REAL-TIME MAINTENANCE POLICIES IN MANUFACTURING SYSTEMS

by

Xi Gu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2016

To my family

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my co-advisor, Professor Jun Ni, for his guidance, care and encouragement through the years. I feel so honored and grateful to pursue my Ph.D. under his supervision. Without his continuous support, this work will not be possible. My sincere thanks also go to my co-advisor, Professor Yoram Koren, for his guidance and insights on my research. Special thanks to Dr. Xiaoning Jin, the discussions with whom have generated new ideas in my dissertation. Many thanks to Professor Mariel Lavieri and Professor Kazuhiro Saitou for spending their precious time serving on my dissertation committee.

I am very grateful for the collaborative environment at the S. M. Wu Manufacturing Research Center. Thanks to George Qiao, Dr. Seungchul Lee, Dr. Shuhuai Lan, Dr. Yang Li, Kai Chen, Baoyang Jiang, Hao Lei, Xinran Liang, Xun Liu, Yangbing Lou, Huanyi Shui, Xin Weng and Kevin Wilt for their help and friendship.

Finally, I would like to dedicate this work to my parents and parents in law. Their unconditional support and encouragement is the source of my strength. Especially, I would like to thank my wife, Weihong, for always believing in me and bringing sunshine into my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Real-Time Maintenance Policies in Manufacturing Systems

by

Xi Gu

Co-Chairs: Jun Ni & Yoram Koren

Effective and timely maintenance actions can sustain and improve both system availability and product quality in automated manufacturing systems. However, arbitrarily stopping machines for maintenance will occupy their production time and, in turn, introduce production losses into the system. Real-time maintenance decision-making in manufacturing systems is complex because it requires the integration of multiple sources of information, such as the system configuration, current machine health condition, real-time buffer level, and system throughout target.

The research presented in this thesis aims at developing tools to support real-time maintenance decision-making in complex manufacturing systems. First, the concept of passive maintenance opportunity window (PMOW) is introduced, which is defined as the machine idle-time induced by the propagation of downtime of the other machines in the system. Real-time PMOWs are predicted in manufacturing systems with serial and non-serial structures. Second, the concept of active maintenance opportunity window (AMOW) is proposed so that machines can be strategically shut down for preventive maintenance (PM) while the system throughput requirement can be still satisfied. A system decomposition method is developed to investigate the transient behavior of manufacturing systems with different configurations, based on which real-time AMOWs are estimated. Third, maintenance policies are examined by integrating the real-time buffer levels and machine degradation profiles. The system throughputs under a control-limit policy and a Markov Decision Process approach are evaluated and compared. Last, the propagation of downtime in serial-parallel manufacturing systems is studied, based on which three resilience metrics (i.e., production loss, throughput recovery time, and total underproduction time) are defined and evaluated. An optimization problem is formulated for the design of a resilient manufacturing system.

The model and methodology developed in this dissertation provide managerial insights on conducting maintenance operations in complex manufacturing systems. The effectiveness of the proposed model and algorithms is validated by case studies with simulations, and measurements in an automotive assembly plant.

CHAPTER I

Introduction

1.1 Motivation

To compete successfully in the market place, leading manufacturing companies are pursuing effective maintenance operations [1]. Effective and timely maintenance policies can sustain and improve both system availability and product quality. However, if maintenance policies are not well-developed, then machine downtime may increase due to the machine degradation and the machine stoppage time for performing maintenance, which incurs huge cost. For example, in a typical automotive assembly line, one minute of downtime could cost as much as \$20,000 [2]. Surveys show that, in the US manufacturing industry, one-third of all maintenance expenditures are wasted due to the inefficient and ineffective utilization of maintenance resources [3].

Typically, large and complex manufacturing systems consist of 30 to 120 machines that produce one or multiple products, and maintenance decision-making in these systems is not a trivial task. First, many factors need to be considered in the maintenance operations, including: (a) current machine conditions (e.g., working, down, idle), (b) maintenance schedule, (c) machine degradation profiles, (d) system configurations, (e) costs of maintenance resources (e.g., labor, spare parts), and (f) throughput target [4]. Second, there are different types of maintenance. Unscheduled maintenance, such as repairs, reacts to the random breakdowns of machines, while scheduled maintenance, such as preventive maintenance (PM), can restore the machine health condition and prevent breakdowns before they actually occur.

Although maintenance can keep machines and equipment operating in good condition, arbitrarily stopping machines for maintenance could cause interruption to regular production and affect system throughput. Therefore, a conflict arises between the production manager and the maintenance manager: the former wants to keep production lines operating to satisfy daily production targets but with little concern about the machine health condition, while the latter wants sufficient stoppage time so that adequate maintenance tasks can be completed. A traditional way to resolve the conflict is to schedule maintenance tasks during non-production shifts or weekends [5]. Such practice is sometimes difficult to meet the required system performance in terms of cost-effectiveness and production efficiency. First, it may incur overtime labor cost. The overtime salaries for the maintenance crews are usually higher than their regular salaries. Second, in large and complex manufacturing systems, there are usually too many maintenance tasks in the queue that not all of them can be completed during scheduled non-production time. Third, such PM policies are usually stationary, and thus they fail to respond quickly to the real-time condition of the systems. Therefore, instead of performing PM only during non-production time, opportunities for PM during regular production time need to be investigated.

To this end, the concept of maintenance opportunity window (MOW) was introduced by Chang et al. [6] as a time window for a specific machine being purposefully shut down to do PM without substantially impacting the throughput of the system. In this research, we further classify the concept of MOWs into two different types. For the first type, machines can be purposefully shut down for maintenance during production by leveraging the surrounding buffers to keep the desired production rate. Such kind of maintenance opportunities is defined as active maintenance opportunity windows (AMOWs). The second type of MOWs is created by the effects of the downtime on other machines. If the repair time for the failed machine is long enough, the downtime will eventually propagate to the surrounding machines, forcing the upstream machines to be blocked and the downstream machines to be starved. Such blockage/starvation time can also be viewed as opportunities for maintenance. We define such MOWs as passive maintenance opportunity windows (PMOWs), which come from the idle duration caused by the downtime of other machines. To find both AMOWs and PMOWs requires investigation on the transient behavior of the manufacturing system when planned downtime (e.g., maintenance) or unplanned downtime (e.g., failure) occurs.

In previous work, Chang et al. [6] used a continuous flow model to investigate the maintenance opportunities in serial lines, and further accounted for the location of the slowest station [7]. However, the model developed in the work had two limitations. First, it assumed that when one machine was down for PM, the other machines would not fail. Second, only serial lines were considered. Therefore, research is still needed to address the transient behavior of systems with unreliable machines and complex configurations.

1.2 Literature Review

1.2.1 Maintenance policies

1.2.1.1 Types of maintenance policies

Maintenance policies have been extensively studied in the literature. These policies are built on the analysis of the criticality (e.g., frequency, downtime) of failure events. For example, Fig. 1.1 shows the frequencies of subsystems of CNC lathes where the failures occur, and the downtime caused by these failures, with the data collected in [8]. In Fig. 1.1, different regions may correspond to different maintenance policies [9]. Basically, these maintenance policies can be categorized into corrective maintenance (CM) and preventive maintenance (PM). According to MIL-STD-721B, corrective maintenance refers to all actions performed as a result of failure, to restore an item to a specific condition, while preventive maintenance includes all actions performed in an attempt to retain an item in specific condition by providing systematic inspection, evaluation, and prevention of incipient failures [10]. In general, the cost of CM is three to four times higher than that of PM [11]. Therefore, PM is usually preferred in real practice, especially when the failure is critical.

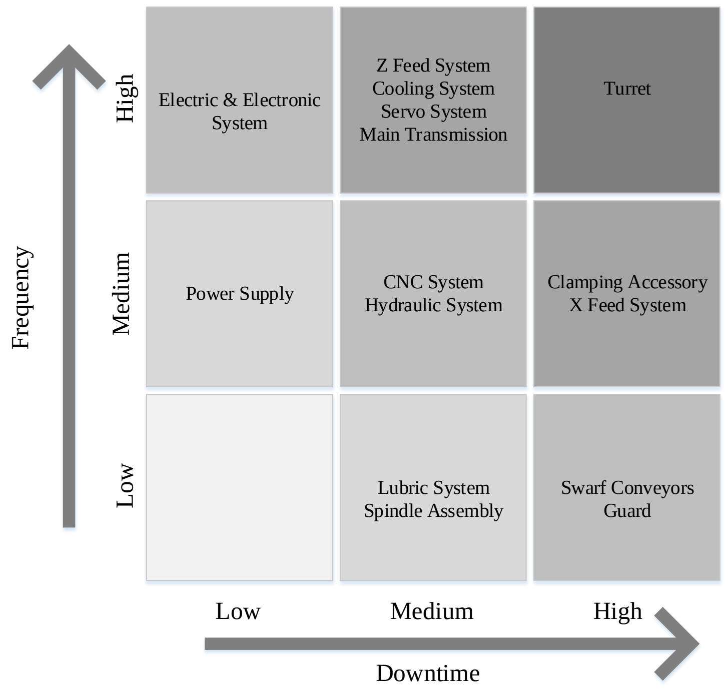


Figure 1.1: The criticality of failures of lathe machines

Moreover, PM can be classified into time-based maintenance (TBM) and condition-based maintenance (CBM) [12]. TBM is a periodic maintenance based on the assumed failure behavior such as mean time between failures (MTBF), while CBM recommends maintenance decisions dynamically based on the information collected through condition monitoring. A CBM program can do diagnosis, prognosis, or both [13]. Usually, CBM is more cost-effective than TBM because it preventively maintains the system only when necessary, and thus can save maintenance cost and improve system availability [14].

The existing CBM models in literature can be classified into two types [15]. The first type of models treats the degradation state of machines as continuous. It is assumed that the reliability function follows certain distribution (e.g., Weibull, exponential; see monographs [16–18]). Optimal maintenance or inspection policies are developed in order to maximize the system availability or to reduce the total cost. For example, Banjevic et al. [19] presented a control-limit policy for a deteriorating system subject to inspections at discrete time points. Grall et al. [20] developed a multi-level control-limit maintenance policy for a stochastically and continuously deteriorating single-unit system. Dieulle et al. [21] investigated the optimal sequential replacement and inspection policy for a system whose deterioration was a Gamma process.

The second type of models described the machine degradation as a Markov process [22]. For example, Stadje and Zuckerman [23] used optimal control to find the optimal threshold condition and the degree of repair for a unit whose degradation process was a discrete-time Markov chain. Chen and Trivedi [24] developed a queueing model to derive the closed-form analytical results. Chen and Trivedi [25] used a semi-Markov Decision Process to jointly optimize the inspection rate and maintenance policy. Xiang [26] used a discrete-time Markov chain model to jointly optimize the control chart and maintenance policy.

1.2.1.2 Maintenance policies in multi-unit systems

Although these CBM policies shed light on the maintenance decision making, they focus primarily on single-unit systems. For a multi-unit system, interactions exist between units, and they can be classified into three different types: economic dependence, structural dependence, and stochastic dependence (See reviews [27–30]). Economic dependence implies that group maintenance actions either save cost or result in higher costs as compared to individual maintenance. Structural dependence applies if components structurally form a part, so that performing maintenance on one component implies maintenance on all components. Stochastic dependence refers to the situation that the condition of one component influences the lifetime distribution of the other components in the system [30].

In a manufacturing system, the machines are usually economically dependent, and the cost of system downtime may be much higher than the maintenance cost. Therefore, there is often a great potential for cost savings by implementing opportunistic maintenance policies [31]. For example, when one machine is down for maintenance, we may have an opportunity to shut down another machine for maintenance with a reduced set-up cost. This type of opportunity is called downtime opportunity [29]. Another opportunity is that when the inventory levels in the buffers are high, appropriate maintenance tasks can be performed on the upstream machines without affecting the end-of-line system throughput [6].

Maintenance policies in multi-unit systems with economic dependence have gained more attention in the recent literatures. For example, Barbera et al. [32] investigated the optimal maintenance policy in a two-unit system by conditioning on the total deterioration of both units. Castanier et al. [33] developed an opportunistic replacement policy for a two-unit system in series to reduce the total maintenance cost. Tian and Liao [34] developed a proportional hazards model (PHM) based CBM policy by assuming the economic dependence of the units. From the perspective of manufacturing systems, Van der Duyn Schouten and Vanneste [35] studied the maintenance policies for the supplier in a supplier-buffer-customer system. Ambani et al. [36] used a continuous-time Markov chain model to develop CBM policies in serial production lines without intermediate buffers. Lee et al. [37] developed a Markovian model to find the optimal inspection policy for a manufacturing system with two machines in series or in parallel. However, the work did not represent the detailed dynamics for a manufacturing system, especially when the system is large and complex.

1.2.2 Manufacturing systems

1.2.2.1 System modeling

To study the effect of maintenance on the system performance, we need to first study the dynamics of manufacturing systems. Modelling, analysis and design of stochastic manufacturing systems have been studied for decades. Different models have been developed, such as continuous-flow model, queueing model, and discrete time Markov chain model (See reviews [5, 38–41], and monographs [42–46]).

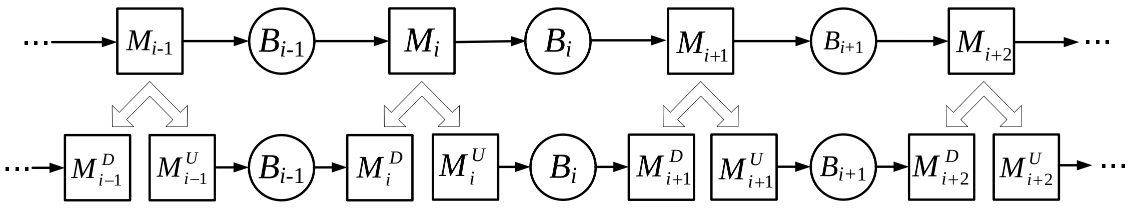
The building block to study manufacturing systems is a two-machine-one-buffer (2M1B) system. Following Gershwin and Berman [47], various two-machine-one-buffer models have been developed, with assuming different probability distribution of the processing times and machine reliabilities. In these models, the system state is usually represented by the distribution of buffer levels, and its evolution depends on the status of the machines. Li et al. [48] pointed out that the exact analysis only existed for 2M1B systems, and compared the performance of six different 2M1B models. Tan and Gershwin [49] developed a methodology to analyze general two-stage Markovian continuous flow systems with a finite buffer. The flexibility of this methodology helps analyze a wide range of systems by specifying the transition rates and the flow rates associated with each state of each stage, and the applicability of the methodology was demonstrated through examples with different problem settings or different parameter distributions [50].

It is always assumed that in a 2M1B system, the upstream machine is never starved and the downstream machine is never blocked. However, for a large manufacturing system, some machine may be starved by its immediately upstream buffer or blocked by its immediately downstream buffer. Therefore, the system state depends on the levels of all the buffers in the system, and the number of states increases dramatically as the number of buffers increases, making exact analysis computationally intractable. Approximation methods have been developed to analyze the system. Most of the approximation methods are based on either machine decomposition or aggregation. For the decomposition method [51], one machine M_i is decomposed into two pseudo-machines M_i^D and M_i^U, where M_i^D is the downstream machine in the 2M1B system $M_{i-1}^U - B_{i-1} - M_i^D$ and M_i^U is the upstream machine in the 2M1B system $M_i^U - B_i - M_{i+1}^D$ (Fig. 1.2(a)). For a continuous flow model, the parameters of the decomposed machines can be solved by DDX algorithm [52]. For the aggregation method, the idea is to aggregate the structures on the upstream and downstream of B_i as two virtual machines, namely, M_i^f and M_{i+1}^b (Fig.1.2(b)), where the superscripts 'f' and 'b' stand for forward and backward aggregation, respectively [46], and the characteristics of the aggregated machines can be estimated by using recursive algorithms.

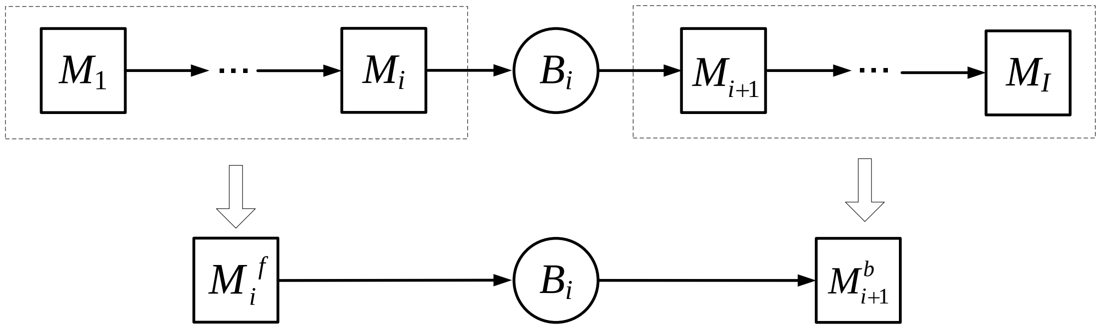
It can be seen from above that, both the decomposition and aggregation methods share the same idea: to view the system from buffers' perspective and represent the system as several 2M1B systems to reduce the dimensionality. However, both methods assume that the system is in the steady state, and are used to obtain the average performance of a manufacturing system [53].

1.2.2.2 Analysis of transient behavior

Unlike the well-studied steady state performance, transient analysis of manufacturing systems has been relatively unexplored. Transient analysis is important if we are interested in the system performance before entering the steady state, or that during a finite interval [54]. Narahari and Viswanadham [54] used a queueing model to analyze the transient behavior of a fail-repair model of a two-machine system without buffers. Mitra [55] developed a fluid model to study the transient behavior of a two-stage system coupled by a buffer, where each stage can have multiple parallel machines. Meerkov and Zhang [56] built a Bernoulli model for a two-machine-one-



(a) Decomposition method



(b) Aggregation method

Figure 1.2: Approximation methods for complex system analysis

buffer system, and found that the second largest eigenvalue of the transition matrix determined transient characteristics of the system. Furthermore, Zhang et al. [57] developed an aggregation method to extend the transient analysis from 2M1B system to serial production lines with Bernoulli machines. Despite the initial work, transient analysis still remains an open area of research, and more studies are required to analyze systems with complex structures, and machines with different reliability models [5].

1.3 Research Objectives

In this research, we will study the propagation of the planned or unplanned downtimes in manufacturing systems. The analysis will provide managerial insights for manufacturing system operations and design. The fundamental challenges and research objectives of this dissertation are summarized as follows:

- Develop a systematic approach to study the transient behavior of manufacturing systems when maintenance is performed. The model should be applicable to analyze the propagation of multiple downtime events in manufacturing systems with different configurations (e.g., serial-parallel, assembly/disassembly, closed-loop).

- Develop real-time maintenance policies in manufacturing systems by integrating the real-time health condition of the machines and the system production information (e.g., buffer contents, short-term production requirement). In addition, some short-term maintenance decision support tools, such as PMOW and AMOW, need to be developed. This objective is from system operations perspective. These developed tools will help maintenance crews make more effective real-time maintenance decisions without affecting the system throughput.

- Design manufacturing systems for resilience, which is the system capability to tolerate the propagation of downtime events. This objective is from system design perspective. We need to quantitiatively investigate the resilience performance of manufacturing systems (i.e., the capability to tolerate planned or unplanned downtimes) by defining resilience metrics. The analysis will help system designers to determine the optimal level of redundancy or flexibility that should be built in the system.

1.4 Outline

The rest of this dissertation is organized as follows.

In Chapter II, passive maintenance opportunity windows (PMOWs) are predicted in manufacturing systems. A deterministic model is built in order to study the PMOW in manufacturing systems with serial or non-serial configurations, and when one or multiple downtime events occur in the system. A case study in an automotive plant is presented to illustrate the results.

In Chapter III, active maintenance opportunity windows (AMOWs) are estimated in manufacturing systems with unreliable machines and finite buffers. A two-period model is developed to calculate the production losses during maintenance and that after maintenance. Numerical case studies are presented to illustrate the advantages of the AMOW-based maintenance policies.

In Chapter IV, we investigate the maintenance policies in manufacturing systems by integrating machine degradation profiles. First, a system decomposition method is developed to evaluate a control-limit policy in multi-stage manufacturing systems. Then, an Markov Decision Process (MDP) approach is used to investigate the optimal maintenance policy.

In Chapter V, three resilience metrics – production loss (PL), throughput recovery time (TRT), and total underproduction time (TUT) – are defined. We investigate how these resilience metrics can be affected by the system built-in redundancy or flexibility. A design optimization problem is formulated to illustrate how these resilience metrics can be considered in the system design stage.

In Chapter VI, the conclusions and contributions of this dissertation are summarized, and future work is proposed.

CHAPTER II

Prediction of Passive Maintenance Opportunity Windows in Manufacturing Systems

2.1 Introduction

Maintenance optimization in multi-unit systems has been studied for decades. Cho and Parlar [28] reviewed papers discussing this problem in the systems where the components might or might not depend on each other, economically or stochastically, and Nicolai and Dekker further considered the structural dependence [30]. Some group maintenance and opportunistic maintenance policies were reviewed by Wang [31]. However, most of the literatures focus on the reliability of a single component and make assumptions about the dependency, while not integrating the detailed system information, such as system configuration and real-time buffer levels. Such information is directly related to how the downtime could propagate to its surrounding machines [58], and hence it plays an important role for developing short-term maintenance policies. Chang et al. [6] used a continuous flow model to investigate the maintenance opportunities in serial lines, and further accounted for the location of the slowest station [7]. Liu et al. [59] investigated the cost of downtime incidents in serial lines, and found that once the downtime on the failure machine exceeded a threshold, the cost would increase linearly with the downtime. This work provides us with insights on how the line physics changes when machine failures occur. However, the above-mentioned works only dealt with serial lines, investigating the problem in complex systems with both serial and non-serial configurations makes it more interesting and challenging. Gu et al. [60] developed an equivalent pseudo serial line method to decompose complex systems so that the problem could be equivalently analyzed in multiple serial lines.

In this chapter, we investigate the PMOW prediction on the bottleneck machine in a complex manufacturing system. The complex system in this work is where the downtime of one machine may propagate to another machine through different branches in the system. As introduced in Chapter I, PMOW is the starvation/blockage time brought by the downtime of other machines in the system. The bottleneck machine is the one whose stoppage could interrupt the system throughput in a strongest manner [61]. Therefore, in practice, it is not preferred to actively shut down bottleneck machines, making the prediction of PMOWs on them more important. Short-term bottleneck machines can be detected by the "turning point" method [62], and Chang et al. [7] showed that, the slowest machine in the system satisfied the concept of turning point and thus could be regarded as the bottleneck machine. Moreover, it is claimed in [59] that, the stoppage of the slowest machine will result in permanent system production losses, demonstrating that the slowest machines are the ones we would least like to stop. Therefore, in this chapter, we treat the bottleneck machine and slowest machine as equivalent.

PMOWs on the bottleneck machine come from the occurrence of random failures on other machines in the system. To determine whether a failure on a non-bottleneck machine will make the bottleneck machine idle, we calculate the critical downtime first. If the downtime on one machine is shorter than its critical downtime, it will not force the bottleneck machine idle. For discrete production lines, Gu et al. [60] calculated the critical downtime analytically in two-machine-one-buffer (2M1B) lines as a building block. They also developed an aggregation method to calculate it in serial lines and extend it to complex systems by using the equivalent pseudo serial line method.

Furthermore, as an event-driven decision support tool, the prediction of PMOW should respond rapidly if there is updated failure information. In this chapter, we not only consider PMOWs under a single failure, but also investigate how to update PMOWs when multiple failures occur. With the prediction and updating of PMOWs, maintenance crews can get prepared in advance, so that proper preventive maintenance tasks can be carried out more effectively.

The remainder of the chapter is organized as follows. In Section 2.3, we investigate the *critical downtime* (DT^*) for machines in serial lines and use the *equivalent pseudo serial line* (EPSL) method to derive DT^* in complex systems. Based on the critical downtime, a mathematical model is developed in Section 2.4 to predict the PMOW on the bottleneck machine. The PMOW is analyzed first under a single failure in complex systems, and then under multiple failures in serial lines. Finally, these results are combined to predict and update PMOWs under multiple failures in complex systems. Section 2.5 presents two numerical case studies where the PMOW prediction model is illustrated and its effectiveness is validated through simulations and real plant data. Section 2.6 contains summaries and future work.

2.2 Model and Assumptions

The assumptions for the manufacturing systems discussed in this chapter are described as follows:

(1) Machine M_i has a constant cycle time T_i.

(2) A machine may have multiple immediately upstream buffers, and it is starved if one of its immediately upstream buffers is empty. The first machine is never starved.

(3) A machine may have multiple immediately downstream buffers, and it is blocked if

one of its immediately downstream buffers is full. The last machine is never blocked.

(4) Buffer B_i has a capacity C_i, and a real-time buffer level $N_i(0)$.

(5) The travelling time at buffers is negligible.

(6) Each buffer can only have one immediately upstream machine and one immediately downstream machine.

Note that, assumption (5) is made in order to simplify the mathematical expressions. The algorithm developed in this chapter can be easily extended to cases that consider the buffer travelling time. Assumptions (2), (3) and (6) define the type of the systems that we discuss in this chapter. Especially, it is assumed that there are no parallel machines in the systems, so that the bottleneck machine does not have partial starvation or blockage.

2.3 Calculation of the Critical Downtime in Manufacturing Systems

The critical downtime (DT^*) is defined as the maximum time that one machine can be down without making the bottleneck machine idle. In Section 2.3.1, the critical downtime in a serial line is analyzed. In Section 2.3.2, an equivalent pseudo serial line (EPSL) method is developed. Finally, Section 2.3.3 summarizes the calculation of the critical downtime in complex systems.

2.3.1 Calculation of the critical downtime in a serial line

We consider an I-machine-$(I-1)$-buffer serial line in Fig. 2.1, where machine M_i is located either on the upstream or the downstream of the bottleneck machine M_b. Backward and forward aggregation methods have been developed in [60] to analytically derive DT_i^*, the critical downtime of machine M_i, when M_i is on the upstream or downstream of machine M_b, respectively.

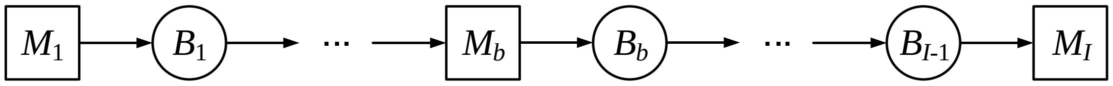


Figure 2.1: An I-machine-$(I-1)$-buffer serial line

The results are:

$$DT_i^* = \begin{cases} \sum_{j=i}^{b-1} N_j(0)T_b - \sum_{j=i}^{b-1} T_j & \text{if } 1 \le i < b \\ 0 & \text{if } i = b \\ \sum_{j=b}^{i-1} (C_j - N_j(0))T_b & \text{if } b < i \le I \end{cases} \tag{2.1}$$

where C_i and $N_i(0)$ are the capacity and real-time contents of buffer B_i.

From Equation (2.1), although the expressions of DT_i^*'s are different based on the relative locations of M_i and M_b, their structures are similar. If M_i is on the upstream of M_b, then $\sum_{j=i}^{b-1} N_j(0)T_b$ is the time for M_b to process all the parts between machine M_i and machine M_b, until M_b becomes starved; and $\sum_{j=i}^{b-1} T_j$ is the time it will take to allow M_b to resume working after M_i is restored from the downtime. Similarly, if M_i is on the downstream of M_b, then $\sum_{j=b}^{i-1}(C_j - N_j(0))T_b$ is the time for M_b to feed the buffer space between machine M_i and M_b before M_b is blocked, which is the same as the critical downtime since resuming M_i will immediately make M_b unblocked. These similarities lead to the construction of an equivalent pseudo serial line (EPSL), which will be discussed next.

