of the multi-stage line. After the step, a new multi-stage line is created with the number of machines reduced by one, the number of buffers reduced by one, and the starvation probability $p_2^s(\infty)$ updated. Continue the process until the number of machines is reduced to be one and all the starvation probabilities $p_i^s(\infty)$, for $i = 2, \cdots, D$, are updated.

An iteration is finished when both one backward aggregation and one forward aggregation are completed. The estimated steady-state performance measures can be obtained after several iterations. The pseudocode for the aggregation method is shown in Figure 4.4. Line 1 and Line 2 are to initialize starvation probability and blockage probability. The iterative procedures of the aggregation method are represented by the loop from line 4 to line 19, among which the backward aggregation is given by the loop from line 5 to line 11 and the forward aggregation is given by the loop from line 12 to line 18. The function that appears in line 9 and line 16 transfers the parameters into the transition matrix by Equation (4.1) and outputs starvation probability and blockage probability by Equation (4.12) and Equation (4.13), respectively.

1: Initialize $p_i^s(\infty) = 0,\ i = 1, \cdots, D$

2: Initialize $p_i^b(\infty) = 0,\ i = 1, \cdots, D$

3: Determine the total number of iterations: $iter$

4: **for** $j = 1, iter$ **do** ▷ Iterations

5: **for** $k = 1, D-1$ **do** ▷ The backward aggregation

6: $l = D - k$ ▷ Set index

7: $m = D - k + 1$

8: $I = \left[p_l, r_l, p_l^s(\infty), p_m, r_m, p_m^b(\infty), T_{l,max}, T_{l,min}, N_l\right]$

9: $\left[\widehat{ST}^{sub}(\infty), \widehat{BL}^{sub}(\infty)\right] = getSubPerformance(I)$

10: $p_l^b(\infty) = \widehat{BL}^{sub}(\infty)$ ▷ Update blockage probability

11: **end for**

12: **for** $k = 1, D-1$ **do** ▷ The forward aggregation

13: $l = k$ ▷ Set index

14: $m = k + 1$

15: $I = \left(p_l, r_l, p_l^s(\infty), p_m, r_m, p_m^b(\infty), T_{l,max}, T_{l,min}, N_l\right)$

16: $\left[\widehat{ST}^{sub}(\infty), \widehat{BL}^{sub}(\infty)\right] = getSubPerformance(I)$

17: $p_m^s(\infty) = \widehat{ST}^{sub}(\infty)$ ▷ Update starvation probability

18: **end for**

19: **end for**

Figure 4.4: Iterative Procedures of the Steady-State Analysis

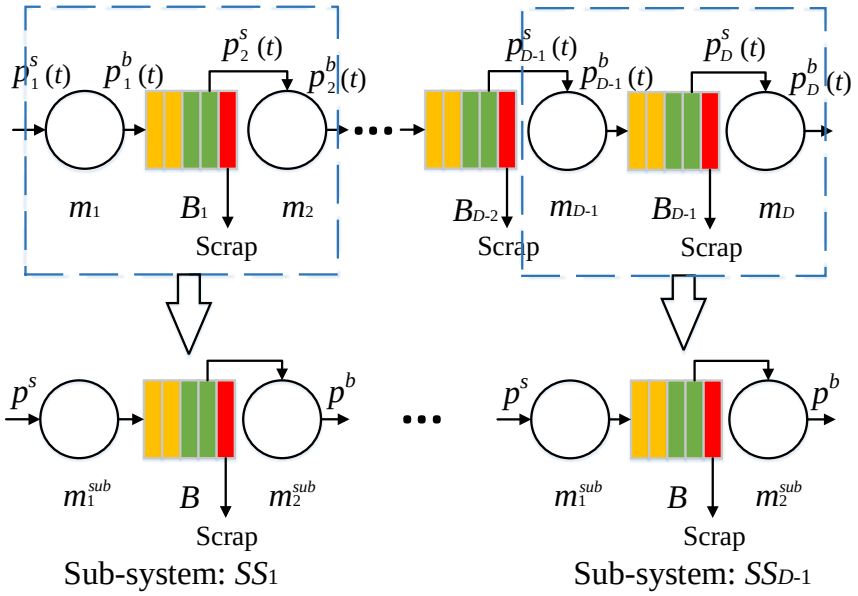


Figure 4.5: A Multi-Stage Line is Decomposed into Subsystems for Transient Analysis

4.4.2 Transient Analysis

With the system parameters and initial system state known, the transient analysis is aimed at obtaining transient performance measures introduced in Section 4.2 for $t \in \mathbb{N}^+$. To perform transient analysis of a multi-stage line, firstly decompose a multi-stage line into several subsystems. Figure 4.5 shows the decomposition, where any two neighboring machines and the buffer between the two machines are isolated to form a subsystem. A multi-stage line with D machines and $(D-1)$ buffers is decomposed into $(D-1)$ subsystems. The subsystem that consists of machine m_i, machine m_{i+1} and buffer B_i is denoted by SS_i, for $i = 1, \cdots, D-1$. In the transient analysis, the starvation probability p^s and the blockage probability p^b for a subsystem change over time, and each subsystem is modeled to be a time-varying Markov chain. The objective of transient analysis is to capture the time-varying transition matrix

of each subsystem over time, so that the transient behavior of both subsystems and the whole multi-stage line can be predicted.

Let $X_i(t)$ denote the row vector of state probabilities of subsystem SS_i in cycle t, for $i = 1, \cdots, D-1$. Denote the transition matrix of subsystem SS_i in cycle t by $Q_i(t)$. The pseudocode for the transient analysis is shown in Figure 4.6, which provides procedures to use the time-varying transition matrices of subsystems to perform transient analysis of the multi-stage line. Given the initial state of a multi-stage line, the initial states of its subsystems are determined. It further determines the state probability $X_i(1)$ for $i = 1, \cdots, D-1$, which is initialized in line 1. With $X_i(1)$ for $i = 1, \cdots, D-1$ known, $p_i^s(1)$ and $p_i^b(1)$ for $i = 1, \cdots, D$ can be obtained. A loop from line 5 to line 18 is then to calculate the transient system state probability of the multi-stage line from cycle $t = 1$ to cycle $t = T$. There are two loops inside the loop. The first loop from line 6 to line 12 is to update the state probabilities of each subsystem, and the second loop from line 13 to line 17 is to update the starvation probabilities and the blockage probabilities. Line 10 is a function to transfer system parameters to transition matrix of a subsystem. The update in line 16 can be achieved by Equation (4.12) and Equation (4.13). When the calculation for all the loops is completed, the state probabilities and performance measures of each subsystem (from SS_1 to SS_{D-1}) for each cycle (from $t = 1$ to $t = T$) are obtained. Then the transient performance measures of the entire multi-stage line are derived.

4.4.3 Comparison between Steady-State and Transient Analysis

Steady-state analysis is aimed at long-term performance. When the system reaches steady state soon and stays in steady state for a long time, the production during the transient stage is negligible. The long-term performance obtained from steady-state analysis can be used to estimate production capacity, make production plan, and

1: Initialize $X_i(1)$, $i = 1, \cdots, D-1$

2: Obtain $p_i^s(1)$, $i = 1, \cdots, D$

3: Obtain $p_i^b(1)$, $i = 1, \cdots, D$

4: Set the length of time: T

5: **for** $j = 1, T-1$ **do**

6: **for** $k = 1, D-1$ **do** ▷ Update state probability

7: $l = D - k$ ▷ Set index

8: $m = D - k + 1$

9: $I = [p_l, r_l, p_l^s(j), p_m, r_m, p_m^b(j), T_{l,max}, T_{l,min}, N_l]$

10: $Q_l(j) = getTransition(I)$

11: $X_l(j+1) = X_l(j)Q_l(j)$

12: **end for**

13: **for** $k = 1, D-1$ **do** ▷ Update starvation and blockage probabilities

14: $l = D - k$ ▷ Set index

15: $m = D - k + 1$

16: Update $p_m^s(j+1)$ and $p_l^b(j+1)$ from $X_l(j+1)$

17: **end for**

18: **end for**

Figure 4.6: Procedures of the Transient Analysis

conduct continuous improvement. Transient analysis is required, when production operates partially or entirely in the transient regime for reasons such as long cycle time, disruptions, etc. The system performance is not stable in transient stage and may be increasing, decreasing or fluctuating in this stage. The transient analysis is aimed at capturing such dynamics. For a simple discrete time Markov chain, system transition can be represented by a transition matrix. Steady-state analysis and transient analysis can be performed by manipulating the transition matrix. However, there is no single matrix that can model system transition for a multi-stage line, and thus the aggregation method is proposed to address the problem.

The procedure for steady-state analysis and the procedure for transient analysis are different from several aspects. First, it is assumed that there exist constant starvation probability, blockage probability and steady state probabilities in steady-state analysis, while in transient analysis those probabilities are changing over time. Second, starvation probability and blockage probability for each subsystem are unknown and initialized to zero in steady-state analysis, while the two probabilities are initially known from the initial system state in transient analysis. Third, the loop from line 4 to line 19 in Figure 4.4 represents the iterative procedure for steady-state analysis, and it converges from performance measures under the initial setting to the performance measures in steady state. The intermediate measures in the iterative procedure have no physical meaning. In contrast, the loop from line 5 to line 18 in Figure 4.6 is to obtain transient behavior. For any j in the loop, transient starvation probability, blockage probability, state probabilities and performance measures for cycle $(j+1)$ are derived. Fourth, the number of iterations for both methods are different. The backward aggregation and forward aggregation are performed *iter* times in steady-state analysis, while there is only one backward aggregation in transient analysis. Finally, the procedure of steady-state analysis cannot analyze transient

Table 4.1: Parameter Setting for Illustrative Example

i	1	2	3	4	5	6	7	8
e_i	0.62	0.75	0.85	0.73	0.68	0.91	0.81	0.72
r_i	0.35	0.44	0.37	0.24	0.27	0.35	0.29	0.46
N_i	5	6	7	7	7	5	6	6
$T_{i,max}$	8	8	10	10	8	7	9	9
$T_{i,min}$	1	1	2	2	1	1	2	1

behavior of a multi-stage line. In contrast, transient analysis can be used to obtain steady-state performance measures by running for a sufficiently large number of cycles as the system reaches steady state, but such a way is not computationally efficient.

4.5 Model Validation

4.5.1 An Illustrative Example

To evaluate the accuracy of the proposed analytical method, the results obtained from the proposed analytical method are compared with simulation. A MATLAB program is constructed to conduct the numerical experiment. First consider a single case with the parameters shown in Table 4.1.

Initially, all the machines are set to be up, and all the buffers are set to be empty. Both steady-state analysis and transient analysis of the aggregation method are performed. For the steady-state analysis of the aggregation method, 6 iterations are conducted to get the steady-state performance measures. Simulation repeats 10,000 times to obtain the average value and 95% confidence interval of each performance measure in each cycle. The result of the numerical study is shown in Figure 4.7. Simulated performance measures are plotted in solid blue line, with the shaded area indicating the 95% confidence interval. The red dashed lines represent the transient perfor-

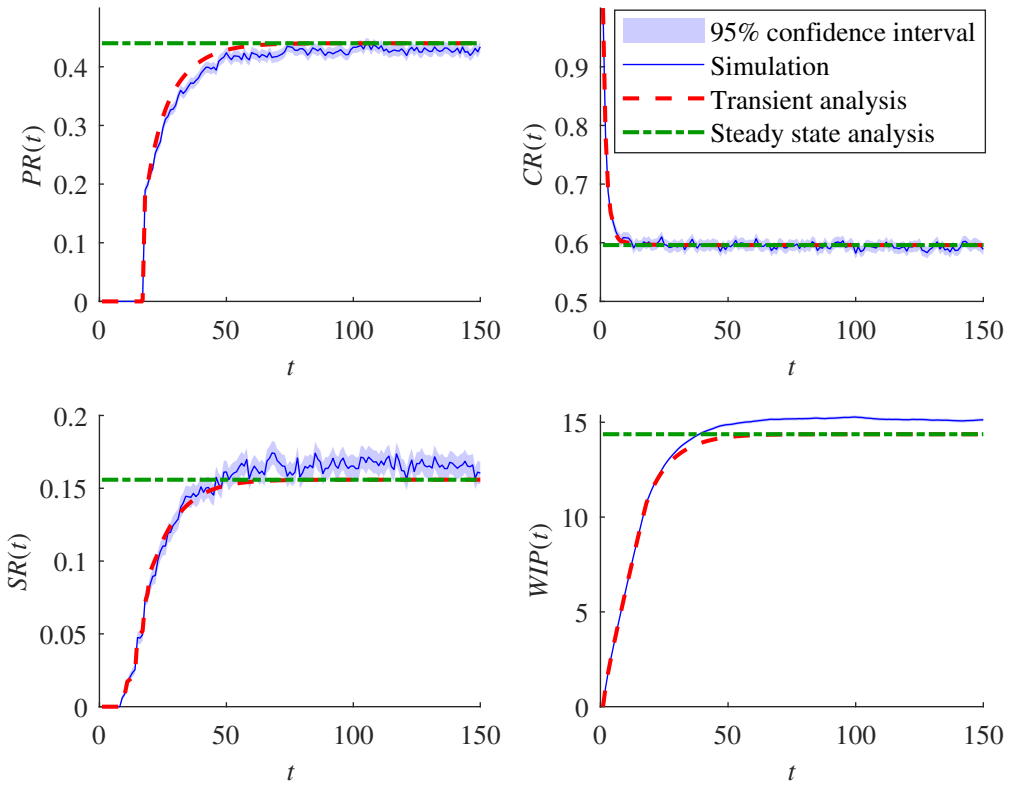


Figure 4.7: Comparison of Performance Measures from the Aggregation Approach and Simulation

mance measures obtained from transient analysis of the aggregation method. The green dash-dotted lines represent the steady-state performance measures obtained from steady-state analysis of the aggregation method. The result of the experiment suggests that the proposed aggregation method can capture both the steady-state and transient behaviors of the multi-stage line accurately.

As is shown in Figure 4.8, the blue solid lines represent the steady-state performance measures obtained from the simulation, while the green dash-dotted lines represent the estimated performance measures after each iteration of the aggregation method. The initial starvation probability and blockage probability for each machine

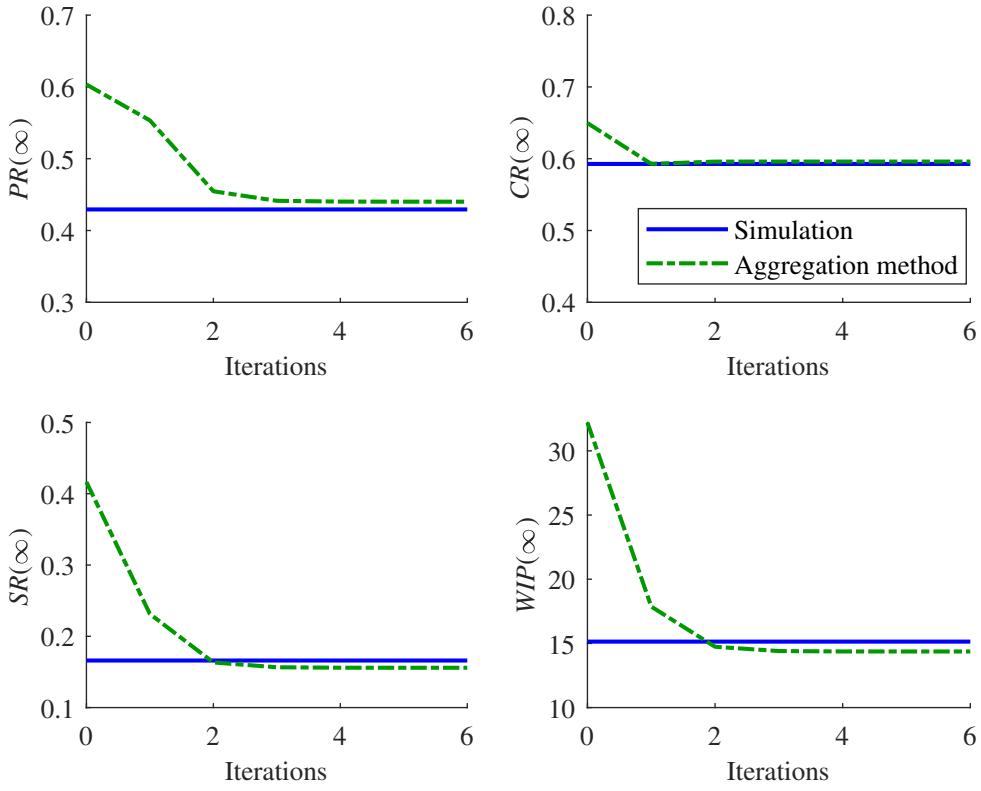


Figure 4.8: Steady-State Performance Measures Estimated in Each Iteration of the Aggregation Method

of the multi-stage line are set to be 0, and the estimated performance measures in iteration 0 in Figure 4.8 represent the estimated performance measures with the initial parameter setting. It suggests that the convergence can be achieved usually within three iterations, and converging performance measures are close to the true values.

4.5.2 Experiment with Random Parameters

To evaluate the accuracy of the proposed method in a more general sense, the experiment with random parameters is conducted. The estimated performance measures obtained through the aggregation method are compared with the ones estimated by

the simulation. Let T_{st} be the threshold of time where one can guarantee that the system in the simulation study can reach the steady state, and let T denote run length of the simulation. Denote by $PM(t)$ the true performance measures, like $PR(t)$, $CR(t)$, $SR(t)$ and $WIP(t)$, and they represented by average performance measures of all repeats of the simulation in cycle t. The true steady-state performance is obtained as follows.

$$PM(\infty) = \frac{1}{T - T_{st} + 1} \sum_{i=T_{st}}^{T} PM(i). \tag{4.14}$$

Denote the steady-state performance measures obtained through the aggregation method by $\widehat{PR}(\infty)$, $\widehat{CR}(\infty)$, $\widehat{SR}(\infty)$, $\widehat{WIP}(\infty)$. The absolute errors of the estimated steady-state performance measures, denoted by $\epsilon_{PR(\infty)}$, $\epsilon_{CR(\infty)}$, $\epsilon_{SR(\infty)}$ and $\epsilon_{WIP(\infty)}$, are provided as follows.

$$\epsilon_{PM(\infty)} = \left| PM(\infty) - \widehat{PM}(\infty) \right|, \tag{4.15}$$

where $\widehat{PM}(\infty)$ represents $\widehat{PR}(\infty)$, $\widehat{CR}(\infty)$, $\widehat{SR}(\infty)$, and $\widehat{WIP}(\infty)$, and $\epsilon_{PM(\infty)}$ represents $\epsilon_{PR(\infty)}$, $\epsilon_{CR(\infty)}$, $\epsilon_{SR(\infty)}$ and $\epsilon_{WIP(\infty)}$. The relative errors of the estimated steady-state performance measures are denoted by $\delta_{PR(\infty)}$, $\delta_{CR(\infty)}$, $\delta_{SR(\infty)}$ and $\delta_{WIP(\infty)}$. Let $\delta_{PM(\infty)}$ stand for $\delta_{PR(\infty)}$, $\delta_{CR(\infty)}$, $\delta_{SR(\infty)}$ and $\delta_{WIP(\infty)}$. The relative errors are defined as follows.

$$\delta_{PM(\infty)} = \frac{\epsilon_{PM(\infty)}}{PM(\infty)}. \tag{4.16}$$

Let T_{tr} be the time before which the transient behaviors of the aggregation method and the simulation are compared. The absolute errors of the estimated transient performance measures are denoted by $\epsilon_{PR(t)}$, $\epsilon_{CR(t)}$, $\epsilon_{SR(t)}$ and $\epsilon_{WIP(t)}$. Let $\epsilon_{PM(t)}$ stand for $\epsilon_{PR(t)}$, $\epsilon_{CR(t)}$, $\epsilon_{SR(t)}$ and $\epsilon_{WIP(t)}$. The absolute errors are defined as follows.

$$\epsilon_{PM(t)} = \frac{\sum_{i=1}^{T_{tr}} \left| PM(i) - \widehat{PM}(i) \right|}{T_{tr}}. \tag{4.17}$$

The relative errors of the estimated transient performance measures are denoted by $\delta_{PR(t)}$, $\delta_{CR(t)}$, $\delta_{SR(t)}$ and $\delta_{WIP(t)}$. Let $\delta_{PM(t)}$ stand for $\delta_{PR(t)}$, $\delta_{CR(t)}$, $\delta_{SR(t)}$ and $\delta_{WIP(t)}$. The relative errors are defined as follows.

$$\delta_{PM(t)} = \frac{\epsilon_{PM(t)}}{PM(\infty)}. \tag{4.18}$$

To test the aggregation method in both steady-state analysis and transient analysis, the parameter settings are randomly selected from the range shown as follows.

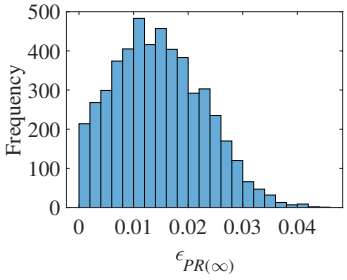
$$\begin{aligned}
D &\in \{4, 5, 6, 7, 8, 9, 10\}, \\
e_i &\in [0.60, 0.99] \qquad \text{for } i = 1, \cdots, D, \\
r_i &\in [0.20, 0.50] \qquad \text{for } i = 1, \cdots, D, \\
N_i &\in \{5, 6, 7\} \qquad \text{for } i = 1, \cdots, D-1, \\
T_{i,max} &\in \{N_i + 1, N_i + 2, N_i + 3\} \text{ for } i = 1, \cdots, D-1, \\
T_{i,min} &\in \{1, 2\} \qquad \text{for } i = 1, \cdots, D-1.
\end{aligned} \tag{4.19}$$

The number of machines is selected from the range $\{4, 5, 6, 7, 8, 9, 10\}$, and this can cover a large number of applications. Machine efficiency is commonly seen in the range of $[0.60, 0.99]$, from which the machine efficiency is randomly chosen in the experiment. The selection of N and $T_{i,min}$ and $T_{i,max}$ covers a large portion of applications. The steady state performance of a production system is worth analyzing, when the buffer is capable of supporting a smooth production and the majority of parts are finally produced with an acceptable quality. Thus, the repair probability is selected from $[0.20, 0.50]$. For the experiments of steady-state analysis, 5,000 random parameter settings are generated. In each parameter setting, the run length is $T = 100,000$, and 40 runs are carried out. The cycles after $T_{st} = 20,001$ are considered in the steady-state analysis. In the transient analysis, 5,000 parameter settings are randomly

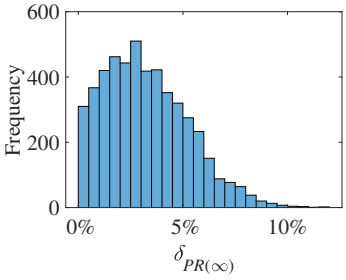
selected. In each parameter setting, the simulation runs $T = 2,000$ cycles and repeats 10,000 times. In addition, set $T_{st} = 1,001$ and $T_{tr} = 400$.

The accuracy of the steady-state analysis and transient analysis is shown in Figure 4.9 and Figure 4.10, respectively. The overall consumption rate can be estimated accurately. The median of the relative error of overall consumption rate is 0.81% in both steady-state analysis and transient analysis, and the relative errors are less than 2.0% for most cases. The estimates of overall production rate and work-in-process have higher errors. For most cases, it can maintain relative errors smaller than 6%, which is also acceptable. The overall scrap rate has the largest error. The median of relative error is 4.1% for steady-state analysis and 3.9% for transient analysis. The absolute error of scrap rate is small, and the large relative error is partially due to the small denominator. The experiment with random parameters suggests that the proposed analytical method, combining the approximate modeling of residence time and the aggregation method, can estimate performance measures of a multi-stage line in high accuracy.

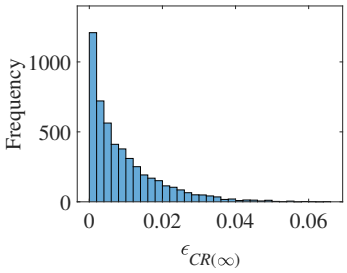
Both the simulation and the aggregation method are developed with MATLAB and run on a computer with Intel(R) Core(TM) i7-8700 CPU, 16 GB RAM, and 64-bit Windows 10 Enterprise operating system. The average time to perform simulation for a parameter setting in steady-state analysis is 39.63 seconds, and it only takes 0.13 seconds on average to perform steady-state analysis of the aggregation method. It shows that the aggregation method is more efficient. In transient analysis, the simulation takes 219.11 seconds on average for each parameter setting, and the aggregation method takes 27.36 seconds. The transient analysis of the aggregation method takes more time than the steady-state analysis, but it is still more efficient than the simulation.