2.3.2 Construction of equivalent pseudo serial lines

In a serial line, one machine is either on the upstream or downstream of another machine, while in a complex system, the relative location of two machines may not be so straightforward. Therefore, a unified form calculation of DT_i^*'s is needed, regardless of whether M_i is on the upstream or downstream of M_b.

In this section, we develop an equivalent pseudo serial line (EPSL) method to obtain DT_i^*'s. If M_i is the machine whose critical downtime needs to be calculated, then an EPSL is constructed in a way such that machine M_i is the first machine and the bottleneck machine M_b is located at the end of the line. Based on the result in [6], in the EPSL, the critical downtime for machine M_i can be calculated by

$$DT_i^* = T_i^{cons} - T_i^{res} \tag{2.2}$$

where T_i^{cons} (time to consume) is the time it takes M_b to process all the buffer contents between M_i and M_b, and T_i^{res} (time to resume) is the time it takes for the first part processed by the repaired M_i to travel to M_b.

In order to construct an EPSL where the bottleneck machine is the last machine, the machines that are located on the downstream of the bottleneck machine need to be reversed to its upstream in the EPSL. For example, in order to analyze the effect of the downtime of M_I in the serial line in Fig. 2.1, $M_b - B_b - \cdots - B_{I-1} - M_I$ are reversed in the pseudo line $M_I^r - B_{I-1}^r - \cdots - B_b^r - M_b^r$, as shown in Fig.2.2.

If a failure occurs on machine M_i $(i > b)$ in the original line, and the failure is long enough, then machines $M_{i-1}, M_{i-2}, \ldots, M_b$ will be blocked successively; and if the failure occurs on the reversed machine M_i^r, then machines $M_{i-1}^r, M_{i-2}^r, \ldots, M_b^r$ will be starved successively. By an appropriate selection of the parameters of the pseudo line, the starvation of M_b^r will be equivalent to the blockage of M_b.

The real line and its EPSL are equivalent in terms of the time to consume and the time to resume. More specifically, the time to fill the empty space in buffers $B_{i-1}, \ldots, B_b$ equals the time to consume the parts in the reversed buffers $B_{i-1}^r, \ldots, B_b^r$; and the time to make M_b unblocked after the downtime of M_i ends is the same as the time to make M_b^r unstarved after M_i^r resumes running. Consequently, the critical downtimes in these two lines are also equivalent.

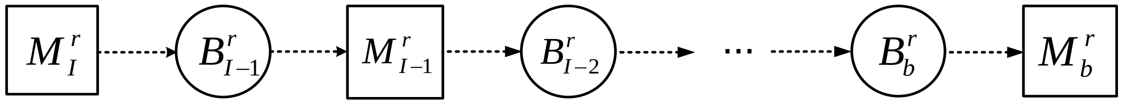


Figure 2.2: A pseudo line for machines on the downstream of M_b

This result is stated in the following theorem.

Theorem 2.1. *The effect of the failure occurring on the downstream of a bottleneck machine can be equivalently analyzed by that on the corresponding upstream machine in the EPSL, through the following transformation:*

$$N_i^r(0) = C_i(0) - N_i(0) \qquad \text{for } i = b, ..., I-1 \tag{2.3a}$$

$$T_i^r = 0 \qquad \text{for } i = b+1, ..., I \tag{2.3b}$$

where $N_i^r(0)$ is the initial level of the reversed buffer B_i^r and T_i^r is the cycle time of the reversed machine M_i^r in the EPSL.

Proof. See Appendix A. □

A numerical example will be given in Section 2.5.1 to illustrate the construction of an EPSL. The calculation of the critical downtime in such a line serves as a basis for the calculation of the critical downtime in a complex system.

2.3.3 Calculation of the critical downtime in a complex system

In this chapter, we define a complex system as follows:

Definition 2.1. A complex manufacturing system is a system where there exist two machines that are connected through multiple routes.

Based on the definition, in a complex manufacturing system, two machines M_i and M_b may be connected through multiple equivalent serial lines (which may be either real or pseudo), and thus the downtime on M_i can propagate to M_b through these different lines.

One example of such systems is illustrated in Fig. 2.3, where machine M_1 is a cutting machine, machines M_2 to M_5 are stamping machines, and the bottleneck machine M_b is a welding machine which joins the parts from both lines together. In such a system, the two stamping lines share the same cutting machine M_1, and if a failure occurs on M_1, the impact of failure can propagate to M_b through both lines $l_{1,1}$ and $l_{1,2}$.

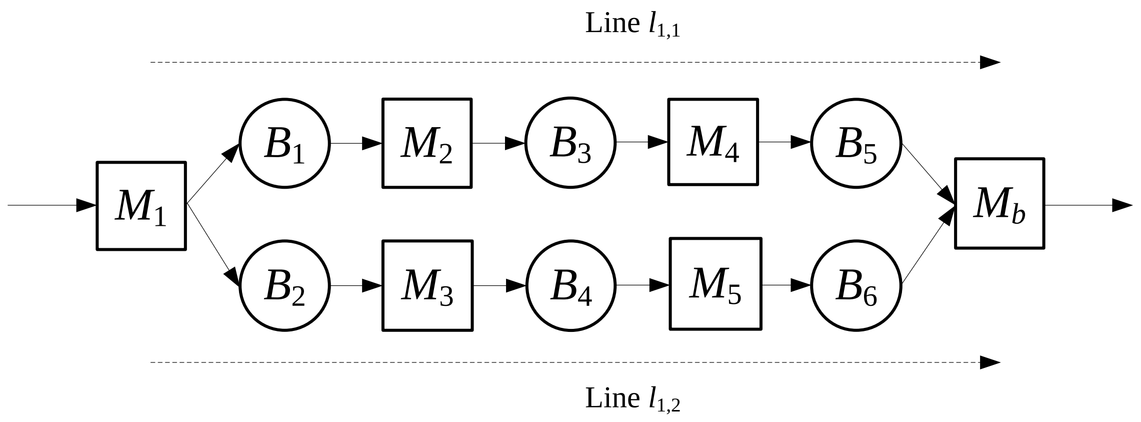


Figure 2.3: An example of non-serial manufacturing systems

Generally, if there are K_i equivalent serial lines connecting M_i and M_b, then in line $l_{i,k}$ $(k = 1, \dots, K_i)$, the critical downtime $DT^*_{i,k}$ can be calculated as $DT^*_{i,k} = T^{cons}_{i,k} - T^{res}_{i,k}$. Then, the critical downtime for the system is the minimum of all $DT^*_{i,k}$'s, i.e.,

$$DT^*_i = \min_{k=1,\dots,K_i} DT^*_{i,k} \tag{2.4}$$

Therefore, if the actual downtime is within DT^*_i , then the bottleneck machine will not be idle in any of the equivalent lines.

2.4 Prediction of Passive Maintenance Opportunity Windows

The above method of critical downtime calculation enables us to predict the passive maintenance opportunity window (PMOW) on the bottleneck machine when failures occur in the system. In this section, the PMOW is first predicted in a single failure scenario, and then updated when sequential failures occur.

2.4.1 Prediction of PMOWs under a single failure in a complex system

Consider that at time 0, a failure occurs on machine M_i and will last for DT_i. Based on Section 2.3.3, the critical downtime of M_i can be calculated as $DT_i^* = \min_{k=1,\dots,K_i}(T_{i,k}^{cons} - T_{i,k}^{res})$. If the actual downtime does not exceed the critical downtime, i.e., $DT_i \leq DT_i^*$, the bottleneck machine M_b will not be starved or blocked. However, if $DT_i > DT_{i,k}^*$ for any $k = 1, \dots, K_i$, M_b may have an idle interval resulting from line $l_{i,k}$. To predict the PMOW on M_b requires the joint consideration of the idle intervals in all of these lines.

First we consider these idle durations independently. In line $l_{i,k}$, M_b will be idle from time $T_{i,k}^{cons}$ to $DT_i + T_{i,k}^{res}$, i.e.,

$$PMOW_k = \left[T_{i,k}^{cons}, DT_i + T_{i,k}^{res}\right) (k = 1, \dots, K_i) \tag{2.5}$$

Also, the length of $PMOW_k$ is

$$|PMOW_k| = \max\left(0, DT_i + T_{i,k}^{res} - T_{i,k}^{cons}\right) = \max\left(0, DT_i - DT_{i,k}^*\right) \tag{2.6}$$

Equation (2.6) demonstrates that if DT_i is greater than the critical downtime $DT_{i,k}^*$, M_b will have an idle duration of $DT_i - DT_{i,k}^*$ in line $l_{i,k}$. It agrees with the result claimed in serial lines [59].

Next, we consider the PMOW resulted from lines $l_{1,1}, \dots, l_{i,K_i}$ jointly. First, these lines are arranged in the ascending order of $T_{i,k}^{cons}$, such that $T_{i,(1)}^{cons} \leq \cdots \leq T_{i,(K_i)}^{cons}$. Then the earliest possible idle interval on machine M_b may occur in line $l_{i,(1)}$, i.e.,

$$PMOW_{(1)} = \left[T_{i,(1)}^{cons}, DT_i + T_{i,(1)}^{res}\right) \tag{2.7}$$

Since M_b has an idle duration of $|PMOW_{(1)}|$, which starts before time $T_{i,(2)}^{cons}$, the additional idle time that is caused by the downtime in $l_{i,(2)}$ will start at $T_{i,(2)}^{cons} + |PMOW_{(1)}|$, while it still ends at $DT_i + T_{i,(2)}^{res}$. Therefore, when the joint effects of lines $l_{i,(1)}$ and $l_{i,(2)}$ on machine M_b are considered, the PMOW on machine M_b is

$$PMOW_{(2)} = PMOW_{(1)} \bigcup \left[T_{i,(2)}^{cons} + |PMOW_{(1)}|, DT_i + T_{i,(2)}^{res}\right) \tag{2.8}$$

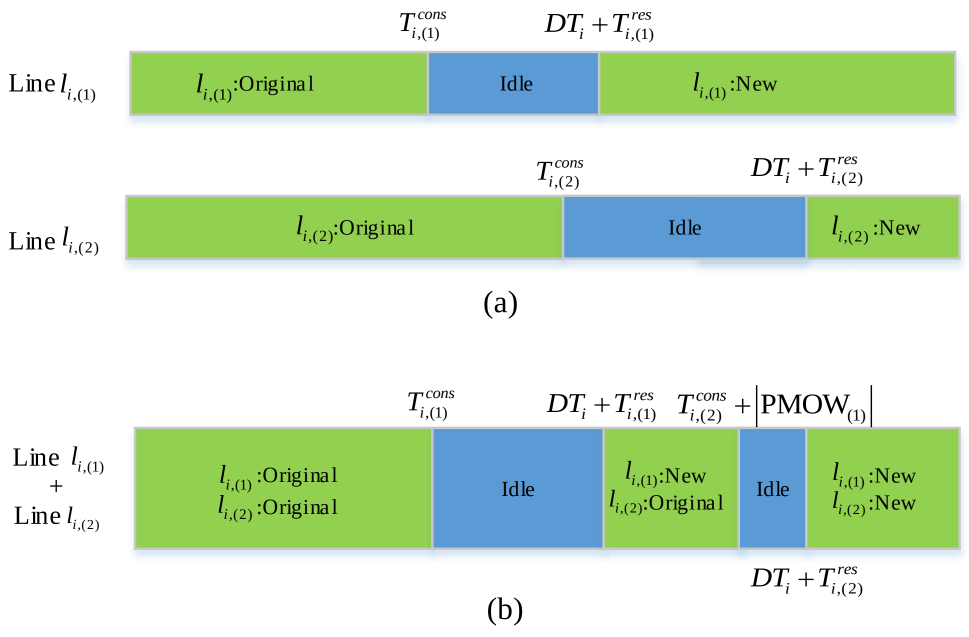


Figure 2.4: The states of the bottleneck machine in lines $l_{i,(1)}$ and $l_{i,(2)}$

Figure 2.4 illustrates the state of the bottleneck machine M_b in lines $l_{i,(1)}$ and $l_{i,(2)}$, when these two lines are considered independently (Fig. 2.4(a)) and jointly (Fig. 2.4(b)). In this figure,"original" means the part that M_b is working on is in the original system before the failure occurs; and "new" means the part is processed by M_i after it resumes running. It shows that, the idle duration on M_b caused in line $l_{i,(1)}$ delays the propagation of the downtime in line $l_{i,(2)}$. The idea of calculating such "delay of propagation" will be frequently used in the following analysis.

This procedure can be applied recursively to predict the PMOW on M_b, which is caused jointly in lines $l_{i,(1)}, \dots, l_{i,(k+1)}$, as

$$PMOW_{(k+1)} = PMOW_{(k)} \bigcup \big[T^{cons}_{i,(k+1)} + |PMOW_{(k)}|, DT_i + T^{res}_{i,(k+1)}\big) \tag{2.9}$$

Moreover, since there are K_i lines connecting M_i and M_b, $PMOW_{(K_i)}$ is the final PMOW caused by the single failure on machine M_i.

As discussed above, in a serial line, the total idle time on the bottleneck machine increases linearly in DT_i as long as DT_i exceeds its threshold DT_i^*. The following theorem demonstrates that the result still holds in a complex system.

Theorem 2.2. *In a complex system, if machine M_i is down for time DT_i, and its critical downtime is DT_i^*, then the total idle time on the bottleneck machine is $|PMOW_i| = (DT_i - DT_i^*)^+$.*

Proof. See Appendix A. □

However, unlike a serial line, where the bottleneck machine could have at most one idle duration, in a complex system, a failure event may cause multiple idle durations on the bottleneck machine. Naturally, one may consider grouping the separate idle durations together and strategically shutting the bottleneck machine down for $|PMOW_i|$. The following proposition shows that, such grouping will not bring additional idle time to the bottleneck machine, as long as it is shut down before the failure event actually propagates to it.

Proposition 2.1. *Let* $j_1 = \min_{k=1,\ldots,K_i} \left\{ k : DT_i > DT^*_{i,(k)} \right\}$*, then for any* $T_0 < T^{cons}_{i,(j_1)}$*, if the bottleneck machine* M_b *is shut down during time* $[T_0, T_0 + |PMOW_i|)$*, it will have no additional idle time afterwards if no future failure occurs.*

Proof. See Appendix A. □

Note that, the maintenance crews can only move the entire or partial idle durations on the bottleneck machine earlier; otherwise the total length of PMOW will be reduced. For example, if a PMOW starts 5 minutes later but the maintenance crews cannot get prepared in 7 minutes, at least 2 minutes of the maintenance opportunities will be wasted. Moreover, whether or not to take the opportunity in advance also depends on the urgency of the maintenance tasks.

2.4.2 Prediction of PMOWs under multiple failures in a serial line

In Section 2.4.1, PMOWs have been predicted under a single failure. However, it is possible that, during the downtime of one machine, another random failure occurs. In this section, we investigate how to update PMOWs in response to multiple failures in a serial line.

Assume that the j^{th} failure F_j occurs on machine M_{i_j} at time T_j, and lasts for DT_{i_j}. The corresponding time to consume and time to resume are denoted as $T^{cons}_{i_j}$ and $T^{res}_{i_j}$, respectively. The updated PMOW on the bottleneck machine M_b at time T_j (after F_j occurs) is denoted as $PMOW(T_j)$. Then, after the first failure F_1, $PMOW(T_1)$ can be predicted by Equation (2.5), as

$$PMOW(T_1) = \left[T_1 + T^{cons}_{i_1}, T_1 + DT_{i_1} + T^{res}_{i_1}\right) \tag{2.10}$$

When the second failure F_2 occurs, $PMOW(T_2)$ can be updated according to the following two cases.

Case 1 $T_2 + T^{cons}_{i_2} \geq T_1 + T^{cons}_{i_1}$:

In this case, the impact of failure F_1 will propagate to M_b before that of F_2, so the occurrence of F_2 will not affect $PMOW(T_1)$. Moreover, the existence of $PMOW(T_1)$ on M_b may delay the time when the impact of F_2 propagates to M_b. We denote $D(F_2)$ as this delay, which can be calculated as

$$D(F_2) = \left[T_1 + T_{i_1}^{cons}, T_1 + DT_{i_1} + T_{i_1}^{res}\right) \cap [T_2, +\infty) \tag{2.11}$$

Then, $PMOW(T_2)$ can be updated as

$$\begin{aligned} PMOW(T_2) = &\left[T_1 + T_{i_1}^{cons}, T_1 + DT_{i_1} + T_{i_1}^{res}\right) \\ &\bigcup \left[T_2 + T_{i_2}^{cons} + |D(F_2)|, T_2 + DT_{i_2} + T_{i_2}^{res}\right) \end{aligned} \tag{2.12}$$

Case 2 $T_2 + T_{i_2}^{cons} < T_1 + T_{i_1}^{cons}$:

The second case occurs when the first idle duration on machine is caused by F_2. Then this idle duration will delay the propagation of F_1, and this delay is calculated as

$$D(F_1) = \left[T_2 + T_{i_2}^{cons}, T_2 + DT_{i_2} + T_{i_2}^{res}\right) \tag{2.13}$$

$PMOW(T_2)$ is then updated as

$$\begin{aligned} PMOW(T_2) = &\left[T_2 + T_{i_2}^{cons}, T_2 + DT_{i_2} + T_{i_2}^{res}\right) \\ &\bigcup \left[T_1 + T_{i_1}^{cons} + |D(F_1)|, T_1 + DT_{i_1} + T_{i_1}^{res}\right) \end{aligned} \tag{2.14}$$

To conclude the two cases, the PMOW after the second failure F_2 can be updated as

$$PMOW(T_2) = \bigcup_{k=1}^{2} \left[T_k + T_{i_k}^{cons} + |D(F_k)|, T_k + DT_{i_k} + T_{i_k}^{res}\right) \tag{2.15}$$

where

$$D(F_1) = \begin{cases} \left[T_2 + T_{i_2}^{cons}, T_2 + DT_{i_2} + T_{i_2}^{res}\right) & \text{if } T_2 + T_{i_2}^{cons} < T_1 + T_{i_1}^{cons} \\ \varnothing & \text{if } T_2 + T_{i_2}^{cons} \geq T_1 + T_{i_1}^{cons} \end{cases}$$

$$D(F_2) = \begin{cases} \varnothing & \text{if } T_2 + T_{i_2}^{cons} < T_1 + T_{i_1}^{cons} \\ \left[T_1 + T_{i_1}^{cons}, T_1 + DT_{i_1} + T_{i_1}^{res}\right) \cap [T_2, +\infty) & \text{if } T_2 + T_{i_2}^{cons} \geq T_1 + T_{i_1}^{cons} \end{cases}$$

Next, we use mathematical induction to generalize the PMOW updating procedure under multiple failures in a serial line. Define function

$$\Theta_s(F_k) := \left[T_k + T_{i_k}^{cons} + |D(F_k)|, T_k + DT_{i_k} + T_{i_k}^{res}\right)$$

which is the additional PMOW on the bottleneck machine that is caused by failure F_k, and $D(F_k)$ is the delay of the propagation of the impact of failure F_k to M_b.

Based on Equation (2.15), it is assumed that the PMOW after failure F_n can be updated as

$$PMOW(T_n) = \bigcup_{k=1}^{n} \Theta_s(F_k) = \Theta_s(F_{(k)}) \tag{2.16}$$

where the order of $F_{(k)}$'s satisfies $T_{(k)} + T_{i_{(k)}}^{cons} + |D(F_{(k)})| \leq T_{(k+1)} + T_{i_{(k+1)}}^{cons} + |D(F_{(k+1)})|$, such that the impact of failure $F_{(k)}$ propagates to M_b before $F_{(k+1)}$.

Then, when a failure F_{n+1} occurs at time T_{n+1}, $PMOW(T_{n+1})$ can be updated through the following steps:

Step 1. Arrange the $n+1$ failures in an ascending order of the time when their impacts propagate to M_b.

Step 1a. Initially, set the delay of propagation for the new failure F_{n+1} as $D(F_{n+1}) = \varnothing$ and its order index $j = 1$.

Step 1b. Determine whether the impact of failure F_{n+1} propagates to M_b before that of failure $F_{(j)}$:

1b.1. If $T_{n+1} + T^{cons}_{i_{n+1}} + |D(F_{n+1})| < T_{(j)} + T^{cons}_{i_{(j)}} + |D(F_{(j)})|$, then F_{n+1} impacts M_b before $F_{(j)}$. Go to Step 1c;

1b.2. If $T_{n+1} + T^{cons}_{i_{n+1}} + |D(F_{n+1})| \geq T_{(j)} + T^{cons}_{i_{(j)}} + |D(F_{(j)})|$, then F_{n+1} impacts M_b after $F_{(j)}$. Update the delay of propagation for F_{n+1} as $D(F_{n+1}) = D(F_{n+1}) \bigcup \Big\{ \Theta_s(F_{(j)}) \cap [T_{n+1}, +\infty) \Big\}$, and set $j = j+1$. If $j = n+1$, then F_{n+1} is the last failure that impacts M_b, and go to Step 1c. Otherwise, repeat Step 1b to compare F_{n+1} with the updated $F_{(j)}$.

Step 1c. Arrange all of the $n+1$ failures in the ascending order of the time when they impact M_b, as

$$\tilde{F}_{(k)} = \begin{cases} F_{(k)} & \text{if } k = 1, \dots, j-1 \\ F_{n+1} & \text{if } k = j \\ F_{(k-1)} & \text{if } k = j+1, \dots, n+1 \end{cases}$$

where the "=" means that the parameters of the updated failure $\tilde{F}$, such as $\tilde{T}$, $\tilde{DT}$, $\tilde{T^{cons}}$ and $\tilde{T^{res}}$, are equivalent to those of the corresponding failure F's.

Step 2. Update the delay of propagation for the failures that impact M_b after F_{n+1}, as $D(\tilde{F_{(k)}}) = \bigcup_{j=1}^{k-1} \Theta_s(\tilde{F}_{(j)}) \bigcap [T_{(k)}, +\infty)$ for $k = j+1, \dots, n+1$.

Step 3. Update the new PMOW as $PMOW(T_{n+1}) = \bigcup_{k=1}^{n+1} \Theta_s(\tilde{F}_{(k)})$. Set $F_{(k)} = \tilde{F}_{(k)} (k = 1, \dots, n+1)$, so that it can be written as

$$PMOW(T_{n+1}) = \bigcup_{k=1}^{n+1} \Theta_s(F_{(k)}) \tag{2.17}$$

Equation (2.17) has the same format as Equation (2.16), which completes the mathematical induction. Therefore, these steps can be used recursively for updating PMOW when a new failure occurs. Moreover, among these three steps, Step 1 is the key. Once the sequence of the failures (in terms of when they start to take effect) is obtained, it is not difficult to update the delays and $PMOW(T_{n+1})$.

2.4.3 Prediction of PMOWs under multiple failures in a complex system

In Sections 2.4.1 and 2.4.2, PMOWs have been predicted under a single failure in a complex system, and under multiple failures in a serial line, respectively. The former prediction is built upon the construction of equivalent pseudo serial lines and the latter updates the PMOWs when sequential failures occur in a serial line. The integration of the two scenarios leads to the prediction of PMOW in a more general scenario, i.e., PMOWs under multiple failures in a complex system. Figure 2.5 illustrates the idea of such integration: once a new machine failure occurs in a complex system, it can be decomposed into multiple failures that occur simultaneously in equivalent serial lines; and then, for each of the equivalent failures, PMOWs can be updated.

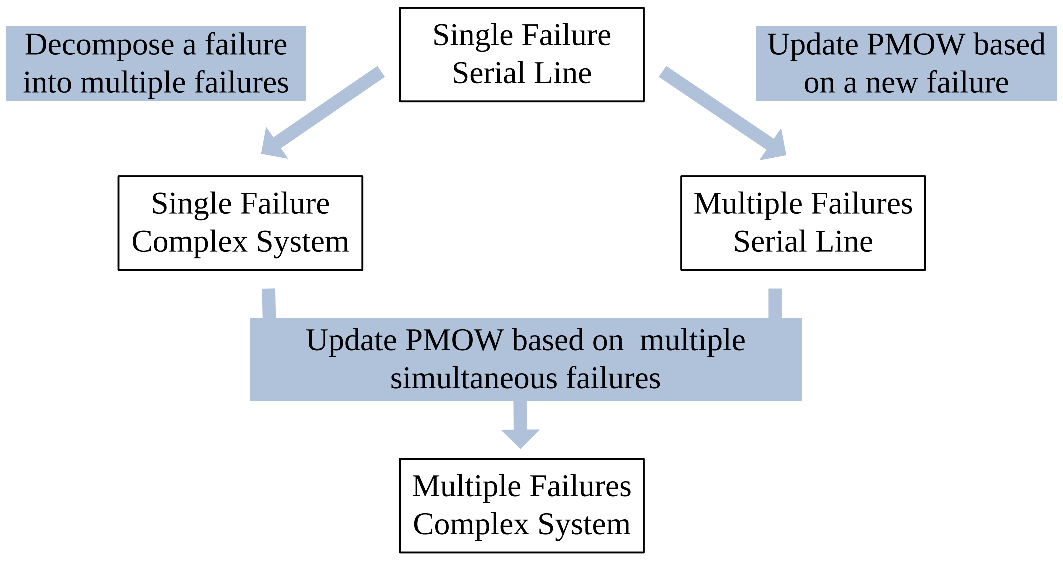

Decompose a failure into multiple failures
Single Failure Serial Line
Update PMOW based on a new failure
Single Failure Complex System
Multiple Failures Serial Line
Update PMOW based on multiple simultaneous failures
Multiple Failures Complex System

Figure 2.5: Prediction of PMOW under multiple failures in a complex system

Assume at time T_1, the first failure F_1 occurs on machine M_{i_1}, which is connected to the bottleneck machine M_b through K_{i_1} equivalent serial lines. Therefore, it can be regarded as there are totally K_{i_1} equivalent failures occurring simultaneously at time T_1, and the PMOW can be predicted based on the analysis in Section 2.4.1. We denote $F_{1,k}$ as an equivalent failure of F_1 in the k^{th} line, and order them by $F_{(k)} = F_{1,(k)}$ ($k = 1, \dots, K_{i_1}$) such that $T^{cons}_{1,(k)} \le T^{cons}_{1,(k+1)}$. From Equation (2.9), PMOW at time T_1 can be calculated as

$$PMOW(T_1) = \bigcup_{k=1}^{K_{i_1}} \Theta_c(F_{(k)}) \tag{2.18}$$

where $\Theta_c(F_{(k)}) = \Big[T_{g(k)} + T^{cons}_{i_{g(k)},h(k)} + |D(F_{(k)})|, T_{g(k)} + DT_{i_{g(k)},h(k)} + T^{res}_{i_{g(k)},h(k)}\Big)$ is the PMOW caused by failure $F_{(k)}$; and $D(F_{(k)}) = \bigcup_{l=1}^{k-1} \Theta_c(F_{(l)})$ is the delay of propagation for failure $F_{(k)}$. $g(k)$ and $h(k)$ are the machine and line indices, such that an equivalent failure $F_{(k)}$ comes from the effect of the actual failure $F_{g(k)}$ in line $h(k)$, i.e., $F_{(k)} = F_{g(k),h(k)}$. Note that, $g(k) = 1$ for all $k = 1, \dots, K_{i_1}$.

Then we assume at time T_2, a second failure F_2 occurs on machine M_{i_2}, which is connected to the bottleneck machine through K_{i_2} lines. The procedure to update PMOW is outlined as follows:

Step 1. Decompose failure F_2 into its equivalent failures $F_{2,(k)}$ ($k = 1, \dots, K_{i_2}$) such that $T^{cons}_{i_2,(1)} \le \cdots \le T^{cons}_{i_2,(K_{i_2})}$.

Step 2. For each $F_{2,(k)}$ ($k = 1, \dots, K_{i_2}$), apply Steps 1–3 in Section 2.4.2 to update $PMOW(T_{2,(k)})$, which is the PMOW caused jointly by $F_{1,(1)}, \dots, F_{1,(K_{i_1})}$ and $F_{2,(1)}, \dots, F_{2,(k)}$.

Step 3. Finally, the PMOW at time T_2 can be updated as

$$PMOW(T_2) = PMOW(T_{2,(K_{i_2})}) = \bigcup_{k=1}^{K_{i_1}+K_{i_2}} \Theta_c(F_{(k)}) \tag{2.19}$$

Furthermore, if a future failure occurs, the PMOW can be updated again by following these steps. Generally, when a failure F_{n+1} occurs at time T_{n+1}, the flowchart to update $PMOW(T_{n+1})$ from $PMOW(T_n)$ is shown in Fig. 2.6.

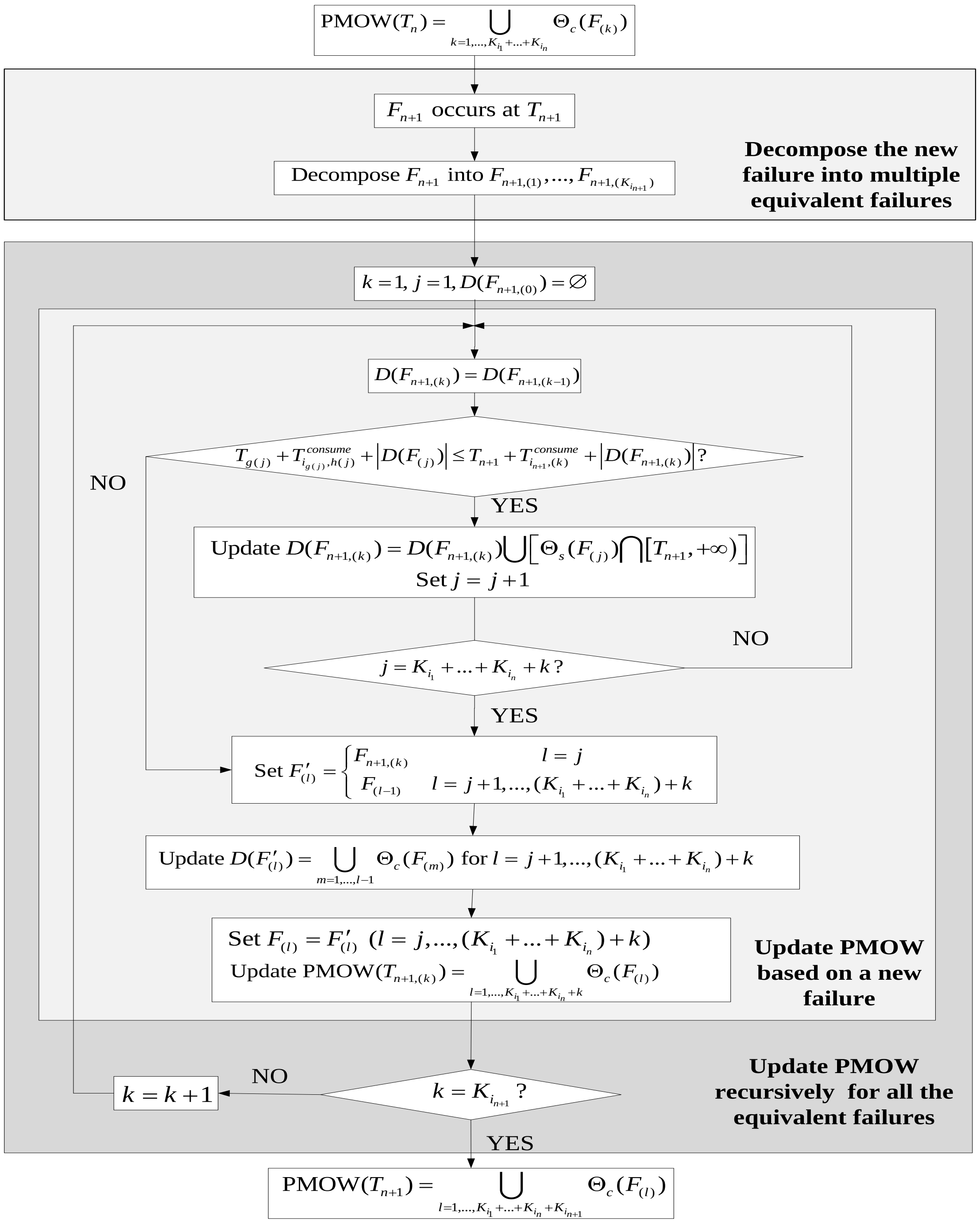

PMOW$(T_n) = \bigcup_{k=1,\dots,K_{i_1}+\dots+K_{i_n}} \Theta_c(F_{(k)})$
F_{n+1} occurs at T_{n+1}
Decompose F_{n+1} into $F_{n+1,(1)},\dots,F_{n+1,(K_{i_{n+1}})}$
Decompose the new failure into multiple equivalent failures
$k=1,\ j=1,\ D(F_{n+1,(0)})=\varnothing$
$D(F_{n+1,(k)}) = D(F_{n+1,(k-1)})$
$T_{g(j)} + T^{consume}_{i_{g(j)},h(j)} + \left|D(F_{(j)})\right| \le T_{n+1} + T^{consume}_{i_{n+1},(k)} + \left|D(F_{n+1,(k)})\right|$?
NO
YES
Update $D(F_{n+1,(k)}) = D(F_{n+1,(k)}) \bigcup \left[\Theta_s(F_{(j)}) \bigcap [T_{n+1},+\infty)\right]$
Set $j = j+1$
NO
$j = K_{i_1} + \dots + K_{i_n} + k$?
YES
Set $F'_{(l)} = \begin{cases} F_{n+1,(k)} & l = j \\ F_{(l-1)} & l = j+1,\dots,(K_{i_1}+\dots+K_{i_n})+k \end{cases}$
Update $D(F'_{(l)}) = \bigcup_{m=1,\dots,l-1} \Theta_c(F_{(m)})$ for $l = j+1,\dots,(K_{i_1}+\dots+K_{i_n})+k$
Set $F_{(l)} = F'_{(l)}$ $(l = j,\dots,(K_{i_1}+\dots+K_{i_n})+k)$
Update PMOW$(T_{n+1,(k)}) = \bigcup_{l=1,\dots,K_{i_1}+\dots+K_{i_n}+k} \Theta_c(F_{(l)})$
Update PMOW based on a new failure
$k = k+1$
NO
$k = K_{i_{n+1}}$?
Update PMOW recursively for all the equivalent failures
YES
PMOW$(T_{n+1}) = \bigcup_{l=1,\dots,K_{i_1}+\dots+K_{i_n}+K_{i_{n+1}}} \Theta_c(F_{(l)})$

Figure 2.6: Flowchart for PMOW prediction under multiple failures in complex systems

2.5 Case Studies

In this section, we present two case studies to illustrate the PMOW prediction algorithm. The first case study predicts the PMOW on the bottleneck machine in a closed-loop system, and the analytical result is validated by simulation. The second case study focuses on PMOW prediction under multiple failures using data from a real automotive plant.

2.5.1 Case study I: PMOW prediction in a closed-loop system

In the first case study, we investigate PMOWs in a closed-loop system as shown in Fig. 2.7. In the closed-loop systems, such as automotive production paint shops, some machines are not only starved by parts, but also by pallets that are used for transporting the parts. In this case study, machines M_1, M_2, M_3 and M_4 are where the parts are processed on the pallets. After machine M_4 completes its work, the finished parts and the pallet will be sent to buffers B_4 and B_0, respectively. Therefore, M_4 will be blocked if either buffer B_4 or B_0 is full. Moreover, although it is assumed that machine M_1 is not starved by parts, it may be starved by pallets (when B_0 is empty).

The machine cycle times, buffer capacities, and their initial contents are shown in Table 2.1.

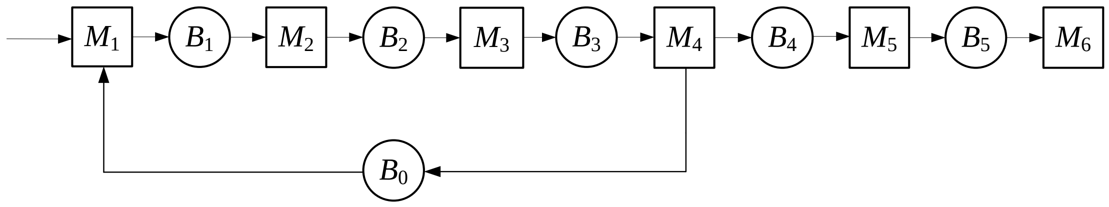


Figure 2.7: The layout of the system for case study I

We investigate the PMOW for different cases of the downtime of machine M_2 (i.e., DT_2). $l_{2,1}$ and $l_{2,2}$ are the two equivalent lines connecting from machine M_2 to M_6, as shown in Fig. 2.8(a) and (b), respectively. The cycle times of the reversed

Table 2.1: System parameters for case study 1

machine	M_1	M_2	M_3	M_4	M_5	M_6
cycle time (sec)	62	60	59	61	60	65
buffer	B_0	B_1	B_2	B_3	B_4	B_5
capacity	5	3	2	4	3	5
initial level	4	2	1	2	1	2

machines, as well as the initial levels of the reversed buffers are shown in Table 2.2. Based on the analysis in Sections 2.2 and 2.3, it can be calculated that $T_{2,1}^{cons} = 390$, $T_{2,1}^{res} = 240$ and $T_{2,2}^{cons} = 325$, $T_{2,2}^{res} = 60$. Therefore, $DT_{2,1}^{*} = 150$ and $DT_{2,2}^{*} = 265$, and $PMOW_2 = [325, DT_2 + 60) \bigcup \Big[390 + |[325, DT_2 + 60)|, DT_2 + 240\Big)$.

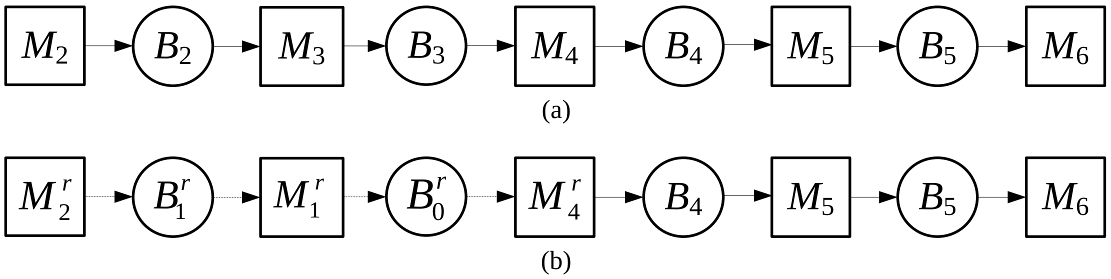


Figure 2.8: Decomposition of the closed-loop system

Table 2.2: Parameters of the reversed machines and buffers

machine	M_1^r	M_2^r	M_4^r	buffer	B_0^r	B_1^r
cycle time (sec)	0	0	0	initial level	1	1

To validate the effectiveness of the prediction algorithm, we have built a simulation model of the closed-loop system by using a commercial simulation software, SIMUL8 [63]. Moreover, in order to make the case more practical, the machine cycle times in the simulations are assumed to follow Gaussian distributions.

First, we study the $|PMOW_2|$ under different DT_2's. The standard deviation of the processing time on each machine is set to be 2%, 5% and 10% of its mean. The total idle time of the bottleneck machine before it completes 20 parts is logged, and its mean and standard deviation (s.d.) are shown in Table 2.3 and Fig. 2.9. These results show that, when $DT_2 > 150$, $|PMOW_2|$ will increase approximately linearly with DT_2. This agrees with the analytical result in Theorem 2.2. Therefore, the PMOW prediction algorithm is applicable to the systems where the variations of the machine cycle times are small. Moreover, since the PMOW prediction is analytical, it is more efficient than simulations, especially for real-time application.

Table 2.3: PMOW prediction for case study I

DT_2(sec)	$\lvert PMOW_2\rvert$ (sec)						
	analytical	2%		5%		10%	
		mean	s.d.	mean	s.d.	mean	s.d.
0	0	0	0	0	0	0	0
50	0	0	0	0	0	0.06	0.95
100	0	0	0	0	0	1.44	5.91
150	0	1.73	2.62	5.91	7.32	23.34	19.96
200	50	50.40	4.04	54.23	10.37	70.87	23.55
250	100	100.31	3.87	104.10	10.13	119.65	23.55
300	150	150.19	3.93	153.95	10.11	170.59	24.83
350	200	200.16	3.96	203.84	10.19	220.80	22.66
400	250	250.22	3.96	254.03	10.18	269.82	25.56
450	300	300.21	3.97	304.01	10.20	320.23	23.00
500	350	350.24	3.99	353.97	10.16	369.90	23.55

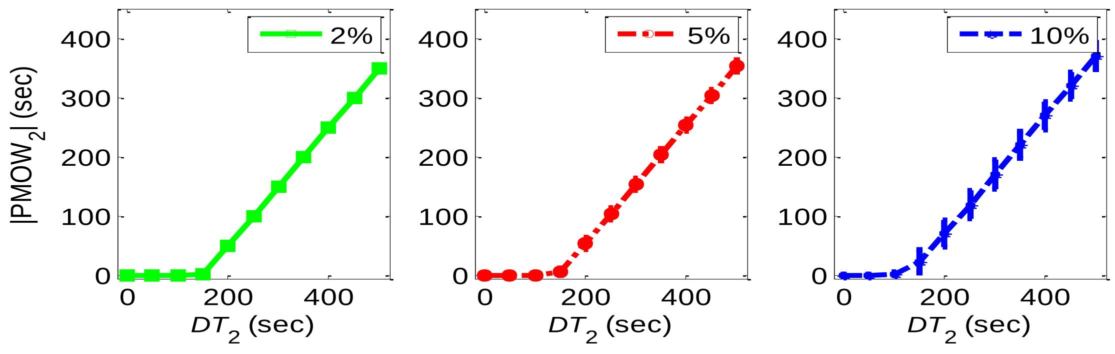


Figure 2.9: The relationship between $|PMOW_2|$ and DT_2

Then we study the working/idle status of the bottleneck machine M_6 under different DT_i's. The results for $DT_2 = 150, 250, 350$ and 450 seconds are shown in Fig. 2.10 (the standard deviation on the processing time is assumed to be 2% of its mean). These results demonstrate that, if DT_2 is within 150 seconds, there is almost no PMOW on M_6. When DT_2 increases to 250 seconds, which is between $DT^*_{2,1}$ and $DT^*_{2,2}$, one idle duration occurs on M_6, and it results from line $l_{2,1}$. However, when DT_2 is greater than $DT^*_{2,2}$, such as 350 and 450 seconds, M_6 will have two separate idle durations (the third one is negligible), where the first one is caused in $l_{2,2}$ and the second is in $l_{2,1}$. When DT_2 continues to increase, the first idle duration keeps growing while the length of the second idle duration remains constant.

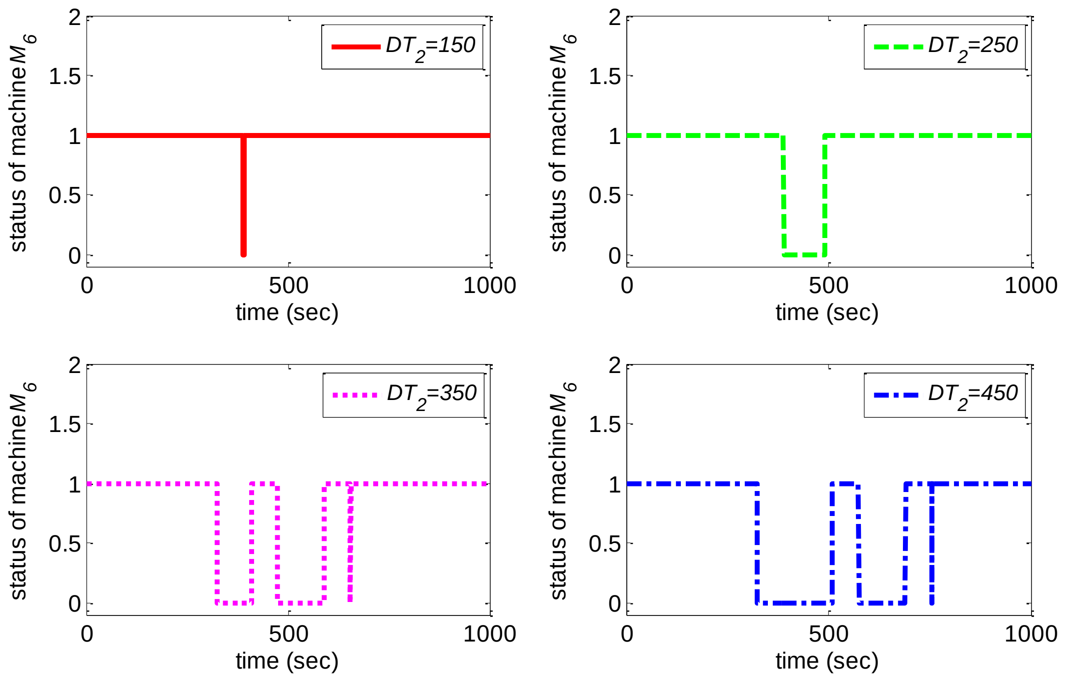


Figure 2.10: Status of the bottleneck machine M_6 over time (1 – working; 0 – idle)

Moreover, this case study shows that, the PMOW prediction tool is also applicable to the case when the downtime DT_i is not deterministic. If the probability distribution of DT_i is known, then the probability distribution of PMOW can be predicted accordingly. Maintenance policies under such probabilistic PMOWs will be investigated in the future.

2.5.2 Case study II: PMOW prediction under multiple failures in a real manufacturing plant

The PMOW prediction tool has been implemented in a real automotive assembly plant to validate its effectiveness in practice. The system in this case study consists of 3 groups of machines, and conveyors between successive groups, as shown in Fig. 2.11(a). The bottleneck machine M_b is the first machine in group 3. Moreover, if a failure occurs, maintenance will be carried out on that machine and the whole group of machines will be stopped while the parts being processed are kept on the machines, until the maintenance is completed. Therefore, here we only consider the PMOWs caused by failures in groups 1 and 2. This system can be modeled as a 3M2B system, as shown in Fig. 2.11(b), with the system parameters in Table 2.4. The cycle time of machine M_i $(i = 1, 2)$ in Fig. 2.11(b) is equal to the summation of the cycle times of all the machines in group i. Moreover, the cycle times of the conveyors will be counted in the calculation of T^{res} .

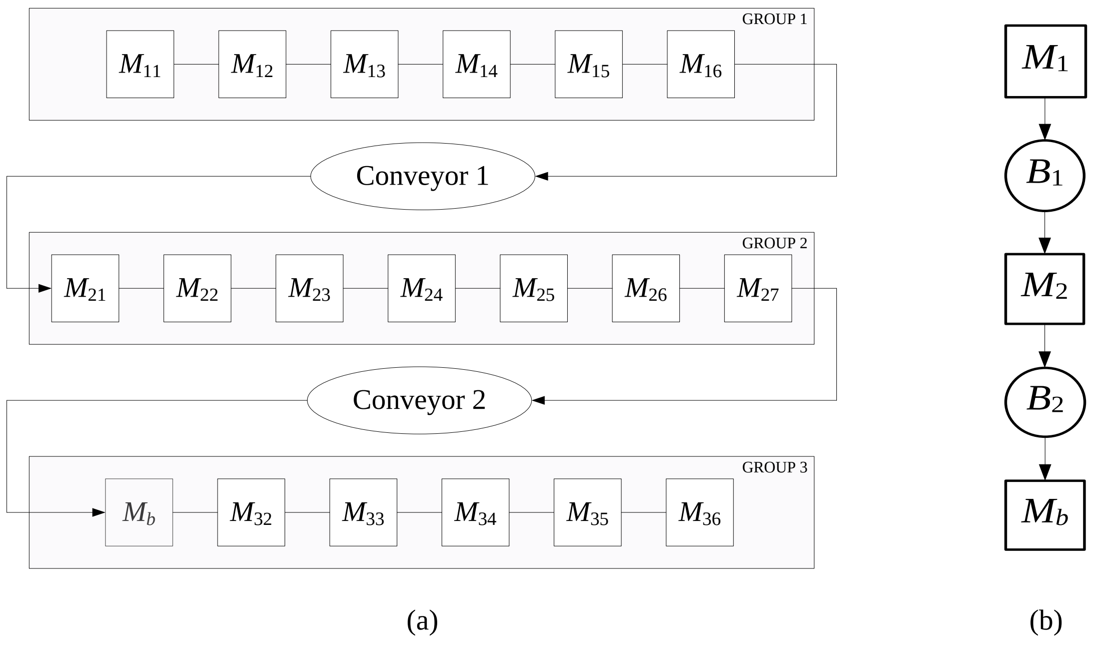


Figure 2.11: System structure and its model for case study II

Table 2.4: Parameters of the 3M2B system

machine/buffer	M_1	B_1	M_2	B_2	M_b
capacity	6	11	7	24	1
cycle time (min)	4.44	0.71	6.23	3.53	1

In order to show how the PMOW was updated under multiple failures, we conducted the following two "failure" events:

(1) F_1: at 1:00 p.m., machine M_{16} was shut down for 15 minutes; and

(2) F_2: at 1:20 p.m., machine M_{14} was shut down for 10 minutes.

The factory information system (FIS) updated the WIP of all buffers and machines every 5 minutes, as shown in Fig. 2.12. Based on the real-time information at 1:00 p.m., the PMOW under F_1 was predicted as [1:25:00, 1:26:13), indicating a 73-s idle duration on the bottleneck machine M_b, which would start from 1:25 p.m. When the second failure F_2 occurred at time 1:20 p.m., the PMOW could be updated as $[1:25:00, 1:26:13) \bigcup [1:32:13, 1:41:13)$, indicating that another 9-min idle duration on M_b would start at 1:32 p.m.

Then, we utilized the real-time plant floor data $after$ 1:20 p.m. (also shown in Fig. 2.12) to validate the prediction (i.e., to see if machine M_b would be idle during $[1:25:00, 1:26:13) \bigcup [1:32:13, 1:41:13)$). From the WIP records in FIS, there were indeed two idle durations on M_b – one was around 1:25 p.m., and the other was around 1:35 to 1:40 p.m. – as predicted. Moreover, with a detailed observation of the WIP in buffer B_2, one could obtain a more accurate estimation of the starting/completion time of the two idle durations. For the first idle duration, it showed that at 1:20 p.m., buffer B_2 had 4 parts, which would allow M_b to process for additional 4 minutes before it got starved; and at 1:25 p.m., B_2 had one "new" part in it, which would be delivered to M_b shortly. Similarly, B_2 had 2 parts at 1:30 p.m., indicating that the second idle duration would start at around 1:32 p.m.; while B_2 had a large WIP at 1:40 p.m., so that the second idle duration on M_b would end soon. These results validated the effectiveness of PMOW prediction algorithm under multiple failures.

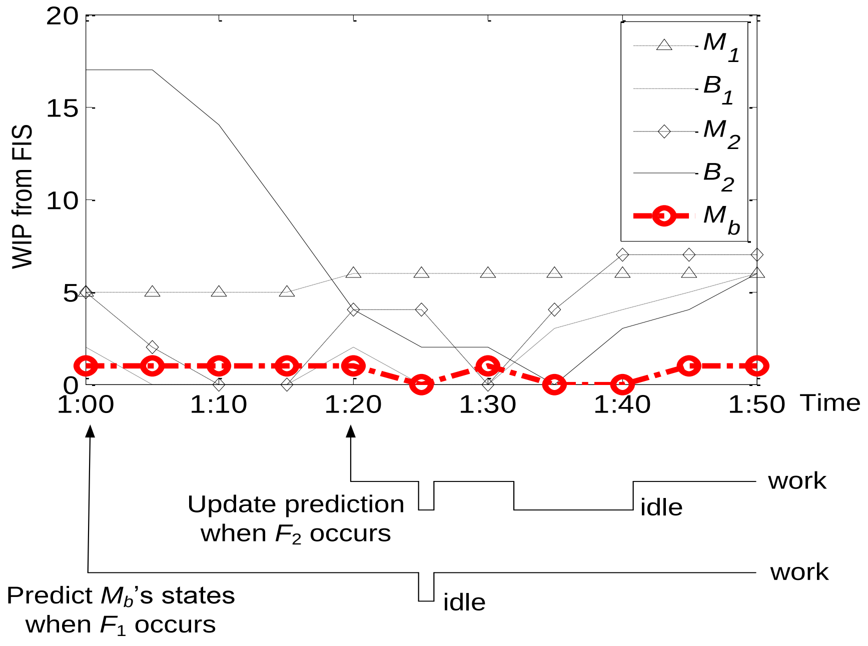


Figure 2.12: PMOW prediction and validation using real-time data from FIS

2.6 Summary

In this chapter, an analytical model has been developed to predict the PMOWs on the bottleneck machine in a manufacturing system. First, through the proposed technique of generating equivalent pseudo serial lines, the critical downtime for each machine in a complex system is calculated. Then, a PMOW prediction algorithm for a complex system is developed when a single failure occurs on one machine. It is found that, in a complex system, the total idle time of the bottleneck machine equals to the difference between the machines actual downtime caused by the failure and its critical downtime. Moreover, the PMOW prediction algorithm under multiple failures has also been investigated. Case studies in simulations and a real automotive plant have been conducted to demonstrate the methodology and insights.

The effectiveness of the PMOW prediction algorithm has also been validated when it is applied to systems with small variations in processing time. The analytical model also provides a sound basis for research on systems with more uncertainties and variations, such as large variation on machine processing time, probabilistic downtime caused by the failure, etc. Such randomness will be taken into consideration in our future work, to make PMOW prediction algorithm more robust for real plant implementation.

The future work related to the PMOW technique includes: (1) estimating PMOWs analytically when the cycle times of machines are not constant; (2) developing methods to correct PMOW based on the feedback information; and (3) integrating maintenance opportunities into the design of manufacturing systems.

CHAPTER III

Estimation of Active Maintenance Opportunity Windows in Manufacturing Systems

3.1 Introduction

In Chapter II, we developed algorithm to predict the passive maintenance opportunity windows (PMOWs), which came from the downtime of other machines in the system. However, the goal for maintenance management is to achieve near-zero downtime [64], and it is desired to perform preventive maintenance tasks before the actual failures occur. Therefore, in addition to "passively" taking advantage of the downtime, we need to look for the opportunities such that one machine can be "actively" shut down for preventive maintenance (PM) during production, without bringing production losses. We call such opportunities active maintenance opportunity windows (AMOWs).

AMOWs come from extra buffer contents or spaces, which will keep the surrounding machines operating for a certain period of time, even if some other machines are undergoing PM. Chang et al. [6] investigated AMOWs by only considering the production losses during the preventive maintenance, while did not calculate the potential production losses due to future machine failures. In this chapter, we estimate the AMOWs by considering the production losses both during the PM and after the

PM.