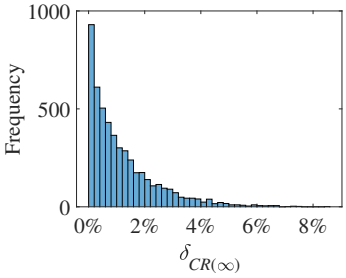
(a) AE of Overall Production Rate



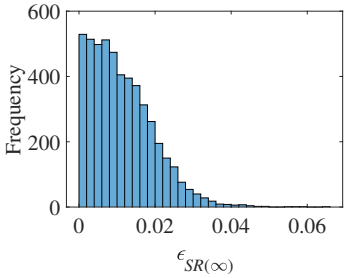
(b) RE of Overall Production Rate



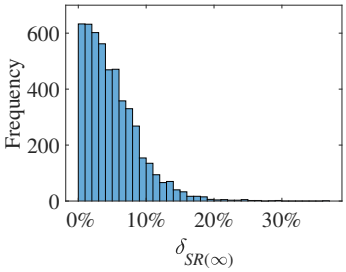
(c) AE of Overall Consumption Rate



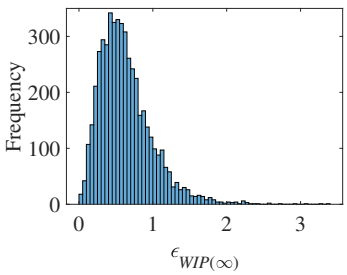
(d) RE of Overall Consumption Rate



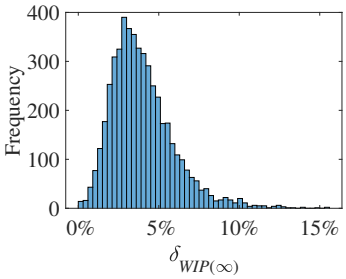
(e) AE of Overall Scrap Rate



(f) RE of Overall Scrap Rate

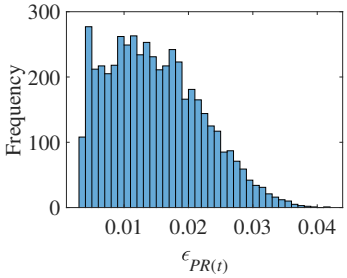


(g) AE of Overall Work-In-Process

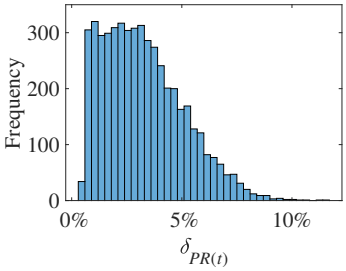


(h) RE of Overall Work-In-Process

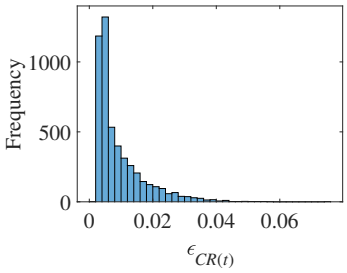
Figure 4.9: Accuracy of the Steady-State Analysis. AE: Absolute Error; RE: Relative Error



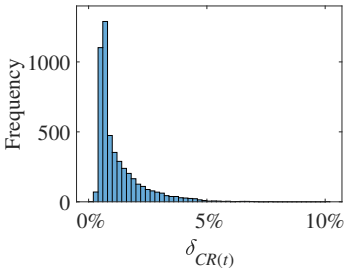
(a) AE of Overall Production Rate



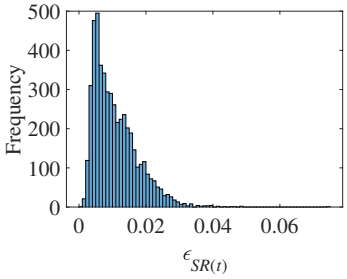
(b) RE of Overall Production Rate



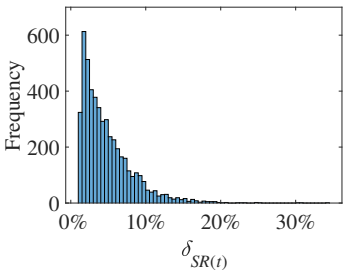
(c) AE of Overall Consumption Rate



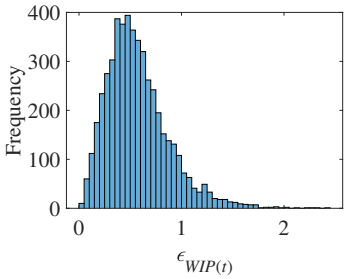
(d) RE of Overall Consumption Rate



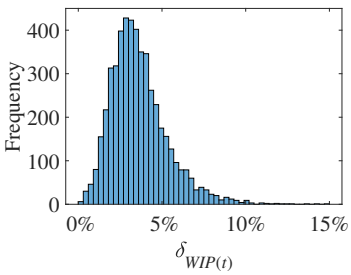
(e) AE of Overall Scrap Rate



(f) RE of Overall Scrap Rate



(g) AE of Overall Work-In-Process



(h) RE of Overall Work-In-Process

Figure 4.10: Accuracy of the Transient Analysis. AE: Absolute Error; RE: Relative Error

Chapter 5

DECOMPOSITION-BASED REAL-TIME CONTROL OF MULTI-STAGE SERIAL LINES

5.1 Problem Formulation

5.1.1 System Description and Assumptions

For simplicity purpose, the term "multi-stage line" is used to represent the multi-stage Bernoulli serial line with residence time constraints for the rest of the chapter. The multi-stage line under study is shown in Figure 5.1. Parts visit each machine and buffer from the left side to the right side, until they finish all the processes or get scrapped from the system. The following assumptions define the machines, the buffers, and their interactions.

(i) The multi-stage line consists of D machines, denoted by $m_1, m_2, \cdots, m_D$, and $(D-1)$ buffers, denoted by $B_1, B_2, \cdots, B_{D-1}$, where $D > 2$.

(ii) All machines are synchronized with a constant processing time (cycle time), which is the time to process a single part on a machine.

(iii) Machines are subject to failures, and each machine is assumed to be an independent Bernoulli machine. The state of a machine is determined at the beginning of a cycle. Before that, the state of machine m_i in cycle t, for $i = 1, \cdots, D$ and $t = 1, 2, \cdots$, is a random variable, denoted by $S_i(t)$, following the Bernoulli distribution with parameter p_i. Specifically, machine m_i is capable of producing a part in cycle t with probability p_i and fails to do so with probability $(1 - p_i)$.

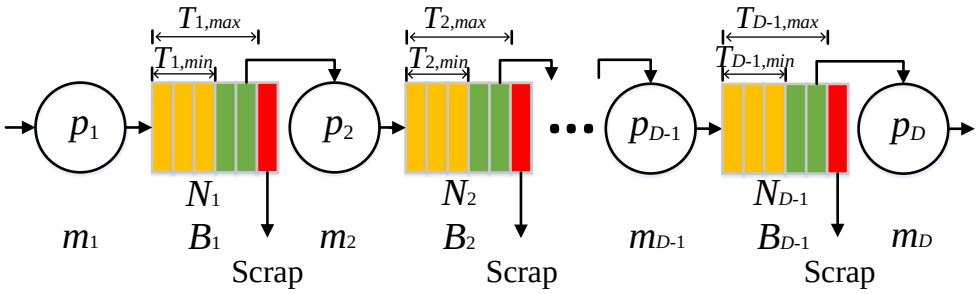


Figure 5.1: Illustration of the Multi-Stage Line with Residence Time Constraints

It can be represented by $P(S_i(t) = 1) = p_i$ and $P(S_i(t) = 0) = 1 - p_i$. At the beginning of cycle t, the machine state is realized, and the realized machine state is denoted by $s_i(t) \in \{0, 1\}$.

(iv) Buffer B_i has a finite capacity N_i $(1 \leq N_i < \infty)$, for $i = 1, 2, \cdots, D-1$, and its buffer occupancy is determined at the end of a cycle and denoted by n_i. First-in-first-out (FIFO) policy is assumed regarding the buffer outflow process.

(v) Each part in a buffer has its residence time, and it is counted as the number of cycles, for which the part has been staying in the buffer. Residence time of a part is determined at the end of a cycle and starts with 0 as the part enters a buffer at the end of a cycle. Residence time of a part in a buffer increases by one each cycle, if the part keeps staying in the same buffer. Let $\tau_{i,j}$ denote the residence time of the jth part in buffer B_i, if such a part exists.

(vi) The maximum allowable residence time for a part in buffer B_i is characterized by $T_{i,max}$, for $i = 1, 2, \cdots, D-1$. A part in buffer B_i will be scrapped when its residence time reaches $T_{i,max}$. Let $T_{i,max} \geq N_i$, otherwise N_i has no effect on the multi-stage line.

(vii) The minimum required residence time for a part in buffer B_i is denoted by $T_{i,min}$, for $i = 1, 2, \cdots, D-1$. A part in buffer B_i is allowed to be processed by machine m_{i+1} only when its residence time reaches or exceeds $T_{i,min}$.

(viii) Machine m_i, for $i = 1, 2, \cdots, D-1$, is blocked during a cycle, if (a) machine m_i is up, (b) buffer B_i is full, (c) machine m_{i+1} does not produce a part in this cycle due to machine failure or blockage, and (d) there will be no part scrapped from buffer B_i. Machine m_D is never blocked. In addition, block-before-service policy is assumed.

(ix) Machine m_i, for $i = 2, \cdots, D$, is starved during a cycle, if machine m_i is up, and no part in buffer B_{i-1} has residence time greater than or equal to $T_{i-1,min}$. Machine m_1 is never starved.

(x) At the end of each cycle, a machine can be stopped to prevent it from producing in the next cycle. One can also have a machine unchanged, and thus the machine will work as a Bernoulli machine in the next cycle. It is always beneficial not to stop the last machine, so only actions on machine m_i, for $i = 1, 2, \cdots, D-1$, are considered. Let $a_i(t) \in \{1, 0\}$, for $i = 1, 2, \cdots, D-1$ and $t = 0, 1, \cdots$, denote the action on machine m_i at the end of cycle t. The action $a_i(t) = 0$ makes machine m_i not work in cycle $(t+1)$. The action $a_i(t) = 1$ represents that machine m_i is unchanged. The action on the whole system is represented by $\boldsymbol{a}(t) = [a_1(t) \quad a_2(t) \quad \cdots \quad a_{D-1}(t)]^T$. The action space is denoted by $\mathcal{A} = \{0, 1\}^{D-1}$.

5.1.2 Performance Measures

To evaluate the multi-stage line, the performance measures of interest are introduced as follows.

- *Production rate of machine m_i, $PR_i(t)$,* for $t = 1, 2, \cdots$ and $i = 1, \cdots, D$: the expected number of parts produced by machine m_i in cycle t;

- *Scrap rate of buffer B_i, $SR_i(t)$,* for $t = 1, 2, \cdots$ and $i = 1, \cdots, D-1$: the expected number of scrapped parts from buffer B_i in cycle t;

- *Scrap rate of the multi-stage line, $SR(t)$* for $t = 1, 2, \cdots$: the expected number of scrapped parts from the multi-stage line in cycle t.

Remark 1. *Both Ju* et al. *(2017b) and Zhang* et al. *(2013) study Bernoulli lines and are consistent with the early work (Li and Meerkov, 2009) in the problem formulation of Bernoulli lines. Ju* et al. *(2017b) and Zhang* et al. *(2013) have a small difference in the definition of performance measures due to the concern of transient analysis. In Ju* et al. *(2017b), performance measures in one cycle are observed in the current cycle and derived from the system state of the previous cycle. In Zhang* et al. *(2013), performance measures in one cycle are derived from the system state in the current cycle and can be observed at the end of the next cycle. One cycle lag of performance measures is the only difference between the two studies. In this chapter, the definition from Ju* et al. *(2017b) is followed.*

Remark 2. *Scrap rate of the multi-stage line is the summation of scrap rates of all buffers. Thus, $SR(t) = \sum_{i=1}^{D-1} SR_i(t)$ for all t.*

For the system under consideration, it is desired to maximize production rate $PR_D(t)$ and minimize scrap rate $SR(t)$ simultaneously. The objective of the study is therefore to maximize $(PR_D(t) - \omega SR(t))$ through the actions defined in assumption (x), where ω is a positive constant to balance the trade-off between production rate $PR_D(t)$ and scrap rate $SR(t)$.

5.1.3 System Dynamics and Optimization Model

Let $\mathcal{H}_i$, for $i = 1, 2, \cdots, D-1$, be a collection of all subsets of set $\{0, 1, \cdots, T_{i,max}-1\}$ that have cardinality smaller than or equal to N_i. Specifically,

$$\mathcal{H}_i = \{h \mid h \subset \{0, 1, \cdots, T_{i,max} - 1\} \text{ and } |h| \leq N_i\}, \tag{5.1}$$

for $i = 1, 2, \cdots, D-1$, which is the state space for buffer B_i. $H_i(t) \in \mathcal{H}_i$, for $t = 0, 1, \cdots$, is defined to be the state of buffer B_i at the end of cycle t, and $H_i(t)$ represents a set of residence times of parts in buffer B_i. The occupancy of buffer B_i at the end of cycle t can be represented by $n_i = |H_i(t)|$. Follow the convention that machine state is determined at the beginning of a cycle, buffer state is determined at the end of a cycle, and the system state is determined at the end of a cycle. Thus, system state is represented by states of all buffers. The state of a multi-stage line at the end of cycle t can be defined by $H(t) = (H_1(t), H_2(t), \cdots, H_{D-1}(t))$, which belongs to the state space of the multi-stage line, denoted by $\mathcal{H} = \otimes_{i=1}^{D-1}\mathcal{H}_i$. In addition, define two other collections as follows.

$$\mathcal{H}_{i,min} = \{H_i \in \mathcal{H}_i \mid \sup H_i \geq T_{i,min}\}, \tag{5.2}$$

$$\mathcal{H}_{i,max} = \{H_i \in \mathcal{H}_i \mid \sup H_i = T_{i,max} - 1\}, \tag{5.3}$$

for $i = 1, 2, \cdots, D-1$. If buffer B_i is not empty, $\sup H_i$ is equal to the residence time of the first part in the buffer and $\tau_{i,1} = \sup H_i$. If buffer B_i is empty, then the set H_i is empty and $\sup H_i = -\infty$. $\mathcal{H}_{i,min}$ is a collection of states of buffer B_i that the first part in the buffer has residence time greater than or equal to $T_{i,min}$, while $\mathcal{H}_{i,max}$ is a collection of states of buffer B_i that the first part in the buffer has residence time equal to $(T_{i,max} - 1)$.

A discounted infinite horizon dynamic optimization problem is considered, and the objective is to maximize discounted cumulative production rate while minimizing

scrap rate in the long term. Given the known initial state $H(0)$, the objective function is

$$\max \quad E\left\{\sum_{t=1}^{\infty}\lambda^{t-1}\left(\tilde{PR}_D\left(t\right)-\omega\tilde{SR}\left(t\right)\right)\right\}, \tag{5.4}$$

where $\tilde{PR}_i\left(t\right)$, for $i=1,2,\cdots,D$, $\tilde{SR}_i(t)$, for $i=1,2,\cdots,D-1$, and $\tilde{SR}(t)$ are random variables, and $\lambda\in[0,1)$ is the discount factor. Then, $PR_i(t)=E\left[\tilde{PR}_i\left(t\right)\right]$, $SR_i(t)=E\left[\tilde{SR}_i\left(t\right)\right]$ and $SR(t)=E\left[\tilde{SR}\left(t\right)\right]$. Start with machine m_D to formulate the system dynamics. The production and scrap of the last machine at time $(t+1)$, for $t=0,1,\cdots$, are represented as follows.

$$\chi_{\mathcal{H}_{D-1,min}}(H_{D-1}(t))S_D(t+1)=\tilde{PR}_D(t+1), \tag{5.5}$$

$$\chi_{\mathcal{H}_{D-1,max}}(H_{D-1}(t))(1-S_D(t+1))=\tilde{SR}_{D-1}(t+1), \tag{5.6}$$

where a characteristic function $\chi_X(x)$ is used. Specifically,

$$\chi_X(x)=\begin{cases}1 & \text{if } x\in X,\\ 0 & \text{otherwise.}\end{cases} \tag{5.7}$$

Equation (5.5) means that machine m_D will finish producing a part at the end of cycle $(t+1)$, if there is at least a part in buffer B_{D-1} with residence time greater than or equal to $T_{D-1,min}$ at the end of cycle t and machine m_D is up during cycle $(t+1)$. Equation (5.6) represents that a part will be scrapped from buffer B_{D-1} if there exists a part in buffer B_{D-1} with residence time equal to $(T_{D-1,max}-1)$ at the end of cycle t and machine is down during cycle $(t+1)$. Then, the state of buffer B_{D-1} is updated as follows.

$$H'_{D-1}\left(t\right)=\begin{cases}H_{D-1}(t) & \text{if } \tilde{PR}_D(t+1)+\tilde{SR}_{D-1}(t+1)=0,\\ H_{D-1}(t)\setminus\sup H_{D-1}(t) & \text{otherwise,}\end{cases} \tag{5.8}$$

$$H''_{D-1}(t)=F\left(H'_{D-1}(t)\right), \tag{5.9}$$

for $t = 0, 1, \cdots$. Equation (5.8) suggests that the part with the largest residence time in buffer B_{D-1} is removed if a part in this buffer is either produced or scrapped. In Equation (5.9), an operator $F()$ on set is introduced. For two sets X and X' such that $X' = F(X)$, $x + 1 \in X'$ is satisfied for any element $x \in X$, and $x - 1 \in X$ is satisfied for any element $x \in X'$. Equation (5.9) means that the residence time of each part increases by one.

In a similar way, the production rate and scrap rate of machine m_{i+1}, for $i = 1, 2, \cdots, D-2$, are expressed as follows.

$$\chi_{\mathcal{H}_{i,min}}(H_i(t))\,\chi_{\mathbb{R}_{>0}}\left(N_{i+1} - |H'_{i+1}(t)|\right) S_{i+1}(t+1)\,a_{i+1}(t) = \tilde{PR}_{i+1}(t+1), \quad (5.10)$$

$$\chi_{\mathcal{H}_{i,max}}(H_i(t))\left(1 - \chi_{\mathbb{R}_{>0}}\left(N_{i+1} - |H'_{i+1}(t)|\right) S_{i+1}(t+1)\,a_{i+1}(t)\right) = \tilde{SR}_i(t+1), \quad (5.11)$$

for $t = 0, 1, \cdots$. If machine m_{i+1} finishes producing a part at the end of cycle $(t+1)$, suggested by Equation (5.10), four conditions should be met. First, there is at least one part in buffer B_i with residence time greater than or equal to $T_{i,min}$ at the end of cycle t. Second, there is no blockage in buffer B_{i+1}. Third, machine m_{i+1} is up during cycle $(t+1)$. Last, machine m_{i+1} is not turned down. If there is one part in buffer B_i with residence time equal to $(T_{i,max} - 1)$ and at least one of the last three conditions above is not satisfied, then a part is scrapped from buffer B_i, suggested by Equation (5.11). Then, update the states of those buffers, shown in Equation (5.12) and Equation (5.13) below.

$$H'_i(t) = \begin{cases} H_i(t) & \text{if } \tilde{PR}_{i+1}(t+1) + \tilde{SR}_i(t+1) = 0, \\ H_i(t) \setminus \sup H_i(t) & \text{otherwise,} \end{cases} \quad (5.12)$$

$$H''_i(t) = F(H'_i(t)), \quad (5.13)$$

for $i = 1, 2, \cdots, D-2$ and $t = 0, 1, \cdots$. Equation (5.12) and Equation (5.13) are similar to Equation (5.8) and Equation (5.9), respectively. If a machine produces

a part, the number of parts in its downstream buffer will increase by one. After considering both inflow and outflow of a buffer, one can determine the state of a buffer in the next cycle as follows.

$$H_{i+1}(t+1) = \begin{cases} H''_{i+1}(t) & \text{if } \tilde{PR}_{i+1}(t+1) = 0, \\ H''_{i+1}(t) \cup \{0\} & \text{otherwise,} \end{cases} \tag{5.14}$$

for $i = 1, 2, \cdots, D-2$ and $t = 0, 1, \cdots$. Equation (5.14) suggests that a new part with residence time equal to 0 is added to buffer B_{i+1} at the end of cycle $(t+1)$ if machine m_{i+1} successfully produces a part at the end of cycle $(t+1)$. In addition,

$$\chi_{\mathbb{R}_{>0}}\left(N_1 - |H'_1(t)|\right) S_1(t+1)\, a_1(t) = \tilde{PR}_1(t+1), \tag{5.15}$$

$$H''_1(t) = F\left(H'_1(t)\right), \tag{5.16}$$

$$H_1(t+1) = \begin{cases} H''_1(t) & \text{if } \tilde{PR}_1(t+1) = 0 \\ H''_1(t) \cup \{0\} & \text{otherwise,} \end{cases} \tag{5.17}$$

for $t = 0, 1, \cdots$. Equation (5.15), Equation (5.16) and Equation (5.17) are for the first machine and first buffer, and they are similar to Equation (5.10), Equation (5.9) and Equation (5.14), respectively. Finally, by Remark 2,

$$\tilde{SR}(t+1) = \sum_{i=1}^{D-1} \tilde{SR}_i(t+1), \tag{5.18}$$

for $t = 0, 1, \cdots$.

5.2 Decomposition-Based Control Framework

5.2.1 Complexity of Multi-Stage Line

The production control problem introduced in Section 5.1 is not able to be analyzed directly due to the large state space. The total number of system states of a multi-stage line, denoted by M, is provided as follows.

$$M = \prod_{i=1}^{D-1} \sum_{j=0}^{N_i} \begin{pmatrix} T_{i,max} \\ j \end{pmatrix}. \tag{5.19}$$

Consider a single buffer first. If the buffer occupancy is fixed to be j, the number of combinations for a buffer is equal to the number of ways to choose j different residence times from $T_{i,max}$ options, which is represented by $\begin{pmatrix} T_{i,max} \\ j \end{pmatrix}$. Then, the total number of system states can be calculated by considering all buffers and all possible buffer occupancy. For example, for a multi-stage line that has 7 machines and 6 buffers with buffer capacity $N_i = 6$ and maximum allowable residence time $T_{i,max} = 8$, for $i = 1, 2, \cdots, 6$, the number of system states is as large as 2.3×10^{14} according to Equation (5.19). To deal with this level of complexity, one common approach is to use reinforcement learning to perform production control by approximately mapping system states and actions to rewards. However, these methods result in a long time training and suffer from interpretability. In addition, the approximation architecture could deteriorate soon as the problem scale continues to increase. To tackle these issues, decomposition-based control, a novel approach, is proposed. Hypothesize that, by leveraging the system decomposition, production performance can be effectively optimized in real time.

5.2.2 Overview of the Decomposition-Based Control Approach

Instead of analyzing and controlling a multi-stage line as a whole, the decomposition-based control is proposed. The concept of the decomposition-based control is shown in Figure 5.2. A multi-stage line is decomposed into subsystems, and structural relationship between subsystems is defined. Under a properly defined structural relationship, each subsystem is assumed to behavior like its corresponding part in the multi-stage line. Each subsystem is modeled independently as an MDP model. Since the state

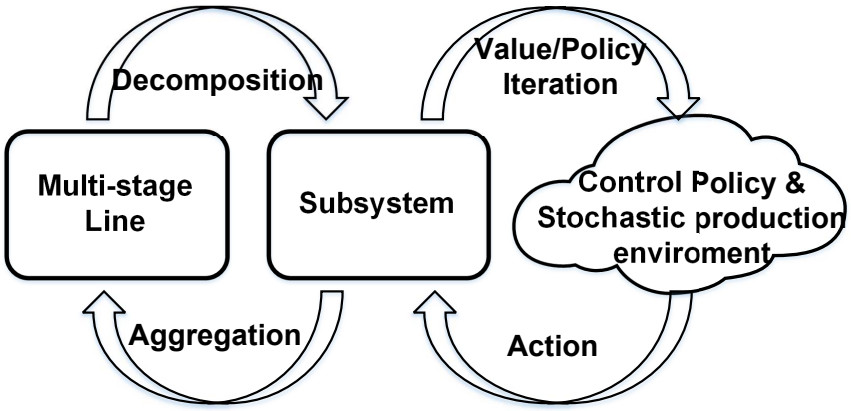
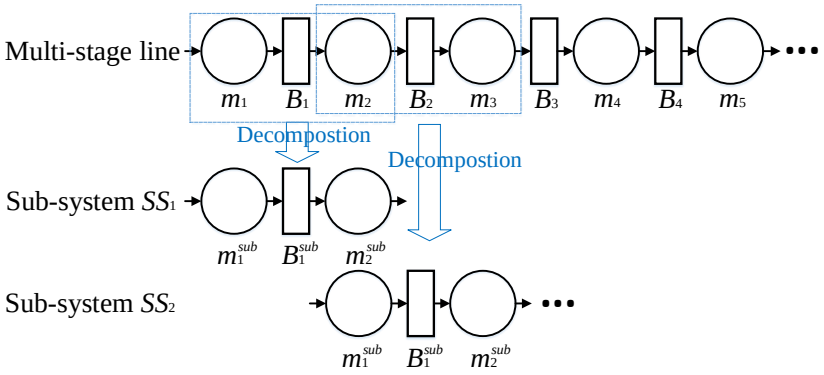


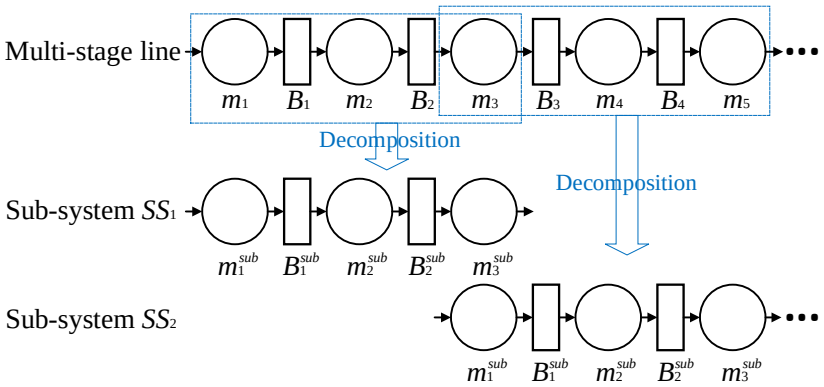
Figure 5.2: Concept of Decomposition-Based Control

space of a subsystem is small enough, the control policy for each subsystem can be derived through value iteration or policy iteration. Each subsystem takes action by observing its local environment. The control policy of a multi-stage line is a combination of all control policies derived from all subsystems. However, as a control policy is implemented, the original structural relationship between subsystems changes. Due to this change, the behavior of a subsystem does not truly represents its corresponding part in the multi-stage line. It requires subsystems to update their relationship according to the current control policy, which is part of the aggregation procedure. A new iteration starts, since the current control policy may not be optimal as the relationship between subsystems is updated. The MDP model for each subsystem with updated relationship is developed, and new control policy is derived. After several iterations, this process converges, and each subsystem has a similar behavior as its corresponding part in the multi-stage line. Control policy for each subsystem can achieve a global improvement.