The production loss caused by preventive maintenance can be evaluated for two periods: the time when PM is performed (the during-PM period), and the time period after PM (the post-PM period). By evaluating the production loss in the two periods, we are able to decide how many buffer contents or empty space can be utilized for AMOW calculation. If we only need to satisfy the system production rate during the AMOW, then all of the buffer contents can be leveraged to do PM, and this case is the same as the one in [6]. However, when random machine failures are considered, the problem becomes more complicated. We need to reserve contents or empty space in the buffers to protect the system against the random failures in the post-PM period. Early work suggests that using all of the contents (or empty space) in buffers may cause a future production loss. For example, simulation results in [56] showed that, with all buffers being empty initially, the production loss in one shift could be as large as 10% of the required number of production. Experiments in [6] also indicated that in a balanced line, 50% of the buffer capacity should be reserved as safety stocks such that there would be no further production loss. Both works investigated the problem using simulations. There are also some other studies using a similar idea to determine safety stocks and PM policies jointly. For example, Srinivasan and Lee [65] developed an (s, S) policy where PM was performed when the inventory level reached a certain threshold. Cheung and Hausman [66] derived an optima finished goods inventory (FGI) when machines had increasing failure rate and the repair times were constant or exponentially distributed. Chelbi and Ait-Kadi [67] determined simultaneously an optimal safety stock level and the PM period. Kyriakidis and Dimitrakos [68] used a Markov decision process to determine when PM should be performed, based on buffer levels as well as machine degradation states. Jin et al. [69] developed an option-based model for a joint production and preventive maintenance system to reduce the risks under stochastic demand. Again, most of these policies are based on steady-state analysis, with the objective of minimizing average cost over a long run.

In this chapter, we develop an AMOW-based PM policy based on real-time production information: buffer levels and machine reliabilities. We estimate AMOWs in manufacturing systems with unreliable machines and finite buffers. The remainder of the chapter is organized as follows. Section 3.2 introduces the model and assumptions. In Section 3.3, the analytical solution to AMOWs for a two-machine-one-buffer (2M1B) system is provided. In Section 3.4, a decomposition method is developed to estimate AMOWs in manufacturing systems with different configurations, and a heuristic algorithm for AMOW estimation in balanced lines is proposed. Section 3.5 summarizes this chapter.

3.2 Model and Assumptions

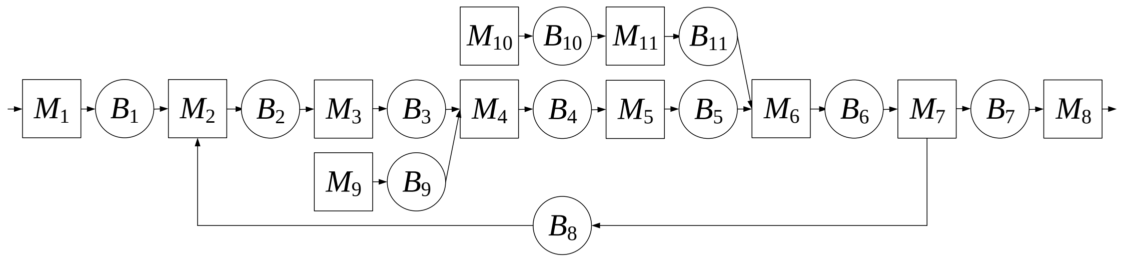


Figure 3.1: A sample manufacturing system

We consider a system consisting of I machines and B buffers. For example, Fig. 3.1 shows an 11-machine-11-buffer (11M11B) system. Considering random machine breakdowns, we assume that the system follows *Bernoulli reliability*, satisfying:
(1) All machines have an identical cycle time, which is denoted as the time unit.
(2) At time k, M_i ($i = 1, 2, ..., I$) is "up" with probability $p_i(k)$, and is "down" with probability $1 - p_i(k)$. If M_i is stopped for PM at time k, $p_i(k) = 0$; otherwise $p_i(k) = p_i$, where p_i is the reliability of M_i. Define $\mathbf{p}(k) := [p_1(k) \ \cdots \ p_I(k)]$.
(3) B_i ($i = 1, 2, \ldots, B$) has capacity C_i. Its buffer level $N_i(k)$ is counted at the end

of time k. Define $\mathbf{C} := [C_1 \ \cdots \ C_B]$, and $\mathbf{N}(k) := [N_1(k) \ \cdots \ N_B(k)]$.
(4) A machine will be starved if one of its immediately upstream buffers is empty. A machine with no upstream buffer is never starved.
(5) A machine will be blocked if one of its immediately downstream buffers is full and the machine on the downstream of that buffer fails to take one part out. A machine with no downstream buffers is never blocked.
(6) The required system throughput is constant over time and equals to the system throughput in the steady state.
(7) At most one AMOW-based PM is performed, and it is performed at the current time unit.

The Bernoulli model, based on assumptions (1) to (5), has been studied by Li and Meerkov [46], and its practical effectiveness has been validated [57, 70, 71]. The reasons for assumption (6) are two-fold. On the one hand, if the required system throughput is greater than the steady-state one, there is no AMOW because the system cannot fulfill its long-term production requirement. On the other hand, if the steady-state system throughput is greater than the required one, PM can take as long as needed because eventually the lost production in the system will be made up. In assumption (7), it is assumed that we do not perform simultaneous AMOWs-based PM tasks on multiple machines, because the objective is to find the maximum possible AMOW, while the consideration of multiple AMOWs makes the AMOWs on each machine relatively smaller. Assumption (7) also indicates that AMOWs in future time is not considered. This is reasonable because future production information is required to estimate future AMOWs. Additionally, the consideration of future AMOWs will also decrease the real-time AMOW. Since there is only one AMOW considered, we always denote the real-time buffer level (i.e., the prior-PM buffer level) as $\mathbf{N}(0)$, and the time afterwards as $k = 1, 2, \ldots$.

The reasons the Bernoulli model is used to study our problem are as follows. First, the Bernoulli models can be represented by a discrete time Markov chain (DTMC). System performance measures, such as production rate (i.e., the expected number of parts produced by the last machine of the system per time unit), and work-in-process (WIP), can be obtained by solving linear equations, requiring less computational effort compared with other reliability models. Moreover, some continuous models, where the mean time to repair (MTTR) and the mean time between failures (MTBF) have other distributions, can be effectively transformed into Bernoulli models by using discretization techniques.

In the following section, AMOWs will be estimated in two-machine-one-buffer (2M1B) lines.

3.3 Estimation of AMOWs in Two-Machine-One-Buffer Lines

3.3.1 Problem formulation

In a 2M1B line, there is only one buffer. For convenience, we denote the buffer as B, its capacity as C and its content at time k as $N(k)$. Given a real-time buffer level $N(0)$, the decision variable n is an indicator of the post-PM buffer level where the machine under PM should be resumed so that production losses are avoided. If PM is performed on M_1, we have $n < N(0)$ and the buffer level decreases at a rate of p_2 (Fig. 3.2(a)), while $n > N(0)$ indicates that PM is performed on M_2, and buffer level increases at a rate of p_1 (Fig. 3.2(b)). Let $PL(N(0), n)$ denote the production loss when the real-time buffer level is $N(0)$ and the post-PM buffer level indicator is n, then the AMOWs in 2M1B lines can be defined as follows:

Definition 3.1. In a 2M1B line with a real-time buffer level $N(0)$, the AMOWs for M_1 and M_2 are $AMOW_1(N(0)) := \max_{n\in\Lambda}(N(0) - n)/p_2$ and $AMOW_2(N(0)) := \max_{n\in\Lambda}(n - N(0))/p_1$, respectively, where $\Lambda = \{n : PL(N(0), n) \leq 0\}$.

Remark 3.1. From Definition 3.1, there is a one-to-one correspondence between the post-PM buffer level indicator n and AMOW. Without loss of generality, we allow the post-PM buffer level indicator n to be negative or greater than the buffer capacity, corresponding to the cases that PM is still being performed when the buffer is already empty or already full, respectively.

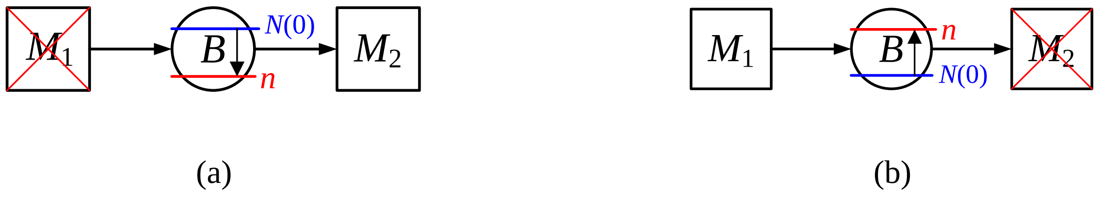


Figure 3.2: Two-machine-one-buffer (2M1B) production lines

In order to obtain the AMOWs in Definition 3.1, the closed-form expression of the production loss $PL(N(0), n)$ needs to be derived first, which is the focus of the next section.

3.3.2 Calculation of production loss

The production loss is the expected total number of the unfulfilled production units, which can be calculated as

$$PL = \sum_{k=1}^{\infty} \left(PR^{req} - PR^{AMOW}(k) \right) \tag{3.1}$$

where PR^{req} is the required system production rate, and $PR^{AMOW}(k)$ is the expected system production rate at time k under the AMOW-based maintenance.

Based on assumption (6), there will be no production loss once the system enters its steady state. Then the production loss can be evaluated over two periods: period 1 is the during-PM period, and period 2 is the duration from PM completion until the system enters its steady state. Note that, the production loss in period 1 depends on both the real-time buffer level $N(0)$ and the post-PM buffer level n, while that in period 2 only depends on n. Therefore, the production loss in periods 1 and 2 can be denoted as $PL^{(1)}_{N(0),n}$ and $PL^{(2)}_{n}$, respectively. The total production loss is the summation of these two:

$$PL(N(0),n) = PL^{(1)}_{N(0),n} + PL^{(2)}_{n} \tag{3.2}$$

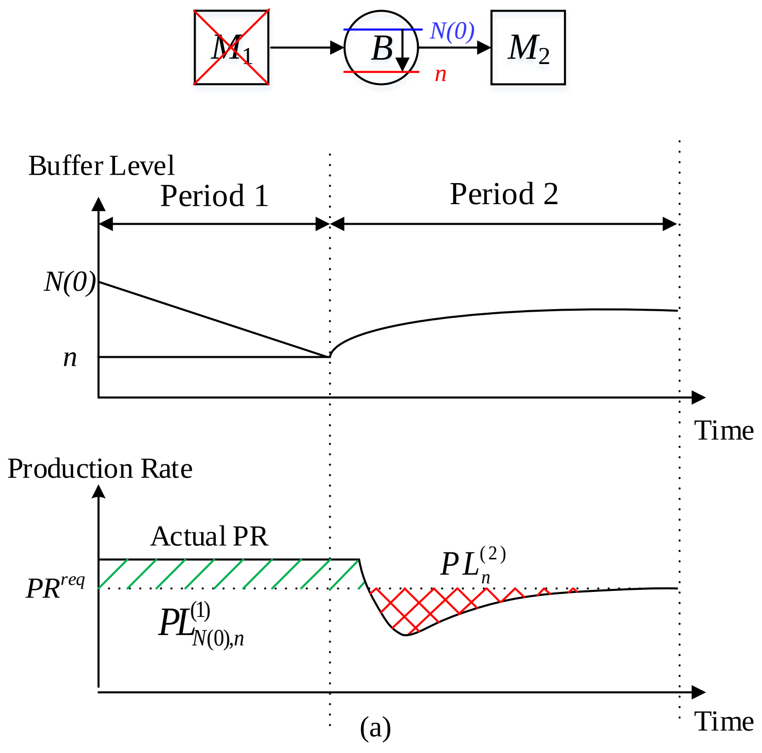
Figures 3.3 (a) and (b) are the illustrations of these two periods when PM is performed on M_1 and M_2, respectively. The area patterned with "/" represents the case that the actual production rate is greater than the required production rate (i.e., a production gain), while that patterned with "×" indicates a production loss.

3.3.2.1 Calculation of the required system production rate (PR^{req})

To calculate the production loss, we first determine the required production rate. The state space of the 2M1B line is $S = \{0, 1, \dots, C\}$. Let $\pi_s = \lim_{k\to\infty} \pi_s(k)$ be the probability that in the steady state, the system is in state s ($s = 0, 1, \dots, C$). Then π_s can be obtained by solving $\boldsymbol{\pi} = \mathbf{P}(p_1, p_2, C+1)\boldsymbol{\pi}$, where $\boldsymbol{\pi} = [\pi_0 \ \cdots \ \pi_C]^T$, and $\mathbf{P}(p_1, p_2, C+1)$ is the transition matrix, given by

$$\mathbf{P}(p_1, p_2, C+1) = \begin{bmatrix} 1-p_1 & p^- & 0 & \cdots & 0 \\ p_1 & 1-p^-p^+ & p^- & \cdots & 0 \\ 0 & p^+ & 1-p^-p^+ & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1-p^-p^+ & p^- \\ 0 & 0 & \cdots & p^+ & 1-p^- \end{bmatrix}_{(C+1)\times(C+1)} \tag{3.3}$$

where $p^+ = p_1(1-p_2)$ and $p^- = (1-p_1)p_2$ denote the probabilities that the buffer level increases and decreases in one time unit, respectively (when the buffer is not empty or full).



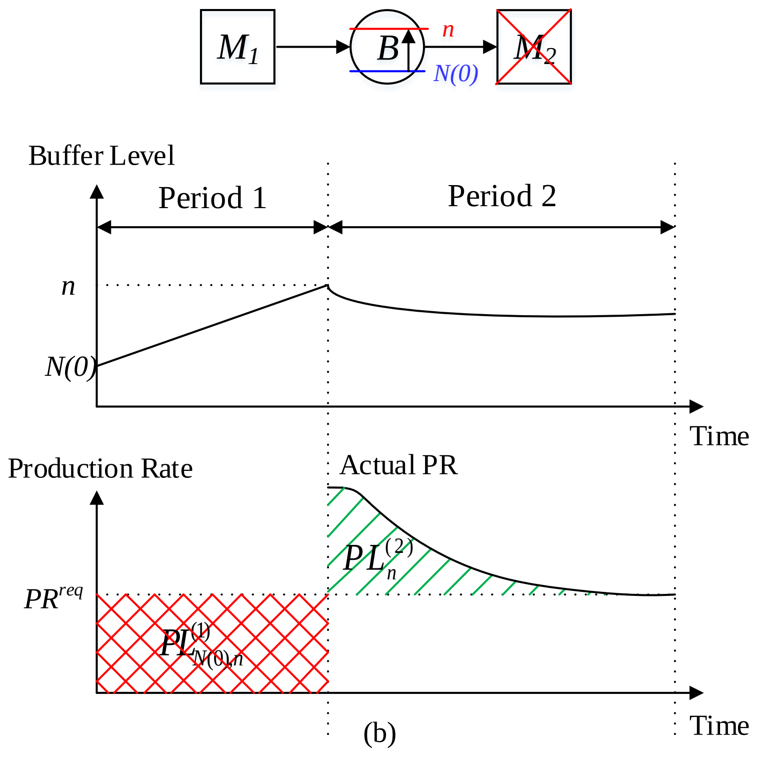


Figure 3.3: Illustrations of the two-period model

Based on [46], the required system production rate can be calculated as

$$PR^{req} = (1-\pi_0)p_2 = \begin{cases} \frac{p_1(1-(p^+/p^-)^C)}{p_2(1-p1/p_2(p^+/p^-)^C)} & \text{if } p_1 \neq p_2 \\ \frac{Cp}{C+1-p} & \text{if } p_1 = p_2 = p \end{cases} \tag{3.4}$$

Then, we analytically calculate the production losses in periods 1 and 2.

3.3.2.2 Calculation of the production loss in period 1 ($PL^{(1)}_{N(0),n}$)

In period 1, stopping M_1 or M_2 for PM has different effects on the system throughput. Performing PM on M_2 interrupts the system throughput immediately. If PM is performed on M_1, however, the interruptive effect on the end-of-line throughput is not immediate. The system has continuous throughput until the buffer becomes empty. Therefore, the production loss in period 1 needs to be discussed separately when PM is performed on M_1 (indicated by $N(0) > n$), or M_2 (indicated by $N(0) < n$).

Case 1 PM is performed on M_1:

Case 1a $n > 0$: In this case, the buffer is not empty for any time k in period 1, and thus $\pi_0(k) = 0$. Consequently, $PL^{(1)}_{N(0),n} = (PR^{req}-p_2)(N(0)-n)/p_2 = -\pi_0(N(0)-n)$. Note that, when $n > 0$, $PL^{(1)}_{N(0),n}$ is smaller than 0, indicating a *production gain* in period 1. The reason for this production gain is that, while the buffer is not empty, the system will keep a constant production rate p_2, which is larger than PR^{req} (as shown in Equation (3.4)).

Case 1b $n \leq 0$: M_2 will be starved when the buffer becomes empty, bringing a production loss of $-(1-\pi_0)n$ afterwards. In this case, the production loss in period 1 can be calculated as $PL^{(1)}_{N(0),n} = -\pi_0 N(0) - (1-\pi_0)n$.

Case 2 PM is performed on M_2:

If M_2 is shut down for PM, there will be no throughput during the entire period 1, and thus the production loss in period 1 is $PL^{(1)}_{N(0),n} = PR^{req}(n - N(0))/p_1 = (1-\pi_0)(n-N(0))p_2/p_1$.

Summarizing both cases above, we obtain the system production loss in period 1 as follows:

$$PL^{(1)}_{N(0),n} = \begin{cases} -\pi_0 N(0) - (1-\pi_0)n & \text{if } n < 0 \\ -\pi_0(N(0) - n) & \text{if } 0 \leq n < N(0) \\ (1-\pi_0)(n - N(0))p_2/p_1 & \text{if } n > N(0) \end{cases} \tag{3.5}$$

3.3.2.3 Calculation of the production loss in period 2 ($PL^{(2)}_n$)

A sample path of how the buffer level $N(k)$ evolves in period 2 is shown in Fig. 3.4. The system dynamics can be represented by $\boldsymbol{\pi}(k+1) = \mathbf{P}(p_1, p_2, C+1)\boldsymbol{\pi}(k)$, where $\mathbf{P}(p_1, p_2, C+1)$ is the time-invariant transition matrix defined in Equation 3.3, and the transition probability $P_{mn} := Pr(N(k+1) = m | N(k) = n)$ $(0 \leq n, m \leq C)$ lies in the $(n+1)$th column and the $(m+1)$th row of $\mathbf{P}(p_1, p_2, C+1)$.

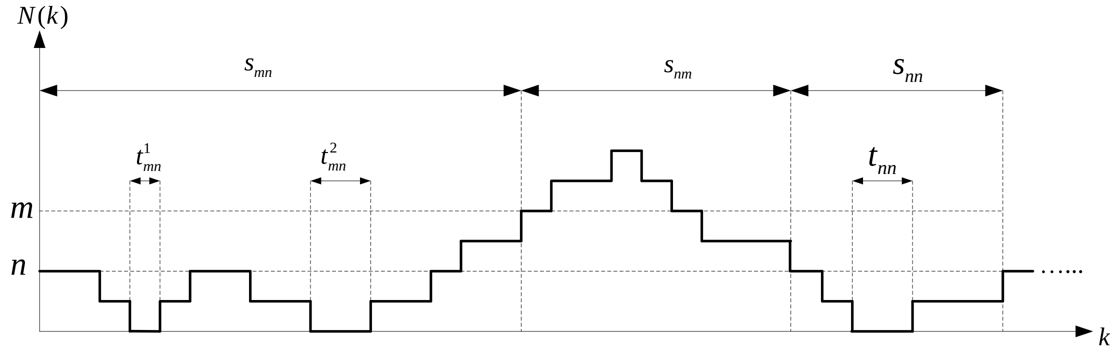


Figure 3.4: A sample path of the buffer level evolution in period 2

We define $s_{mn} := \min\{k : N(k+s) = m, k > 0 | N(s) = n\}$ and $t_{mn} := \sum_{k=0}^{s_{mn}} \mathbb{1}\{N(k) = 0\}$, where $\mathbb{1}\{X\}$ is an indicator function (i.e., $\mathbb{1}\{X\} = 1$ if X is true). In other words, s_{mn} is the time duration from state n to the first time the system reaches state m, and t_{mn} is total time that the buffer is empty during s_{mn} . For the example in Fig. 3.4, $s_{mn} = 16$, $s_{nm} = 10$, $s_{nn} = 6$; and $t_{mn} = t^1_{mn} + t^2_{mn} = 3$, $t_{mn} = 0$, $t_{nn} = 2$. If we denote $T_{mn} = \mathbb{E}[t_{mn}]$ and $S_{mn} = \mathbb{E}[s_{mn}]$, then all T_{mn}'s and

S_{mn}'s can be calculated by solving the equations $T_{mn} = \mathbb{1}\{n = 0\} + \sum_{l \neq m} P_{ln} T_{ml}$ and $S_{mn} = 1 + \sum_{l \neq m} P_{ln} S_{ml}$ for all $n, m \in \{0, 1, \ldots, C\}$. Then T_{mn} can be calculated as:

If $p_1 \neq p_2$

$$T_{mn} = \begin{cases} 0 & \text{if } m < n \\ \frac{(p^-/p^+)^n - (p^-/p^+)^m}{p_1(1-p^-/p^+)} & \text{if } m > n \\ \frac{(p^-)^n}{p_1(p^+)^{n-1}} & \text{if } m = n > 0 \\ 1 & \text{if } m = n = 0 \end{cases} \tag{3.6a}$$

If $p_1 = p_2$

$$T_{mn} = \begin{cases} 0 & \text{if } m < n \\ \frac{m-n}{p} & \text{if } m > n \\ 1-p & \text{if } m = n > 0 \\ 1 & \text{if } m = n = 0 \end{cases} \tag{3.6b}$$

And S_{mn} can be calculated as:

If $p_1 \neq p_2$

$$S_{mn} = \begin{cases} \frac{1}{p^-}\left(\frac{n-m}{p^- - p^+} - \frac{(p^+/p^-)^{C+1-n} - (p^+/p^-)^{C+1-m}}{p^-(1-p^+/p^-)^2}\right) & \text{if } m < n \\ \frac{(p^-/p^+)^n - (p^-/p^+)^m}{p_1(1-p^-/p^+)} + \frac{m-n}{p^+ - p^-} - \frac{(p^-/p^+)^n - (p^-/p^+)^m}{p^+(1-p^-/p^+)^2} & \text{if } m > n \\ \left(\frac{p^+}{p^-}\right)^{C-n} \frac{1-(p^-/p^+)^C}{1-p^-/p^+} + \frac{p^-(p^-/p^+)^{n-1}}{p_1} & \text{if } m = n > 0 \\ 1 + \frac{p_1(1-(p^+/p^-)^C)}{p^- - p^+} & \text{if } m = n = 0 \end{cases} \tag{3.7a}$$

If $p_1 = p_2$

$$S_{mn} = \begin{cases} \frac{(C-m+1)(C-m))-(C-n+1)(C-n))}{2p(1-p)} & \text{if } m < n \\ \frac{m-n}{p} + \frac{m(m-1)-n(n-1)}{2p(1-p)} & \text{if } m > n \\ C+1-p & \text{if } m = n > 0 \\ \frac{C+1-p}{1-p} & \text{if } m = n = 0 \end{cases} \tag{3.7b}$$

These equations lead to the following lemma.

Lemma 3.1. $\pi_0 = T_{nn}/S_{nn}$ *for all* $n = 0, \dots, C$.

Lemma 3.1 demonstrates that, the proportion of the expected time that the buffer is empty during two consecutive visits to state n, is equal to the probability that the buffer is empty in the steady-state. This result can also be proved by the renewal theory, and leads to the following lemma.

Lemma 3.2. *The production loss from state n to state m is equal to the production loss during the time the system first reaches state m from state n.*

Proof. See Appendex B. □

Next, we derive the expression for PL_n, the production loss starting from state n until the system reaches its steady state. Assume k_s is the time point that the system enters its steady state, satisfying $\pi_m(k_s) = \pi_m$. Then PL_n can be calculated as

$$PL_n = PL_n(k_s) = \sum_{m=0}^{C} PL_{mn}(k_s)\pi_m(k_s) = \sum_{m=0}^{C} (T_{mn} - \pi_0 S_{mn}) p_2 \pi_m \quad (n = 0, \dots, C) \tag{3.8}$$

which can be further calculated as:

If $p_1 \neq p_2$

$$PL_n = \frac{a + bn + c(p^+/p^-)^{C-n}}{p_2(1 - p_1/p_2(p^+/p^-)^C)^2} \tag{3.9a}$$

where

$$a = (-2p_1C - p^+)(p^+/p^-)^C + \frac{p_1(1-(p^+/p^-)^C}{1-p^+/p^-},$$
$$b = p_1(p^+/p^-)^C - p_2,$$
$$c = p^+ + \frac{p_1\left((p^+/p^-) - (p^+/p^-)^{C+1}\right)}{1-p^+/p^-}.$$

If $p_1 = p_2 = p$

$$PL_n = \frac{3(C+1-p)n^2 - 3(2C^2+3C-2PC-P+1)n + C(C+1)(2C+1)}{6(C+1-p)^2} \tag{3.9b}$$

The following theorem demonstrates the properties of PL_n.

Theorem 3.1. *The production loss function PL_n ($n = 0, ..., C$) is a convex and decreasing function of n, and $-C < PL_n < C$.*

Proof. See Appendix B. □

Theorem 3.1 indicates that, a higher n results in a smaller production loss (or a larger production gain) in period 2. Moreover, the production loss/gain is bounded by the buffer capacity.

Then, we come back to $PL_n^{(2)}$, the production loss in period 2. Based on remark 3.1, if the n in $PL_n^{(2)}$ is larger than C, then period 2 starts with a buffer level C (a full buffer), and if $n < 0$, period 2 starts with a buffer level 0 (an empty buffer). Therefore, $PL_n^{(2)}$ can be expressed as

$$PL_n^{(2)} = \begin{cases} PL_0 & n \le 0 \\ PL_n & 0 < n \le C \\ PL_C & n > C \end{cases} \tag{3.10}$$

where the closed-form expressions of PL_n are shown in Equation (3.9).

3.3.3 AMOW estimation based on the production loss

Next, we compute the AMOWs in general 2M1B lines. For any real-time buffer level $N(0)$, we define the upper and lower bounds of the feasible post-PM buffer levels as $N^U(N(0))$, and $N^L(N(0))$, respectively, i.e.,

$$N^U(N(0)) = \arg\max_n \big\{n : PL(N(0), n) \leq 0\big\} \tag{3.11}$$

$$N^L(N(0)) = \arg\min_n \big\{n : PL(N(0), n) \leq 0\big\} \tag{3.12}$$

Once these bounds are calculated, the AMOWs can be obtained from Definition 3.1, as

$$AMOW_1(N(0)) = (N(0) - N^L(N(0)))/p_2 \tag{3.13}$$

$$AMOW_2(N(0)) = (N^U(N(0)) - N(0))/p_1 \tag{3.14}$$

Clearly, the AMOWs for both machines are functions of $N(0)$. The following theorem demonstrates the monotonicity of the AMOWs with respect to $N(0)$.

Theorem 3.2. *$AMOW_1(N(0))$ and $AMOW_2(N(0))$ are non-decreasing in the real-time buffer level, $N(0)$. Moreover, $AMOW_2(N(0)) = AMOW_2(N^*(0))$ when $N(0) > N^*(0)$ where $N^*(0) = \min\{N(0) : PL(N(0), C) \leq 0\}$.*

Proof. See Appendix B. □

Theorem 3.2 shows the structures of $AMOW_1$ and $AMOW_2$ with regard to the real-time buffer level $N(0)$. It indicates that, the buffer with more contents can offer longer AMOWs to both machines. More specifically, if the PM is performed on M_1, a higher buffer level can keep M_2 running for a longer time, and thus the production loss in period 1 will be smaller; if the PM is performed on M_2, then based on Theorem 3.1, a higher buffer level will result in a smaller production loss in period 2. This result further demonstrates that AMOWs exist from the extra contents in the buffer. The second part of Theorem 3.2 states that, as $N(0)$ reaches a threshold value $N^*(0)$, $AMOW_2$ will be constant as $N(0)$ increases. In the following section, we will further investigate the properties of AMOWs through numerical examples.

3.3.4 Model exploration and insights

In this section, we present numerical studies to explore the structural properties of the lower and upper bounds, with which the AMOWs can be calculated according to Equations (3.13) and (3.14). We test the AMOW estimation methods in various system settings and obtain meaningful insights from numerical studies.

In our preliminary experiments, we take three cases $p_1 = p_2$, $p_1 > p_2$, and $p_1 < p_2$ as typical scenarios in a 2M1B line.

Case 1 $p_1 = p_2 = p$:

The key to finding the AMOW for any given real-time buffer level $N(0)$, is to calculate the production loss for the post-PM buffer level n. The numerical examples for $p = 0.95, 0.80$, and $C = 20, 50$, are investigated as shown in Fig. 3.5, where the area of the feasible solutions satisfying $PL(N(0), n) \leq 0$ is shaded. The diagonal line D is the line $n = N(0)$, so the shaded area below (resp., above) line D indicates the existence of $AMOW_1$ (resp., $AMOW_2$). For each $N(0)$, the lower and upper bounds of the shaded region are the lower bound (N^L) and upper bound (N^U) for the post-PM buffer levels, respectively. (In this section, the "$N(0)$" in $N^L(N(0))$ and $N^U(N(0))$ are omitted for brevity.) Moreover, in Fig. 3.5, $AMOW_1(N(0))$ (resp., $AMOW_2(N(0))$) is illustrated as the vertical distance between its corresponding N^L (resp., N^U) and line D, multiplied by a scale parameter $1/p_1$ (resp., $1/p_2$). For example, in the line where $p_1 = p_2 = 0.95$ and $C = 20$, if the real-time buffer level $N(0) = 15$, it can be seen from Fig. 3.5 that $N^L = 9$ and $N^U = 18$. Then we obtain

$AMOW_1(15) = (15 - 9)/0.95 = 6.316$ and $AMOW_2(15) = (18 - 15)/0.95 = 3.158$.

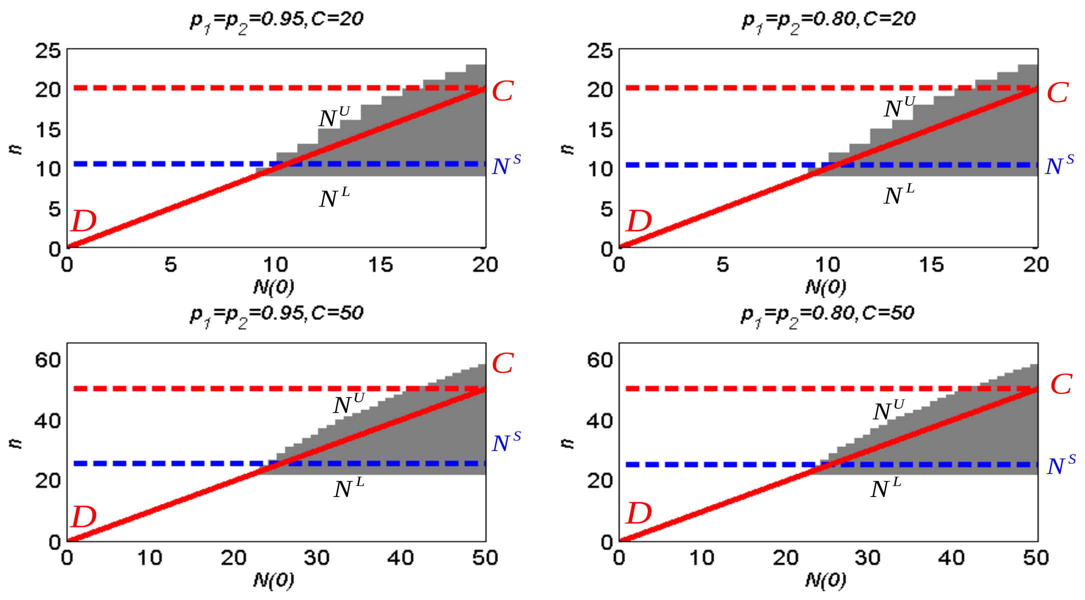


Figure 3.5: The structures of N^U and N^L in balanced lines

Figure 3.5 indicates that N^L is constant when $N(0)$ varies. The reason is that, when PM is performed on M_1, the production loss mainly comes from period 2. However, for PM on M_2, period 1 also contributes to the production loss, resulting in an N^U dependent on $N(0)$. Figure 3.5 also demonstrates that, when N^U is smaller than the buffer capacity C, it increases at least as fast as the diagonal line D. When N^U exceeds C, $N^U - C$ becomes a constant. This result agrees with Theorem 3.2.

Moreover, Fig. 3.5 also shows that N^L is less than the steady-state buffer level N^S (see [46] for its calculation), which is the dashed line between 0 and the buffer capacity. The following proposition addresses the relationship between N^L and N^S in balanced lines.

Proposition 3.1. *In a 2M1B line where $p_1 = p_2$, the lower bound for the post-PM buffer level N^L and the steady-state buffer level N^S satisfy $N^L \leq \lceil N^S \rceil$, where $\lceil N^S \rceil$ is the smallest integer that is greater than or equal to N^S.*

Proof. See Appendix B. □

Last but not least, Fig. 3.5 illustrates that, for a given $N(0)$, N^L's are the same for $p = 0.90$ and 0.80; and so are the N^U's. As the machine reliability decreases, the scale parameter $1/p$ increases, and accordingly, both $AMOW_1$ and $AMOW_2$ increase. This is because the parts in the buffer can keep a less reliable machine running for a longer time.

To summarize the results above, in a balanced line where $p_1 = p_2 = p$: (1)AMOWs exist for both machines when $N(0)$ exceeds half of the buffer capacity; (2) for $AMOW_1$, there exists a lower bound for the post-PM buffer level N^L, which is independent of $N(0)$ and approximately half of the buffer capacity; (3) for AMOW2, the upper bound for the post-PM buffer level N^U is dependent on $N(0)$; and (4) for a given $N(0)$, AMOWs increase as the machine reliability decreases.

Case 2 $p_1 > p_2$:

The $PL(N(0), n)$'s for $p_1 = 0.95, 0.96$, $p_2 = 0.94$, and $C = 20$ are plotted in Fig. 3.6. The shaded area above line D indicates that M_2 has almost no AMOW. This is because M_2 is more critical for both short-term and long-term performance, which should be less preferred to be actively stopped. Fig. 3.6 also shows that there exists a fixed lower bound N^L, which is positive but smaller than N^S. This is similar to the situation in a balanced line. In fact, the following result holds for production lines where $p_1 \geq p_2$.

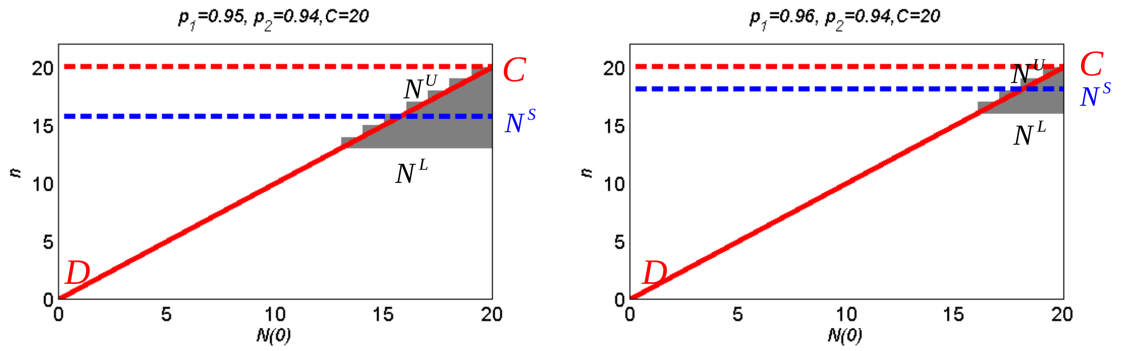


Figure 3.6: The structures of N^U and N^L when $p_1 > p_2$

Proposition 3.2. *In a 2M1B line where $p_1 \geq p_2$, $N^L(N(0)) \geq 0$ for any $N(0)$.*

Proof. See Appendix B. $\square$

Proposition 3.2 demonstrates that if $p_1 \geq p_2$, M_1 should be resumed no later than the buffer becomes empty; otherwise, M_2 will be starved and thereby interrupting the throughput.

Figure 3.6 also implies that, N^L increases as p_1 increases. However, based on one's intuition, if p_1 gets larger, it is easier for M_1 to catch up with M_2 eventually, so M_1 should have a longer AMOW. We now introduce *a slack constraint* to interpret such counterintuitive phenomenon.

We investigate the behavior of N^U and N^L when introducing a slack constraint such that a small production loss is allowed, namely, the constraint $\Lambda = \{n : PL(N(0), n) \leq 0\}$ in Definition 3.1 is replaced by $\Lambda = \{n : PL(N(0), n) \leq \Delta\}$, where Δ is a positive number representing the allowed production loss. Figure 3.7 shows N^U and N^L of the two systems when the slackness is set as $\Delta = 0.1$ and 0.5. Comparing Fig. 3.6 and Fig. 3.7, one can see that N^L decreases as Δ increases, indicating that there exist a longer AMOW for M_1 if a larger production loss is allowed. Moreover, N^L decreases as p_1 increases, indicating that a larger p_1 results a longer $AMOW_1$, which now agrees with one's intuition.

The following results summarize the characteristics of AMOWs in a 2M1B line where $p_1 > p_2$, and $PL(N(0), n) \leq \Delta$ ($\Delta \geq 0$): (1) M_2 has almost no AMOWs; (2) the lower bound N^L w.r.t. PM on M_1 is constant, which is smaller than N^S; and (3) N^L is non-increasing in Δ and p_1.

Case 3: $p_1 < p_2$

The last case is for the lines where $p_1 < p_2$. In this case, M_2 is more critical in period 1 while M_1 is more critical from the long-term perspective. With a fixed $p_2 = 0.96$, the feasible solution areas for $PL(N(0), n) \leq 0$ under varying $p_1 = 0.95, 0.90, 0.85$, and 0.80, are shown in Fig. 3.8.

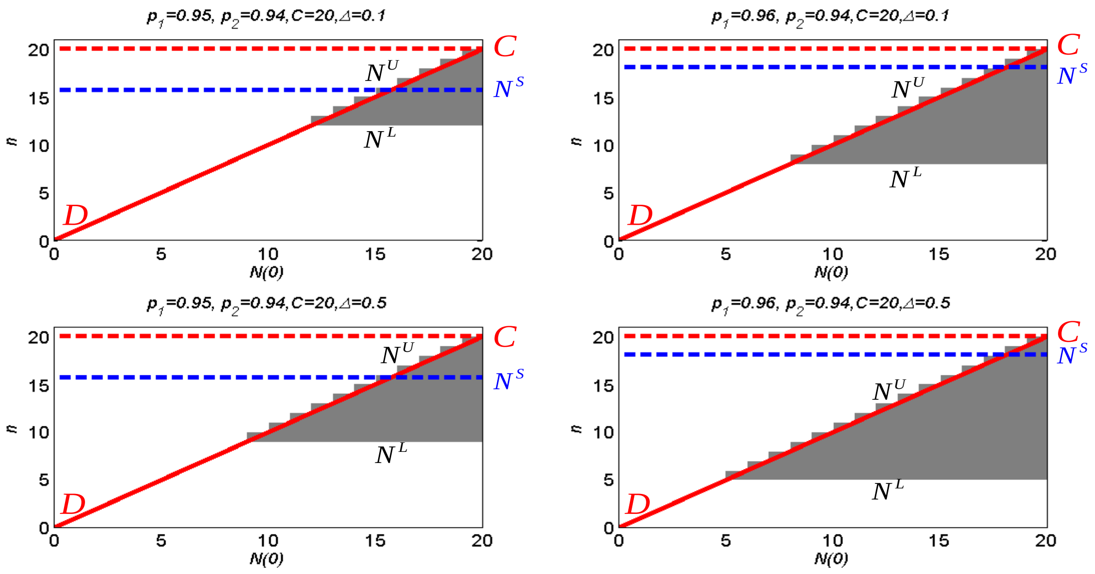


Figure 3.7: The structures of N^U and N^L when $p_1 > p_2$ and $PL(N(0), n) \leq \Delta$

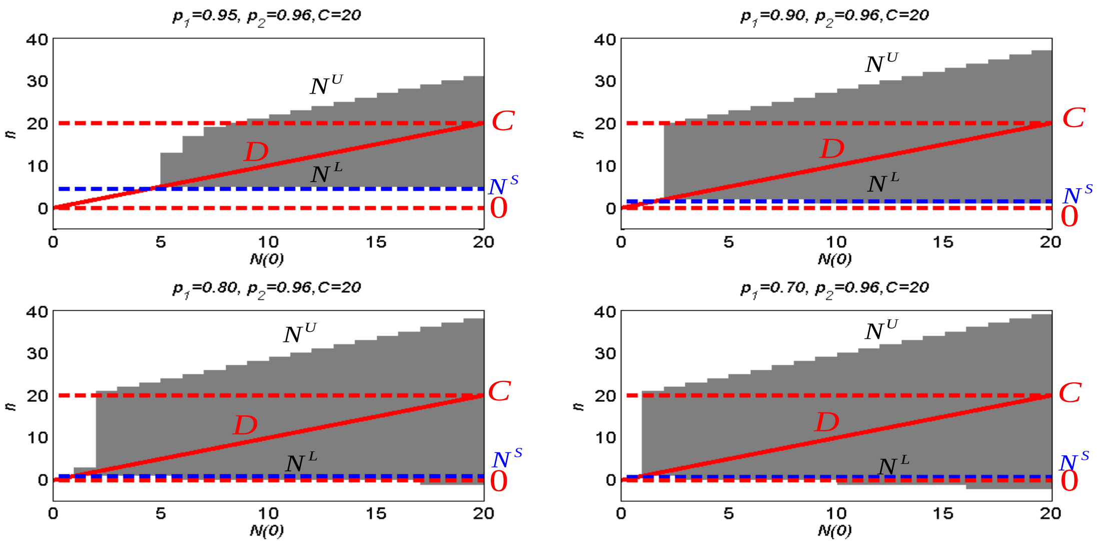


Figure 3.8: The structures of N^U and N^L when $p_1 < p_2$

As shown in Fig. 3.8, given a fixed p_2, the change of p_1, the reliability of M_1, could affect both $AMOW_1$ and $AMOW_2$, and result in changes of the feasible solution area. If p_1 is close to p_2, the structure is similar to the one in a balanced line, i.e., N^L is constant while N^U has a step-like shape. As p_1 decreases, N^U increases, which agrees with one's intuition that as p_1 gets smaller, the system production rate depends more on M_1. This provides more maintenance opportunities to M_2.

However, one may argue that there should be no AMOW for M_1, especially when p_1 is much smaller than p_2, while in Fig. 3.8 there is quite an amount of shaded area below the diagonal line D, and in the last two cases there are even some negative N^L's. The explanation for the counterintuitive behavior is that, the real-time AMOW estimation only considers meeting the future production requirement, while any production loss in the past is neglected. For example, if the real-time buffer level $N(0)$ is found to be greater than its steady-state buffer level N^S, one of the possible reasons is that M_2 has suffered a downtime previously, while this downtime may have already induced some production losses. We introduce a tight constraint by considering such a previous production loss.

In order to compensate for a previous production loss, we investigate the problem of *a tighter constraint* on the production loss. In practice, we can specify a required production gain that is needed to compensate for the production loss in the past, based on the production information. Here, it is assumed that (1) if $N(0) \leq N^S$, no production losses are allowed ($PL(N(0), n) = 0$); (2) if $N(0) > N^S$, a production gain of $N(0) - N^S$ is required to compensate for the previous production loss. In other words, the constraint in Definition 3.1 is changed into $\Lambda = \{n : PL(N(0), n) \leq \Delta\}$, where $\Delta = \min(0, -(N(0) - N^S))$. However, under this constraint, there is almost no AMOW for either M_1 or M_2, so we relax Δ as $\Delta = \min(0, -(N(0) - N^S) + 0.1)$. The results are shown in Fig. 3.9. It demonstrate that, under the tight constraint, there is almost no $AMOW_1$, because M_1 is less reliable than M_2. Moreover, when p_1

decreases, N^U gets larger, indicating a larger $AMOW_2$. The reason is that, when p_1 is smaller, M_2 is more likely to eventually catch up with M_1 in period 2.

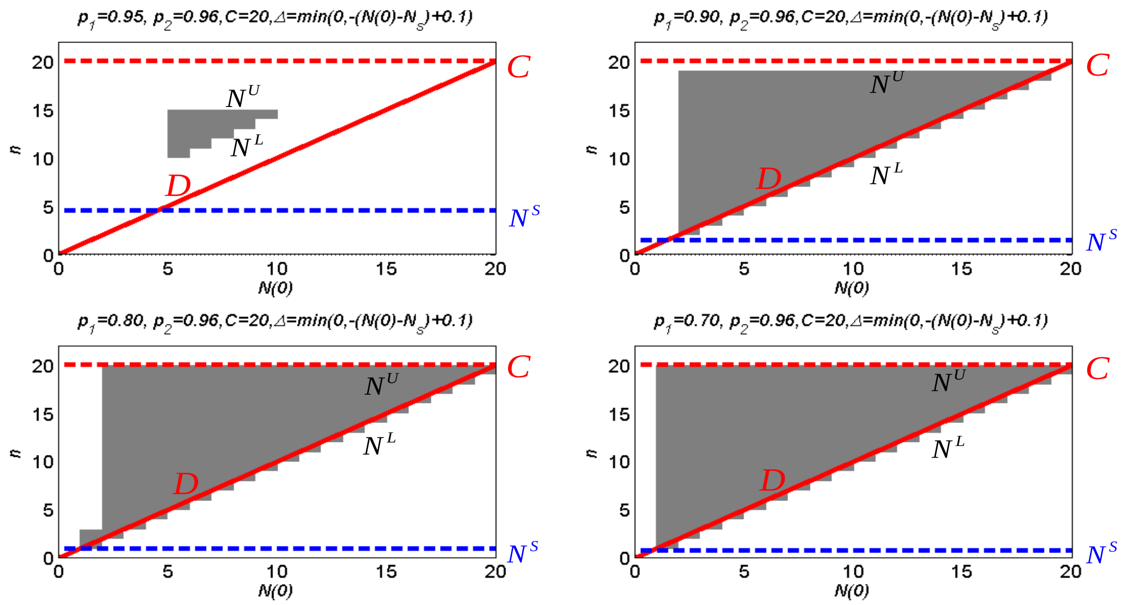


Figure 3.9: The structures of N^U and N^L when $p_1 < p_2$, $PL(N(0), n) \leq \Delta$ and $\Delta = \min(0, -(N(0) - N^S) + 0.1)$

In summary, in a line where $p_1 < p_2 = p$, and $PL(N(0), n) \leq \Delta$: (1) if $\Delta = 0$, there exist AMOWs for both M_1 and M_2. If p_1 is smaller, $AMOW_1$ is smaller while $AMOW_2$ is larger; and (2) if $\Delta = \min(0, -(N(0) - N^S) + \delta)$, M_1 has no AMOW, while $AMOW_2$ increases as p_1 decreases or δ increases (δ is a small positive number).

This section shows that, in a general 2M1B line with a real-time buffer level $N(0)$, the upper and lower bounds of the post-PM buffer level n can be estimated based on the analytical $PL(N(0), n)$, and the AMOWs for both machines can be calculated correspondingly. In the next section, we extend the AMOW estimation problem to manufacturing systems consisting of more than two machines.

3.4 Estimation of AMOWs in Manufacturing Systems

3.4.1 AMOW estimation based on system decomposition

Our objective is to generalize the findings of maintenance opportunities from a 2M1B line to systems with I ($I > 2$) machines (e.g., the 11M11B system shown in Fig. 3.1). However, as the number of machines, I, increases, the interaction between machines and buffers is significantly complicated, and the size of the state space $|S| = \prod_{i=1}^{B}(1 + C_i)$ increases dramatically, which makes the exact analysis of the problem intractable. To reduce the computational complexity in the analysis, some approximation methods have been developed, such as the decomposition method [44] and the aggregation method [46]. The traditional aggregation method is used to analyze the system performance in the steady state. Recently, Zhang et al. [57] developed a recursive procedure to obtain the reliabilities of the aggregated machines in transients, and evaluated the transient performance of serial lines. Here, we extend the work in [57] to system with more complex structures.

In this chapter, we use decomposition method to study AMOW in manufacturing systems. The idea is to view the system from each buffer's perspective, and decompose its upstream (resp., downstream) machine as a virtual machine which is non-starved (resp., non-blocked). For instance, for buffer B_i, we decompose its upstream and downstream machines as two virtual machines, namely, $M^{NS_i}_{U(B_i)}$ and $M^{NB_i}_{D(B_i)}$, where the superscripts 'NS_i' (resp., 'NB_i') stands for non-starvation (resp., non-blockage) for buffer B_i, and $U(B_i)$ (resp., $D(B_i)$) is the index of the machine on the upstream (resp., downstream) of B_i. We also denote $U(M_i)$ (resp., $D(M_i)$) as the set of the indices of the buffers on the upstream (reps., downstream) of machine M_i.

Then we set equations to calculate the reliabilities of the decomposed machines. Note that, machine will be starved for buffer B_i if one of its immediately downstream buffers (except B_i) is full, and the following machine fails to take one part out.

Therefore,

$$\begin{aligned} p_{U(B_i)}^{NS_i}(k) = p_{U(B_i)}(k) \cdot & \prod_{j_1 \in U(M_{U(B_i)})} \left(\begin{bmatrix} 0 & 1 & \cdots & 1 \end{bmatrix} \boldsymbol{\pi}_{j_1}(k-1) \right) \\ & \cdot \prod_{\substack{j_2 \in D(M_{U(B_i)}) \\ j_2 \neq i}} \Big(\begin{bmatrix} 1 & \cdots & 1 & p_{D(B_{j_2})}^{NB_{j_2}}(k) \end{bmatrix} \boldsymbol{\pi}_{j_2}(k-1) \Big) \end{aligned} \tag{3.15}$$

Similarly, for the downstream decomposed machine $M_{U(i)}^{NB_i}$, we have

$$\begin{aligned} p_{D(B_i)}^{NB_i}(k) = p_{D(B_i)}(k) \cdot & \prod_{\substack{j_1 \in U(M_{U(B_i)}) \\ j_1 \neq i}} \left(\begin{bmatrix} 0 & 1 & \cdots & 1 \end{bmatrix} \boldsymbol{\pi}_{j_1}(k-1) \right) \\ & \cdot \prod_{j_2 \in D(M_{U(B_i)})} \Big(\begin{bmatrix} 1 & \cdots & 1 & p_{D(B_{j_2})}^{NB_{j_2}}(k) \end{bmatrix} \boldsymbol{\pi}_{j_2}(k-1) \Big) \end{aligned} \tag{3.16}$$

Equations (3.15) and (3.16) can be built for each buffer. Therefore, for a system consisting of B buffers, $2B$ equations can be built, from which we can get the reliabilities for the $2B$ decomposed machines. Once these reliabilities (i.e., $p_{U(B_i)}^{NS_i}$ and $p_{D(B_i)}^{NB_i}$; $i = 1, \ldots, B$) are obtained, the buffer states at time k can be updated as

$$\boldsymbol{\pi}_i(k) = \mathbf{P}(p_{U(B_i)}^{NS_i}(k), p_{D(B_i)}^{NB_i}(k), C_i)\boldsymbol{\pi}_i(k-1) \quad \forall i \tag{3.17}$$

Then we calculate the maintenance opportunity window $AMOW_i(\mathbf{N}(0))$ for any machine M_i, given any prior-PM buffer level $\mathbf{N}(0)$. Denote T_i as the time duration that M_i is stopped for PM, then the reliabilities of the undecomposed machines can be updated as

$$p_i(k) = \begin{cases} 0 & \text{if } k \leq T_i \\ p_i & \text{if } k > T_i \end{cases} \text{and } p_j(k) = p_j \text{ for } j \neq i \tag{3.18}$$

By substituting Equation (3.18) into Equations (3.15) and (3.16), and then following Equations (3.15) to (3.17) recursively, we can calculate the reliabilities of all decomposed machines at $k = 1, 2, \dots$ until we find a $K >> T_i$ such that

$$\max\left(\mid p_{U(B_i)}^{NS_i}(K) - p_{U(B_i)}^{NS_i}(K-1) \mid , \mid p_{D(B_i)}^{NB_i}(K) - p_{D(B_i)}^{NB_i}(K-1) \mid \right) \le \epsilon \quad \forall i$$

Then the production loss under T_i can be calculated as

$$PL(\mathbf{N}(0), T_i) = K \cdot PR(K) - \sum_{j=1}^{K} PR(j) \tag{3.19}$$

Remark 3.2. In a complex system, we directly use the PM time as the decision variable, instead of the post-PM buffer level that is used in the 2M1B system. In a 2M1B system, the buffer level decreases (resp., increases) when PM is performed on the upstream (resp., downstream) machine, so the optimal post-PM buffer level can be directly translated into AMOW. However, in a complex system, even if the optimal post-PM buffer levels can be found, it is difficult to translate them into AMOWs, because of the complex dynamics of the system.

Remark 3.3. ϵ is a predetermined small positive number used as the termination criteria for stopping the iterations. We set $\epsilon = 10^{-8}$ in the following examples.

The flowchart for AMOW estimation is shown in Fig. 3.10. In the following sections, we use the algorithm to investigate the AMOW in serial lines and systems with complex structures.

3.4.2 AMOW estimation in serial lines

The recursive algorithm can be utilized to estimate the AMOW for any machine in a serial line. As an illustration, we consider eight different 5M4B lines, namely, Line 1 to 8, with different machine reliabilities, as shown in Table 3.1. The buffer capacities for all lines are $\mathbf{C} = [10\ 10\ 10\ 10]$.