In the following subsections, the system decomposition and modeling of subsystems will be introduced in details, and a novel aggregation based procedure will be provided to generate the control policy.



(a) Two-Machine-One-Buffer Subsystems



(b) Three-Machine-Two-Buffer Subsystems

Figure 5.3: System Decomposition with Two- or Three-Machine Subsystems

5.2.3 System Decomposition

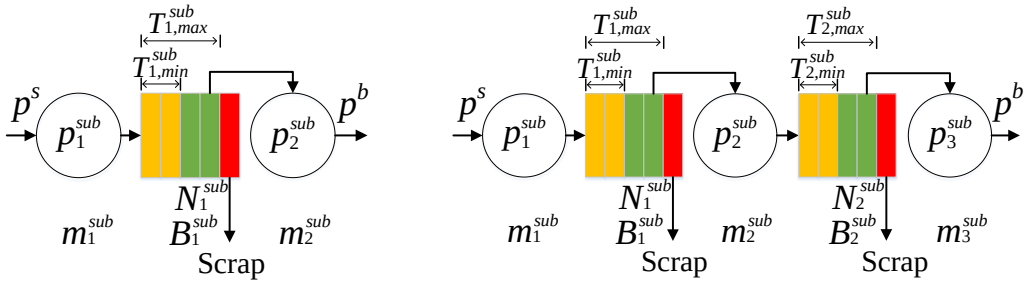
A subsystem, isolated from a multi-stage line, serves as a building block to support the decomposition-based control, and it can be a two-machine-one-buffer subsystem or a three-machine-two-buffer subsystem, shown in Figure 5.3. Figure 5.3a shows how a multi-stage line with D machines is decomposed into $(D-1)$ two-machine-one-buffer subsystems. Each subsystem consists of two machines, m_1^{sub} and m_2^{sub}, and a buffer B_1^{sub}. The ith subsystem of a multi-stage line is denoted by SS_i, for $i = 1, 2 \cdots, D-1$. The control model for a two-machine-one-buffer subsystem is to determine when to

turn machine m_1^{sub} down based on the state of the subsystem. Figure 5.3b shows how a multi-stage line with D machines is decomposed into $(D-1)/2$ three-machine-two-buffer subsystems. Each three-machine-two-buffer subsystem consists of three machines, denoted by m_1^{sub}, m_2^{sub} and m_3^{sub}, and two buffers, denoted by B_1^{sub} and B_2^{sub}. Similar to a two-machine-one-buffer subsystem, the control model for a three-machine-two-buffer subsystem is to control machine m_1^{sub} and m_2^{sub} according to the state of the subsystem.

Decompose a serial line into subsystems, since the system as a whole is infeasible to analyze directly due to its large state space. Decomposition, as an approximation based method, can compromise modeling accuracy, when too many subsystems are involved. Also, to consider the complexity of the subsystem itself, the balance is found by using three-machine-two-buffer subsystem as a general building block in the decomposition method. One two-machine-one-buffer subsystem will be utilized to handle the systems with even number of machines.

5.2.4 Descriptive Model of Subsystem

The relevant parameters to model a subsystem are presented in Figure 5.4. The ith machine in a subsystem is denoted by m_i^{sub}, and it is a Bernoulli machine with parameter p_i^{sub}. The ith buffer in a subsystem is denoted by B_i^{sub}. Buffer B_i^{sub} is described by buffer capacity N_i^{sub}, maximum allowable residence time $T_{i,max}^{sub}$ and minimum required residence time $T_{i,min}^{sub}$. Neighboring subsystems are mutually influenced. Such influence is modeled by starvation probability, p^s, and blockage probability, p^b. The probability that machine m_1^{sub} is not able to produce due to the starvation from its upstream buffer is denoted by p^s. If buffer B_1^{sub} has available space, the probability that the first machine can produce is $p_1^{sub}(1-p^s)$. Machine m_2^{sub} in a two-machine-one-buffer subsystem and machine m_3^{sub} in a three-machine-two-buffer subsystem are



(a) A Two-Machine-One-Buffer Subsystem (b) A Three-Machine-Two-Buffer Subsystem

Figure 5.4: Models for Subsystems

shared by its downstream subsystem, which is illustrated in Figure 5.3. The probability p^b represents the probability that machine m_2^{sub} in a two-machine-one-buffer subsystem or machine m_3^{sub} in a three-machine-two-buffer subsystem is not allowed to work due to downstream blockage or the control policy of the downstream subsystem. Thus, if there is at least one part in buffer B_1^{sub} of a two-machine-one-buffer subsystem or buffer B_2^{sub} of a three-machine-two-buffer subsystem with residence time larger than or equal to the minimum required residence time, the probability that the part can be produced and leave the subsystem is $p_2^{sub}(1-p^b)$ and $p_3^{sub}(1-p^b)$ for two-machine-one-buffer subsystem and three-machine-two-buffer subsystem, respectively. Assumption (ix) suggests that machine m_1 is never starved, so p^s is always equal to 0 for the first subsystem. Similarly, the last machine of the last subsystem is never blocked, and thus p^b in the last subsystem is always equal to 0.

5.2.5 Markov Decision Model for Subsystem

When a two-machine-one-buffer subsystem is isolated, the subsystem can be viewed as a two-machine serial line with two Bernoulli machines with parameters $p_1^{sub}(1-p^s)$

and $p_2^{sub}(1-p^b)$, respectively. Similarly, a three-machine-two-buffer subsystem can be viewed as a three-machine serial line with three Bernoulli machines with parameters $p_1^{sub}(1-p^s)$, p_2 and $p_3^{sub}(1-p^b)$, respectively. The modeling of two-machine-one-buffer subsystem shares similarities with the modeling of three-machine-two-buffer subsystem. In this subsection, only how to model a three-machine-two-buffer subsystem is shown without repeating it for two-machine-one-buffer subsystem.

- Decision epochs: $t = 0, 1, \cdots$.

- System state: $h^{sub}(t) = (n_1^{sub}, \tau_1^{sub}, n_2^{sub}, \tau_2^{sub}) \in \mathcal{H}^{sub}$. n_1^{sub} and n_2^{sub} are buffer occupancy of buffer B_1^{sub} and buffer B_2^{sub}, respectively. τ_1^{sub} and τ_2^{sub} are residence time of the first part in buffer B_1^{sub} and buffer B_2^{sub}, respectively, if the buffer is not empty. Let $\tau_i^{sub} = 0$, for $i = 1, 2$, if $n_i^{sub} = 0$. The state space of a subsystem is denoted by $\mathcal{H}^{sub}$.

- Action: $\boldsymbol{a}^{sub}(t) = \begin{bmatrix} a_1^{sub}(t) & a_2^{sub}(t) \end{bmatrix}^T \in \mathcal{A}^{sub}$, where $a_i^{sub}(t) \in \{1, 0\}$, for $i = 1, 2$, at any time t. The action $a_i^{sub}(t) = 0$ makes machine m_i^{sub} not work in cycle $(t+1)$, and the action $a_i^{sub}(t) = 1$ keeps machine m_i^{sub} unchanged. The action space of a subsystem is denoted by $\mathcal{A}^{sub}$.

- Reward: the reward at time $(t-1)$ is denoted by $r\left(h^{sub}(t-1), \boldsymbol{a}^{sub}(t-1)\right)$. Specifically,

$$r\left(h^{sub}(t-1), \boldsymbol{a}^{sub}(t-1)\right) = \tilde{PR}^{sub}(t) - \omega \tilde{SR}^{sub}(t), \tag{5.20}$$

  where $\tilde{PR}^{sub}(t)$ and $\tilde{SR}^{sub}(t)$ are random variables representing the production of the last machine m_3^{sub} and the scrap from both buffers, respectively.

- Expected total discounted reward of policy π^{sub}:

$$v^{\pi^{sub}} = E^{\pi^{sub}}\left\{ \sum_{t=0}^{\infty} \lambda^t r\left(h^{sub}(t), \boldsymbol{a}^{sub}(t)\right) \right\}, \tag{5.21}$$

where $\lambda \in [0, 1)$ is the discount.

The optimal control policy of a subsystem can be expressed as

$$\pi^* \in \arg\max_{\pi^{sub}} E^{\pi^{sub}} \left\{ \sum_{t=0}^{\infty} \lambda^t r \left(h^{sub}(t), \boldsymbol{a}^{sub}(t) \right) \right\}. \tag{5.22}$$

Remark 3. *The state of a buffer is defined only by the buffer occupancy and the residence time of the first part in the buffer. Following the approximate method detailed in Ju* et al. *(2017b), the optimization problem can be treated as an MDP with an exact stochastic model, and standard methods, such as the value iteration and the policy iteration, can be used to solve the problem.*

Let $\pi^{sub} : \mathcal{H}^{sub} \to \mathcal{A}^{sub}$ be a mapping from state to action under control policy π^{sub}. As control policy π^{sub} is implemented, the subsystem reaches steady state. Let $\mu : \mathcal{H}^{sub} \to [0, 1]$ be a mapping from state to its steady-state probability under control policy π^{sub}. $\widehat{PR}^{sub}$, $\widehat{SR}^{sub}$, $\widehat{ST}^{sub}$ and $\widehat{BL}^{sub}$ denote the estimated long-term performance measures of the subsystem under control policy π^{sub}, and they are defined and derived as follows.

- *Estimated production rate* $\widehat{PR}^{sub}$: the expected number of parts produced by the last machine of the subsystem in a cycle, and specifically,

$$\widehat{PR}^{sub} = \sum_{h^{sub} \in \mathcal{H}_{PR}^{sub}} \mu \left(h^{sub} \right) p_3^{sub} \left(1 - p^b \right), \tag{5.23}$$

  where

$$\mathcal{H}_{PR}^{sub} = \left\{ h^{sub} \in \mathcal{H}^{sub} \mid \tau_2^{sub} \geq T_{2,min}^{sub} \right\}. \tag{5.24}$$

  The subset of state space, $\mathcal{H}_{PR}^{sub}$, represents all states where the residence time of the first part in buffer B_2^{sub} is equal to or larger than $T_{2,min}^{sub}$. It is suggested by Equation (5.23) that one part can be produced for a subsystem in a state in $\mathcal{H}_{PR}^{sub}$ if machine m_3^{sub} is up and there is no blockage to the machine.

- *Estimated scrap rate* $\widehat{SR}^{sub}$: the expected number of scrapped parts from the subsystem in a cycle, and specifically,

$$\begin{aligned}\widehat{SR}^{sub} = & \sum_{h^{sub}\in\mathcal{H}^{sub}_{SR,1}} \mu\left(h^{sub}\right)\left(1-\begin{bmatrix}0 & p_2^{sub}\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\right) \\ & + \sum_{h^{sub}\in\mathcal{H}^{sub}_{SR,2}} \mu\left(h^{sub}\right)\begin{bmatrix}0 & p_2^{sub}\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\left(1-p_3^{sub}\left(1-p^b\right)\right) \\ & + \sum_{h^{sub}\in\mathcal{H}^{sub}_{SR,3}} \mu\left(h^{sub}\right)\left(1-p_3^{sub}\left(1-p^b\right)\right),\end{aligned} \tag{5.25}$$

where

$$\mathcal{H}^{sub}_{SR,1} = \left\{h^{sub}\in\mathcal{H}^{sub} \mid \tau_1^{sub} = T^{sub}_{1,max}-1\right\}, \tag{5.26}$$

$$\mathcal{H}^{sub}_{SR,2} = \left\{h^{sub}\in\mathcal{H}^{sub} \mid \tau_1^{sub} = T^{sub}_{1,max}-1, n_2^{sub}=N_2^{sub}, \tau_2^{sub} < T^{sub}_{2,max}-1\right\}. \tag{5.27}$$

$$\mathcal{H}^{sub}_{SR,3} = \left\{h^{sub}\in\mathcal{H}^{sub} \mid \tau_2^{sub} = T^{sub}_{2,max}-1\right\}. \tag{5.28}$$

Equation (5.25) is the summation of three terms. The first term represents the case that a part is scrapped from buffer B_1^{sub} due to failure of machine m_2^{sub} or an action that turns machine m_2^{sub} down. In the second term, machine m_2^{sub} is capable of working, but a part is scrapped from buffer B_1^{sub} due to blockage of buffer B_2^{sub}. The third term represents a scrap from buffer B_2^{sub} caused by machine m_3^{sub}.

- *Estimated starvation probability* $\widehat{ST}^{sub}$: the probability that the last machine of the subsystem is not able to produce due to starvation, and specifically,

$$\widehat{ST}^{sub} = \sum_{h^{sub}\in\mathcal{H}^{sub}_{ST}} \mu\left(h^{sub}\right), \tag{5.29}$$

where

$$\mathcal{H}^{sub}_{ST} = \left\{h^{sub}\in\mathcal{H}^{sub} \mid \tau_2^{sub} < T^{sub}_{2,min}\right\}. \tag{5.30}$$

The estimated starvation probability $\widehat{ST}^{sub}$ is the probability that buffer B_2^{sub} has no part with residence time larger than or equal to $T_{2,min}^{sub}$.

- *Estimated blockage probability* $\widehat{BL}^{sub}$: the probability that the first machine of the subsystem is not able to produce due to blockage or control policy, and specifically,

$$\begin{aligned}\widehat{BL}^{sub} = & \sum_{h^{sub}\in\mathcal{H}^{sub}} \mu\left(h^{sub}\right)\left(1-\begin{bmatrix}1 & 0\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\right) \\ & + \sum_{h^{sub}\in\mathcal{H}_{BL,1}^{sub}} \mu\left(h^{sub}\right)\begin{bmatrix}1 & 0\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\left(1-\begin{bmatrix}0 & p_2^{sub}\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\right) \\ & + \sum_{h^{sub}\in\mathcal{H}_{BL,2}^{sub}} \mu\left(h^{sub}\right)\begin{bmatrix}1 & 0\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\begin{bmatrix}0 & p_2^{sub}\end{bmatrix}\pi^{sub}\left(h^{sub}\right)\left(\left(1-p_3^{sub}\right)+p_3^{sub}p^b\right),\end{aligned} \tag{5.31}$$

where

$$\mathcal{H}_{BL,1}^{sub} = \left\{h^{sub}\in\mathcal{H}^{sub} \mid n_1^{sub}=N_1^{sub}, \tau_1^{sub} < T_{1,max}^{sub}-1\right\}, \tag{5.32}$$

$$\mathcal{H}_{BL,2}^{sub} = \left\{h^{sub}\in\mathcal{H}^{sub} \mid n_1^{sub}=N_1^{sub}, \tau_1^{sub} < T_{1,max}^{sub}-1, n_2^{sub}=N_2^{sub}, \tau_2^{sub} < T_{2,max}^{sub}-1\right\}. \tag{5.33}$$

The first term of Equation (5.31) is the probability that machine m_1^{sub} is blocked by the control policy that directly turns machine m_1^{sub} down. The second term represents the case when buffer B_1^{sub} is full and machine m_2^{sub} cannot produce a part from buffer B_1^{sub} due to the control policy or failure on machine m_2^{sub}. The third term gives the situation where both buffer B_1^{sub} and buffer B_2^{sub} are full and machine m_3^{sub} cannot produce a part due to the control policy or failure.

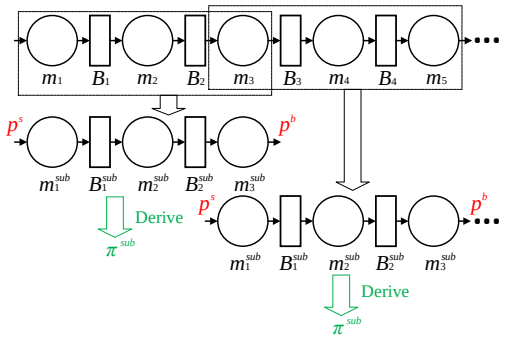
In a similar way, the MDP model of a two-machine-one-buffer subsystem can be built, and the performance measures of a two-machine-one-buffer subsystem can be derived.

5.2.6 Aggregation Procedure

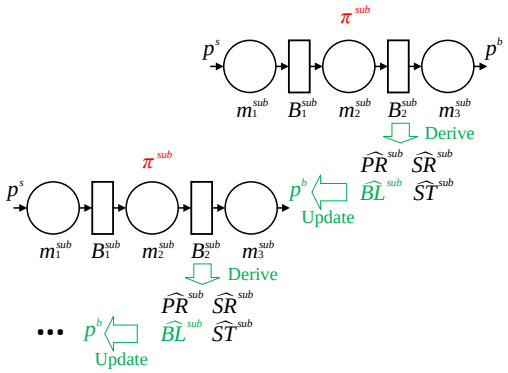
The structural relationship between neighboring subsystems is defined by starvation probability p^s and blockage probability p^b. If p^s and p^b are accurate, the behavior of a subsystem will be similar to its corresponding part in the multi-stage line. The control policy of each subsystem is derived from its MDP model as p^s and p^b are assumed to be known. However, as the control policy of each subsystem is implemented, it changes the relationship between neighboring subsystems. Thus, it requires the relationship to be updated. The update of relationship further requires each subsystem to derive an updated control policy. Thus, an iterative method, the aggregation procedure, is proposed to update the relationship between neighboring subsystems and the control policy of each subsystem.

The aggregation procedure, shown in Figure 5.5, includes the backward aggregation and the forward aggregation. Figure 5.5a shows that a multi-stage line is decomposed into several subsystems, and control policy π^{sub} is derived for each subsystem as the starvation probability p^s and the blockage probability p^b of each subsystem are assumed to be known and fixed. Figure 5.5b and Figure 5.5c illustrate the backward aggregation and the forward aggregation, respectively. In this process, the control policy π^{sub} is fixed, and p^b and p^s are updated through the backward aggregation and forward aggregation, respectively. In addition, performance measures, including $\widehat{PR}^{sub}$, $\widehat{SR}^{sub}$, are derived.

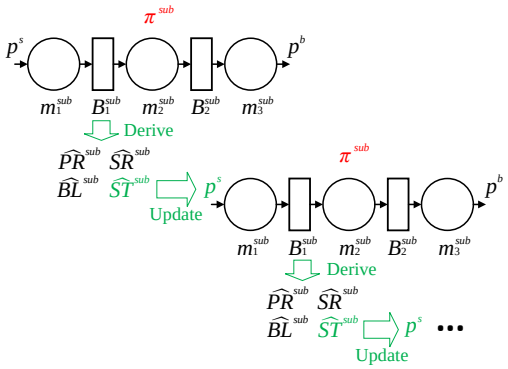
The backward aggregation, shown in Figure 5.5b, starts with the last subsystem and moves backward. The blockage probability $\widehat{BL}^{sub}$, derived by Equation (5.31) from the a subsystem, is used to update p^b of its upstream neighboring subsystem, and this process continues until p^b of the first subsystem is updated. The forward aggregation, shown in Figure 5.5c, is similar to the backward aggregation but starts



(a) Fix p^s and p^b, and Derive Control Policy π^{sub}



(b) The Backward Aggregation. Fix Control Policy π^{sub} and Update p^b



(c) The Forward Aggregation. Fix Control Policy π^{sub} and Update p^s

Figure 5.5: The Aggregation Procedure

with the first subsystem. The starvation probability $\widehat{ST}^{sub}$, derived by Equation (5.29) from a subsystem, is used to update p^s of its downstream neighboring subsystem. This forward aggregation continues until p^s of the last subsystem is updated.

Figure 5.6 provides the pseudocode of the decomposition-based control approach. Line 1 is to decompose the multi-stage line into subsystems. Line 2 initializes the control policy for each subsystem, and the initial control policy never turns machines down. The decomposition-based control consists of several iterations to finally derive

1: Decompose multi-stage line into K subsystems
2: Initialize control policy for each subsystem
3: **while** Stop criteria is not satisfied **do**
4: Initialize p^s and p^b to 0
5: **while** Stop criteria is not satisfied **do**
6: **for** $k = 1, K-1$ **do**
7: Derive performance of subsystem SS_{K-k+1}
8: Update p^b of subsystem SS_{K-k}
9: **end for**
10: **for** $k = 1, K-1$ **do**
11: Derive performance of subsystem SS_k
12: Update p^s of subsystem SS_{k+1}
13: **end for**
14: **end while**
15: **for** $k = 1, K-1$ **do**
16: Obtain control policy for subsystem SS_k
17: **end for**
18: **end while**

Figure 5.6: The Iterative Procedure for Decomposition-Based Control

the control policy for each subsystem, and the iterations are presented from line 3 to line 18. It is a loop from line 3 to line 18. Inside the loop, the steps from line 6 to line 9 represent the backward aggregation, and the steps from line 10 to line 13 represent the forward aggregation. As p^s and p^b of each subsystem are updated, the new control policy for each subsystem is derived, shown from line 15 to line 17. A

stop criteria is set for the loop, and it can be a certain number of iterations or the indication of convergence of p^s and p^b.

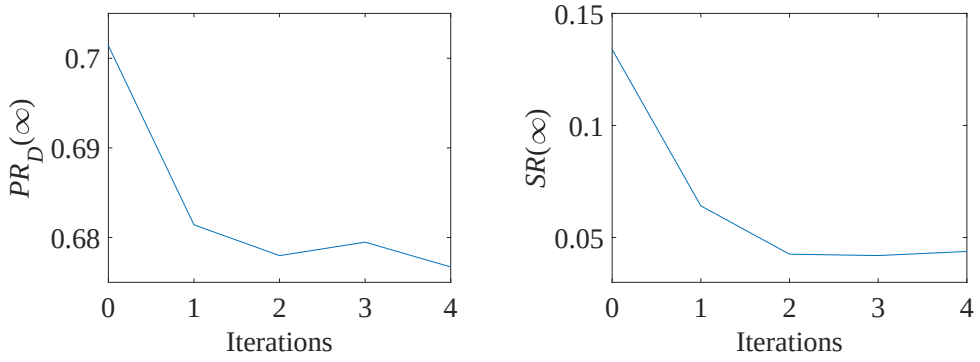
5.2.7 Convergence

A parameter setting is selected as follows to numerically study the convergence of the aggregation procedure.

$$\begin{aligned} &D = 7, \\ &p_1 = 0.9, p_2 = 0.87, p_3 = 0.85, p_4 = 0.83, p_5 = 0.8, p_6 = 0.77, p_7 = 0.75, \\ &N_i = 6, T_{i,max} = 8, T_{i,min} = 2, \quad \text{for } i = 1, \cdots, D-1, \\ &\omega = 1.3. \end{aligned} \tag{5.34}$$

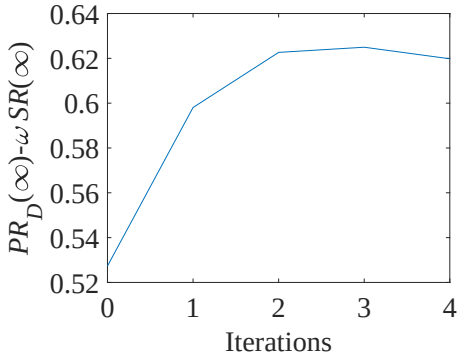
The discount, λ, is set to be 0.95. A set of control policies for subsystems are obtained in each iteration, and the steady-state performance measures are compared under those control policies through simulation. The simulation repeats 1,000 times, and the steady-state performance measures are shown in Figure 5.7. The horizontal axis represents the iterations, and the vertical axis represents the performance measures. Iteration 0 shows the performance measures where the initial control policy is implemented. The result suggests that the decomposition-based control can soon improve the performance in a small number of iterations. The performance measures oscillate withing a small zone primarily due to the random error from simulation. The oscillation of production rate looks more obvious, because the control policy do not change the production rate much.