Note that, in Line 1, all the aggregated 2M1B lines are balanced, i.e., $p_i^{NS} = p_{i+1}^{NB}$

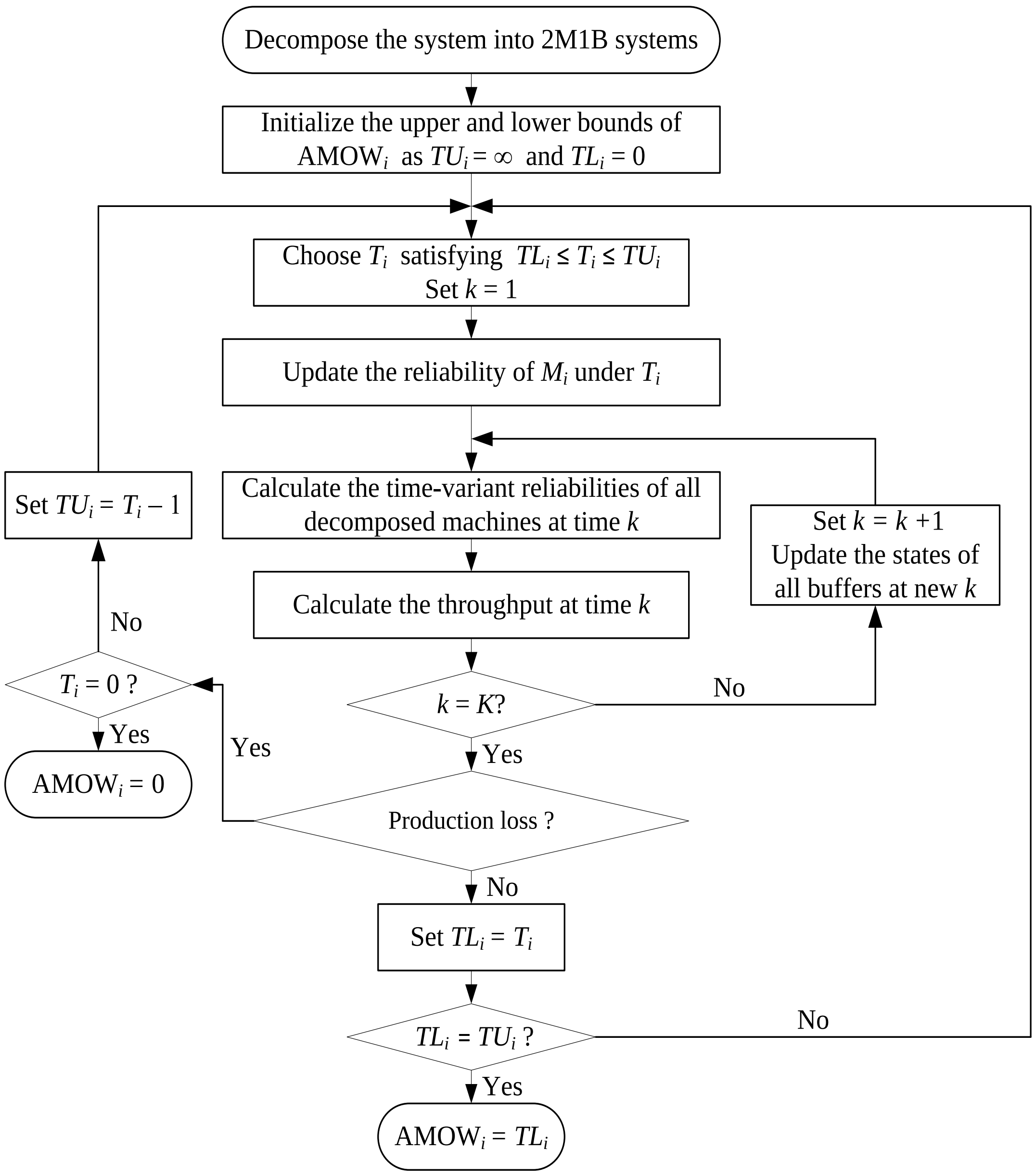

Decompose the system into 2M1B systems
Initialize the upper and lower bounds of AMOW$_i$ as $TU_i = \infty$ and $TL_i = 0$
Choose T_i satisfying $TL_i \le T_i \le TU_i$ Set $k = 1$
Update the reliability of M_i under T_i
Calculate the time-variant reliabilities of all decomposed machines at time k
Calculate the throughput at time k
$k = K$?
No
Set $k = k + 1$ Update the states of all buffers at new k
Yes
Production loss ?
Yes
$T_i = 0$?
No
Set $TU_i = T_i - 1$
Yes
AMOW$_i$ = 0
No
Set $TL_i = T_i$
$TL_i = TU_i$?
No
Yes
AMOW$_i$ = TL_i

Figure 3.10: Flowchart for AMOW estimation

Table 3.1: Machine reliabilities in eight different lines

Line	machine reliabilities				
	p_1	p_2	p_3	p_4	p_5
1	0.8943	0.9038	0.9038	0.9038	0.8943
2	0.9	0.9	0.9	0.9	0.9
3	0.95	0.9	0.9	0.9	0.9
4	0.9	0.9	0.95	0.9	0.9
5	0.9	0.9	0.9	0.9	0.95
6	0.85	0.9	0.9	0.9	0.9
7	0.9	0.9	0.85	0.9	0.9
8	0.9	0.9	0.9	0.9	0.85

for $i = 1, 2, 3, 4$ (In serial lines, any machine cannot have more than one upstream buffers or downstream buffers, so $p_i^{NS_i}$ and $p_{i+1}^{NB_i}$ can be simplified as p_i^{NS} and p_{i+1}^{NB}, respectively, without introducing any confusion). Here, we call such a line a balanced line, and the way to construct it can be found in [46]. All the machines in Line 2 have an identical reliability. For Lines 3 through 5, one machine in each line is more reliable than the other machines. The more reliable machine is located in the beginning of Line 3, in the middle of Line 4 and at the end of Line 5. Lines 6 through 8 are similar to Lines 3 through 5 respectively, except that the more reliable machine ($p = 0.95$) is replaced by a machine that is less reliable than the others ($p = 0.85$).

For each line, the AMOWs are estimated under seven cases with different real-time buffer levels $\mathbf{N}(0)$, as shown in Table 3.2. We denote Case 1 as the baseline case; Case 2 and Case 3 as Pair 1 (where only the initial buffer levels of B_4 are changed); Case 4 and Case 5 as Pair 2 (where only the initial buffer levels in B_1 are changed); and Case 6 and Case 7 as Pair 3 (where buffer levels in B_2 and B_3 are changed). The comparison of the AMOWs in Case 1 and the three pairs are plotted in the three columns in Fig. 3.11. In each subplot, the horizontal axis is the machine index i, and the vertical axis is the AMOW calculated by the recursive algorithm.

Figure 3.11 reveals how system parameters affect AMOWs in long lines. First, AMOWs depend on the machine reliabilities. The machine with a higher (resp., lower)

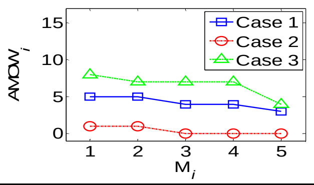
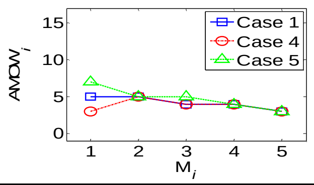
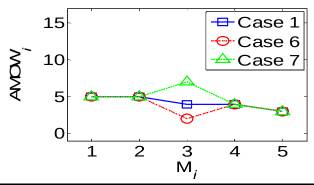
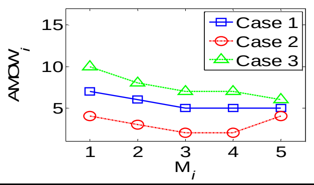
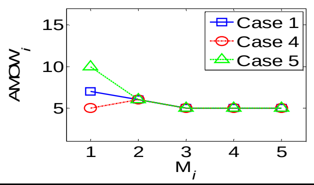
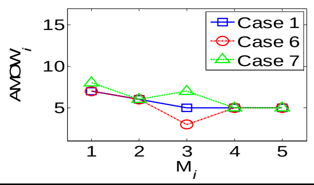
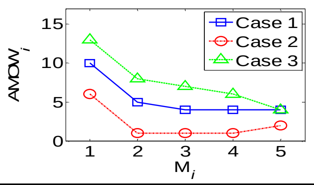
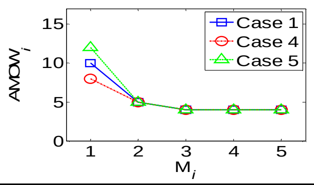
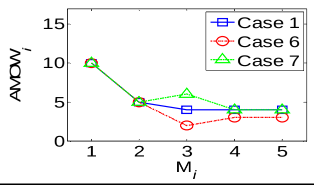
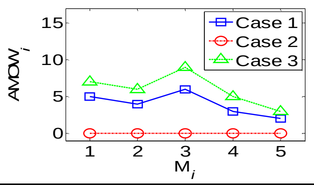
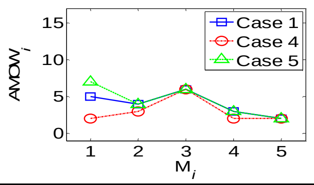
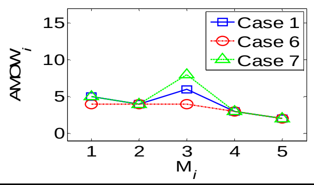
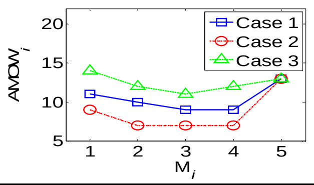
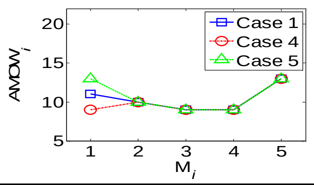
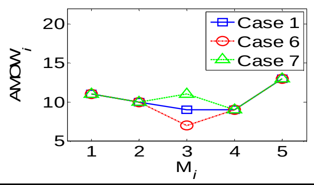
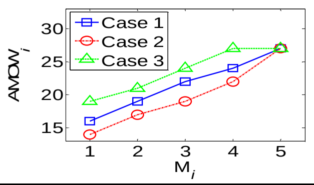
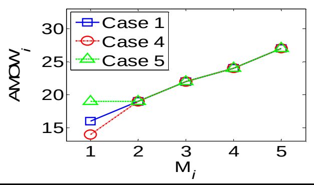
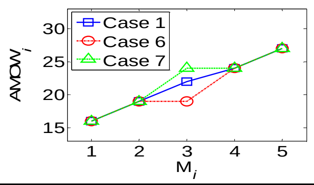
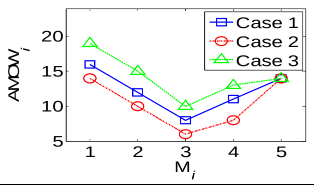
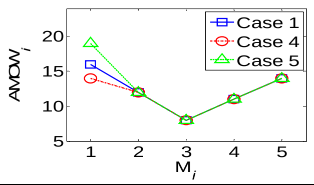
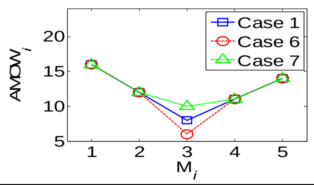
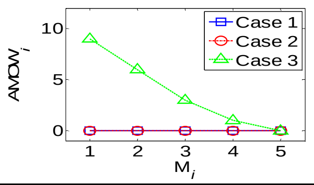
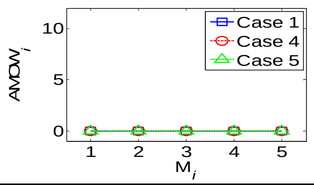
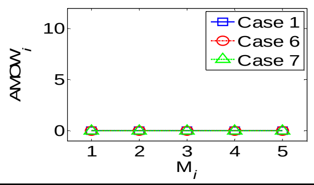
	Comparison of AMOWs in Case 1 and Pair 1	Comparison of AMOWs in Case 1 and Pair 2	Comparison of AMOWs in Case 1 and Pair 3
Line 1			
Line 2			
Line 3			
Line 4			
Line 5			
Line 6			
Line 7			
Line 8			

Figure 3.11: Comparison of the AMOWs in 5M4B lines

Table 3.2: Real-time buffer levels in seven cases

case	$\mathbf{N}(0)$	case	$\mathbf{N}(0)$	case	$\mathbf{N}(0)$
1	[6 6 6 6]	2	[6 6 6 4]	3	[6 6 6 8]
		4	[4 6 6 6]	5	[8 6 6 6]
		6	[6 8 4 6]	7	[6 4 8 6]

reliability has a longer (reps., shorter) AMOW (See Lines 3 through 7). Second, the AMOW is also sensitive to the position of the machine. The machine closer to the most (resp., least) reliable machine has a longer (resp., shorter) AMOW than the distant machines (See Lines 3 through 7). Last, the real-time buffer levels play an important role in AMOWs. A higher prior-PM level in one buffer results in longer AMOWs, especially for its upstream machines (See Columns 1 and 2). Also, different prior-PM levels in a distant upstream buffer have little influence on the AMOWs of downstream machines (See Column 2).

Note that, not all real-time buffer levels result in AMOWs. For example, no AMOW exists in Line 8 in any case but Case 3. In Line 8, the least reliable machine is located at the end of the line, and thus Line 8 has a high steady-state work-in-process (which can be calculated as [8.39 8.37 8.37 8.37]). Among all the cases, only the real-time buffer levels in Case 3 are sufficient to provide AMOWs.

Based on these results, the monotonicity property in Theorem 3.2 can be extended to long lines, as Theorem 3.3 below.

Theorem 3.3. *In an I-machine-$(I-1)$-buffer line, $AMOW_i(\mathbf{N}_B(0)) \geq AMOW_i(\mathbf{N}_A(0))$ for all $i = 1, \ldots, I$, if $\mathbf{N}_B(0) > \mathbf{N}_A(0)$.*

Proof. See Appendix B. $\square$

The above results give insight on the structures and values of AMOWs in long lines. Since the production lines are usually well balanced, we propose a heuristic algorithm to estimate AMOWs in balanced lines in the next section.

3.4.3 Heuristic algorithm for AMOW estimation in balanced lines

Although AMOWs can be effectively estimated by the recursive algorithm in Section 3.4.2, it is computationally intensive, especially when the line is long, the buffer size is large, or the parameter ϵ is small. Therefore, it is not computationally efficient for real-time applications. In this section, we propose a heuristic algorithm to estimate AMOWs in balanced lines more efficiently.

As mentioned in Section 3.4.2, an I-machine-$(I-1)$-buffer balanced line can be analyzed by $I-1$ decomposed 2M1B balanced systems, where the reliabilities of all decomposed machines are the same, as $p_1 = p_i^{NS} = p_{i+1}^{NB} = p_I$ $(i = 2, \ldots, I-1)$. To estimate the AMOW of M_i $(i = 1, \ldots, I)$ in such a line, we consider the prior-PM levels of both its upstream and downstream buffers.

First, we estimate the AMOW provided by its *downstream* buffers, and denote it as $AMOW_i^d(\mathbf{N}(0)$. From Section 3.3.4, in a balanced 2M1B line, the lower bound of the post-PM buffer level is approximately half of the buffer capacity. Therefore, for the last buffer B_{I-1}, if its real-time buffer level $N_{I-1}(0)$ is larger than $C_{I-1}/2$, the excessive parts can provide the decomposed machine M_{I-1}^{NS} an AMOW of $(N_{I-1}(0)-C_{I-1}/2)/p_1$. Within this time period, all of the machines M_1 to M_{I-1} in the original line can be stopped for maintenance. After this time period, the excessive parts in buffer B_{I-2} can offer M_{I-2}^{NS} an AMOW, which is equal to $(N_{I-2}(0) - C_{I-2}/2)/p_1$. Based on this argument, $AMOW_i^d(\mathbf{N}(0)$ can be calculated from the excessive contents in any downstream buffer B_j $(j = i, \ldots, I-1)$:

$$AMOW_i^d(\mathbf{N}(0) = \sum_{j=i}^{I-1}(N_j(0) - C_j/2)/p_1 \tag{3.20}$$

Then we consider $AMOW_i^u(\mathbf{N}(0)$, the AMOW provided by the upstream buffers of M_i. From Section 3.4.2, the prior-PM levels in the distant upstream buffers have little impact on the AMOW. Taking this fact into consideration, we group all the upstream buffers together to build a virtual buffer $\widetilde{B}_{i-1}$, whose capacity and real-time buffer level are $\widetilde{C}_{i-1} = \left[\sum_{j=1}^{i-1} p_1^{i-1-j} C_j\right]$, and $\widetilde{N}_{i-1}(0) = \left[\sum_{j=1}^{i-1} p_1^{i-1-j} N_j(0)\right]$, respectively. p^{i-1-j} is the weighting parameter, which increases when j increases (i.e., when buffer B_j is closer to machine M_i). Then the AMOW can be calculated in a 2M1B line $M_{i-1}^{NS} - \widetilde{B}_{i-1} - M_i^{NB}$, where both machines have reliability p_1. We assume that when M_i is down for $AMOW_i^d(\mathbf{N}(0))$, the buffer level in $\widetilde{B}_{i-1}$ increases at a rate of p_1. Then the buffer level of $\widetilde{B}_{i-1}$ after $AMOW_i^d(\mathbf{N}(0))$ can be updated as

$$\widetilde{N}'_{i-1} = \min\left\{\widetilde{C}_{i-1}, \left[\widetilde{N}_{i-1}(0) + \sum_{j=i}^{I-1}(N_j(0) - C_j/2)\right]\right\},$$

with which $AMOW_i^u(\mathbf{N}(0))$ can be estimated. To be more specific, $AMOW_i^u(\mathbf{N}(0))$ can be approximated by the AMOW of the downstream machine in a 2M1B balanced line, with the machine reliability p_1, the buffer capacity $\widetilde{C}_{i-1}$, and the real-time buffer level $\widetilde{N}'_{i-1}$.

Finally, the heuristic AMOW can be expressed as the summation of the AMOWs provided by both the upstream and downstream buffers, as

$$AMOW_i(\mathbf{N}(0)) = \max\left(0, \lfloor AMOW_i^d(\mathbf{N}(0)) + AMOW_i^u(\mathbf{N}(0))\rfloor\right) \tag{3.21}$$

In order to evaluate the performance of the heuristic algorithm, we use it to estimate the AMOWs of each machine in Line 1 described in Section 3.4.2, and compare them with the AMOWs calculated from the recursive algorithm. The AMOWs in all seven cases of the real-time buffer levels are calculated, and the results are shown in Table 3.3.

Table 3.3 shows that in most case, the heuristic AMOWs are smaller than the ones calculated from the recursive algorithm, which indicates that the heuristic algorithm is conservative. The exception lies for machines M_1, M_3 and M_5 in Case 2, where the heuristic AMOWs are larger. However, when taking a detailed look at the production

Table 3.3: The AMOWs from the recursive algorithm (R) and the heuristic algorithm (H) in Line 1

case	$AMOW_1$		$AMOW_2$		$AMOW_3$		$AMOW_4$		$AMOW_5$	
	R	H	R	H	R	H	R	H	R	H
1	5	4	5	4	4	4	4	3	3	2
2	1	$2^{(a)}$	1	1	0	$1^{(b)}$	0	0	0	$1^{(c)}$
3	8	5	7	5	7	6	7	5	4	3
4	3	2	5	3	4	3	4	2	3	2
5	7	5	5	4	5	4	4	3	3	3
6	5	4	5	4	2	2	4	3	3	2
7	5	4	5	4	7	6	4	3	3	3
production losses under heuristic AMOWs: $(a)0.21, (b)0.02, (c)0.00$										

loss when one machine is shut down for its corresponding heuristic AMOWs, we find it is smaller than 0.21 production unit. Therefore, in this example, we conclude that the AMOWs calculated from the heuristic algorithm result no or small production losses.

Next, we investigate the performance of the heuristic algorithm in a balanced line with more general cases of real-time buffer level. We construct a 10M9B balanced line (Line 9) with $p_1 = p_{10} = 0.8923$, $p_i = 0.9010$ $(i = 2, \ldots, 9)$, and $C_i = 10$ $(i = 1, \ldots, 9)$. The real-time buffer level for each buffer can take values among 0, 5, and 10. Therefore, there are $3^9 = 19683$ cases of the prior-PM buffer in total. In each case, the heuristic AMOWs for all ten machines are calculated. We measure the extra production loss under the heuristic AMOWs as follows. In some cases (e.g., initially all buffers are near empty), even if the machine is not shut down for PM, the system still has production losses because the initial buffer levels are below the steady-state buffer levels. Such production losses are mainly attributed to the insufficient buffer contents, rather than AMOWs. Therefore, if a system has a positive production loss even when no PM is performed, the extra production loss is measured by the production loss under heuristic AMOWs minus the production loss when no PM is performed. Mathematically, the extra production loss ΔPL can be calculated as

$$\Delta PL = \min\left(0, \sum_{k=1}^{\infty}\left(PR^{0}(k) - PR^{req}\right)\right) + \sum_{k=1}^{\infty}\left(PR^{req} - PR^{AMOW}(k)\right) \qquad (3.22)$$

The histograms of the extra production losses under the heuristic AMOWs are plotted in Fig. 3.12. It shows that, the heuristic AMOW does not cause extra production losses in the system in most of the cases. Moreover, both the production gain (i.e., the absolute value of the extra production loss when it is negative) and loss are small, indicating that, the heuristic AMOW successfully takes advantages of the extra buffer contents (in terms of small production gains) while still satisfying the throughput requirement (in term of small production losses). These results further validate the effectiveness of the heuristic algorithm.

In summary, the proposed heuristic algorithm is conservative in general, and able to effectively predict the AMOWs in balanced lines given the real-time buffer levels. It should be noted that, the heuristic may not work so well in some unbalanced systems. One reason is that the lower bound of the post-PM buffer level in an unbalanced 2M1B line cannot be approximated by half of the buffer capacity, especially when some slackness is introduced. However, the AMOWs in unbalanced lines can still be estimated by the recursive algorithm, and heuristics will be investigated in the future.

3.4.4 AMOW estimation in complex systems

In the previous analysis, we assume that the constraint is to meet the long-term system production rate, which is equal to the steady-state production rate (Assumption (7)). Sometimes, however, the short-term throughput is of more practical interest. To this end, we need to estimate AMOWs without sacrificing the throughput requirement during a specific time horizon. This problem can also be solved by the AMOW algorithm listed in Section 3.4.1, with $K = K'$ and Equation (3.19) replaced

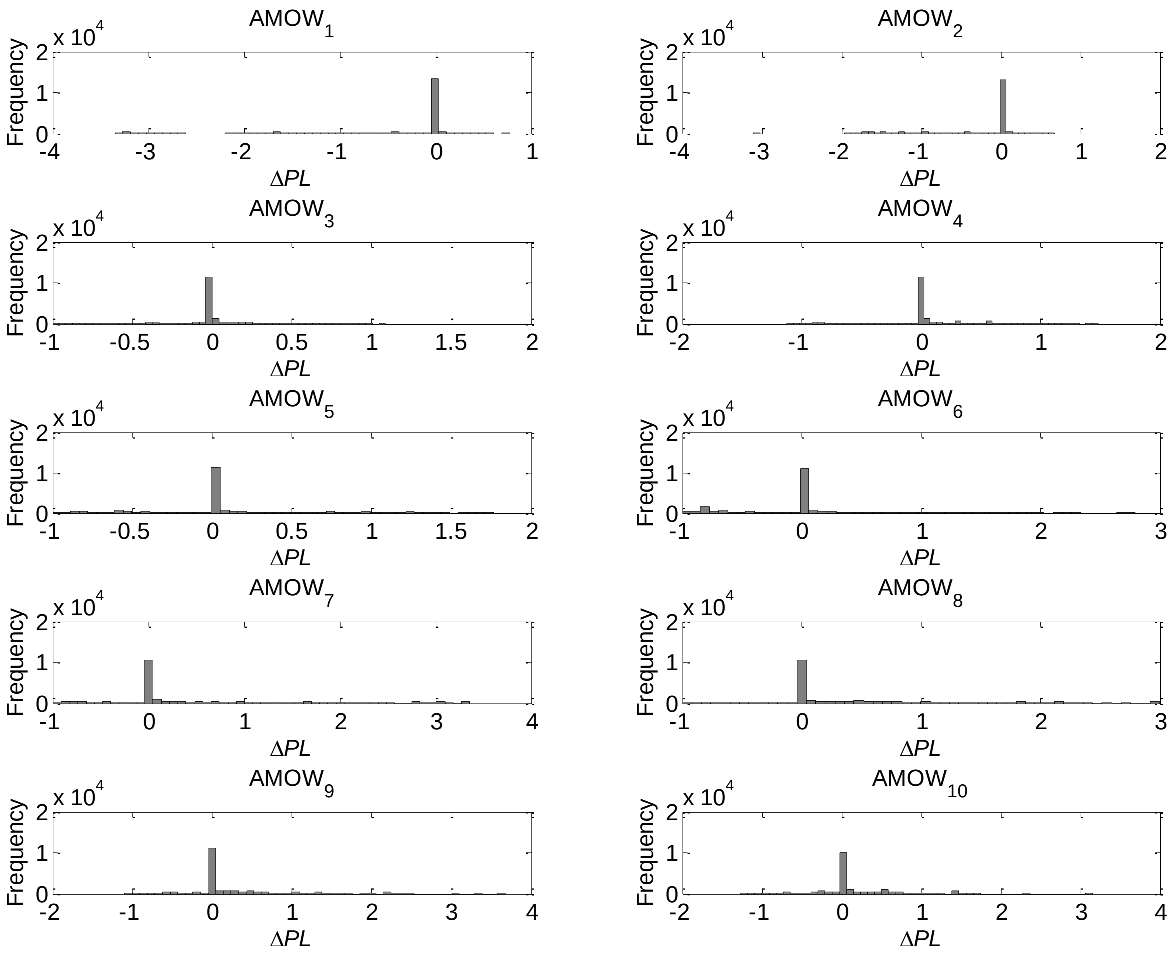


Figure 3.12: Extra production losses under the heuristic AMOWs in Line 9

by

$$PL(\mathbf{N}(0), T_i) = K' \cdot PR^{req} - \sum_{j=1}^{K'} PR(j) \tag{3.23}$$

where K' is the specified time horizon and PR^{req} is the required average production rate during the horizon.

The next numerical example estimates AMOWs for specific horizons. The system we study here is the one in Fig. 3.1, whose buffer capacities are shown in Table 3.4.

Table 3.4: Buffer capacities of the sample system

buffer	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}
capacity	10	10	10	10	10	10	10	15	12	10	12

Three cases of the real-time system information (denoted as Cases 1, 2 and 3) are studied here. The buffer contents and machine reliabilities of the three cases are shown in Table 3.3. Moreover, in each of the three cases, we estimate the AMOWs under three throughput requirements (denoted as Requirements 1, 2 and 3). Their time horizons are 500, 100 and 20, and the required production rate is 0.9 for all cases. The first requirement is a relatively long-term requirement and the last requirement is a relatively short-term one.

Table 3.5: Real-time information of the system under different cases

Case 1	buffer	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}
	level	6	5	5	5	5	4	5	8	6	5	6
	machine	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
	reliability	.94	.95	.95	.96	.95	.95	.95	.96	.95	.94	.95
Case 2	buffer	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}
	level	6	5	5	5	5	4	5	8	6	5	6
	machine	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
	reliability	.98	.95	.95	.91	.91	.91	.95	.96	.96	.98	.94
Case 3	buffer	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}
	level	6	4	3	2	6	7	7	8	3	8	8
	machine	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
	reliability	.94	.95	.95	.96	.95	.95	.95	.96	.95	.94	.95

The given system can be decomposed into eleven 2M1B systems, where machines

M_2, M_4, M_6 and M_7 are decomposed into three machines, and machines M_3, M_5 and M_{11} are decomposed into two machines. By following the procedure in Fig. 3.10, the AMOWs in each case can be estimated. Moreover, we compare the estimated AMOWs with the baseline policy, where the remaining time in the horizon is used for maintenance after the production requirement has been satisfied. The results are plotted in Fig. 3.13, which indicates that the AMOW-based maintenance policy will provide us with more time for maintenance than the policy that PM is performed after the production requirement is satisfied.

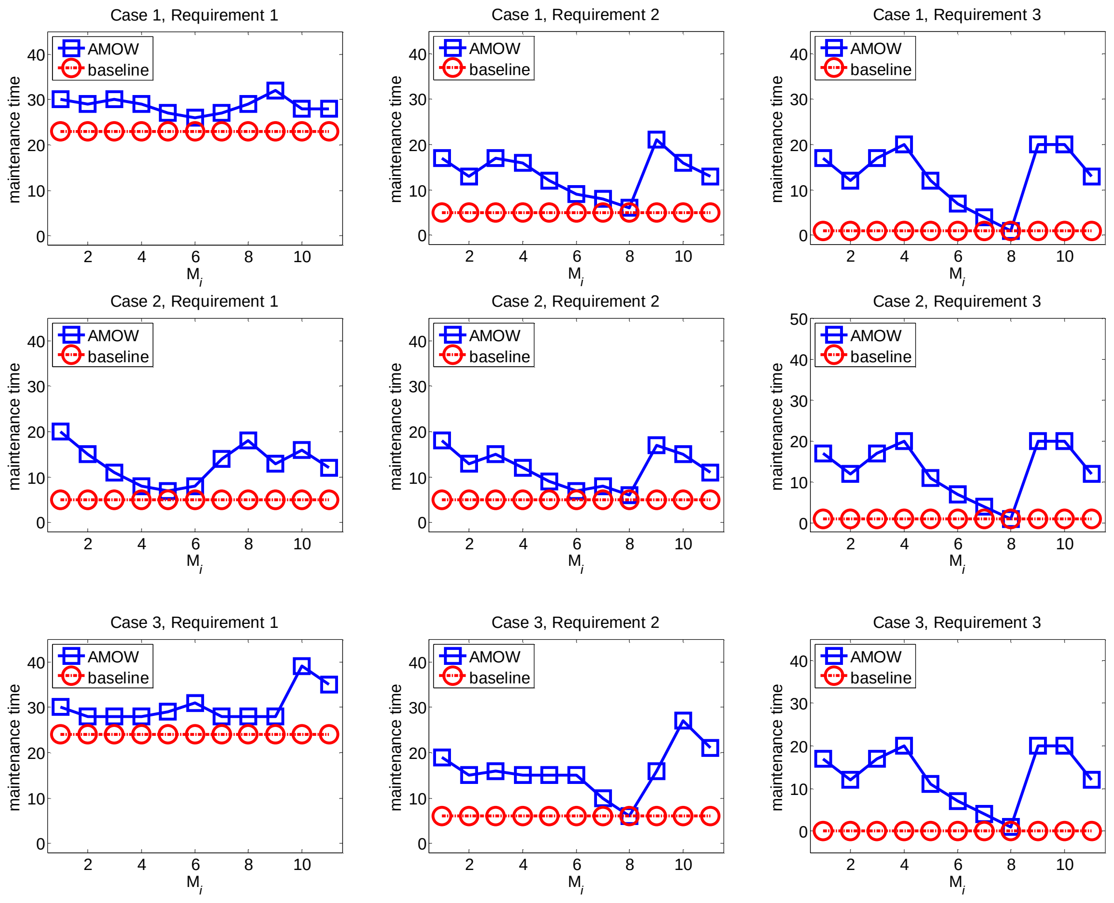


Figure 3.13: Comparison of AMOW and the baseline policy in the sample system

Figure 3.13 also shows how the AMOWs in the system are affected by the throughput requirement and its horizon. The end-of-line machine (i.e., M_8) is more critical for the short-term system throughput, and thus it has a shorter AMOW under Requirement 3. However, when the horizon is long, the less reliable machines (e.g., M_4, M_5 and M_6 in Case 2) in the system will be more important, and they will have shorter AMOWs. Also, the machines in the system with a higher reliability (Cases 1 and 3) have longer AMOWs than those in a less reliable system (Case 2), especially when the horizon of the throughput requirement is long. Also, since the AMOW-based policy performs PM earlier than the baseline policy, it can help to reduce the risk of machine breakdown due to degradation.

3.5 Summary

This chapter presents the concept of active maintenance opportunity windows (AMOWs) to seek real-time opportunities of performing preventive maintenance during production time. We have proposed analytical models to estimate how long one machine can be shut down for maintenance while still satisfying the system throughput requirement. We first build a Bernoulli model to estimate AMOWs analytically in general 2M1B lines. The results show that, in a balanced line, both machines have AMOWs when the real-time buffer level exceeds half of the buffer capacity. In an unbalanced line, the less reliable machine has a shorter AMOW. A recursive procedure based on decomposition method is used to investigate AMOWs in manufacturing systems with different configurations. We find that the AMOWs depend on multiple factors, such as throughput requirement, machine reliability, machine position and real-time buffer levels. We also propose a heuristic approach to estimate AMOWs in balanced lines, where the real-time contents in both upstream and downstream buffers are considered. The performance of the heuristic approach is evaluated with numerical studies.

However, as a decision support tool, AMOW does not make any decisions on how to schedule the maintenance tasks. To further develop maintenance policies in terms of scheduling and prioritization, we need to integrate the estimated AMOW with other information of the system, such as the degradation of machines, the list of PM tasks and their impact on machine reliabilities, the availability of maintenance resources, and the skill levels of maintenance crews. This integration will affect the estimation of AMOW, and make the problem more challenging.

CHAPTER IV

Maintenance Decision-Making in Manufacturing Systems

4.1 Introduction

In the previous chapters, we studied the propagation of a planned or an unplanned downtime in manufacturing systems consisting of machines whose reliabilities are constant over time. Based on the analysis of the system transient behavior, real-time maintenance decision support tools, including PMOW and AMOW, have been developed. In order to make optimal real-time maintenance decisions, one should consider other factors, such as machine degradation profiles, duration of maintenance tasks and their impact on the machine reliabilities, etc. In this chapter, we consider a maintenance decision-making problem by integrating such information.

In many models for single-unit systems, the optimal policy is a control limit policy, where maintenance or replacement is carried out when the degree of deterioration is greater than a critical level (or the machine reliability is below a critical level). The sufficient conditions for the optimality of control limit policies were discussed when the changes of state are Markovian [72] or Semi-Markovian [73, 74]. Douer and Yechiali [75] investigated the control limit policy where the effect of maintenance was stochastic. Stadje and Zuckerman [23] studied the policy with a general degree of repair.

However, in multi-unit systems, components are dependent in different ways, so that the maintenance decision on one machine depends not only on the machine itself, but also on the status of other machines in the system. Ambani et al. [36] used a continuous-time Markov chain model to develop CBM policy in serial production lines. Lee et al. [37] developed a Markovian model to find the optimal inspection policy for a manufacturing system with two machines in series or in parallel. The systems studied in their models consisted of no buffers. The buffer size and level, however, also play an important role in maintenance decision-making [35, 68, 76]. Van der Duyn Schouten and Vanneste [35] studied the optimal maintenance policy in an installation-buffer-production system, where the installation was subjected to random failures and was where the maintenance decision should be made. Kyriakidis and Dimitrakos [68] generalized the problem in [35] to the case where the installation had a non-stationary deterioration. Meller and Kim [76] investigated how the system cost and optimal buffer size were impacted by preventive maintenance (PM). In these studies, the production equipment (i.e., the downstream machine) was assumed to be perfectly reliable, and maintenance decision was only made on the installation (i.e.,the upstream machine), which limited their practical application.

Additionally, most of the previous work used a queueing model to study the problem, where the repair time was assumed to be exponentially distributed, and would resettle every time when the system dynamics changed. Such memoryless assumption cannot effectively capture the remaining maintenance time, which may be unrealistic in many cases.

In this chapter, we consider a serial line consisting of multiple machines and buffers, where each machine is subject to a degradation process. The maintenance decision should be made on each machine based on the real-time condition of the system, including the machine conditions and buffer levels. Moreover, the machine states consist of both the machine degradation states (when the machine is working) and the remaining maintenance time (when the machine is under maintenance).

The remaining chapter is structured as follows. In Section 4.2, the mathematical model and assumptions are introduced. In Section 4.3, a control limit policy is developed as the baseline policy, and it is evaluated in multi-stage systems. In Section 4.4, the optimal policy is investigated by using a Markov Decision Process (MDP) approach. Section 4.5 presents case studies to illustrate the structure of the optimal policy, and compare it with the baseline policy. Section 4.6 summarizes this chapter.

4.2 Model and Assumptions

4.2.1 Modeling of machine degradation

We consider a machine whose state can only change at discrete time $t = 1, 2, \ldots$. The degradation state of the machine is also discrete, and denoted by d ($d = 1, \ldots, D+1$), where a larger d represents a more severe degradation level. The assumptions of the machine and maintenance models are summarized as follows:

(1) When the degradation state of the machine is d ($d = 2, \ldots, D$), one can choose to perform a preventive maintenance (PM) or not.

(2) If PM is performed, the machine should be stopped for $T^{(d-1)}$ units of time to complete the PM, and the system will be restored to the "as good as new" state (i.e., degradation state 1).

(3) Given that PM is not performed, then during the machine cycle time, a random failure may occur with a probability $f^{(d)}$, which is dependent on the machine degradation state d ($d = 1, ..., D$). The random failure will be handled by a minimal repair (MR) [77], which restores the system to the "as bad as old" state (i.e., degradation state d, the state right before the random failure occurs). It is assumed the minimal

repair always takes one unit of time regardless of the machine degradation state.

(4) Given that PM is not performed and no random failure occurs, there may be a degradation. When the machine degradation state is d ($d = 1, ..., D$), the degradation state in the next time unit becomes $d + 1$ with probability $q^{(d)}$, and remains d with probability $\overline{q^{(d)}}$ (i.e.,$1 - q^{(d)}$). For simplicity, we assume $q^{(d)} = q \ \forall d$.

(5) When the degradation state of the machine is $D + 1$, a corrective maintenance (CM) must be performed. The CM takes $T^{(D)}$ units of time, and restores the system to the "as good as new" state (i.e., degradation state 1).

In addition, we assume the following two conditions hold.

Condition 5.1 $f^{(d+1)} > f^{(d)}$ for $0 < d \leq D$.

Condition 5.2 $T^{(d+1)} \geq T^{(d)}$ for $0 < d < D$ and $T^{(d+1)} - T^{(d)} \geq T^{(d)} - T^{(d-1)}$ for $1 < d < D$.

These two conditions state that, a random failure is more likely to occur when the machine is in a more severe degradation state. Also, it will take a longer time for the PM to be completed when the machine is more degraded. Moreover, the PM duration is a convex function on the degradation state d.

The state of the machine can be defined as $\mathcal{S} = \mathcal{S}^{\mathcal{D}} \times \mathcal{S}^{\mathcal{M}}$, where $\mathcal{S}^{\mathcal{D}}$ and $\mathcal{S}^{\mathcal{M}}$ consist of the degradation state of the machine and the remaining maintenance time on the machine. Note that, $\mathcal{S}^{\mathcal{D}}$ and $\mathcal{S}^{\mathcal{M}}$ are mutually exclusive because the formal indicates that the machine is not under maintenance while the latter represents the cases where the machine is under maintenance. Therefore, the state index of one machine can be defined as

$$S := \begin{cases} d & \begin{array}{l}\text{if the machine is not under maintenance,} \\ \text{and its degradation state is } d \ (d = 1, \ldots, D)\end{array} \\ \\ D + 1 + T^{(D)} - t & \begin{array}{l}\text{if the machine is under maintenance,} \\ \text{and its remaining maintenance time is } t, \ (t = 1, \ldots, T^{(D)})\end{array} \end{cases} \tag{4.1}$$

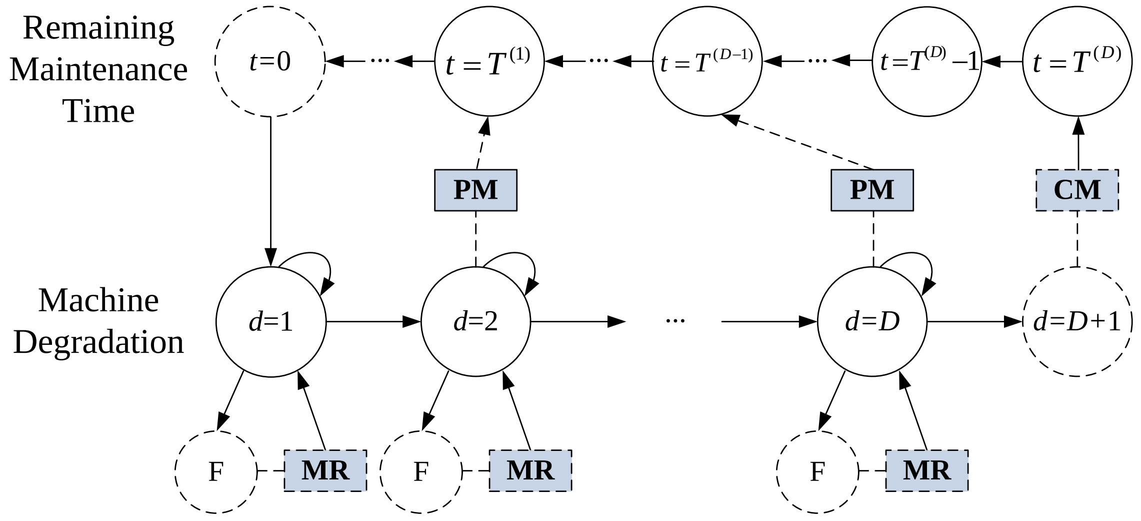


Figure 4.1: Machine degradation and maintenance model

Table 4.1 shows the state indices and their corresponding degradation level/remaining maintenance time. Note that, when the machine is in its failure state, CM will be carried out immediately, resulting a remaining maintenance time $T^{(D)}$. Hence, the degradation state $D + 1$ is the same as the state that has a remaining maintenance time $T^{(D)}$. Additionally, the system enters the degradation state 1 as soon as the maintenance ends, so state 1 also represents the case where the remaining maintenance time $t = 0$. From Table 4.1, S can take $D + T^{(D)}$ possible values.

Table 4.1: State index of the single machine

state	degradation level	remaining maintenance time
1	1	0
2	2	NA
$\vdots$	$\vdots$	$\vdots$
D	D	NA
$D+1$	$D+1$	$T^{(D)}$
$D+2$	NA	$T^{(D)}-1$
$\vdots$	$\vdots$	$\vdots$
$D+T^{(D)}$	NA	1

4.2.2 Modeling of system

We still consider the I-machine-$(I-1)$-buffer system that is depicted in Fig. 2.1. The subscript "i" is used to represent the corresponding parameters for machine M_i. Specifically, the degradation state for M_i ($i = 1, \dots, I$) is represented by $1, \dots, D_i$, and when the degradation state of machine M_i is d_i ($d_i = 1, \dots, D_i$), its probabilities for a random failure and machine degradation are $f_i^{(d_i)}$ and q_i, respectively; and its PM durations is $T_i^{(d_i-1)}$. Moreover, the time for CM is $T_i^{(D_i)}$, and that for the minimal repair is one unit time, regardless of the degradation states. The blockage and starvation rules, are the same as in Chapter III. This system can be modeled as a discrete time controlled Markov chain, and its dynamics will be further discussed in Sections 4.3 and 4.4.

4.3 Baseline Maintenance Policy: A Control Limit Policy

In this section, we evaluate the control limit policy for a single machine, as well as in multi-stage manufacturing systems.

4.3.1 Evaluation of the control limit policy for a single machine

For the single machine problem, the policy can be analyzed by a (delayed) renewal process, as shown in Fig. 4.2. It is well-known that a control limit policy exists for a single machine, i.e., the machine should be repaired when its degradation state reaches a threshold value.

Let $TTD^{(d)}$ be the expected time to degradation from state d to state $d+1$, then it can be calculated by the following equation:

$$TTD^{(d)} = 1 + \overline{f^{(d)}}\big(\bar{q} \cdot TTD^{(d)} + q \cdot 0\big) + f^{(d)} \cdot TTD^{(d)}$$
$$\Longrightarrow TTD^{(d)} = 1/(\overline{f^{(d)}} \cdot q)$$

Then, the total number of units produced when the machine is in degradation state d can be calculated as $\overline{f^{(d)}} \cdot TTD^{(d)} = 1/q$.

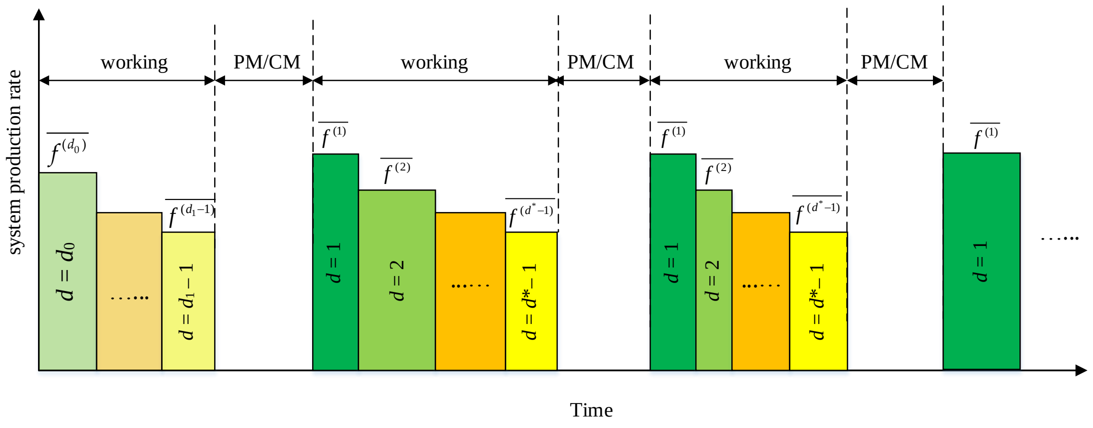


Figure 4.2: Production rate for a single degrading machine

Let d_0 be the initial state of the system. If $d_0 = 1$, the process is a renewal process, and the expected performance in each cycle is the same (a cycle is the time from the completion of one PM until the completion of the next PM). Let PR_d be the expected system production rate when the maintenance is performed at the degradation state d. Then PR_d can be calculated as

$$PR_d = \frac{\sum_{d'=1}^{d-1} 1/q}{\sum_{d'=1}^{d-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d-1)}} \tag{4.2}$$

Therefore, d^*, the critical degradation state where the maintenance should be performed, can be calculated as

$$d^* = \arg\max_d PR_d \tag{4.3}$$

However, if $d_0 \neq d^*$, the process is a delayed renewal process, where except for the first cycle, all other cycles start with degradation state 1 (Fig. 4.2). Therefore, the policy starting from the second cycle is to always perform maintenance at degradation state d^*. We only need to determine the degradation state at which the maintenance should be performed in the first cycle, which is denoted as d_1. Then, d_1 can be calculated as

$$d_1 = \arg\max_d \left[\sum_{d'=d_0}^{d-1} 1/q - PR_{d^*} \left(\sum_{d'=d_0}^{d-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d-1)} \right) \right] \tag{4.4}$$

where $\sum_{d'=d_0}^{d-1} 1/q$ and $\sum_{d'=d_0}^{d-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d-1)}$ are the expected number of the production units in the first cycle and the expected duration of the first cycle, respectively. Thus, the expression in the square brackets is the excessive number of parts produced in the first cycle.

Theorem 4.1. *$d_1 = \max(d_0, d^*)$.*

Proof. See Appendix C $\square$

Theorem 4.1 shows that, if the initial degradation state d_0 is not more severe than the threshold state d^*, then the policy is to perform maintenance when the machine degrades to state d^*. However, if the initial state $d_0 \geq d^*$, the optimal control action is to perform maintenance immediately.

4.3.2 Evaluation of the control limit policy in multi-stage systems

Next, we denote the control limit policy as a baseline policy and evaluate it in a serial line consisting of multiple machines. To be more specific, the control limit policy is to perform maintenance on one machine when its degradation state reaches its threshold value d_i^* that is calculated from Equation (4.3). The objective is to calculate PR_{cl}, the expected system production rate under the control limit policy.

4.3.2.1 Evaluation of the control limit policy in two-machine-one-buffer (2M1B) systems

First we evaluate the control limit policy in a two-machine-one-buffer (2M1B) system $M_1 - B - M_2$. The Markov chain model under the control limit policy is illustrated in Fig. 4.3. The state of the 2M1B system can be represented by a three-tuple (S_1', S_2', N), where N is the buffer level, S_i' is the modified state of machine M_i, which removes all the transient states under the control limit policy, and is defined as

$$S_i' := \begin{cases} S_i & \text{if } S_i = 1, \dots, d_i^* - 1 \\ d_i^* + T_i^{(d_i^*-1)} - t_i & \text{if the remaining maintenance time is } t_i \ (t_i = 1, \dots, T_i^{(d_i^*-1)}) \end{cases} \tag{4.5}$$

Then the total number of states of the system is $\left(d_1^* + T_1^{(d_1^*-1)}\right)\left(d_2^* + T_2^{(d_2^*-1)}\right)(C+1)$, where C is the buffer capacity. We define the state index as

$$I(S_1', S_2', N) = (S_1' - 1)(d_2^* + T_2^{(d_2^*-1)})(C+1) + (S_2' - 1)(C+1) + N + 1 \tag{4.6}$$

Table 4.2 illustrates the state index $I(S_1', S_2', N)$ and the corresponding machine and buffer states.

Definition 4.1. Denote $\mathbf{\Pi}(\mathbf{P})$ as the steady-state distribution associated with tran-

Table 4.2: State index of the 2M1B system

$I(S_1', S_2', N)$	S_1'	S_2'	N
1	1	1	0
2	1	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$C+1$	1	1	C
$C+2$	1	2	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2(C+1)$	1	2	C
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(d_2^* + T_2^{(d_2^*-1)})(C+1)$	1	$d_2^* + T_2^{(d_2^*-1)}$	C
$(d_2^* + T_2^{(d_2^*-1)})(C+1)+1$	2	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2(d_2^* + T_2^{(d_2^*-1)})(C+1)$	2	$d_2^* + T_2^{(d_2^*-1)}$	C
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(d_1^* + T_1^{(d_1^*-1)})(d_2^* + T_2^{(d_2^*-1)})(C+1)$	$d_1^* + T_1^{(d_1^*-1)}$	$d_2^* + T_2^{(d_2^*-1)}$	C

sition matrix $\mathbf{P}$, such that $||\mathbf{\Pi}(\mathbf{P})||_1 = 1$ and $\mathbf{\Pi}(\mathbf{P}) = \mathbf{P} \cdot \mathbf{\Pi}(\mathbf{P})$.

Remark 4.1. Mathematically, if P_{mn} is the element at the $(n+1)^{th}$ column and the $(m+1)^{th}$ row of matrix $\mathbf{P}$, then

$$\mathbf{\Pi}(\mathbf{P}) = \begin{bmatrix} P_{00}-1 & P_{01} & \cdots & P_{0,C-1} & P_{0C} \\ P_{10} & P_{11}-1 & \cdots & P_{1,C-1} & P_{1C} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ P_{C-1,0} & P_{C-1,1} & \cdots & P_{C-1,C-1}-1 & P_{C-1,C} \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{4.7}$$

Numerically, given any $\boldsymbol{\pi}_0$, $\mathbf{\Pi}(\mathbf{P})$ can be calculated as $\mathbf{\Pi}(\mathbf{P}) = \mathbf{P}^K \cdot \boldsymbol{\pi}_0$ recursively by $K = 1, 2, ...$ until $|\mathbf{P}^K \cdot \boldsymbol{\pi}_0 - \mathbf{P}^{K-1} \cdot \boldsymbol{\pi}_0| < \epsilon$, where ϵ is a predetermined small number.