To numerically study the convergence in a more general sense, vectors $\boldsymbol{p^s}_i$ and $\boldsymbol{p^b}_i$, for $i = 0, 1, \cdots$, are introduced. Let $\boldsymbol{p^s}_i$ and $\boldsymbol{p^b}_i$, for $i = 0, 1, \cdots$, be a vector of the starvation probabilities and a vector of blockage probabilities from all subsystems under the control policy obtained from the ith iteration, respectively. Specifically,



(a) Production Rate

(b) Scrap Rate



(c) Reward

Figure 5.7: Steady-State Performance Measures with Control Policies Obtained in Each Iteration

$$\boldsymbol{p^s}_i = \begin{bmatrix} p_1^s & p_2^s & \cdots \end{bmatrix}^T, \tag{5.35}$$

$$\boldsymbol{p^b}_i = \begin{bmatrix} p_1^b & p_2^b & \cdots \end{bmatrix}^T, \tag{5.36}$$

where p_j^s and p_j^b, for $j = 1, 2, \cdots$, are the starvation probability p^s and blockage probably p^b of subsystem SS_j, respectively. The distance of $\boldsymbol{p^s}_i$ and $\boldsymbol{p^s}_{i-1}$ and the distance of $\boldsymbol{p^b}_i$ and $\boldsymbol{p^b}_{i-1}$ are denoted by d_i^s and d_i^b for $i = 1, 2, \cdots$, respectively, and

defined as follows.

$$d_i^s = \left(\boldsymbol{p^s}_i - \boldsymbol{p^s}_{i-1}\right)^T \left(\boldsymbol{p^s}_i - \boldsymbol{p^s}_{i-1}\right), \tag{5.37}$$

$$d_i^b = \left(\boldsymbol{p^b}_i - \boldsymbol{p^b}_{i-1}\right)^T \left(\boldsymbol{p^b}_i - \boldsymbol{p^b}_{i-1}\right). \tag{5.38}$$

The convergence can be observed, if d_i^s and d_i^b are getting close to 0 as i increases.

To numerically show the convergence of the aggregation procedure of the decomposition-based control, 2,000 parameter settings are randomly generated from the range of parameter settings as follows.

$$\begin{aligned}
p_1 &\in [0.85, 0.99], \\
p_i &\in [0.65, 0.99] \text{ for } i = 2, \cdots, D, \\
N_i &\in \{5, 6, 7\} \text{ for } i = 1, \cdots, D-1, \\
T_{i,max} &\in \{N_i + 1, N_i + 2, N_i + 3\} \text{ for } i = 1, \cdots, D-1, \\
T_{i,min} &\in \{1, 2\} \text{ for } i = 1, \cdots, D-1, \\
\omega &\in [0.7, 1.7].
\end{aligned} \tag{5.39}$$

Parameters are selected with equal probability from the range. Let the number of machines be 9. The number of iterations is set to be 8. Both d_8^s and d_8^b at the end of the iteration are obtained for each parameter setting. The experiment result shows that 99.95% of all cases have d_8^s smaller than 10^{-3} and 100.00% of the cases result in d_8^b smaller than 10^{-3}. It indicates that the performance measures converge within a small interval after a certain number of iterations.

5.3 Numerical Experiments and Performance Comparison

5.3.1 RL Control for Comparison

The decomposition-based control is compared with a feature-based reinforcement learning control (RL control). In the RL control, a feature-based architecture is used to handle the large state space.

Let $r\left(H\left(t-1\right),\boldsymbol{a}(t-1)\right)$ be the reward function of the multi-stage line at time $(t-1)$. Specifically,

$$r\left(H\left(t-1\right),\boldsymbol{a}(t-1)\right)=P\tilde{R}_{D}\left(t\right)-\omega\tilde{SR}\left(t\right). \tag{5.40}$$

Given the initial system state $H\left(0\right)$, the optimal expected total discounted reward is expressed as follows.

$$v^{*}\left(H\left(0\right)\right)=\max_{\pi}E^{\pi}\left\{\sum_{i=0}^{\infty}\lambda^{i}r\left(H\left(i\right),\boldsymbol{a}(i)\right)\right\}, \tag{5.41}$$

which, however, is impossible to obtained due to the large state space of the problem. An approximate lookahead function $\hat{v}(\phi(H(t)),\boldsymbol{\beta})$ with parameters $\boldsymbol{\beta}$ is introduced to replace $v^{*}\left(H\left(t\right)\right)$. Function $\phi(H(t))$ maps system state $H(t)$ to the feature, and $\hat{v}(\phi(H(t)),\boldsymbol{\beta})$ can be obtained through training. The buffer occupancy of each buffer and the residence time of the first part in each buffer are important measures to capture system dynamics, and thus they are taken as candidates of features. To further explore features, a preliminary analysis of features is performed with parameters given as follows.

$$
\begin{aligned}
&D = 4, \\
&p_1 = 0.9, p_2 = 0.83, p_3 = 0.75, p_4 = 0.7, \\
&N_i = 6, \text{ for } i = 1, 2, 3 \\
&T_{i,max} = 8 \text{ for } i = 1, 2, 3, \\
&T_{i,min} = 0 \text{ for } i = 1, 2, 3, \\
&\omega = 0.9, \lambda = 0.95.
\end{aligned}
\tag{5.42}
$$

Let the initial buffer occupancy of each buffer be 2 and residence time of the first part in each buffer be 6. The effect of initial buffer occupancy is studied. Change the initial buffer occupancy from 1 to 6 for each buffer each time with all other parameters fixed. For each initial buffer occupancy, 4,000 initial system states are randomly generated, and simulation runs for 50 cycles starting with each initial system state. The average total discounted rewards, $\sum_{t=1}^{50} \lambda^{t-1} \left(PR\left(t\right) - \omega SR\left(t\right) \right)$, starting with different initial buffer occupancy are compared. The result is shown in Figure 5.8. It suggests that, to have a large average total discounted reward, the buffer occupancy should not be either too small or too large. A small buffer occupancy results in a high probability of starvation for the downstream machines, and it reduces the production rate. In contrast, a large buffer occupancy requires long time to have all the parts in the buffer processed, and the risk of scrap increases.

Following the same way with parameters given in Equation (5.42), the effect of initial residence time of the first part in the buffer is studied. The initial buffer occupancy is set to be 4 for each buffer, and the initial residence time of the head part in each buffer is set to be 3. Change the residence time from 3 to 7 and plot the average total discounted reward in Figure 5.9. A trend can be seen that the average total discounted reward decreases as the initial residence time of head part in buffer

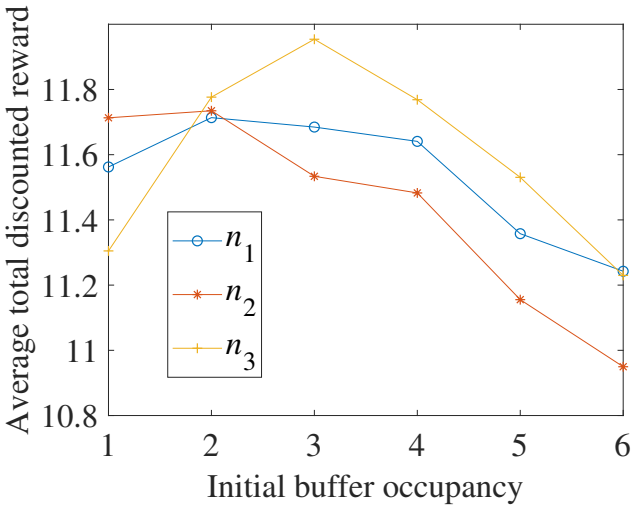


Figure 5.8: The Average Total Discounted Reward with Different Initial Buffer Occupancy

increases. A large residence time results in a high risk of scrap, and thus a small residence time is always preferred.

According to the simulation study, three features are adopted for each buffer, and they are the buffer occupancy n_i, the square of buffer occupancy n_i^2 and residence time of the first part in the buffer $\tau_{i,1}$. Thus, the features for the multi-stage line are provided by

$$\phi\left(H\left(t\right)\right) = \left[\phi_1 \quad \phi_2 \quad \cdots \quad \phi_{3D-2}\right]^T, \tag{5.43}$$

where ϕ_1 is a constant term, and ϕ_{3i-1}, ϕ_{3i} and ϕ_{3i+1} are features of buffer B_i, for $i = 1, 2, \cdots, D-1$. Specifically,

$$\begin{aligned} \phi_{3i-1} &= n_i, \\ \phi_{3i} &= n_i^2, \\ \phi_{3i+1} &= \begin{cases} \tau_{i,1}, & \text{if } n_i \neq 0 \\ 0, & \text{if } n_i = 0 \end{cases}, i = 1, 2, \cdots, D-1. \end{aligned} \tag{5.44}$$

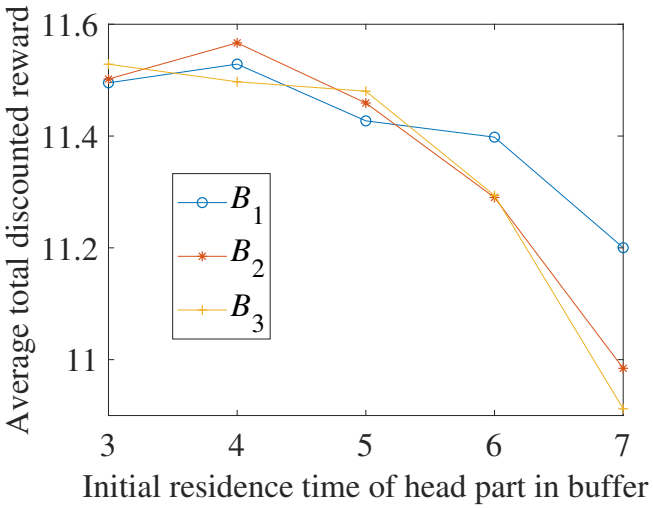


Figure 5.9: The Average Total Discounted Reward with Different Initial Residence Time of Head Part in Buffer

Then, the lookahead function, following a linear feature-based architecture, is expressed as follows.

$$\hat{v}\left(\phi\left(H\left(t\right)\right),\boldsymbol{\beta}\right)=\boldsymbol{\beta}^{T}\phi\left(H\left(t\right)\right). \tag{5.45}$$

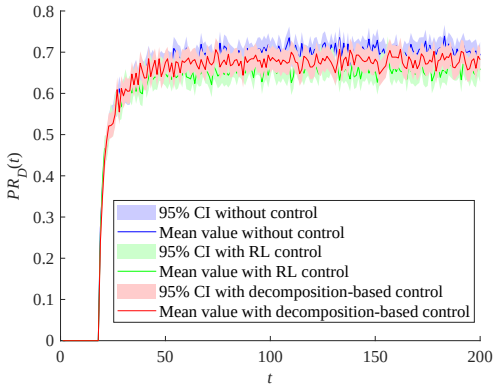
Parameter $\boldsymbol{\beta}$ in Equation (5.45) can be estimated in training through simulation. The optimal action can be expressed as

$$\boldsymbol{a}^{*}(t-1)\in\arg\max_{\boldsymbol{a}(t-1)\in\mathcal{A}}E\left\{r\left(H\left(t-1\right),\boldsymbol{a}(t-1)\right)+\lambda\hat{v}\left(\phi\left(H\left(t\right)\right),\boldsymbol{\beta}\right)\right\}. \tag{5.46}$$

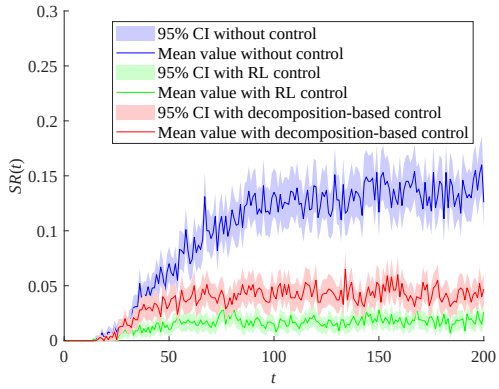
5.3.2 Simulation Experiment with a Single Case

To show how the decomposition-based control improves the multi-stage line, the parameter setting in Equation (5.34) is used. The simulation runs 200 cycles with all buffers empty initially and repeats 1,000 times. The multi-stage line is decomposed into three-machine-two-buffer subsystems.

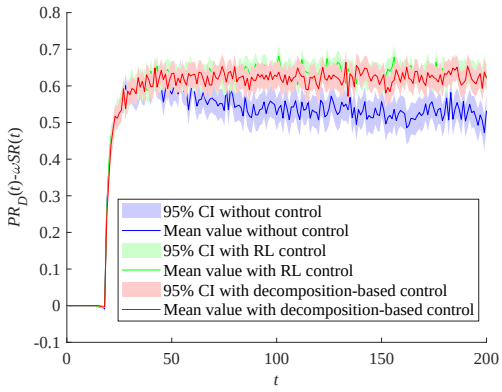
The result of the simulation experiment is shown in Figure 5.10. In each one of the three plots in Figure 5.10, the horizontal axis represents the time from cycle 0



(a) Production Rate



(b) Scrap Rate



(c) Reward

Figure 5.10: Comparison of Performance Measures

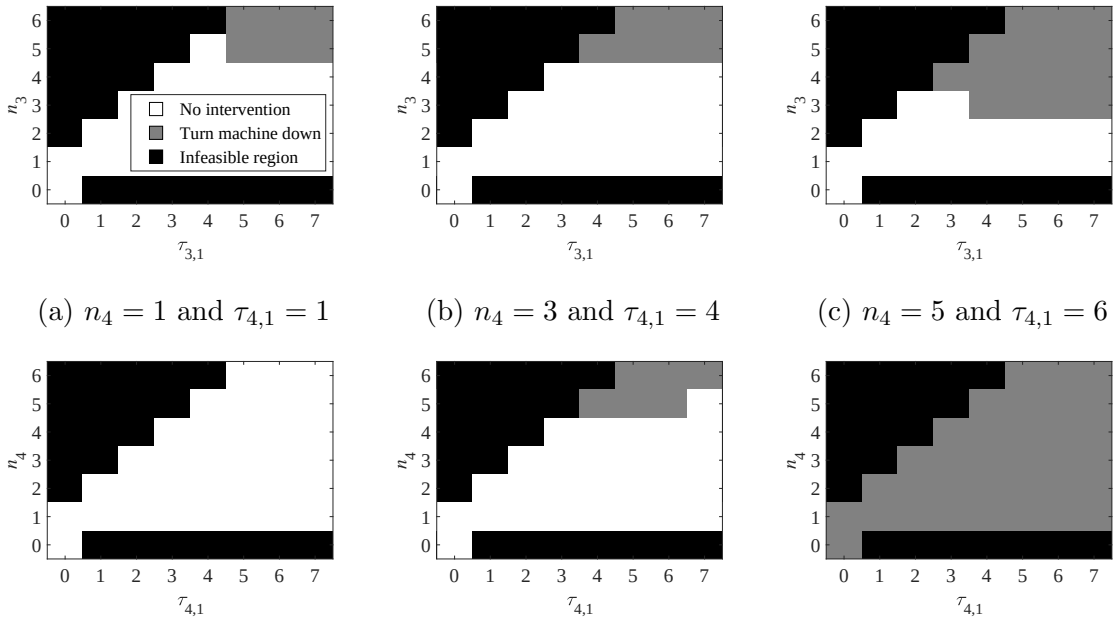
to cycle 200, and the vertical axis represents the performance measures. There are three plots representing three performance measures, and they are production rate $PR_D(t)$, scrap rate $SR(t)$ and reward $(PR_D(t) - \omega SR(t))$. The average performance measures and 95% confidence intervals without control are plotted by blue lines and blue shaded areas, respectively. Similarly, the green color and red color are used for the RL control and the decomposition-based control, respectively.

Production rates with two control methods and without control are plotted in Figure 5.10a, and it shows no significant difference of production rates among the three

methods. The two control methods slightly reduce the production rate. Among the two control methods, the decomposition-based control maintains a higher production rate. The two control methods have significant improvement to the scrap rate, shown in Figure 5.10b, and in this case, RL control shows to reduce more scrap rate. The result suggests that both control methods can significantly reduce scrap rate without sacrificing too much production rate. Figure 5.10c shows the rewards of the three methods. The rewards under RL control and decomposition-based control are higher than the reward without control. The rewards of RL control and decomposition-based control are almost overlapped, and RL control results in a slightly higher reward in this case. In terms of computing time, the decomposition-based control is much more computationally efficient than the RL control. The experiment runs on a server with Intel(R) Core(TM) i7-5930K CPU, sufficiently large RAM and Linux operating system. In this single experiment, the decomposition-based control takes 19 seconds to generate the control policy, while the RL control needs as much as 10,222 seconds for training.

After the last iteration of the aggregation procedure, each subsystem has its control policy. The control policies for machine m_3 and machine m_4 are partially presented in Figure 5.11 and Figure 5.12, respectively. Machine m_3 and machine m_4 are assigned to the second subsystem, which consists of machine m_3, machine m_4, machine m_5, buffer B_3 and buffer B_4. Both machine m_3 and machine m_4 take actions by observing the states of buffer B_3 and buffer B_4.

Figure 5.11 presents the control policy for machine m_3. First fix the state of buffer B_4. The relationship between the action that machine m_3 takes and the state of buffer B_3 is illustrated in Figure 5.11a, Figure 5.11b and Figure 5.11c. The horizontal axis represents the residence time of the first part in buffer B_3, and the vertical axis represents the buffer occupancy of buffer B_3. Given that the state of buffer B_4 is

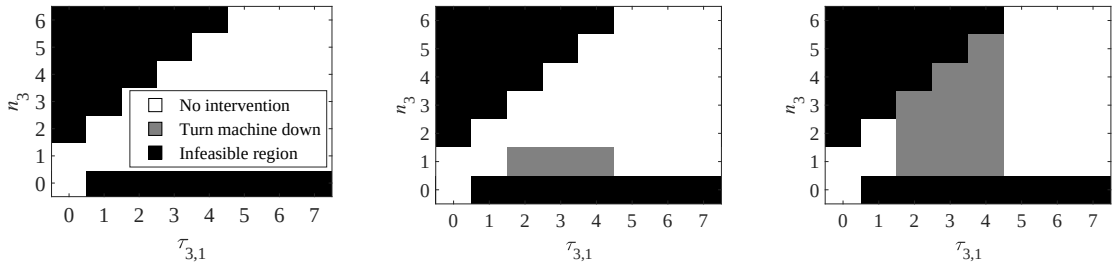


(a) $n_4 = 1$ and $\tau_{4,1} = 1$ (b) $n_4 = 3$ and $\tau_{4,1} = 4$ (c) $n_4 = 5$ and $\tau_{4,1} = 6$

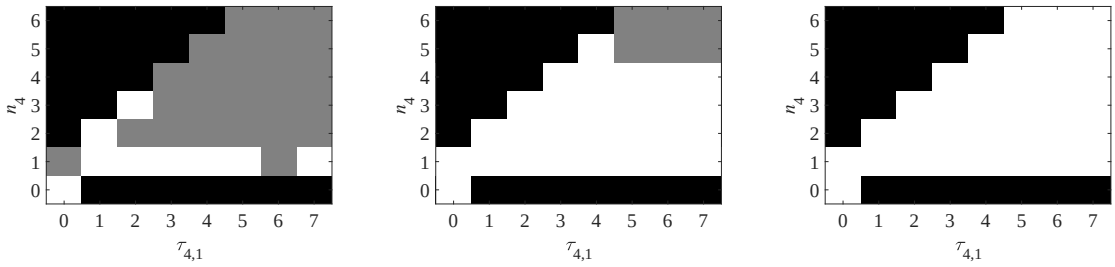
(d) $n_3 = 1$ and $\tau_{3,1} = 2$ (e) $n_3 = 3$ and $\tau_{3,1} = 4$ (f) $n_3 = 5$ and $\tau_{3,1} = 6$

Figure 5.11: Control Policy Obtained from the Decomposition-Based Control for Machine m_3

fixed, a state for the subsystem is represented by a block in the figure. The black blocks are the infeasible region that the subsystem will never visits. In the feasible region, a block is colored to be white or gray, indicating two actions. The white color means that machine m_3 will be unchanged, while the gray color indicates that machine m_3 will be turn down manually. Figure 5.11a shows the case when buffer B_4 has a low buffer occupancy and a small residence time of the first part. It can be observed that machine m_3 is turned down only when there is a high buffer occupancy in buffer B_3. Figure 5.11b shows a control policy when buffer B_4 has a median buffer occupancy and a median residence time of the first part, and the control policy is similar to the control policy shown in Figure 5.11a. When buffer B_4 reaches a high occupancy and has a large residence time of the first part, machine m_3 is more likely



(a) $n_4 = 1$ and $\tau_{4,1} = 1$ (b) $n_4 = 3$ and $\tau_{4,1} = 4$ (c) $n_4 = 5$ and $\tau_{4,1} = 6$



(d) $n_3 = 1$ and $\tau_{3,1} = 2$ (e) $n_3 = 3$ and $\tau_{3,1} = 4$ (f) $n_3 = 5$ and $\tau_{3,1} = 6$

Figure 5.12: Control Policy Obtained from the Decomposition-Based Control for Machine m_4

to be turned down to maintain a lower buffer occupancy for buffer B_3, shown in Figure 5.11c.

Then, fix the state of buffer B_3 and present the control policy with respect to the state of buffer B_4. The result is shown in Figure 5.11d, Figure 5.11e and Figure 5.11f. Figure 5.11d indicates that machine m_3 always keeps unchanged whatever state buffer B_4 is when buffer B_3 has a low buffer occupancy and a small residence time of the first part. In contrast, machine m_3 is always turned down whatever state buffer B_4 is when buffer B_3 has a high buffer occupancy and a large residence time of the first part, which is shown in Figure 5.11f. Figure 5.11e indicates that when buffer B_3 has a median buffer occupancy and a median residence time of the first part, machine m_3 is turned down only when buffer B_4 reaches a high buffer occupancy.

From Figure 5.11, three main features related to the decision making of machine m_3 can be observed. First, buffer occupancy of buffer B_3 and buffer B_4 plays an important role in machine m_3's decision making. Machine m_3 is more likely to be turned down when buffer B_3 and/or buffer B_4 have/has a high buffer occupancy. Such actions prevent the subsystem from a potential scrap by turning machine m_3 down and stopping new parts from entering the subsystem. Since the buffer occupancy is high, the action that turns machine m_3 down will not cause too much loss of production. Second, buffer B_3 has a larger influence on machine m_3's decision making than buffer B_4. It can be observed that the lookup tables shown in Figure 5.11a, Figure 5.11b and Figure 5.11c does not change too much mutually, while the lookup tables in Figure 5.11d, Figure 5.11e and Figure 5.11f shows a large difference. Machine m_3 is closer to buffer B_3 than buffer B_4, and it explains why buffer B_3 has a larger influence on machine m_3's decision making. Last, the control policy is not sensitive to the residence time of the first part in either buffer B_3 or buffer B_4, and the boundary that separates the white region and gray region does not show the property of monotonicity.

Figure 5.12 presents the control policy for machine m_4. In each plot, the black blocks represent the infeasible region. Within the feasible region, the white block indicates the action that no intervention is given, while the gray block indicates the action to turn machine m_4 down. First fix the state of buffer B_4. When buffer B_4 has a low buffer occupancy and a small residence time of the first part, machine m_4 is always kept unchanged. In such a situation, there is no risk of scrap from buffer B_4, and letting machine m_4 work can potentially increase production rate. When buffer B_4 has a median buffer occupancy and a median residence time of the first part, machine m_4 is turned down when buffer B_3 has a small buffer occupancy and a small residence time for the first part. This action can decrease the risk of scrap from buffer B_4 without increasing the risk of scrap from buffer B_3. It can be observed that

the feasible region with $\tau_{3,1}$ smaller than 2 is white, and the actions in those states in fact do not make any difference. The reason is that machine m_4 cannot produce a part from buffer B_3 when the residence time of the first part in buffer B_3 is smaller than $T_{3,min}$. When buffer B_4 has a high buffer occupancy and a large residence time of the first part, machine m_4 produces when the residence time of the first part in buffer B_3 is large. In this case, machine m_4 has to do a trade-off by considering scrap from both buffer B_3 and buffer B_4.

Then, fix the state of buffer B_3. Figure 5.12d suggests that machine m_4 is more likely to be turned down when B_3 has a low buffer occupancy and a small residence time of the first part. Figure 5.12e indicates that, in the cases that B_3 has a median buffer occupancy and a median residence time of the first part, machine m_4 is turned down only when buffer occupancy of buffer B_4 is high. Machine m_4 does so due to the trade-off of scrap in buffer B_3 and buffer B_4. Figure 5.12f suggests that machine m_4 is unchanged whatever state buffer B_4 is when buffer B_3 has a high buffer occupancy and a large residence time of the first part.

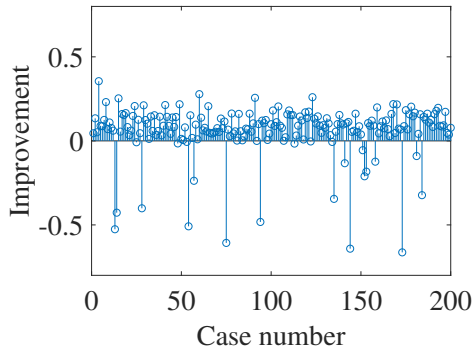
When Figure 5.11 is compared with Figure 5.12, it can be observed that the action on machine m_3 and the action on machine m_4 play different roles in improving the systems. Machine m_3 is the first machine of its subsystem, and it decides to allow a part to enter the subsystem or prevent a part from entering the subsystem. Machine m_4 is in the middle of buffer B_3 and buffer B_4. Its responsibility is to balance the risk of scrap from buffer B_3 and buffer B_4.

5.3.3 Simulation Experiment with Randomly Selected Parameter Settings

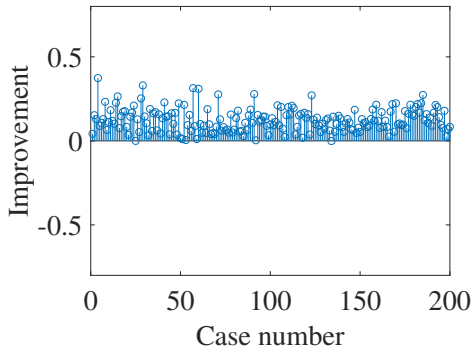
To evaluate the performance of the decomposition-based control in a more general sense, parameter settings of a multi-stage line are randomly selected from a predefined range, and the performance measures of a system without control, with RL control

and with decomposition-based control are compared. The range of parameter settings is given in Equation (5.39). Parameters are selected with equal probability from the range. 200 parameter settings are randomly selected for multi-stage lines with $D = 5$ machines and $D = 7$ machines, respectively. In each parameter setting, the average steady-state performance measures of a system without control, with RL control and with decomposition-based control are obtained through simulation and compared mutually. The simulation starts with empty buffers. The average reward of each cycle from cycle 201 to cycle 400 among 100 repeats, which is the mean value of 20,000 observations, is calculated and compared. In the decomposition-based control, multi-stage line is decomposed into three-machine-two-buffers subsystems.

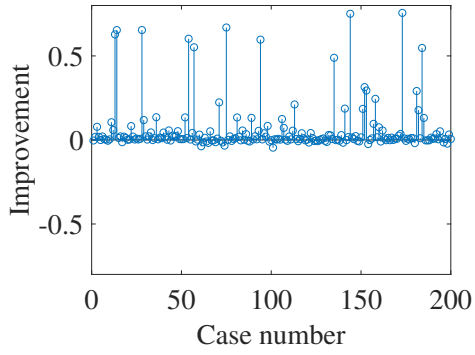
The results of the simulation experiment for the multi-stage lines with 5 machines and 7 machines are shown in Figure 5.13 and in Figure 5.14, respectively. Figure 5.13a and Fig 5.13b show the improvement of reward by RL control and the decomposition-based control for the multi-stage line with 5 machines, respectively. In most cases among 200 random parameter settings, the RL control can improve the system, but it could happen in some cases that the RL control makes the performance worse. In contrast, the decomposition-based control is more robust, and all 200 cases can be improved. Figure 5.13c shows a pairwise comparison where the improvement of decomposition-based control minus the improvement of RL control for each case is presented, and the result suggests that the decomposition-based control outperforms the RL control. Considering the average reward without control is 0.525, such an improvement is significant. The same comparison is performed for the multi-stage line with 7 machines as well. Figure 5.14a, compared with Figure 5.13a, shows more negative improvement. It suggests that as the number of machines increases the RL control is more likely to fail to work. In contrast, Figure 5.14b suggests that the decomposition-based control can still maintain a good performance. Figure 5.14c,



(a) Improvement of RL Control



(b) Improvement of Decomposition-Based Control

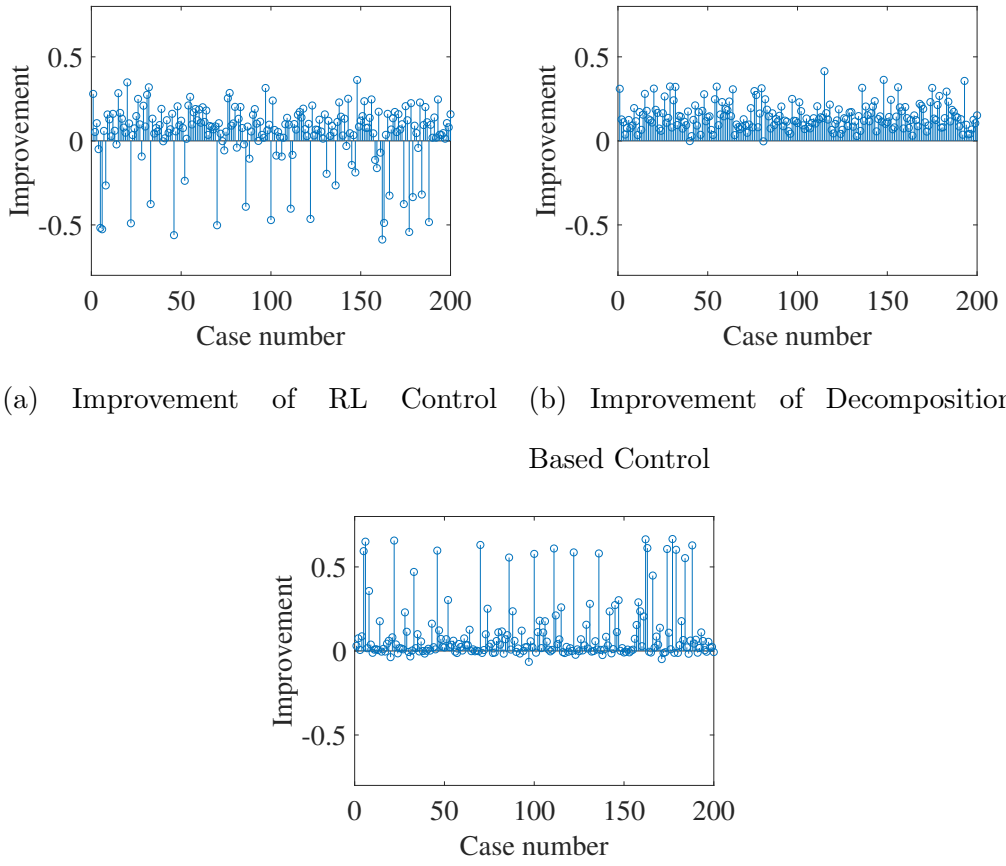


(c) Improvement of Decomposition-Based Control minus Improvement of RL Control

Figure 5.13: Improvement of Average Reward for Multi-Stage Lines with 5 Machines. The Average Reward without Control is 0.525.

compared with Figure 5.13c, shows that the strength of decomposition-based control over the RL control is more significant as the the number of machines increases. The average reward without control is 0.464, and it shows a significant improvement of the decomposition-based control.

The control methods are developed with MATLAB and run on a server with Intel(R) Core(TM) i7-5930K CPU, sufficiently large RAM and Linux operating sys-



(a) Improvement of RL Control

(b) Improvement of Decomposition-Based Control

(c) Improvement of Decomposition-Based Control minus Improvement of RL Control

Figure 5.14: Improvement of Average Reward for Multi-stage Lines with 7 Machines. The Average Reward without Control is 0.464.

tem. It takes time to perform training for RL control and perform the aggregation procedure of decomposition-based control. When there are 5 machines, the average computing time is 1,037.8 seconds for RL control and 34.6 seconds for decomposition-based control. As the total number of machines increases to 7, the average computing time is 14,506.3 seconds for RL control and 70.8 seconds for decomposition-based control. The result suggests that the decomposition-based control is much more compu-

Table 5.1: Average Reward of Different Methods

D	Average reward without control	Average reward with decomposition-based control	Relative improvement of decomposition-based control	Average reward with RL control	Relative improvement of RL control
5	0.525	0.648	23.4%	0.588	12%
7	0.464	0.608	31.0%	0.506	9.1%
9	0.403	0.563	39.7%	–	–
11	0.385	0.543	41.0%	–	–

Table 5.2: Computing Time of Different Methods (Second)

D	Aggregation procedure of decomposition-based control	Training of RL control
5	34.6	1,037.8
7	70.8	14,506.3
9	170.7	–
11	211.6	–

tationally efficient than the RL control. When the number of machines increases, the computing time of the RL control increases much faster than the decomposition-based control.

Serial lines with more machines are tested, and the result is summarized in Tables 5.1 and 5.2. Table 5.1 provides the reward of each method under each setting. The decomposition-based control shows a good performance and also outperforms the RL control and the case under no control. The computing time is presented in Table 5.2. The computing time of the aggregation procedure of the decomposition-based control is much smaller than the training time of RL control and also much less sensitive to the number of machines than the RL control.

Chapter 6

FAST RESPONSE TO MACHINE FAILURES IN SEMICONDUCTOR ASSEMBLY LINES WITH RESIDENCE TIME CONSTRAINTS

6.1 Background

Semiconductor manufacturing consists of wafer fabrication and assembly, also referred to as front-end and back-end, respectively (Zhang *et al.*, 2020; Li *et al.*, 2012; Chen and Lo, 2012). Semiconductor assembly line is a flexible manufacturing system. Products of different types from different orders, having customized operations and qualified machines, are processed in the same assembly line at the same time. The high flexibility, on the other hand, increases the complexity of production system analysis and control. A lot of research has been devoted to a better way to schedule semiconductor assembly so that the throughput is maximized, due date of each order is met, and production requirements are well satisfied (Chung *et al.*, 2014; Lin and Chen, 2015; Hsieh and Cheng, 2018). At Intel, a master schedule is created every shift, specifying when and where each lot of products should be processed within the next three weeks. Creating such a master schedule, considering both the feasibility and optimality, is time-consuming.

However, the semiconductor assembly may not always go as planned, and an assembly line can easily be disrupted by machine failures. Machine repair takes from several minutes to hours, causing the master schedule to be sub-optimal or even infeasible. Specifically, machine failures can result in delay. Since those lots directly impacted by machine failures may not arrive at downstream operations in time, it further leads to delay at downstream operations. In addition, products in

a semiconductor assembly line are subject to residence time constraints. Due to oxidation and moisture absorption issues, they cannot stay in buffer too long (Han and Kim, 2017). Residence time constraints are considered in a master schedule to guarantee product quality, but machine failures may lead to violation of residence time constraints. When long machine failure occurs, neither sticking to the original master schedule nor immediately remaking a new master schedule is a good way to deal with it.

This chapter is aimed at fast response to machine failures in semiconductor assembly lines. There are two practical requirements. First, the master schedule should be retained if possible, considering the difficulty of remaking a master schedule. Second, the computation time should be small enough. In this chapter, machine failures in semiconductor assembly lines are classified into two categories, short machine failure and long machine failure. To handle short machine failure, extra time is added to each operation of each lot to make the master schedule robust. The assembly line can recover on its own without intervention. When long machine failure occurs, a mixed integer programming model is formulated to adjust the master schedule. The original master schedule is taken as a warm start, and a short period schedule is obtained with CPLEX Optimizer for the semiconductor assembly line to follow immediately. In this way, the semiconductor assembly line can respond to long machine failure fast without replacing the whole master schedule, or it can give master scheduler enough time to remake a new master schedule. Thus, the negative impact of machine failure is minimized. Data from shop floor are collected. Using those data, a simulation model is developed with Python and SimPy package to simulate a real-world semiconductor assembly line and evaluate the proposed method. The experiment results show that the proposed method can achieve fast response to machine failures in semiconductor assembly lines.

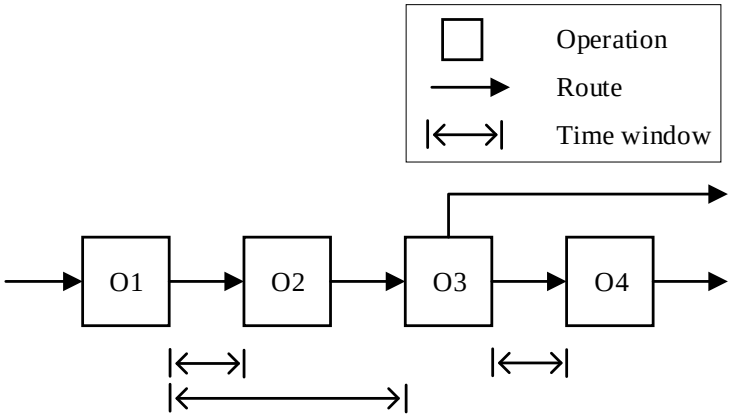


Figure 6.1: The Layout of the Segment of a Semiconductor Assembly Line under Study

6.2 System Description

6.2.1 Layout

In this chapter, a segment of a semiconductor assembly line, presented in Figure 6.1, is studied. There are four operations, denoted by O1, O2, O3 and O4, respectively. Products of different types may have different requirements in terms of operations. Some products go over all four operations, and the others only need to visit the first three operations. Products are subject to residence time constraints, defined by time windows. A time window specifies the maximum time a product can stay in a certain area containing one or more than one buffer.

6.2.2 Product, Product Group and Lot

Each product belongs to a product group. Products of the same product group share great similarity and require the same operations. The properties of product

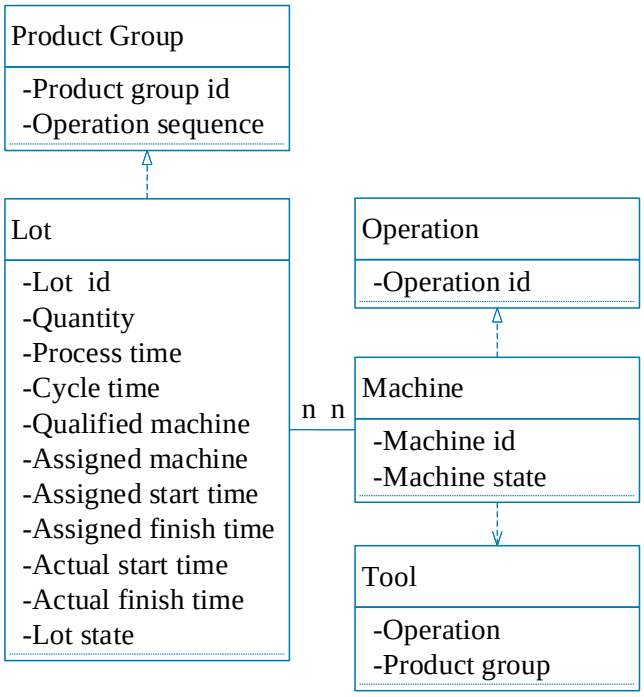


Figure 6.2: The Properties and Relationship of Product Group, Lot, Operation, Machine and Tool

group include product group id and operation sequence, shown in Figure 6.2. In the assembly line under study, there are 20 different product groups.

A lot consists of hundreds of products of the same product group, denoted by letter L plus a number, such as L1, L2, etc. Products in the same lot are processed and moved together. Lot is the minimum unit flowing in the assembly line. A lot carries properties presented in Figure 6.2. Each lot has a unique id. Different lots may carry different quantities of products, leading to different process time. A lot has a set of qualified machines capable of processing the lot. Lots of the same product group may have minor difference and thus can have different sets of qualified machines. A lot is assigned to one machine for each operation, chosen from the set of qualified machines. When a lot finishes the process at an operation, there are some supplementary works

to do on the lot. It also takes time to move the lot to the next operation. The cycle time refers to the time length from the lot finishing an operation to the lot becoming available in buffer for the next operation. Thus, at any time, a lot can be waiting in a buffer, getting process on a machine, or spending cycle time. There is a predetermined schedule that specifies the assigned start time and assigned finish time of each operation for each lot. If the actual finish time of a lot is later than the assigned finish time, then the lot is delayed.

6.2.3 Operation, Machine and Tool

An operation is a collection of machines. A machine is denoted by letter M plus a number, such as M1, M2, etc. A machine can only process lots one at a time. When a machine is vacant, it chooses from its buffer a lot that is ready to go or a delayed lot with the smallest assigned start time. To process a lot, a machine needs to have a tool installed. The tool type and the product group of the lot should match. The machine keeps using the same tool, if the following lots are of the same product group. Otherwise, conversion of tools is required, and it takes time to set up a tool.

Figure 6.3 shows the layout of an operation that consists of several machines. For a lot at this operation, there is a set of qualified machines, which is a subset of all machines at the operation. The lot chooses one qualified machine according to the schedule to have the operation done.

6.2.4 Schedule and Disruption

A schedule is made every shift, specifying when a lot should be released into the system, when the lot should be processed at each operation, and what machine is assigned to process the lot. Making a feasible and satisfactory schedule is complex, and it usually takes a long time. The production of the following three weeks is then

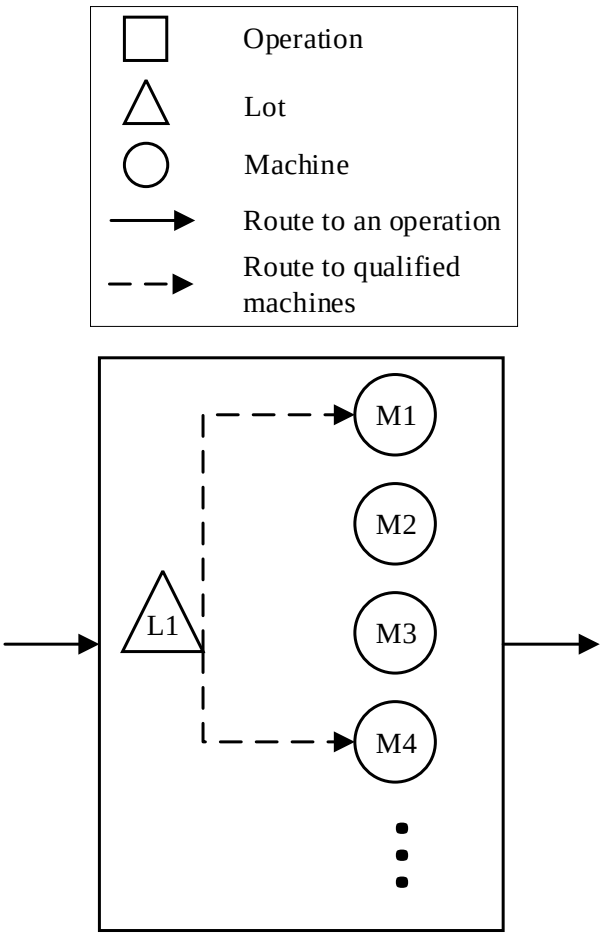


Figure 6.3: The Layout of an Operation

carried out according to the schedule. However, the production can be disrupted by machine failures. When a machine fails to work, lots assigned to this machine are waiting until the machine is repaired. Delay could happen to many lots. It also causes violation of residence time constraints.

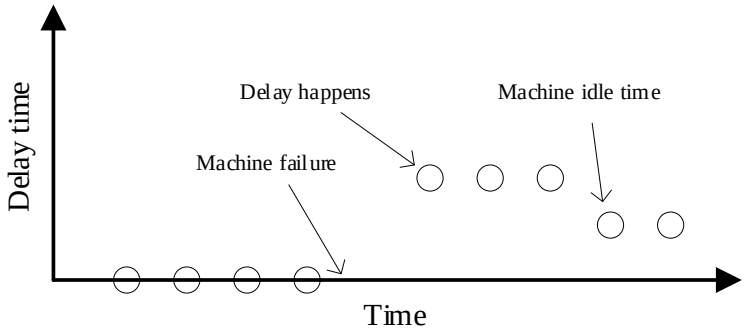
6.3 Method

6.3.1 Short Machine Failure

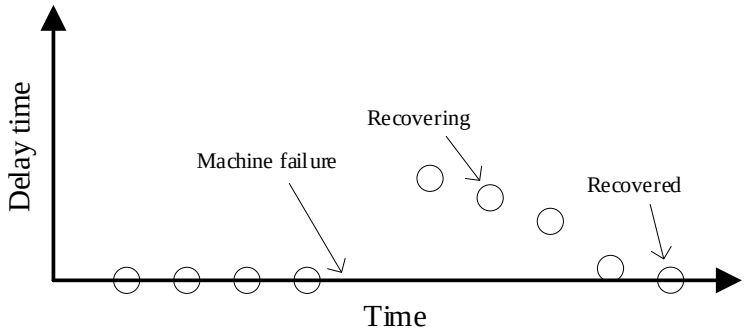
There are two categories of machine failures, short machine failure and long machine failure, which are dealt with separately. To address short machine failure, which lasts usually less than one hour, a small amount of extra time is assigned to each operation for each lot in the schedule. Thus, the assembly line can recover soon on its own after the machine is repaired. Figure 6.4 presents a situation where a machine fails to work. Each circle represents a lot that is assigned to the machine experiencing short machine failure. The horizontal axis is the time that a lot finishes the operation, and the vertical axis stands for the delay time. Machine failure occurs after the fourth lot is processed. Without extra time added, delay time of lots lasts long and reduces only when there exists machine idle time, shown in Figure 6.4a. In contrast, Figure 6.4b shows the case with extra time added, and it starts recovering when the machine is repaired. The delay time of lots is decreasing until it finally reaches zero.

6.3.2 Long Machine Failure

A schedule even with extra time added to each operation for each lot cannot be robust enough to handle long machine failure, and a proper adjustment is required. The computation time should be short enough so that the factory floor can soon switch to the adjusted schedule without too much delay. The adjustment can allow the assembly line to recover faster. Figure 6.5 illustrates the adjustment for an operation. When a machine fails, some lots are directly impacted, shown in Figure 6.5a. Without proper control, those lots and the following lots assigned to the same machine can have a long delay. The control takes advantage of machine idle time in the master schedule so that the total delay time is minimized and the residence time constraints



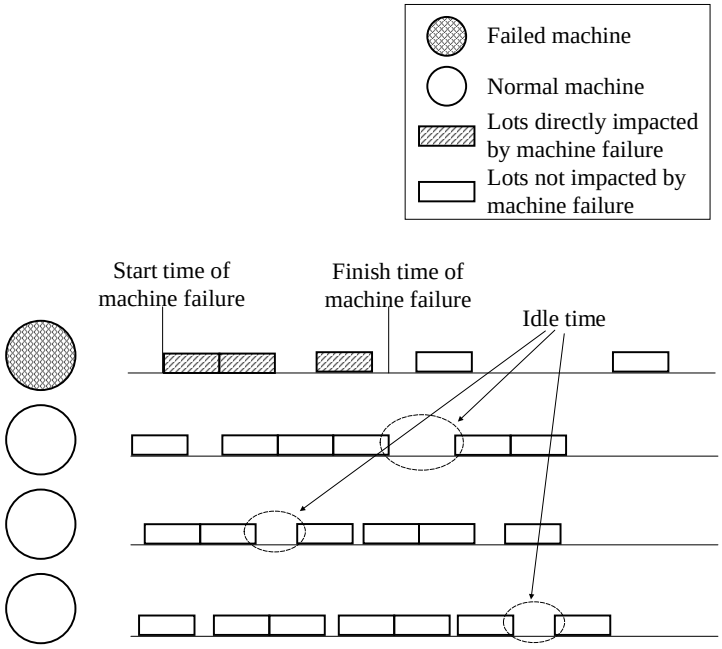
(a) Without Extra Time Added



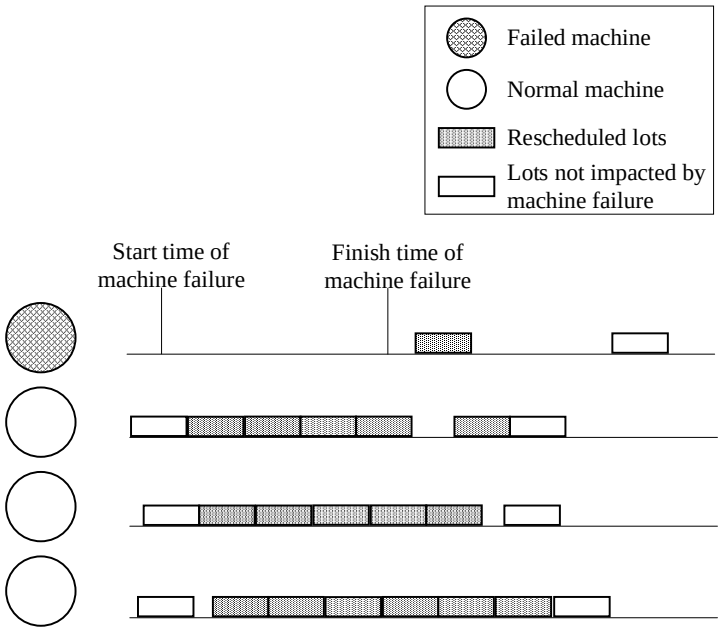
(b) With Extra Time Added

Figure 6.4: Illustration of Short Machine Failure

are satisfied. After a short period, the production can catch up with the master schedule, shown in Figure 6.5b. If the assembly line cannot recover via the control, the quickly adjusted schedule gives the master scheduler time to remake a new master schedule.



(a) Without Extra Time Added



(b) With Extra Time Added

Figure 6.5: Illustration of adjustment for an Operation

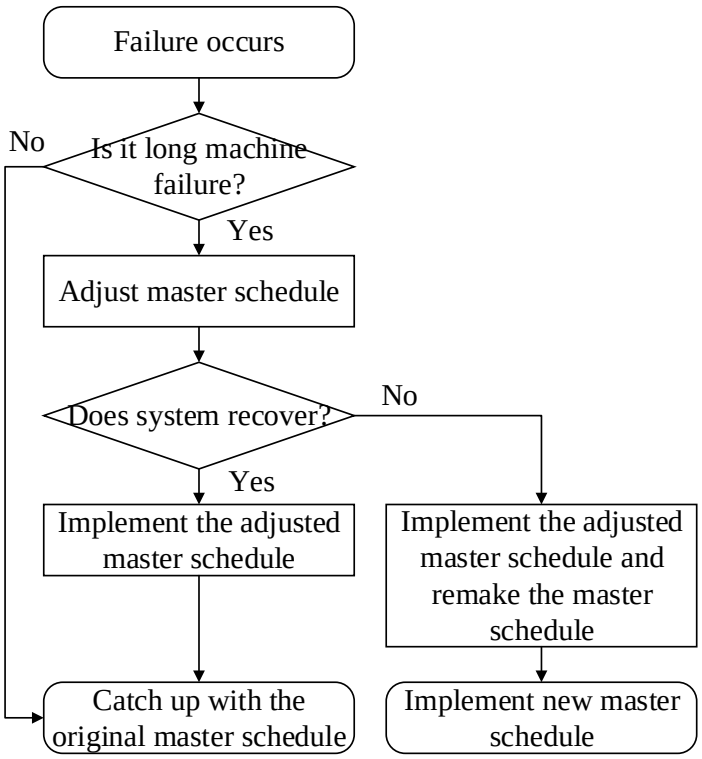


Figure 6.6: The Flowchart to Handle Machine Failure

Figure 6.5 only shows the control for the operation with failed machine. The adjusted schedule also impacts downstream operations. Some lots may delay at one operation and not be able to start the next operation in time. Therefore, similar adjustment is carried out for the downstream operations.

Figure 6.6 shows how a machine failure is handled. If it is a short machine failure, the system can recover on its own. If it is a long machine failure, the master schedule is adjusted so the system can catch up with the original master schedule or has enough time to remake a new master schedule.

Model of an Operation

Assume a machine fails, and the repair is expected to be long. The decision horizon is determined and should be longer than the repair time. Let I be the total number of lots that are or will be at the operation within the decision horizon. Let K be the total number of machines at the operation. Model parameters are first introduced. $o_{i,k} \in \mathbb{R}_{\geq 0}$, for $i = 1, 2, \cdots, I$ and $k = 1, 2, \cdots, K$, specifies if the kth machine is a qualified machine for the ith lot and how long it takes to process. If $o_{i,k} = 0$, the kth machine is not qualified for the ith lot. If $o_{i,k} > 0$, the kth machine is qualified and the process time is $o_{i,k}$. Let e_i, for $i = 1, 2, \cdots, I$, be the time when the ith lot enters the buffer and is ready for the operation. Denote by α_i, for $i = 1, 2, \cdots, I$, the product group of the ith lot. $s_{k,\alpha_i,\alpha_{i\prime}}$ denotes the conversion time of the kth machine from product group α_i to product group $\alpha_{i\prime}$. Let t_k^{r}, for $k = 1, 2, \cdots, K$, be the time when the kth machine becomes available for the first time in the decision horizon. If the kth machine is the failed machine, then t_k^{r} is the time when the machine is repaired. Let p_i, for $i = 1, 2, \cdots, I$, be the time when the ith lot is scheduled to finish the process according to the original master schedule. Let τ_i be the time, by which the ith lot has to finish the process due to residence time constraints. M is a large number. Decision variables are defined, including $x_{i,k} \in \mathbb{R}_{\geq 0}$, for $i = 1, 2, \cdots, I$ and $k = 1, 2, \cdots, K$, $z_{i,i\prime} \in \{0, 1\}$, for $i = 1, 2, \cdots, I$, $i\prime = 1, 2, \cdots, I$ and $i \neq i\prime$, $a_{i,k} \in \{0, 1\}$, for $i = 1, 2, \cdots, I$ and $k = 1, 2, \cdots, K$, and $d_i \in \mathbb{R}_{\geq 0}$, for $i = 1, 2, \cdots, I$. $x_{i,k} = 0$ means the kth machine is not used to perform the operation for the ith lot. If $x_{i,k} > 0$, the ith lot is assigned to the kth machine and the process starts at time $x_{i,k}$. $z_{i,i\prime} \in \{0, 1\}$ is a binary decision variable. $a_{i,k} = 1$ if the ith lot is assigned to the kth machine. Otherwise, $a_{i,k} = 0$. d_i is the delay time of the ith lot. Thus, the mixed integer programming model is developed as follows.

$$\min \quad f\left(x_{i,k}, z_{i,i\prime}, a_{i,k}, d_i\right) = \sum_{i=1}^{I} d_i, \tag{6.1}$$

s.t.

$$\sum_{k=1}^{K} a_{i,k} = 1, \qquad \text{for } i = 1, 2, \cdots, I, \tag{6.2}$$

$$x_{i,k} \leq M a_{i,k}, \qquad \text{for } i = 1, 2, \cdots, I, \text{ and } k = 1, 2, \cdots, K, \tag{6.3}$$

$$a_{i,k} \leq o_{i,k}, \qquad \text{for } i = 1, 2, \cdots, I, \text{ and } k = 1, 2, \cdots, K, \tag{6.4}$$

$$e_i \leq \sum_{k=1}^{K} x_{i,k}, \qquad \text{for } i = 1, 2, \cdots, I, \tag{6.5}$$

$$x_{i\prime,k} - x_{i,k} \leq M\left(1 - z_{i,i\prime}\right) - 1 + 2M(1 - a_{i\prime,k}) + 2M(1 - a_{i,k}),$$
$$\text{for } i = 1, 2, \cdots, I, \ i\prime = 1, 2, \cdots, I, \ i \neq i\prime, \text{ and } k = 1, 2, \cdots, K, \tag{6.6}$$

$$x_{i\prime,k} - \left(x_{i,k} + o_{i,k} + s_{k,\alpha_i,\alpha_{i\prime}}\right) \geq -M z_{i,i\prime} - 2M(1 - a_{i\prime,k}) - 2M(1 - a_{i,k}),$$
$$\text{for } i = 1, 2, \cdots, I, \ i\prime = 1, 2, \cdots, I, \ i \neq i\prime, \text{ and } k = 1, 2, \cdots, K, \tag{6.7}$$

$$x_{i,k} \geq \left(t_k^{\mathrm{r}} + s_{k,\beta_k,\alpha_i}\right) a_{i,k}, \qquad \text{for } i = 1, 2, \cdots, I, \text{ and } k = 1, 2, \cdots, K, \tag{6.8}$$

$$\sum_{k=1}^{K} x_{i,k} \leq \tau_i, \qquad \text{for } i = 1, 2, \cdots, I, \tag{6.9}$$

$$\sum_{k=1}^{K} \left(x_{i,k} + o_{i,k} a_{i,k}\right) - p_i \leq d_i, \qquad \text{for } i = 1, 2, \cdots, I, \tag{6.10}$$

$$x_{i,k} \in \mathbb{R}_{\geq 0}, \ a_{i,k} \in \{0, 1\} \qquad \text{for } i = 1, 2, \cdots, I, \text{ and } k = 1, 2, \cdots, K,$$

$$z_{i,i\prime} \in \{0, 1\} \qquad \text{for } i = 1, 2, \cdots, I, \ i\prime = 1, 2, \cdots, I, \text{ and } i \neq i\prime$$

$$d_i \in \mathbb{R}_{\geq 0} \qquad \text{for } i = 1, 2, \cdots, I.$$

Since exactly one machine is required for each lot, and thus the constraint given by Equation (6.2) holds. The process time is always greater than one minute. Therefore, if $o_{i,k} > 0$, $a_{i,k}$ can be either 1 or 0. Otherwise, $a_{i,k} = 0$. M is always greater than $x_{i,k}$. The ith lot can be assigned to the kth machine, only when the kth machine is

qualified for the ith lot. It means that $x_{i,k}$ and $a_{i,k}$ can be nonzero only when $o_{i,k}$ is nonzero. Thus, it leads to constraints (6.3) and (6.4). e_i is the time when the ith lot enters the buffer, and the start time of a lot should be later than its arrival time. Thus, it leads to Equation (6.5).

Constraints (6.6) and (6.7) restrict that a machine works for lots one at a time. $x_{i,k}$ and $x_{i\prime,k}$ should be mutually compared only when both the ith lot and the $i\prime$th lot are assigned to the kth machine, which means that both $x_{i,k}$ and $x_{i\prime,k}$ are greater than zero. Therefore, Equations (6.6) and (6.7) always hold, if either $x_{i,k}$ or $x_{i\prime,k}$ is equal to zero. If both $x_{i,k}$ and $x_{i\prime,k}$ are positive, Equations (6.6) and (6.7) become Equations (6.11) and (6.12), respectively, which are presented as follows.

$$x_{i\prime,k} - x_{i,k} \leq M\left(1 - z_{i,i\prime}\right) - 1, \tag{6.11}$$

$$x_{i\prime,k} - \left(x_{i,k} + o_{i,k} + s_{k,\alpha_i,\alpha_{i\prime}}\right) \geq -M z_{i,i\prime}. \tag{6.12}$$

Since both i and $i\prime$ traverse set $\{1, 2, \cdots, I\}$, one only needs to compare $x_{i,k}$ and $x_{i\prime,k}$ when the ith lot starts process earlier than the $i\prime$th lot. If the $i\prime$th lot starts earlier, then $z_{i,i\prime} = 1$ and Equations (6.11) and (6.12) become Equations (6.13) and (6.14), respectively, presented as follows.

$$x_{i\prime,k} - x_{i,k} \leq -1, \tag{6.13}$$

$$x_{i\prime,k} - \left(x_{i,k} + o_{i,k} + s_{k,\alpha_i,\alpha_{i\prime}}\right) \geq -M, \tag{6.14}$$

which always hold. If the ith lot starts earlier, then one needs to require that the $i\prime$th lot should start at least after the ith lot finishes the process. In this case, $z_{i,i\prime} = 0$, and Equations (6.11) and (6.12) become Equations (6.15) and (6.16), respectively.

$$x_{i\prime,k} - x_{i,k} \leq M - 1, \tag{6.15}$$

$$x_{i\prime,k} - \left(x_{i,k} + o_{i,k} + s_{k,\alpha_i,\alpha_{i\prime}}\right) \geq 0. \tag{6.16}$$

Equation (6.15) always holds, and Equation (6.16) suggests that if $x_{i\prime,k}$ is greater than $x_{i,k}$ then it should be at least greater than $(x_{i,k} + o_{i,k} + s_{k,\alpha_i,\alpha_{i\prime}})$.

Any lot assigned to the kth machine should start later than t_k^{r} plus conversion time, which is presented in Equation (6.8). If the ith lot is not assigned to the kth machine, then both $x_{i,k}$ and $a_{i,k}$ are equal to zero and Equation (6.8) still holds. Otherwise, $x_{i,k}$ should be greater than or equal to $(t_k^{\mathrm{r}} + s_{k,\beta_k,\alpha_i})$. Equation (6.9) has residence time constraints to be satisfied. Equation (6.10) gives the delay time, which is to be minimized in objective function (6.1).

Thus, a mixed integer programming model is developed. The adjustment of the master schedule for the operation with long machine failure can be obtained by solving the model.

Warm Start

When a machine has a long failure, it causes long delay time to lots assigned to the machine. Lots assigned to other machines of the same operation still follow the original master schedule. It is how the production is carried out without real-time intervention, which is not preferred but could be a feasible solution to the model. This solution is taken as a warm start for the solver to speed up computation.

To address such a real-time decision making problem, one may prefer to reach a good solution quickly rather than search for the optimal solution with long computation time. Warm start acts as a benchmark for an algorithm to solve the model, and thus the output is always better than the benchmark. Therefore, even with very short computation time provided, an adjustment at least better than no control can be obtained.

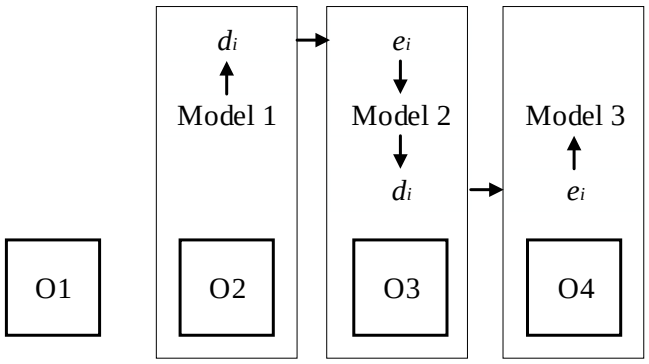


Figure 6.7: All Operations are Coordinated

Adjustment of Downstream Operations

Even though the objective function (6.1) is minimized, there could be some lots with positive delay time d_i. Those lots do not finish the operation in time and thus do not arrive at downstream operations in time. Thus, the downstream operations should also be adjusted according to the adjustment of the upstream operations. Figure 6.7 shows the process of the adjustment for the downstream operations. Assume long machine failure occurs to the second operation. The first model is developed for the second operation. The delay time d_i of the second operation is used to update the arrival time e_i of the third operation. The same model provided in Section 6.3.2 is then developed with the updated e_i and solved for the third operation. The output d_i is then imported to the model for the fourth operation. Thus, a model is developed for a single operation each time, and all operations are coordinated in a decentralized way.

Table 6.1: System Configuration of the Semiconductor Assembly Line

Setting	Value
Number of operations	4
Number of machines at operation 1	11
Number of machines at operation 2	3
Number of machines at operation 3	8
Number of machines at operation 4	15
Number of product groups	20
Span of schedule	3 weeks
Number of lots	528

6.4 Experiment

6.4.1 System Configuration

The system configuration of the semiconductor assembly line in this experiment is presented in Table 6.1. There are four operations, and each operation consists of several machines.

A simulation model is developed with Python and SimPy package. The parameters for building the simulation model are estimated from a real-world semiconductor assembly line. A 3-week master schedule for more than 500 lots of 20 different product groups is available. Both short and long machine failures are randomly generated in simulation run. Figure 6.8 presents the delay time of all lots at the first operation in a simulation run. The horizontal axis gives the assigned start time of a lot, and the vertical axis shows the delay time. Most lots are processed in time, and their delay time is close to zero. When a machine fails for around 20 minutes, it could cause a lot to be delayed for around 20 minutes. Around 4 minutes is assigned as extra time to each lot. Thus, the system is recovering, until the delay time finally reaches zero.

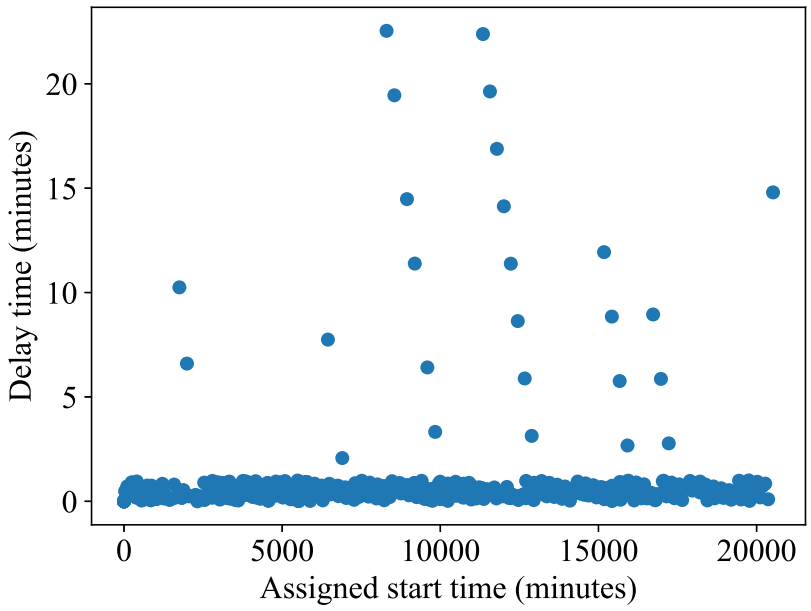


Figure 6.8: Lot Delay Time over 3 Weeks

The first and second operations are subject to long machine failure. The mixed integer programming model, introduced in Section 6.3.2, is developed with Python and solved by CPLEX Optimizer. An initial solution is given to the CPLEX Optimizer as a warm start. The maximum computation time is set to be 2 minutes. The best solution is exported, if the optimal solution is not obtained within the maximum computation time. The experiment is conducted on a server with Intel(R) Core(TM) i7-5930K CPU, sufficiently large RAM and Linux operating system. At most 6 threads are assigned to the CPLEX Optimizer.

6.4.2 An Illustrative Example

Let one machine at the second operation fail for 4 hours at around 3,100 minutes. Table 6.2 presents the original master schedule of the second operation for 12 hours after 3,100 minutes. Three machines at the second operation are M59, M60 and M62.

Table 6.2: Original Master Schedule (Minute)

Machine	Lot	Ready time	Actual start time	Actual finish time	Assigned finish time	Delay time
M59	L406	3119	3173	3215.5	3216	0
M59	L372	3230	3230	3273.5	3274	0
M59	L82	3358	3358	3402.5	3403	0
M59	L94	3411	3411	3455.5	3456	0
M59	L381	3450	3456	3499.5	3500	0
M59	L136	3624	3624	3668.5	3669	0
M59	L84	3605	3669	3713.5	3714	0
M59	L65	3852	3852	3896.5	3897	0
M60	L378	2629	3208	3241.6	3242	0
M60	L396	3163	3242	3248.9	3249	0
M60	L322	1919	3249	3297.5	3298	0
M60	L463	2445	3298	3345.5	3346	0
M60	L166	3377	3377	3421.5	3422	0
M60	L389	3425	3425	3473.5	3474	0
M60	L90	3658	3658	3702.5	3703	0
M62	L108	2636	3167	3211.5	3212	0
M62	L285	3130	3212	3256.5	3257	0
M62	L210	2328	3257	3301.5	3302	0
M62	L168	2575	3302	3346.5	3347	0
M62	L12	1991	3347	3394.5	3395	0
M62	L371	2790	3395	3438.5	3439	0
M62	L373	3010	3439	3482.5	3483	0
M62	L222	2715	3483	3531.5	3532	0
M62	L277	3702	3702	3746.5	3747	0

Table 6.3: Impact of Long Machine Failure on Master Schedule (Minute)

Machine	Lot	Ready time	Actual start time	Actual finish time	Assigned finish time	Delay time
M59	L406	3119	3173	3215.5	3216	0
M59	L372	3230	3230	3273.5	3274	0
M59	L82	3358	3358	3402.5	3403	0
M59	L94	3411	3411	3455.5	3456	0
M59	L381	3450	3456	3499.5	3500	0
M59	L136	3624	3624	3668.5	3669	0
M59	L84	3605	3669	3713.5	3714	0
M59	L65	3852	3852	3896.5	3897	0
M60	L378	2629	3208	3241.6	3242	0
M60	L396	3163	3242	3248.9	3249	0
M60	L322	1919	3249	3297.5	3298	0
M60	L463	2445	3298	3345.5	3346	0
M60	L166	3377	3377	3421.5	3422	0
M60	L389	3425	3425	3473.5	3474	0
M60	L90	3658	3658	3702.5	3703	0
M62	L108	2636	3407	3451.5	3212	239.5
M62	L285	3130	3451.5	3496.0	3257	239.0
M62	L210	2328	3496	3540.5	3302	238.5
M62	L168	2575	3540.5	3585.0	3347	238.0
M62	L12	1991	3585	3632.5	3395	237.5
M62	L371	2790	3632.5	3676.0	3439	237.0
M62	L373	3010	3676.0	3719.5	3483	236.5
M62	L222	2715	3719.5	3767.9	3532	235.9
M62	L277	3702	3767.9	3812.4	3747	65.4

Table 6.4: Adjusted Master Schedule (Minute)

Machine	Lot	Ready time	Actual start time	Actual finish time	Assigned finish time	Delay time
M59	L463	2445	3437	3484.5	3346	138.5
M59	L168	2575	3348	3392.5	3347	45.5
M59	L108	2636	3215.5	3260	3212	48
M59	L406	3119	3173	3215.5	3216	0
M59	L285	3130	3303.5	3348	3257	91
M59	L372	3230	3260	3303.5	3274	29.5
M59	L166	3377	3392.5	3437	3422	15
M59	L381	3450	3484.5	3528	3500	28
M60	L322	1919	3248.5	3297	3298	0
M60	L12	1991	3341.5	3389	3395	0
M60	L210	2328	3297	3341.5	3302	39.5
M60	L378	2629	3208	3241.6	3242	0
M60	L396	3163	3241.6	3248.5	3249	0
M60	L82	3358	3389	3433.5	3403	30.5
M60	L94	3411	3433.5	3478	3456	22
M60	L389	3425	3478	3526.4	3474	52.4
M60	L136	3624	3624	3668.5	3669	0
M60	L277	3702	3702	3746.5	3747	0
M60	L65	3852	3852	3896.5	3897	0
M62	L222	2715	3494	3542.5	3532	10.5
M62	L371	2790	3407	3450.5	3439	11.5
M62	L373	3010	3450.5	3494	3483	11
M62	L84	3605	3605	3649.5	3714	0
M62	L90	3658	3658	3702.5	3703	0

Ready time of a lot in the table is the time when the lot arrives in buffer and is ready for the operation. If there is no disruption, the actual start time of the operation goes according to the schedule, and the actual finish time is a little earlier than the assigned finish time. The delay time is zero for each lot.

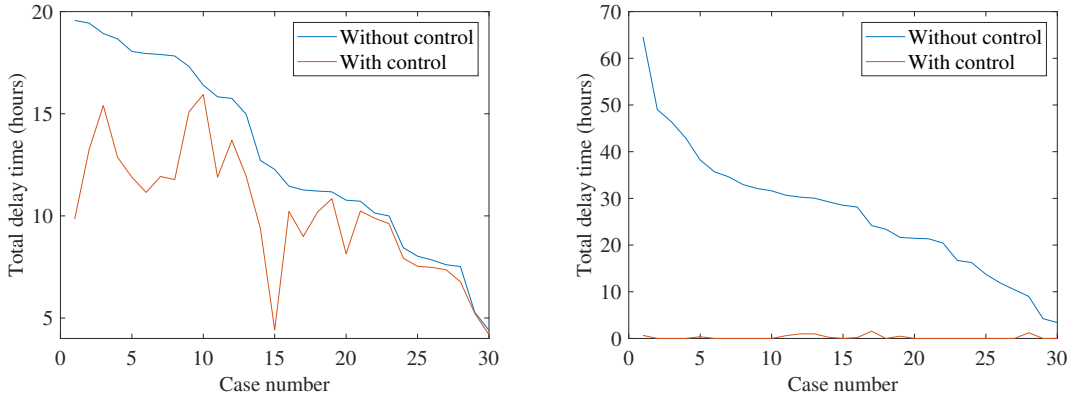
Table 6.3 shows what will happen without control if machine M62 fails for 4 hours. All lots assigned to machine M62 will wait in buffer until the machine is repaired. The machine failure causes around 4 hours delay to 8 lots. The total delay time is 32.79 hours. Production control for 12 hours decision horizon is carried out, and the result is presented in Table 6.4. The total delay time is 9.55 hours, achieving 70.9% reduction.

The delay time of the second operation after adjustment is taken as input for the adjustment of the third operation, and it results in 8 minutes total delay time at the third operation. Then, the 8 minutes delay time does not further cause delay at the fourth operation.

6.4.3 Simulation Experiment with Randomly Generated Long Machine Failure

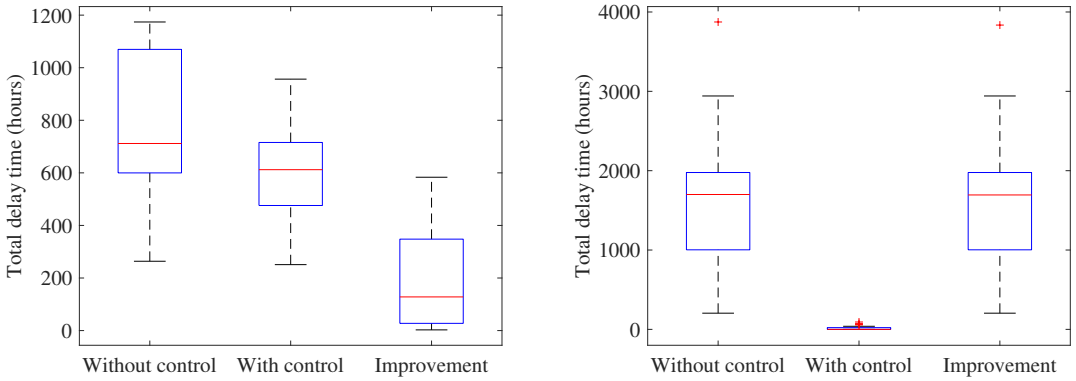
Long machine failures from 200 minutes to 400 minutes are randomly generated in simulation experiment. 30 cases test the long machine failure on the first operation, and another 30 cases test second operation. Figure 6.9 presents the total delay time with and without control. All 30 cases in each plot are sorted by the total delay time without control in descending order. The second operation has much more idle time, and thus the production control shows a better improvement. At both operations, total delay time with control is always smaller than the one without control. Box plots of two operations are presented in Figure 6.10.

Another purpose of fast response to machine failures is to reduce the number of lots violating residence time constraints. Here is an experiment focusing on the



(a) The First Operation (b) The Second Operation

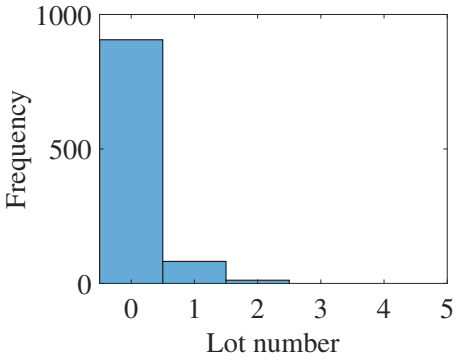
Figure 6.9: Total Delay Time with and without Control with Randomly Generated Long Machine Failure



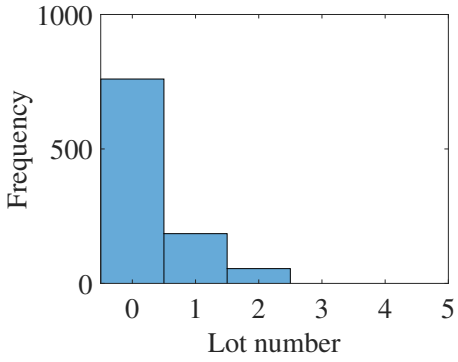
(a) The First Operation (b) The Second Operation

Figure 6.10: Box Plot of Total Delay Time with and without Control

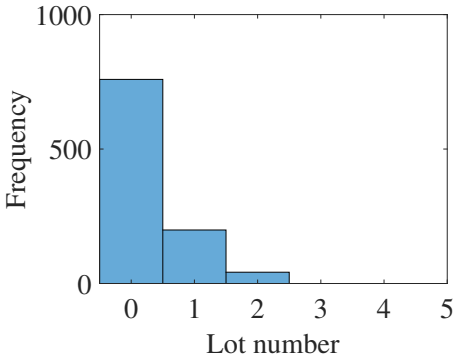
time window between the first operation and the second operation. Machine failure is set from 100 minutes to 600 minutes. For each machine failure, 1,000 cases are randomly generated, and Figure 6.11 shows how frequently residence time constraint violation could occur. The horizontal axis is the number of lots violating residence time constraints, and the vertical axis gives the frequency. As the machine failure time



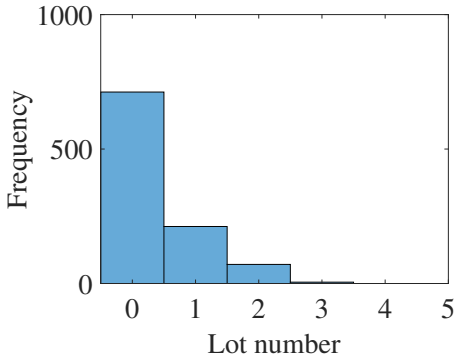
(a) 100 Minutes Machine ailure



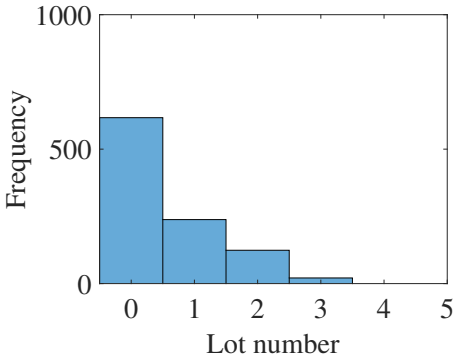
(b) 200 Minutes Machine Failure



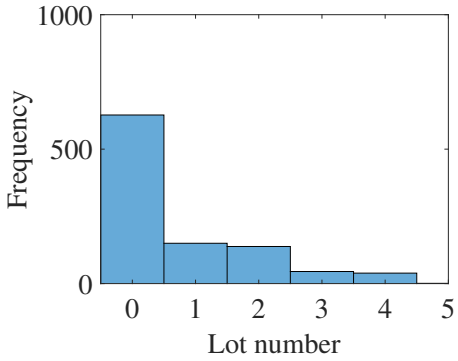
(c) 300 Minutes Machine Failure



(d) 400 Minutes Machine Failure



(e) 500 Minutes Machine Failure



(f) 600 Minutes Machine Failure

Figure 6.11: Histogram of Lots Having Time Window Violation

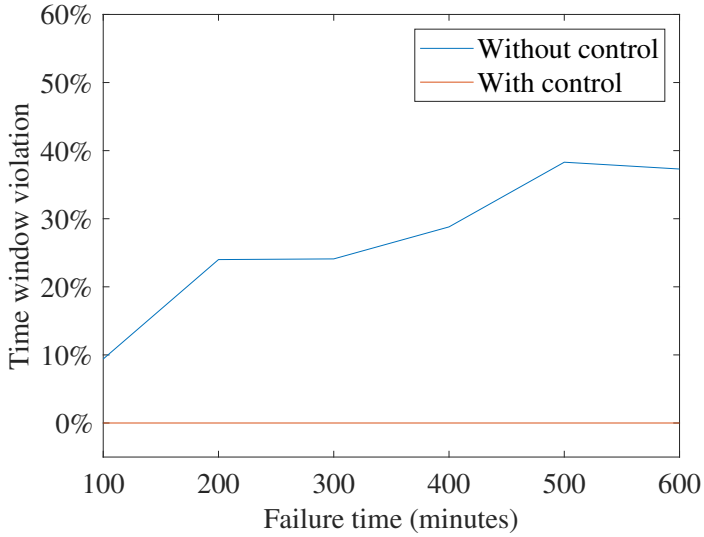


Figure 6.12: Percentage of Cases that Have Time Window Violation

increases, the number of cases with zero residence time constraint violation decreases and more lots could violate the constraint. The production control can improve this performance measure, as shown in Figure 6.12. No residence time constraint violation occurs among all 6,000 cases with production control.

6.4.4 Control in Run Time

The mixed integer programming model is integrated into the simulation model to simulate the scenario in real-world factory, where a long machine failure occurs and the production control is carried out. The factory carries out production according to a schedule, but due to uncertainty the system dynamics can deviate from the schedule. Thus, before any adjustment, one needs to obtain parameters of the mixed integer programming model from real-time system state.

- *Lot set within decision horizon.* Machines may have short delay, and thus the lot set to be considered cannot be directly obtained from the master schedule. By checking the lot being processed on each machine, the lot set can be determined.

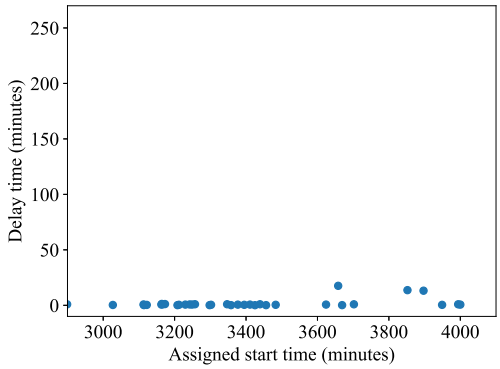
- *The time when a machine becomes available* t_k^r. Due to short machine failure, the actual available time could be later than what suggests in master schedule. By checking lots being processed and their start time, t_k^r for each machine can be determined.

- *The time that lot arrives in buffer* e_i. Lots may not arrive in buffer in time. The actual e_i can be later than what suggests in master schedule. By checking the upstream machines and their delay time, e_i of each lot can be estimated.

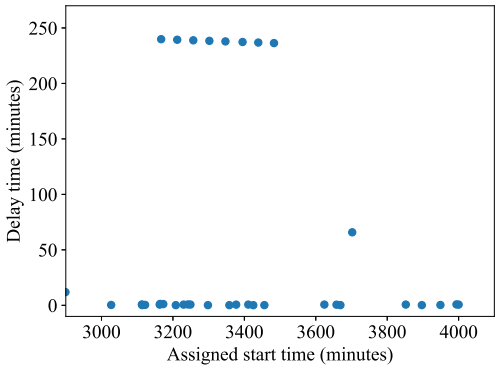
- *The residence time constraint* τ_i. If the upstream operation is delayed, the actual τ_i could be greater than what suggests in master schedule. The upstream operations should be considered to get the value of τ_i.

All other parameters are independent of real-time system state and can be easily obtained from master schedule.

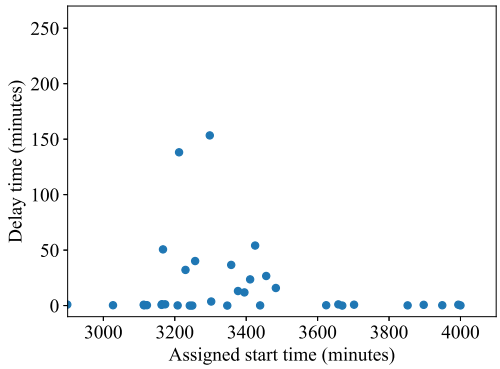
After the mixed integer programming model is integrated into the simulation, a long machine failure is created and the results with and without control are compared. Figure 6.13 presents the results of three different settings. In the first setting shown in Figure 6.13a, there is no long machine failure. Each spot stands for a lot with its assigned start time and delay time. Since there is no long machine failure, delay time is usually around zero minutes. Figure 6.13b shows the setting with long machine failure but no control. The machine failure lasts 4 hours, and many lots are negatively impacted. The impact on lots is reduced by production control, shown in Figure 6.13c. Figure 6.14 compares the three settings from the perspective of throughput. It starts from 2,900 minutes, and the horizontal axis shows the number of lots processed by the time given by the vertical axis. The setting without long machine failure always leads to the highest throughput, and production control can minimize the negative impact of long machine failure on throughput.



(a) With No Long Time Machine Failure



(b) With Long Machine Failure but No Control



(c) With Long Machine Failure and Control

Figure 6.13: Delay Time of the Second Operation

6.5 Discussion

The idle time shown in Figure 6.5 is essential. The essence of the adjustment is to reduce delay time by making use of the idle time. Those idle time exists for two reasons. First, the operation may not be the bottleneck of the production system. Machines may often be idle waiting for lots to come. Second, the idle time could also be created in the master schedule purposely to trade efficiency for flexibility. If there is not much idle time, an operation may not be able to recover. In this

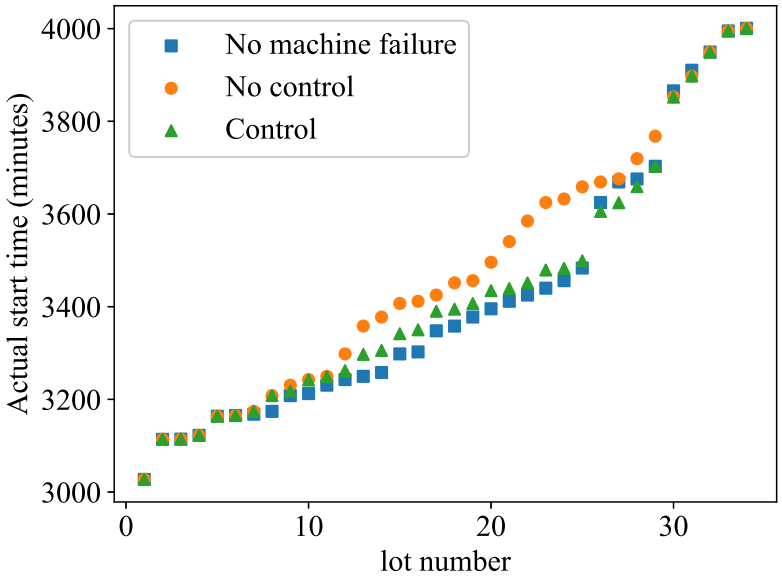


Figure 6.14: Throughput

situation, the proposed method can still minimize total delay time and the residence time constraint violation. In addition, a fast response to machine failures provides the master scheduler with enough time to remake a new master schedule.

The decision horizon is determined depending on length of machine failure and idle time. The long machine failure could last several hours or a whole day. The mixed integer programming needs to consider all lots directly impacted. A short decision horizon means there may not be enough idle time to make use of, while a long decision horizon results in a high problem complexity.

In this chapter, it is assumed that how long the machine will fail is known. When a machine fails to work, its repair time can be estimated according to the failure type. This estimation may not always be accurate. If the machine is capable of getting back to work early, it does not impact the adjusted master schedule. If the repair

is delayed, based on the information about how long it could be delayed, production control can be performed again with similar process.

Chapter 7

CONCLUSIONS

This dissertation focuses on production systems with constrained time window. The study starts with a two-machine serial production line with residence time constraints. An analytical model is introduced to investigate its transient performance. Compared to simulations, the analytical results are verified to possess high accuracy in evaluating the transient behavior. Systematic properties are explored to help have a better understanding of such systems. An algorithm for real-time control is proposed to optimize the system performance. Then, a novel modeling approach is introduced for performance evaluation of multi-stage serial lines with residence time constraints. Specifically, the two-machine-one-buffer subsystem is defined as a building block, and a Markov chain model is developed to analyze the two-machine-one-buffer subsystem. The aggregation method is then developed to use subsystems to approximately evaluate the performance of a multi-stage serial line. Compared with simulation, the proposed aggregation method is verified to possess high accuracy. Due to a large state space, it is difficult to perform real-time control for multi-stage serial production lines. The decomposition-based control is proposed to address the problem. The simulation experiment suggests that the proposed method can improve system performance. Compared with a general-purpose reinforcement learning based control method, the decomposition-based control can achieve a better system performance improvement and a significant reduction on computing time. It provides production engineers with an effective and quantitative tool to perform real-time control of production systems with residence time constraints. Semiconductor assembly line is an example, where machines are disrupted by random failure and products are restricted by time win-

dows. To optimize the operation of a semiconductor assembly line, machine failures are classified into the short machine failure and long machine failure and dealt with separately. Fast response to machine failures is achieved to reduce lot delay time and residence time constraint violation.

There are several directions for future research. Production systems may have different structures, such as distributed systems and assembly systems. Those structures and their properties can be explored. In addition, residence time constraints may have different forms, which can be further studied. Bernoulli machines and geometric machines are used to model machine reliability in this dissertation. It is worth studying production control in a manufacturing environment with more general machine reliability models. In terms of semiconductor assembly line, algorithms to solve the mixed integer programming model can be studied. Besides, more practical requirements observed in semiconductor assembly lines can be considered in modeling. Re-entrance is commonly seen. In some operations, lots with small size can merge to form a single large lot. Sometimes, a large lot splits into multiple small lots. Semiconductor assembly lines are also faced with different disruptions, other than machine failures. It is worth studying fast response in semiconductor assembly line with those requirements considered.

REFERENCES

Abumaizar, R. J. and J. A. Svestka, "Rescheduling job shops under random disruptions", International journal of production research **35**, 7, 2065–2082 (1997).

Ahumada, O. and J. R. Villalobos, "Operational model for planning the harvest and distribution of perishable agricultural products", International Journal of Production Economics **133**, 2, 677–687 (2011).

Albornoz, V. M., M. C. González-Araya, M. C. Gripe and S. V. Rodríguez, "A mixed integer linear program for operational planning in a meat packing plant", in "ICORES", pp. 254–261 (2015).

Amorim, P., H. Meyr, C. Almeder and B. Almada-Lobo, "Managing perishability in production-distribution planning: a discussion and review", Flexible Services and Manufacturing Journal **25**, 3, 389–413 (2013).

Angius, A., M. Colledani, A. Horváth and S. B. Gershwin, "Analysis of the lead time distribution in closed loop manufacturing systems", IFAC-PapersOnLine **49**, 12, 307–312 (2016).

Barbosa, J., P. Leitão, E. Adam and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems: The adacor evolution", Computers in industry **66**, 99–111 (2015).

Beliën, J. and H. Forcé, "Supply chain management of blood products: A literature review", European Journal of Operational Research **217**, 1, 1–16 (2012).

Bertsekas, D., *Dynamic programming and optimal control* (Athena Scientific, Belmont, MA, 2012).

Brodheim, E. and G. P. Prastacos, "The long island blood distribution system as a prototype for regional blood management", Interfaces **9**, 5, 3–20 (1979).

Cao, P. and J. Xie, "Optimal control of an inventory system with joint production and pricing decisions", IEEE Transactions on Automatic Control **61**, 12, 4235–4240 (2015).

Chen, A. and R. H.-Y. Lo, *Semiconductor packaging: materials interaction and reliability* (Taylor & Francis, 2012).

Chen, G., C. Wang, L. Zhang, J. Arinez and G. Xiao, "Transient performance analysis of serial production lines with geometric machines", IEEE Transactions on Automatic Control **61**, 4, 877–891 (2016).

Chung, B.-s., J. Lim, I.-B. Park, J. Park, M. Seo and J. Seo, "Setup change scheduling for semiconductor packaging facilities using a genetic algorithm with an operator recommender", IEEE Transactions on Semiconductor Manufacturing **27**, 3, 377–387 (2014).

Colledani, M. and S. B. Gershwin, "A decomposition method for approximate evaluation of continuous flow multi-stage lines with general markovian machines", Annals of Operations Research **209**, 1, 5–40 (2013).

Dumas, M. B. and M. Rabinowitz, "Policies for reducing blood wastage in hospital blood banks", Management Science **23**, 10, 1124–1132 (1977).

Galloway, J. E. and B. M. Miles, "Moisture absorption and desorption predictions for plastic ball grid array packages", in "InterSociety Conference on Thermal Phenomena in Electronic Systems, I-THERM V", pp. 180–186 (IEEE, 1996).

Gershwin, S. B., "An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking", Operations research **35**, 2, 291–305 (1987).

Gershwin, S. B., *Manufacturing systems engineering* (Prentice Hall, Upper Saddle River, NJ, USA, 1994).

Giret, A., E. Garcia and V. Botti, "An engineering framework for service-oriented intelligent manufacturing systems", Computers in Industry **81**, 116–127 (2016).

Gupta, D., C. T. Maravelias and J. M. Wassick, "From rescheduling to online scheduling", Chemical Engineering Research and Design **116**, 83–97 (2016).

Han, B. and D.-S. Kim, "Moisture ingress, behavior, and prediction inside semiconductor packaging: A review", Journal of Electronic Packaging **139**, 1, 010802–1–010802–11 (2017).

Hsieh, L. Y. and C.-B. Cheng, "Efficient due-date quoting and production scheduling for integrated circuit packaging with reentrant processes", IEEE Transactions on Components, Packaging and Manufacturing Technology **8**, 8, 1487–1495 (2018).

Huh, J., I. Park, S. Lim, B. Paeng, J. Park and K. Kim, "Learning to dispatch operations with intentional delay for re-entrant multiple-chip product assembly lines", Sustainability **10**, 11, 4123 (2018).

Jaegler, Y., A. Jaegler, P. Burlat, S. Lamouri and D. Trentesaux, "The conwip production control system: a systematic review and classification", International Journal of Production Research **56**, 17, 5736–5757 (2018).

Ju, F., *Battery Manufacturing Management System for Electric Vehicles*, Ph.D. thesis, University of Wisconsin-Madison (2015).

Ju, F., J. Li and W. Deng, "Selective assembly system with unreliable bernoulli machines and finite buffers", IEEE transactions on automation science and engineering **14**, 1, 171–184 (2016).

Ju, F., J. Li and W. Deng, "Selective assembly system with unreliable bernoulli machines and finite buffers", IEEE Transactions on Automation Science and Engineering **14**, 1, 171–184 (2017a).

Ju, F., J. Li and J. Horst, "Transient analysis of bernoulli serial line with perishable products", in "Proceedings of IFAC Symposium on Information Control Problems in Manufacturing", pp. 1670–1675 (2015).

Ju, F., J. Li and J. A. Horst, "Transient analysis of serial production lines with perishable products: Bernoulli reliability model", IEEE Transactions on automatic control **62**, 2, 694–707 (2017b).

Kang, N., F. Ju and L. Zheng, "Transient analysis of geometric serial lines with perishable intermediate products", IEEE Robotics and Automation Letters **2**, 1, 149–156 (2017a).

Kang, N., C. Zhao, J. Li and L. Zheng, "A sub-optimal control policy in a two-product door manufacturing line with geometric reliability machines", IEEE Robotics and Automation Letters **2**, 1, 157–164 (2017b).

Kang, Y. and F. Ju, "Integrated analysis of productivity and machine condition degradation: A geometric-machine case", in "2016 IEEE International Conference on Automation Science and Engineering (CASE)", pp. 1128–1133 (IEEE, 2016).

Kang, Y. and F. Ju, "Maintenance decision model for a two-machine production line with multistage degradation machines", in "ASME 2018 13th International Manufacturing Science and Engineering Conference", pp. V003T02A023–V003T02A023 (American Society of Mechanical Engineers, 2018).

Kang, Y., H. Yan and F. Ju, "Real-time production performance analysis using machine degradation signals: a two-machine case", in "2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)", pp. 1501–1506 (IEEE, 2018).

Kang, Y., H. Yan and F. Ju, "Performance evaluation of production systems using real-time machine degradation signals", IEEE Transactions on Automation Science and Engineering **17**, 1, 273–283 (2019).

Kärkkäinen, M., "Increasing efficiency in the supply chain for short shelf life goods using rfid tagging", International Journal of Retail & Distribution Management **31**, 10, 529–536 (2003).

Kim, H.-J. and J.-H. Lee, "Three-machine flow shop scheduling with overlapping waiting time constraints", Computers & Operations Research **101**, 93–102 (2019).

Klemmt, A. and L. Mönch, "Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing", in "Proceedings of the 2012 winter simulation conference (WSC)", pp. 1–10 (IEEE, 2012).

Lee, J.-H. and J. Li, "Performance evaluation of bernoulli serial lines with waiting time constraints", IFAC-PapersOnLine **50**, 1, 1087–1092 (2017).

Lee, J.-H., J. Li and J. A. Horst, "Serial production lines with waiting time limits: Bernoulli reliability model", IEEE Transactions on Engineering Management **65**, 2, 316–329 (2018).

Leitão, P., "Agent-based distributed manufacturing control: A state-of-the-art survey", Engineering Applications of Artificial Intelligence **22**, 7, 979–991 (2009).

Levantesi, R., A. Matta and T. Tolio, "Performance evaluation of continuous production lines with machines having different processing times and multiple failure modes", Performance evaluation **51**, 2-4, 247–268 (2003).

Li, J. and S. M. Meerkov, *Production systems engineering* (Springer Science & Business Media, New York, NY, USA, 2009).

Li, L., Z. Sun, M. Zhou and F. Qiao, "Adaptive dispatching rule for semiconductor wafer fabrication facility", IEEE Transactions on Automation Science and Engineering **10**, 2, 354–364 (2012).

Liberopoulos, G. and P. Tsarouhas, "Reliability analysis of an automated pizza production line", Journal of Food Engineering **69**, 1, 79–96 (2005).

Lin, J. T. and C.-M. Chen, "Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing", Simulation Modelling Practice and Theory **51**, 100–114 (2015).

Liu, L., H.-y. Gu and Y.-g. Xi, "Robust and stable scheduling of a single machine with random machine breakdowns", The International Journal of Advanced Manufacturing Technology **31**, 7, 645–654 (2007).

Lu, Y. and F. Ju, "Smart manufacturing systems based on cyber-physical manufacturing services (cpms)", IFAC-PapersOnLine **50**, 1, 15883–15889 (2017).

Lu, Y., F. Riddick and N. Ivezic, "The paradigm shift in smart manufacturing system architecture", in "IFIP International Conference on Advances in Production Management Systems", pp. 767–776 (Springer, 2016).

Mason, S., S. Jin and C. Wessels, "Rescheduling strategies for minimizing total weighted tardiness in complex job shops", International Journal of Production Research **42**, 3, 613–628 (2004).

Mastrolilli, M. and L. M. Gambardella, "Effective neighbourhood functions for the flexible job shop problem", Journal of scheduling **3**, 1, 3–20 (2000).

Meerkov, S. M., N. Shimkin and L. Zhang, "Transient behavior of two-machine geometric production lines", IEEE Transactions on Automatic Control **55**, 2, 453–458 (2010).

Meerkov, S. M. and L. Zhang, "Transient behavior of serial production lines with bernoulli machines", IIE Transactions **40**, 3, 297–312 (2008).

Naebulharam, R. and L. Zhang, "Bernoulli serial lines with deteriorating product quality: performance evaluation and system-theoretic properties", International Journal of Production Research **52**, 5, 1479–1494 (2014).

Nahmias, S., "Perishable inventory theory: A review", Operations research **30**, 4, 680–708 (1982).

Pahl, J. and S. Voß, "Integrating deterioration and lifetime constraints in production and supply chain planning: A survey", European Journal of Operational Research **238**, 3, 654–674 (2014).

Pan, C., M. Zhou, Y. Qiao and N. Wu, "Scheduling cluster tools in semiconductor manufacturing: Recent advances and challenges", IEEE Transactions on Automation Science and Engineering **15**, 2, 586–601 (2017).

Papadopoulos, C. T., J. Li and M. E. O'Kelly, "A classification and review of timed markov models of manufacturing systems", Computers & Industrial Engineering **128**, 219–244 (2019).

Park, I.-B., J. Huh, J. Kim and J. Park, "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities", IEEE Transactions on Automation Science and Engineering **17**, 3, 1420–1431 (2019).

Qiao, F., L. Li, Y. Ma and B. Shi, "Single machine oriented match-up rescheduling method for semiconductor manufacturing system", in "International Conference on Intelligent Robotics and Applications", pp. 217–226 (Springer, 2012).

Qiao, F., Y. Ma, M. Zhou and Q. Wu, "A novel rescheduling method for dynamic semiconductor manufacturing systems", IEEE Transactions on Systems, Man, and Cybernetics: Systems **50**, 5, 1679–1689 (2018).

Raafat, F., "Survey of literature on continuously deteriorating inventory models", Journal of the Operational Research society **42**, 1, 27–37 (1991).

Shi, C. and S. B. Gershwin, "Part waiting time distribution in a two-machine line", IFAC Proceedings Volumes **45**, 6, 457–462 (2012).

Shi, C. and S. B. Gershwin, "Part sojourn time distribution in a two-machine line", European Journal of Operational Research **248**, 1, 146–158 (2016).

Shin, M., H. Lee, K. Ryu, Y. Cho and Y.-J. Son, "A two-phased perishable inventory model for production planning in a food industry", Computers & Industrial Engineering **133**, 175–185 (2019).

Stephan, P., G. Meixner, H. Koessling, F. Floerchinger and L. Ollinger, "Product-mediated communication through digital object memories in heterogeneous value chains", in "Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on", pp. 199–207 (IEEE, 2010).

Stricker, N., A. Kuhnle, R. Sturm and S. Friess, "Reinforcement learning for adaptive order dispatching in the semiconductor industry", CIRP Annals **67**, 1, 511–514 (2018).

Tee, T. Y. and Z. Zhong, "Integrated vapor pressure, hygroswelling, and thermo-mechanical stress modeling of qfn package during reflow with interfacial fracture mechanics analysis", Microelectronics reliability **44**, 1, 105–114 (2004).

Thürer, M., N. O. Fernandes, M. Stevenson, T. Qu and C. D. Li, "Centralised vs. decentralised control decision in card-based control systems: comparing kanban systems and cobacabana", International Journal of Production Research **57**, 2, 322–337 (2019).

Wang, F., F. Ju and N. Kang, "Transient analysis and real-time control of geometric serial lines with residence time constraints", IISE Transactions **51**, 7, 709–728 (2019).

Wang, F., F. Ju and Y. Lu, "A study on performance evaluation and status-based decision for cyber-physical production systems", in "2017 13th IEEE Conference on Automation Science and Engineering (CASE)", pp. 1000–1005 (IEEE, 2017).

Wang, F., Y. Lu and F. Ju, "Condition-based real-time production control for smart manufacturing systems", in "2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)", pp. 1052–1057 (IEEE, 2018a).

Wang, J., Y. Hu and J. Li, "Transient analysis to design buffer capacity in dairy filling and packing production lines", Journal of Food Engineering **98**, 1, 1–12 (2010).

Wang, J., C. Pan, H. Hu, L. Li and Y. Zhou, "A cyclic scheduling approach to single-arm cluster tools with multiple wafer types and residency time constraints", IEEE Transactions on Automation Science and Engineering **16**, 3, 1373–1386 (2018b).

Waschneck, B., A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp and A. Kyek, "Deep reinforcement learning for semiconductor production scheduling", in "2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)", pp. 301–306 (IEEE, 2018).

Weyer, S., M. Schmitt, M. Ohmer and D. Gorecky, "Towards industry 4.0-standardization as the crucial challenge for highly modular, multi-vendor production systems", Ifac-Papersonline **48**, 3, 579–584 (2015).

Xie, X. and J. Li, "Modeling, analysis and continuous improvement of food production systems: A case study at a meat shaving and packaging line", Journal of food engineering **113**, 2, 344–350 (2012).

Yang, F., N. Wu, Y. Qiao, M. Zhou and Z. Li, "Scheduling of single-arm cluster tools for an atomic layer deposition process with residency time constraints", IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**, 3, 502–516 (2016).

Yoon, S., C. Jang and B. Han, "Nonlinear stress modeling scheme to analyze semiconductor packages subjected to combined thermal and hygroscopic loading", Journal of Electronic Packaging **130**, 2, 024502–1–024502–5 (2008).

Zhang, F., J. Song, Y. Dai and J. Xu, "Semiconductor wafer fabrication production planning using multi-fidelity simulation optimisation", International Journal of Production Research **58**, 21, 6585–6600 (2020).

Zhang, L., C. Wang, J. Arinez and S. Biller, "Transient analysis of bernoulli serial lines: Performance evaluation and system-theoretic properties", IIE Transactions **45**, 5, 528–543 (2013).

Zhu, Q., N. Wu, Y. Qiao and M. Zhou, "Scheduling of single-arm multi-cluster tools with wafer residency time constraints in semiconductor manufacturing", IEEE Transactions on Semiconductor Manufacturing **28**, 1, 117–125 (2014).

APPENDIX A

TRANSITION EQUATIONS IN CHAPTER 3

Following Section 3.2.1, the rest of the transitions are shown here. We start with the simple state $(0, 0, s_1, s_2)$ in the $(t+1)$-th cycle, where $s_1, s_2 = 0, 1$, meaning that the buffer is empty and the states for both machines are s_1 and s_2, respectively. All possible transitions to state $(0, 0, s_1, s_2)$ are illustrated as follows.

(a) $(0, 0, 0, 1)$ and $(0, 0, 0, 0)$. In these two cases, machine m_1 is down and buffer B is empty in cycle t. B will remain empty in the $(t+1)$-th cycle regardless of the state of machine m_2.

(b) $(1, \tau_1, 0, 1)$, for $\tau_1 = T_{min}, T_{min}+1, \cdots, T_{max}-1$. In this scenario, there is one part whose residence time has reached or exceeded T_{min} at the beginning of cycle t. Machine m_2 is up. Thus the part will be consumed by machine m_2. In addition, as m_1 is down, no part will enter the buffer, which will leave buffer B empty in cycle $(t+1)$.

(c) $(1, T_{max}-1, 0, 0)$. Such a state specifies that there is one part in the buffer with residence time $(T_{max}-1)$, and machine m_2 is down during the t-th cycle. Thus the part will have to be scrapped at the end of the t-th cycle, since it fails to be consumed by machine m_2 and its residence time will reach T_{max} at the beginning of the next cycle. Moreover, machine m_1 is down in cycle t, indicating that no new part enters buffer B. It leaves buffer B empty in cycle $(t+1)$.

Transitions regarding state $(0, 0, s_1, s_2)$ in cycle $(t+1)$ can be obtained using the following equation

$$\begin{aligned} x(0,0,s_1,s_2,t+1) =& x(0,0,0,1,t)P^{(1)}_{0,s_1}P^{(2)}_{1,s_2} \\ &+ x(0,0,0,0,t)P^{(1)}_{0,s_1}P^{(2)}_{0,s_2} \\ &+ \sum_{\tau_1=T_{min}}^{T_{max}-1} x(1,\tau_1,0,1,t)P^{(1)}_{0,s_1}P^{(2)}_{1,s_2} \\ &+ x(1,T_{max}-1,0,0,t)P^{(1)}_{0,s_1}P^{(2)}_{0,s_2}, \end{aligned} \tag{A.1}$$

where $s_1, s_2 = 0, 1$.

Next, state $(1, 0, s_1, s_2)$ in cycle $(t+1)$, for $s_1, s_2 = 0, 1$, shows that there is one part in buffer B at the beginning of the $(t+1)$-th cycle and its residence time is 0. It implies that the part is produced by m_1 at the end of cycle t. Thus, m_1 must be up in cycle t. Similar to $x(0, 0, s_1, s_2, t+1)$, the system evolution for $x(1, 0, s_1, s_2, t+1)$ can be represented as

$$\begin{aligned} x(1,0,s_1,s_2,t+1) =& x(0,0,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2} \\ &+ x(0,0,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2} \\ &+ \sum_{\tau_1=T_{min}}^{T_{max}-1} x(1,\tau_1,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2} \\ &+ x(1,T_{max}-1,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2}, \end{aligned} \tag{A.2}$$

where $s_1, s_2 = 0, 1$.

Then, consider the state $(1, i, s_1, s_2)$ in cycle $(t+1)$, for $1 \leq i \leq T_{max} - 1$ and $s_1, s_2 = 0, 1$. There is one part in the buffer but its residence time could be any feasible value except 0. It can be transferred from one of the following items.

(a) $(1, i-1, 0, 0)$. This means that in cycle t there is a part in buffer B with residence time $(i-1)$ and both machines are down.

(b) $(1, i-1, 0, 1)$, where $i - 1 < T_{min}$. This means that in cycle t there is a part in buffer B with residence time $(i-1)$ smaller than T_{min}, and m_1 is down. Though m_2 is up, this part cannot leave buffer B and enter m_2.

(c) $(2, \tau_1, 0, 1)$, for $\tau_1 = \max(i, T_{min}), \max(i, T_{min}) + 1, \cdots, T_{max} - 1$. The second part in buffer B has residence time $(i-1)$.

(d) $(2, T_{max} - 1, 0, 0)$. The second part in buffer B has residence time $(i-1)$.

Using operator $\Phi(n, \tau_1, \tau_2)$, the transition equation for state $(1, i, s_1, s_2)$, $1 \leq i \leq T_{max} - 1$, could be expressed as:

$$\begin{aligned} x(1, i, s_1, s_2, t+1) =& x(1, i-1, 0, 0, t) P^{(1)}_{0,s_1} P^{(2)}_{0,s_2} \\ &+ x(1, i-1, 0, 1, t) P^{(1)}_{0,s_1} P^{(2)}_{1,s_2} \mathbf{1}_{\mathbb{N}^+}(T_{min} + 1 - i) \\ &+ \sum_{j=\max(i, T_{min})}^{T_{max}-1} x(2, j, 0, 1, t) P^{(1)}_{0,s_1} P^{(2)}_{1,s_2} \Phi(2, j, i-1) \\ &+ x(2, T_{max} - 1, 0, 0, t) P^{(1)}_{0,s_1} P^{(2)}_{0,s_2} \Phi(2, T_{max} - 1, i-1), \end{aligned} \tag{A.3}$$

where $1 \leq i \leq T_{max} - 1$ and $s_1, s_2 = 0, 1$.

Transitions for the rest of states could also be obtained in a similar way, shown as below.

$$\begin{aligned} x(j, j-1, s_1, s_2, t+1) =& x(j-1, j-2, 1, 0, t) P^{(1)}_{1,s_1} P^{(2)}_{0,s_2} \\ &+ x(j-1, j-2, 1, 1, t) P^{(1)}_{1,s_1} P^{(2)}_{1,s_2} \mathbf{1}_{\mathbb{N}^+}(T_{min} + 2 - j) \\ &+ \sum_{i=\max(j-1, T_{min})}^{T_{max}-1} x(j, i, 1, 1, t) P^{(1)}_{1,s_1} P^{(2)}_{1,s_2} \Phi(j, i, j-2) \\ &+ x(j, T_{max} - 1, 1, 0, t) P^{(1)}_{1,s_1} P^{(2)}_{0,s_2} \Phi(j, T_{max} - 1, j-2), \end{aligned} \tag{A.4}$$

where $2 \leq j \leq N$, and $s_1, s_2 = 0, 1$;

$$
\begin{aligned}
x(N,i,s_1,s_2,t+1) =& x(N-1,i-1,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2} \\
&+ x(N-1,i-1,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&+ x(N,i-1,0,0,t)P^{(1)}_{0,s_1}P^{(2)}_{0,s_2} \\
&+ x(N,i-1,0,1,t)P^{(1)}_{0,s_1}P^{(2)}_{1,s_2}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&+ \sum_{j=\max(i,T_{min})}^{T_{max}-1} x(N,j,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2}\Phi(N,j,i-1) \\
&+ x(N,T_{max}-1,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2}\Phi(N,T_{max}-1,i-1) \\
&+ x(N,i-1,1,0,t)P^{(1)}_{1,s_1}P^{(2)}_{0,s_2} \\
&+ x(N,i-1,1,1,t)P^{(1)}_{1,s_1}P^{(2)}_{1,s_2}\mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i),
\end{aligned}
\tag{A.5}
$$

where $N \leq i \leq T_{max}-1$, and $s_1, s_2 = 0, 1$.

APPENDIX B

PROOF OF LEMMA 1 IN CHAPTER 3

Proof. Assume without loss of generality that the initial state for both system ρ and system ζ is $(\{\}, \phi_1^1, \phi_1^2)$.

Basis step. If $\phi_1^1 = 0$ in cycle $t = 1$, the state in cycle $t = 2$ for both system ρ and system ζ is $(\{\}, \phi_2^1, \phi_2^2)$. If $\phi_1^1 = 1$, the state in cycle $t = 2$ for both system ρ and system ζ is $(\{2\}, \phi_2^1, \phi_2^2)$. $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ hold for $t = 2$.

Induction step. Assume that $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ hold for $1 \leq t < k$, where $k = 2, 3, \cdots$. We want to prove that $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ hold for $t = k$. Three potential events may change φ^ρ and φ^ζ. They are the scrap, the consumption on machine m_1 and the production on machine m_2.

First we consider the scrap. Since $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ are satisfied in cycle $t = k - 1$ and both system ρ and system ζ have the same T_{max}, the scrap in cycle $t = k$ can happen to both system ρ and system ζ, or only to a unique part in system ζ. Therefore, $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ hold in cycle $t = k$ if scrap happens.

Next we consider the consumption on machine m_1. Assume for a contradiction that a new part enters the buffer of system ρ in cycle $t = k$ but no part enters the buffer of system ζ. The two systems share the same sample path, so system ζ is blocked and system ρ is not blocked in cycle $t = k - 1$. We have that $\left|\varphi^\zeta\right| = N^\zeta$ and $|\varphi^\rho| < N^\rho$ in cycle $t = k - 1$. It is a contradiction to $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ for $1 \leq t < k$. Assume for a contradiction that a new part enters system ζ but no part enters system ρ because of blockage, and then $|\varphi^\rho| + 1 < \left|\varphi^\zeta\right|$ in cycle $t = k$. From $|\varphi^\rho| + 1 < \left|\varphi^\zeta\right|$ in cycle $t = k$, we have that $|\varphi^\rho| < \left|\varphi^\zeta\right|$ in cycle $t = k - 1$. From the blockage that happens to system ρ, we have that $\left|\varphi^\zeta\right| < N^\zeta$ and $|\varphi^\rho| = N^\rho$ in cycle $t = k - 1$. From $N^\rho + 1 = N^\zeta$, we have that $|\varphi^\rho| + 1 > \left|\varphi^\zeta\right|$ in cycle $t = k - 1$, which is a contradiction to $|\varphi^\rho| < \left|\varphi^\zeta\right|$ in cycle $t = k - 1$. If one part enters the buffer of system ρ and one part enters the buffer of system ζ in cycle $t = k$, $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ still hold. Therefore, $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ hold in cycle $t = k$ when consumption on machine m_1 happens.

Finally, we consider the production on machine m_2. Assume for a contradiction that $\varphi^\rho \nsubseteq \varphi^\zeta$ in cycle $t = k$ after a part in system ζ is produced on machine m_2 at the end of cycle $t = k - 1$. Let the arrival time of the part in system ζ produced at the end of cycle $t = k - 1$ be φ. We have that $\varphi \in \varphi^\rho$ and $\varphi \in \varphi^\zeta$ during cycle $t = k - 1$, but $\varphi \in \varphi^\rho$ and $\varphi \notin \varphi^\zeta$ during cycle $t = k$. From the assumption, we have that in cycle $t = k - 1$, $\varphi = \varphi^\zeta_{[1]}$, and $\varphi = \varphi^\rho_{[j]}$ for some $j > 1$. Then $\varphi^\rho_{[1]} < \varphi^\rho_{[j]} = \varphi^\zeta_{[1]}$. $\varphi^\rho_{[1]} \in \varphi^\rho$ but $\varphi^\rho_{[1]} \notin \varphi^\zeta$ in cycle $t = k - 1$. It is a contradiction to $\varphi^\rho \subseteq \varphi^\zeta$ for $1 \leq t < k$. Assume for a contradiction that production on machine m_2 happens in system ρ but not in system ζ at the end of cycle $t = k - 1$, and then $|\varphi^\rho| + 1 < \left|\varphi^\zeta\right|$ at the beginning of cycle $t = k$. From the assumption, we have that $t - \varphi^\rho_{[1]} \geq T_{min}$ and $t - \varphi^\zeta_{[1]} < T_{min}$ in cycle $t = k - 1$. It implies that $\varphi^\zeta_{[1]} > \varphi^\rho_{[1]}$. Then $\varphi^\rho_{[1]} \in \varphi^\rho$ but $\varphi^\rho_{[1]} \notin \varphi^\zeta$ in cycle $t = k - 1$. It is a contradiction to $\varphi^\rho \subseteq \varphi^\zeta$ for $1 \leq t < k$. If one part in system ρ is produced on machine m_2 and one part in system ζ is produced on machine m_2 at the end of cycle $t = k - 1$, $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ still hold in cycle $t = k$. Therefore, $\varphi^\rho \subseteq \varphi^\zeta$ and $|\varphi^\rho| + 1 \geq \left|\varphi^\zeta\right|$ hold in cycle $t = k$ when production on machine m_2 happens.

After considering all the possible events in cycle $t = k$, we have that $\varphi^{\rho} \subseteq \varphi^{\zeta}$ and $|\varphi^{\rho}| + 1 \geq \left|\varphi^{\zeta}\right|$ hold in cycle $t = k$. The proof is completed. $\square$

APPENDIX C

TRANSITION EQUATIONS IN CHAPTER 4

We start with the simple state $(0,0,s_1^{sub},s_2^{sub})$ in the $(t+1)$-th cycle, for $s_1^{sub},s_2^{sub}=0,1$. It represents the system state that the buffer is empty and the states for both machines are s_1^{sub} and s_2^{sub}, respectively.

Transitions regarding state $(0,0,s_1^{sub},s_2^{sub})$ in cycle $t+1$ can be obtained using the following equation

$$\begin{aligned}
x(0,0,s_1^{sub},s_2^{sub},t+1) =& x(0,0,0,1,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ x(0,0,1,1,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ x(0,0,0,0,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{0,s_2^{sub}} \\
&+ x(0,0,1,0,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{0,s_2^{sub}} \\
&+ \sum_{\tau_1=T_{min}}^{T_{max}-2} x(1,\tau_1,0,1,t)(1-p^b)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ \sum_{\tau_1=T_{min}}^{T_{max}-2} x(1,\tau_1,1,1,t)p^s(1-p^b)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ x(1,T_{max}-1,0,0,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{0,s_2^{sub}} \\
&+ x(1,T_{max}-1,1,0,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{0,s_2^{sub}} \\
&+ x(1,T_{max}-1,0,1,t)P^{(1)}_{0,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ x(1,T_{max}-1,1,1,t)p^sP^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}},
\end{aligned} \tag{C.1}$$

for $s_1^{sub},s_2^{sub}=0,1$.

State $(1,0,s_1^{sub},s_2^{sub})$ in cycle $t+1$, for $s_1^{sub},s_2^{sub}=0,1$, represents the state that there is one part in buffer B and its residence time is 0. It implies the part is produced by machine m_1^{sub} at the end of cycle t. Thus, machine m_1^{sub} must be up in cycle t, and starvation does not happen to machine m_1^{sub}.

The system evolution for $x(1,0,s_1^{sub},s_2^{sub},t+1)$ can be represented as

$$\begin{aligned}
x(1,0,s_1^{sub},s_2^{sub},t+1) =& x(0,0,1,0,t)(1-p^s)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{0,s_2^{sub}} \\
&+ x(0,0,1,1,t)(1-p^s)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ \sum_{\tau_1=T_{min}}^{T_{max}-2} x(1,\tau_1,1,1,t)(1-p^s)(1-p^b)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}} \\
&+ x(1,T_{max}-1,1,0,t)(1-p^s)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{0,s_2^{sub}} \\
&+ x(1,T_{max}-1,1,1,t)(1-p^s)P^{(1)}_{1,s_1^{sub}}P^{(2)}_{1,s_2^{sub}},
\end{aligned} \tag{C.2}$$

for $s_1^{sub},s_2^{sub}=0,1$.

The rest of the transitions are shown as below.

$$
\begin{aligned}
&x(j, j-1, s_1^{sub}, s_2^{sub}, t+1) \\
&\quad = x(j-1, j-2, 1, 0, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \\
&\qquad + x(j-1, j-2, 1, 1, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+2-j) \\
&\qquad + x(j-1, j-2, 1, 1, t)(1-p^s)p^b P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(j-1-T_{min}) \\
&\qquad + \sum_{i=\max(j-1,T_{min})}^{T_{max}-2} x(j, i, 1, 1, t)(1-p^s)(1-p^b)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j, i, j-2) \\
&\qquad + x(j, T_{max}-1, 1, 1, 0, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \Phi(j, T_{max}-1, j-2) \\
&\qquad + x(j, T_{max}-1, 1, 1, 1, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j, T_{max}-1, j-2),
\end{aligned}
\tag{C.3}
$$

for $2 \leq j \leq N$, and $s_1^{sub}, s_2^{sub} = 0, 1$;

$$
\begin{aligned}
&x(j, i, s_1^{sub}, s_2^{sub}, t+1) \\
&\quad = x(j-1, i-1, 1, 0, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \\
&\qquad + x(j-1, i-1, 1, 1, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&\qquad + x(j-1, i-1, 1, 1, t)(1-p^s)p^b P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(i-T_{min}) \\
&\qquad + x(j, i-1, 0, 0, t)P^{(1)}_{0,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \\
&\qquad + x(j, i-1, 0, 1, t)P^{(1)}_{0,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&\qquad + x(j, i-1, 0, 1, t)p^b P^{(1)}_{0,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(i-T_{min}) \\
&\qquad + x(j, i-1, 1, 0, t)p^s P^{(1)}_{1,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \\
&\qquad + x(j, i-1, 1, 1, t)p^s P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&\qquad + x(j, i-1, 1, 1, t)p^s p^b P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(i-T_{min}) \\
&\qquad + \sum_{k=\max(i,T_{min})}^{T_{max}-2} x(j, k, 1, 1, t)(1-p^s)(1-p^b)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j, k, i-1) \\
&\qquad + x(j, T_{max}-1, 1, 1, 0, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \Phi(j, T_{max}-1, i-1) \\
&\qquad + x(j, T_{max}-1, 1, 1, 1, t)(1-p^s)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j, T_{max}-1, i-1) \\
&\qquad + \sum_{k=\max(i,T_{min})}^{T_{max}-2} x(j+1, k, 0, 1, t)(1-p^b)P^{(1)}_{0,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j+1, k, i-1) \\
&\qquad + \sum_{k=\max(i,T_{min})}^{T_{max}-2} x(j+1, k, 1, 1, t)p^s(1-p^b)P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j+1, k, i-1) \\
&\qquad + x(j+1, T_{max}-1, 0, 0, t)P^{(1)}_{0,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \Phi(j+1, T_{max}-1, i-1) \\
&\qquad + x(j+1, T_{max}-1, 0, 1, t)P^{(1)}_{0,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j+1, T_{max}-1, i-1) \\
&\qquad + x(j+1, T_{max}-1, 1, 0, t)p^s P^{(1)}_{1,s_1^{sub}} P^{(2)}_{0,s_2^{sub}} \Phi(j+1, T_{max}-1, i-1) \\
&\qquad + x(j+1, T_{max}-1, 1, 1, t)p^s P^{(1)}_{1,s_1^{sub}} P^{(2)}_{1,s_2^{sub}} \Phi(j+1, T_{max}-1, i-1),
\end{aligned}
\tag{C.4}
$$

for $2 \leq j \leq N-1$, $j \leq i \leq T_{max}-1$, and $s_1^{sub}, s_2^{sub} = 0, 1$;

$$
\begin{aligned}
&x(N, i, s_1^{sub}, s_2^{sub}, t+1) \\
&\quad = x(N-1, i-1, 1, 0, t)(1-p^s) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{0, s_2^{sub}} \\
&\qquad + x(N-1, i-1, 1, 1, t)(1-p^s) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&\qquad + x(N-1, i-1, 1, 1, t)(1-p^s) p^b P^{(1)}_{1, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(i-T_{min}) \\
&\qquad + x(N, i-1, 0, 0, t) P^{(1)}_{0, s_1^{sub}} P^{(2)}_{0, s_2^{sub}} \\
&\qquad + x(N, i-1, 0, 1, t) P^{(1)}_{0, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&\qquad + x(N, i-1, 0, 1, t) p^b P^{(1)}_{0, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(i-T_{min}) \\
&\qquad + \sum_{j=\max(i, T_{min})}^{T_{max}-2} x(N, j, 1, 1, t)(1-p^s)(1-p^b) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \Phi(N, j, i-1) \\
&\qquad + x(N, T_{max}-1, 1, 0, t)(1-p^s) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{0, s_2^{sub}} \Phi(N, T_{max}-1, i-1) \\
&\qquad + x(N, T_{max}-1, 1, 1, t)(1-p^s) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \Phi(N, T_{max}-1, i-1) \\
&\qquad + x(N, i-1, 1, 0, t) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{0, s_2^{sub}} \\
&\qquad + x(N, i-1, 1, 1, t) P^{(1)}_{1, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(T_{min}+1-i) \\
&\qquad + x(N, i-1, 1, 1, t) p^b P^{(1)}_{1, s_1^{sub}} P^{(2)}_{1, s_2^{sub}} \mathbf{1}_{\mathbb{N}^+}(i-T_{min}),
\end{aligned}
\tag{C.5}
$$

for $N \leq i \leq T_{max}-1$, and $s_1^{sub}, s_2^{sub} = 0, 1$.