Let $\pi^S_{(S_1', S_2', N)}$ be the stationary distribution of state (S_1', S_2', N), and $\boldsymbol{\pi}^S$ be the column vector whose $I(S_1', S_2', N)^{th}$ element is $\pi^S_{(S_1', S_2', N)}$. Then based on Definition 4.1,

$\boldsymbol{\pi}^S = \mathbf{\Pi}(\mathbf{P})$, where

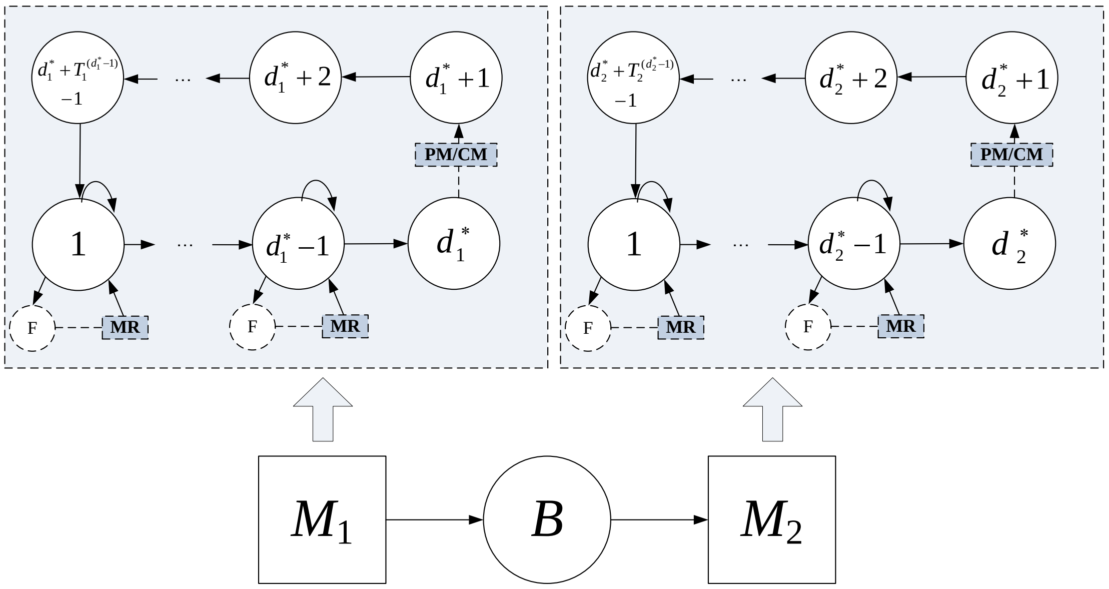


Figure 4.3: Baseline policy in a 2M1B system

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}_1 & & & & \mathbf{B}_{d_1^*+T_1^{(d_1^*-1)}} \\ \mathbf{B}_1 & \ddots & & & \\ & \ddots & \mathbf{A}_{d_1^*-1} & & \\ & & \mathbf{B}_{d_1^*-1} & & \\ & & & \ddots & \\ & & & \mathbf{B}_{d_1^*+T_1^{(d_1^*-1)}-1} & \end{bmatrix} \tag{4.8a}$$

$$=: \mathbf{P}(\overline{f_1^{(1:d_1^*-1)}}, \overline{f_2^{(1:d_2^*-1)}}, q_1, q_2, T_1^{(d_1^*-1)}, T_2^{(d_2^*-1)}) \tag{4.8b}$$

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{C}_{i1} & & & & & \mathbf{D}_{i,d_2^*+T_2^{(d_2^*-1)}} \\ \mathbf{D}_{i1} & \ddots & & & & \\ & \ddots & \mathbf{C}_{i,d_2^*-1} & & & \\ & & \mathbf{D}_{i,d_2^*-1} & & & \\ & & & \ddots & & \\ & & & & \mathbf{D}_{i,d_2^*+T_2^{(d_2^*-1)}-1} & \end{bmatrix} \tag{4.8c}$$

$$\mathbf{B}_i = \begin{bmatrix} \mathbf{E}_{i1} & & & & & \mathbf{F}_{i,d_2^*+T_2^{(d_2^*-1)}} \\ \mathbf{F}_{i1} & \ddots & & & & \\ & \ddots & \mathbf{E}_{i,d_2^*-1} & & & \\ & & \mathbf{F}_{i,d_2^*-1} & & & \\ & & & \ddots & & \\ & & & & \mathbf{F}_{i,d_2^*+T_2^{(d_2^*-1)}-1} & \end{bmatrix} \tag{4.8d}$$

$$\mathbf{C}_{ij} = \begin{bmatrix} f_1^{(i)} & f_1^{(i)}\overline{f_2^{(j)}} \cdot \overline{q_2} & & & \\ \overline{f_1^{(i)}} \cdot \overline{q_1} & c_0 & \ddots & & \\ & \overline{f_1^{(i)}} f_2^{(j)} \cdot \overline{q_1} & \ddots & f_1^{(i)}\overline{f_2^{(j)}} \cdot \overline{q_2} & \\ & & \ddots & c_0 & f_1^{(i)}\overline{f_2^{(j)}} \cdot \overline{q_2} \\ & & & \overline{f_1^{(i)}} f_2^{(j)} \overline{q_1} & f_2^{(j)} + \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}} \cdot \overline{q_1} \cdot \overline{q_2} \end{bmatrix} \tag{4.8e}$$

$$\mathbf{D}_{ij} = \begin{cases} \begin{bmatrix} 0 & f_1^{(i)}\overline{f_2^{(j)}}q_2 & & \\ & \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}} \cdot \overline{q_1}q_2 & \ddots & \\ & & \ddots & f_1^{(i)}\overline{f_2^{(j)}}q_2 \\ & & & \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}} \cdot \overline{q_1}q_2 \end{bmatrix} & \text{if } j < d_2^* \\ \begin{bmatrix} f_1^{(i)} & & & \\ \overline{f_1^{(i)}} \cdot \overline{q_1} & \ddots & & \\ & \ddots & f_1^{(i)} & \\ & & \overline{f_1^{(i)}} \cdot \overline{q_1} & 1 \end{bmatrix} & \text{if } j \geq d_2^* \end{cases} \tag{4.8f}$$

$$\mathbf{E}_{ij} = \begin{cases} \begin{bmatrix} 0 & & & & \\ \overline{f_1^{(i)}}q_1 & \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}}q_1\overline{q_2} & & & \\ & \overline{f_1^{(i)}}f_2^{(j)}q_1 & \ddots & & \\ & & \ddots & \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}}q_1\overline{q_2} & 0 \\ & & & \overline{f_1^{(i)}}f_2^{(j)}q_1 & \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}}q_1\overline{q_2} \end{bmatrix} & \text{if } i < d_1^* \\ \begin{bmatrix} 1 & \overline{f_2^{(j)}} \cdot \overline{q_2} & & \\ 0 & f_2^{(j)} & \ddots & \\ & & \ddots & \overline{f_2^{(j)}} \cdot \overline{q_2} \\ & & & f_2^{(j)} \end{bmatrix} & \text{if } i \geq d_1^* \end{cases} \tag{4.8g}$$

$$\mathbf{F}_{ij} = \begin{cases} \begin{bmatrix} 0 & & & \\ \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}} q_1 q_2 & & & \\ & \ddots & & \\ & & & \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}} q_1 q_2 \end{bmatrix} & \text{if } i < d_1^*,\ j < d_2^* \\ \begin{bmatrix} 0 & & & \\ \overline{f_1^{(i)}} q_1 & & & \\ & \ddots & & \\ & & \overline{f_1^{(i)}} q_1 & 0 \end{bmatrix} & \text{if } i < d_1^*,\ j \geq d_2^* \\ \begin{bmatrix} 0 & \overline{f_2^{(j)}} q_2 & & \\ & & \ddots & \\ & & & \overline{f_2^{(j)}} q_2 \\ & & & 0 \end{bmatrix} & \text{if } i \geq d_1^*,\ j < d_2^* \\ \mathbf{I} & \text{if } i \geq d_1^*,\ j \geq d_2^* \end{cases} \tag{4.8h}$$

and $c_0 = f_1^{(i)} f_2^{(j)} + \overline{f_1^{(i)}} \cdot \overline{f_2^{(j)}} \cdot \overline{q_1} \cdot \overline{q_2}$.

All of the matrices in Equation (4.8) are square matrices, and their dimensions are summarized in Table 4.3.

Table 4.3: Matrix and dimension

matrix	dimension
$\mathbf{P}$	$(d_1^* + T_1^{(d_1^*-1)})(d_2^* + T_2^{(d_2^*-1)})(C+1)$
$\mathbf{A}, \mathbf{B}$	$(d_2^* + T_2^{(d_2^*-1)})(C+1)$
$\mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}$	$C+1$

In these matrices, i and j represent the states of machines M_1 and M_2, respectively. $\mathbf{A}_i$ $(i < d_1^*)$ is the matrix of transition probabilities that M_1 is working but not degrading while $\mathbf{B}_i$ is the matrix of transition probabilities that M_1 is degrading or under maintenance. Moreover, $\mathbf{A}_i$ consists of $\mathbf{C}_{ij}$ $(j < d_2^*)$ and $\mathbf{D}_{ij}$, and $\mathbf{B}_i$ consists of $\mathbf{E}_{ij}$ $(j < d_2^*)$ and $\mathbf{F}_{ij}$, where $\mathbf{C}_{ij}$ and $\mathbf{E}_{ij}$ consist of transition probabilities that M_2 is working but not degrading, while $\mathbf{D}_{ij}$ and $\mathbf{F}_{ij}$ consist of transition probabilities that M_2 is degrading or under maintenance.

Then the steady-state system production rate under the control limit policy can be calculated as

$$\begin{aligned} PR_{cl} &= \Pr(N > 0,\, M_2 \text{ is working}) \\ &= \sum_{S_2'=1}^{d_2^*-1} \sum_{N=1}^{C} \Pr(N, S_2') \cdot \Pr(M_2 \text{ is working}|S_2') \\ &= \sum_{S_2'=1}^{d_2^*-1} \sum_{S_1'=1}^{d_1^*+T_1^{(d_1^*-1)}} \sum_{N=1}^{C} \Pr(S_1', S_2', N) \cdot \Pr(M_2 \text{ is working}|S_2') \\ &= \sum_{S_2'=1}^{d_2^*-1} \sum_{S_1'=1}^{d_1^*+T_1^{(d_1^*-1)}} \sum_{N=1}^{C} \pi^S_{(S_1',S_2',N)} \overline{f_2^{(S_2')}} \end{aligned} \tag{4.9}$$

Moreover, when the degradation state of machine M_2 is S_2' $(S_2' = 1, \ldots, d_2^* - 1)$, the probability that M_2 is not starved is

$$ns_2^{(S_2')} = \frac{\Pr(N > 0,\, S_2')}{\Pr(S_2')} = \frac{\sum_{S_1'=1}^{d_1^*+T_1^{(d_1^*-1)}} \sum_{N=1}^{C} \pi^S_{(S_1',S_2',N)}}{\sum_{S_1'=1}^{d_1^*+T_1^{(d_1^*-1)}} \sum_{N=0}^{C} \pi^S_{(S_1',S_2',N)}} \tag{4.10}$$

Similarly, one can calculate the consumption rate of the system, which takes the form

$$
\begin{aligned}
CR_{cl} =& \Pr(M_1 \text{ is working and not blocked}) \\
=& \left(\sum_{N=0}^{C-1} \Pr(N) + \Pr(N = C,\ M_2 \text{ is working}) \right) \cdot \Pr(M_1 \text{ is working}) \\
=& \left(\sum_{S_2'=1}^{d_2^* + T_2^{(d_2^*-1)}} \sum_{N=0}^{C-1} \Pr(N, S_2') + \sum_{S_2'=1}^{d_2^*-1} \Pr(N = C, S_2') \cdot \Pr(M_2 \text{ is working}|S_2') \right) \\
& \cdot \Pr(M_1 \text{ is working}) \\
=& \sum_{S_1'=1}^{d_1^*-1} \left(\sum_{S_2'=1}^{d_2^* + T_2^{(d_2^*-1)}} \sum_{N=0}^{C-1} \pi^S_{(S_1',S_2',N)} + \sum_{S_2'=1}^{d_2^*-1} \pi^S_{(S_1',S_2',C)} \overline{f_2^{(S_2')}} \right) \Pr(M_1 \text{ is working}|S_1') \\
=& \sum_{S_1'=1}^{d_1^*-1} \left(\sum_{S_2'=1}^{d_2^* + T_2^{(d_2^*-1)}} \sum_{N=0}^{C-1} \pi^S_{(S_1',S_2',N)} + \sum_{S_2'=1}^{d_2^*-1} \pi^S_{(S_1',S_2',C)} \overline{f_2^{(S_2')}} \right) \overline{f_1^{(S_1')}}
\end{aligned}
\tag{4.11}
$$

and the probability that machine M_1 is not blocked when its degradation state is S_1' ($S_1' = 1, \ldots, d_1^* - 1$) is

$$
nb_1^{(S_1')} = \frac{\Pr(S_1',\ M_1 \text{ is not blocked})}{\Pr(S_1')} = \frac{\sum_{S_2'=1}^{d_2^* + T_2^{(d_2^*-1)}} \sum_{N=0}^{C-1} \pi^S_{(S_1',S_2',N)} + \sum_{S_2'=1}^{d_2^*-1} \pi^S_{(S_1',S_2',C)} \overline{f_2^{(S_2')}}}{\sum_{S_2'=1}^{d_2^* + T_2^{(d_2^*-1)}} \sum_{N=0}^{C} \pi^S_{(S_1',S_2',N)}}
\tag{4.12}
$$

The analysis for a 2M1B system can be utilized as a building block to evaluate the control limit policy in multi-stage systems, which will be studied in the next section.

4.3.2.2 Evaluation of the control limit policy in multi-stage systems

Similar as in Chapter III, we use the decomposition method to approximately study the performance of a multi-stage system under the control limit policy. We decompose every machine (except M_1 and M_I) into its non-starvation and non-blockage "dummy" parts, so that in the decomposed 2M1B systems, all of the upstream machines are non-starved and all of the downstream machines are non-blocked. Then these systems can be analyzed by the method in Section 4.3.2.1.

Here, the key is to find the probability that each machine is non-blocked or non-starved when it is in a specific degradation state. Let $nfs_i^{(d_i)}$(resp., $nbs_i^{(d_i)}$) be the probability that, given machine M_i is in state d_i ($d_i = 1, \ldots, d_i^* - 1$), it is not starved (resp., not blocked) and no random failure occurs on it.

Then, $nfs_i^{(d_i)}$ can be calculated as

$$\begin{aligned}
nfs_i^{(d_i)} =& \Pr\{M_i \text{ is up and not starved} \mid S_i = d_i\} \\
=& \frac{\Pr\{M_i \text{ is up and not starved, } S_i = d_i\}}{\Pr\{S_i = d_i\}} \\
=& \frac{\Pr\{M_i \text{ is up} \mid M_i \text{ is not starved, } S_i = d_i\} \cdot \Pr\{M_i \text{ is not starved, } S_i = d_i\}}{\Pr\{S_i = d_i\}} \\
=& \Pr\{M_i \text{ is up} \mid S_i = d_i\} \cdot \frac{\Pr\{M_i \text{ is not starved, } S_i = d_i\}}{\Pr\{S_i = d_i\}} \\
=& \Pr\{M_i \text{ is up} \mid S_i = d_i\} \cdot \Pr\{M_i \text{ is not starved} \mid S_i = d_i\} \\
=& \overline{f_i^{(d_i)}} ns_i^{(d_i)}
\end{aligned} \tag{4.13}$$

where $ns_i^{(d_i)}$ is the non-starvation probability when M_i is in state d_i, and it can be calculated by using Equation (4.10) in the 2M1B system $M_{i-1}^{NS} - B_{i-1} - M_i^{NB}$.

Simiarly, $nfb_i^{(d_i)}$ can be calculated as

$$\begin{aligned}
nfb_i^{(d_i)} =& \Pr\{M_i \text{ is up and not blocked} \mid S_i = d_i\} \\
=& \Pr\{M_i \text{ is up} \mid S_i = d_i\} \cdot \Pr\{M_i \text{ is not blocked} \mid S_i = d_i\} \\
=& \overline{f_i^{(d_i)}} nb_i^{(d_i)}
\end{aligned} \tag{4.14}$$

where $nb_i^{(d_i)}$ is the non-blockage probability when M_i is in state d_i, and it can be calculated by applying Equation (4.12) in the 2M1B system $M_i^{NS} - B_i - M_{i+1}^{NB}$.

The following recursive procedure is developed to calculate the system production rate in the steady state.

Recursive Procedure 4.1. We solve the following equations recursively for $a = 0, 1, 2, \dots$:

Step 1. Set the initial and boundary conditions:

initial conditions: $nb_i^{(d_i)}(0) = 1$, and $ns_i^{(d_i)}(0) = 1$, $\forall i, d_i$;

boundary conditions: $nb_I^{(d_I)}(a) = 1$, and $ns_1^{(d_1)}(a) = 1$, $\forall a, d_1, d_I$.

Step 2a. For $i = I-1, \dots, 1$, do Steps $2a.1 - 2a.3$ recursively :

2a.1. Set $nfb_{i+1}^{(d_{i+1})}(a+1) = \overline{f_{i+1}^{(d_{i+1})}} nb_{i+1}^{(d_{i+1})}(a+1)$ $\forall d_{i+1}$;

2a.2. Calculate $\boldsymbol{\pi}_i^S$, the distribution of buffer B_i, as

$$\boldsymbol{\pi}_i^S(a+1/2) = \mathbf{\Pi}\Big(\mathbf{P}\big(nfs_i^{(1:d_i^*-1)}(a), nfb_{i+1}^{(1:d_{i+1}^*-1)}(a+1), q_i, q_{i+1}, T_i^{(d_i^*-1)}, T_{i+1}^{(d_{i+1}^*-1)}\big)\Big).$$

2a.3. Substituting $\boldsymbol{\pi}_i^S(a+1/2)$ into Equation (4.12), calculate $nb_i^{(d_i)}(a+1)$.

Step 2b. For $i = 1, \dots, I-1$, do Steps $2b.1 - 2b.3$ recursively:

2b.1. Set $nfs_i^{(d_i)}(a+1) = \overline{f_i^{(d_i)}} ns_i^{(d_i)}(a+1)$ $\forall d_i$;

2b.2. Calculate $\boldsymbol{\pi}_i^S$, the distribution of buffer B_i, as

$$\boldsymbol{\pi}_i^S(a+1) = \mathbf{\Pi}\Big(\mathbf{P}\big(nfs_i^{(1:d_i^*-1)}(a+1), nfb_{i+1}^{(1:d_{i+1}^*-1)}(a+1), q_i, q_{i+1}, T_i^{(d_i^*-1)}, T_{i+1}^{(d_{i+1}^*-1)}\big)\Big).$$

2b.3. Substituting $\boldsymbol{\pi}_i^S(a+1)$ into Equation (4.10), calculate $ns_{i+1}^{(d_{i+1})}(a+1)$.

Step 3. Compare $\boldsymbol{\pi}_i^S(a)$ and $\boldsymbol{\pi}_i^S(a+1)$. Given a predetermined $\epsilon > 0$, if $|\boldsymbol{\pi}_i^S(a) - \boldsymbol{\pi}_i^S(a+1)| \leq \epsilon$ for all i, go to Step 4. Otherwise, set $a = a+1$ and go back to Step 2.

Step 4. Based on $\boldsymbol{\pi}_{I-1}^S(a+1)$, use Equation (4.9) to calculate PR_{cl}, the steady-state production rate of the system under the control limit policy.

The approximation method based on system decomposition is used to calculate the system production rate under the control limit policy in seven different lines, whose parameters are shown in Table 4.4. All machines in Lines 1 to 4 are identical, while the machines in Lines 5 to 7 have different degradation profiles. (In addition to the changes in the buffer capacities, Line 6 is constructed by adding M_3 to the two existing machines in Line 5, and Line 7 is constructed by adding M_4 and M_5 to Line 6.) For each machine M_i, the critical degradation state d_i^* can be calculated based on Equation (4.3): in Lines 1 to 4, $d_i^* = 3\ \forall i$; and in Lines 5 to 7, $d_1^* = 3$, $d_2^* = 4$, $d_3^* = 3$, $d_4^* = 3$, $d_5^* = 2$.

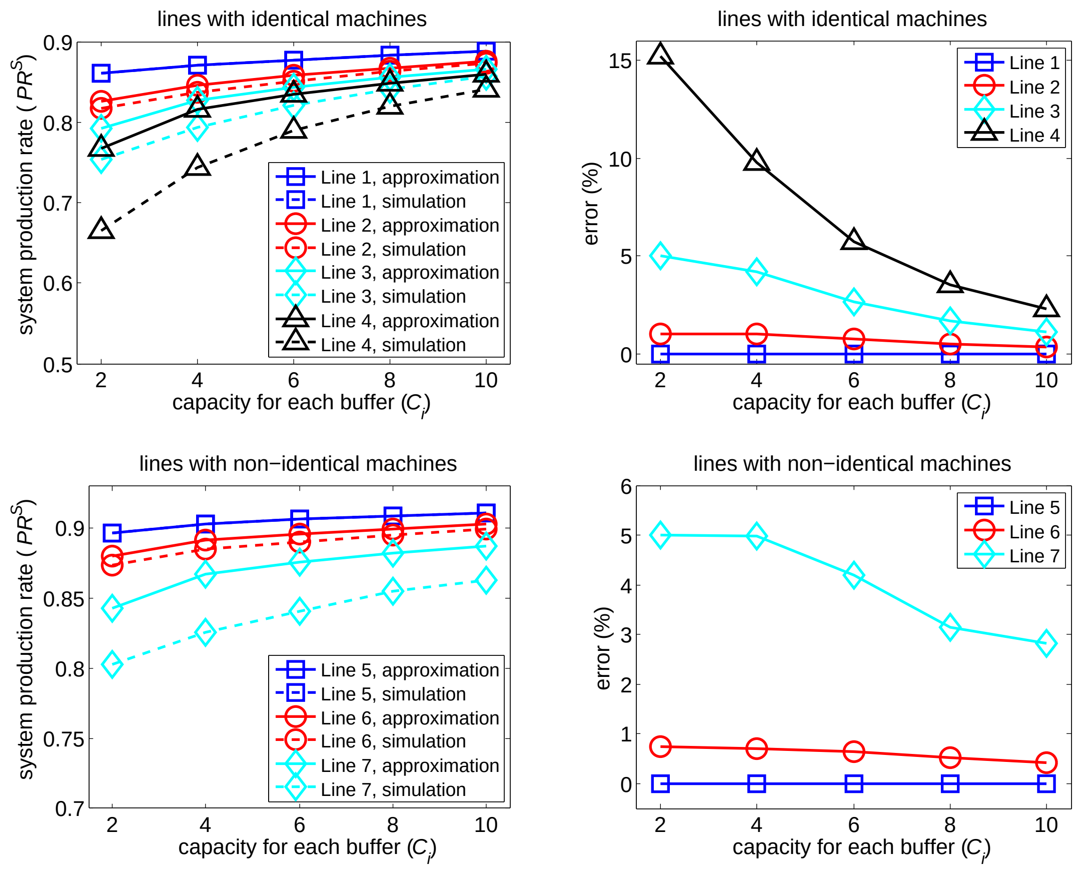
To validate the effectiveness of the approximation method, we use Monte Carlo method to simulate the system production rates at every time unit of these seven lines. The simulated average system production rate is calculated by $PR_{sim} = \sum_{r=1}^{R}\sum_{k=1}^{K} PR_r(k)/(KR)$, where R is the number of repetition, K is the total time in each repetition, and $PR_r(k)$ is the system production rate at time k in repetition r. In the numerical results, we set $K = 100000$ and $R = 100$.

Moreover, we set the capacity of each buffer C_i =2, 4, 6, 8, and 10. Under each of the buffer capacities, the steady-state system production rates by using both the approximation method and Monte Carlo simulation method are plotted in Figure 4.4. It can be seen that, no matter the machines are identical (Lines 1 to 4) or not (Lines 5 to 7), the accuracy of the approximation method decreases as the number of machines increases, while it increases as the buffer capacity increases.

Table 4.4: Line parameters

Line	type size	failure probability	degradation probability	maintenance time
1	2M1B	$f_i^{(1:D_i)}$ =[.02 .05 .1 .15]	$q_i = .01$	$T_i^{(1:D_i)}$ =[8 10 15 20]
2	3M2B			
3	5M4B			
4	10M9B			
5	2M1B	$f_1^{(1:D_1)}$ =[.02 .05 .1 .15] $f_2^{(1:D_2)}$ =[.01 .03 .04 .1]	$q_1 = .002$ $q_2 = .005$	$T_1^{(1:D_1)}$ =[25 30 35 40] $T_2^{(1:D_2)}$ =[15 17 20 25]
6	3M2B	$f_3^{(1:D_3)}$ =[.01 .025 .05 .1]	$q_3 = .0025$	$T_3^{(1:D_3)}$ =[15 20 25 30]
7	5M4B	$f_4^{(1:D_4)}$ =[.02 .04 .06 .1] $f_5^{(1:D_5)}$ =[.01 .04 .1 .2]	$q_4 = .003$ $q_5 = .004$	$T_4^{(1:D_4)}$ =[20 25 30 40] $T_5^{(1:D_5)}$ =[15 17 20 25]

Figure 4.4: Comparison of system production rates



The numerical results show that the approximation method can be used to effectively evaluate the control limit policy in I-machine-$(I-1)$-buffer serial lines, especially when the number of machines I is small or the buffer capacity is large. Moreover, based on the system decomposition method in Section 3.4, the approximation method can also be extended to evaluate the control limit policy in systems with more complex structures. However, this policy only considers the degradation states of independent machines, and thereby is not optimal from a system point of view. In the next section, we will study the optimal policy by using a Markov Decision Process (MDP) approach.

4.4 Optimal Maintenance Policy: A Markov Decision Process Approach

We implement the machine degradation model in Section 4.2.1 to investigate the optimal maintenance policies. In this section, we only use a 2M1B system as an example. Similar to the analysis in Section 4.3.2.1, the system state can be represented by (S_1, S_2, N), where $1 \le S_i \le D_i + T_i^{(D_i)}$ $(i = 1, 2)$ and $0 \le N \le C$. Therefore, the total number of the states is $(C+1)\prod_{i=1}^{2}\left(D_i + T_i^{(D_i)}\right)$.

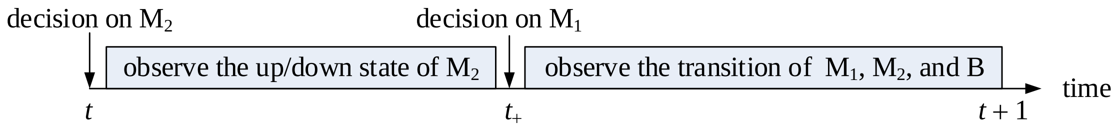

decision on M$_2$
decision on M$_1$
observe the up/down state of M$_2$
observe the transition of M$_1$, M$_2$, and B
time
t
t_+
$t+1$

Figure 4.5: The decision-making and state transition processes at time t

There are three actions on each machine: preventive maintenance, stop (for one cycle time), and default (i.e., keep its working or maintenance state), where the first two actions are only feasible when the machine is not under maintenance. Based on the assumption of the starvation/blockage rule, the blockage of machine M_1 depends on the state of machine M_2 (when the buffer is full), while the starvation of M_2 does not depend on the state of M_1. Therefore, the decision is made on M_2 first, and then on M_1. Figure 4.5 shows the process during one time interval. At time t, one decision is made on machine M_2, and if the action is "keep M_2 working" (which is only feasible when $S_2 \leq D_2$), then a Bernoulli trial is followed to determine whether a random failure occurs on M_2 or not. Based on the observation of the up/down state of M_2, a decision is made on M_1 at time t_+, after which the actual transition of the states of M_1, M_2, and the buffer B start. Whether the machines degrade or not will be known at the beginning of time $t+1$.

Note that, when the decision is made on M_1, the up/down state of M_2 is already known. There are two cases, namely, case a and case b, corresponding to the situation where M_2 is working or not working, respectively. Also, if M_2 is not working, its state at time $t+1$ is the same as that at time t (i.e., $S_2(t+1) = S_2(t)$), and thus $S_2(t+1)$ can be determined before time t_+. Therefore, in case b, the action on M_1 can be made based upon $S_2(t+1)$. The determination of the state transitions in both cases is shown in Fig. 4.6.

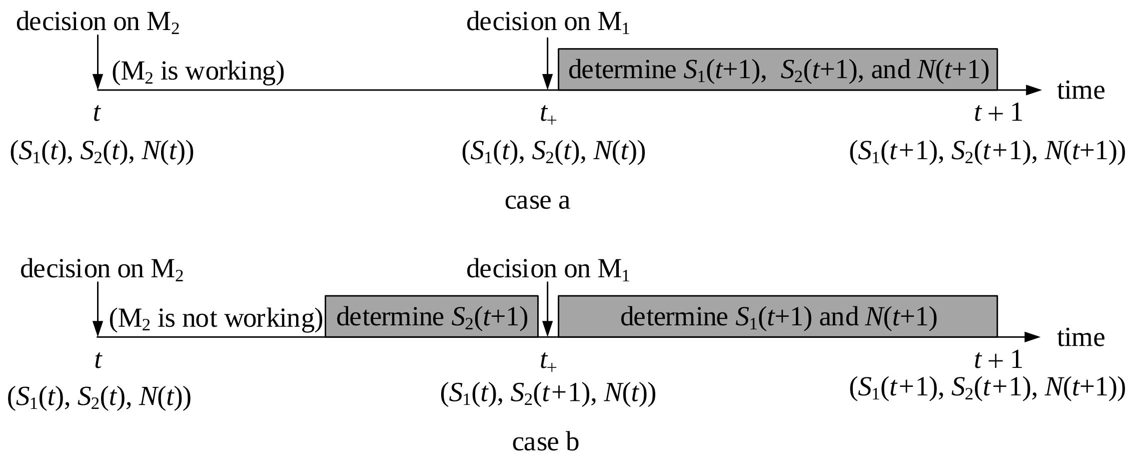


Figure 4.6: Determination of the state transitions at time t

Let a_2 be the action on machine M_2, and a_1^a and a_1^b be the actions on M_1 in case a, and case b, respetively. The actions "preventive maintenance", "stop", and "default" are denoted as “M”, “S” and “D”, respectively. Note that, action a_1^a is always set as the same as a_1^b if case a does not exist (i.e., when M_2 is not working). Moreover, if M_1 (resp., M_2) is blocked (resp., starved), then actions “S” and “D” are the same, which will be denoted as “D”.

Denote the reward as “1” if there is a production unit. Let $V_t(S_1, S_2, N)$ be the maximal total reward when the remaining time is t. Then,

$$V_t(S_1,S_2,N) = \begin{cases} \max\begin{cases} V_t^b(S_1,S_2^{++},N) \\ V_t^b(S_1,S_2,N) \\ \overline{f_2^{(S_2)}}V_t^a(S_1,S_2,N) + f_2^{(S_2)}V_t^b(S_1,S_2,N) \end{cases} & S_2 \le D_2, N \ge 1 \\ \max\begin{cases} V_t^b(S_1,S_2^{++},N) \\ V_t^b(S_1,S_2,N) \end{cases} & S_2 \le D_2, N = 0 \\ V_t^b(S_1,S_2^{+},N) & S_2 > D_2 \end{cases} \tag{4.15}$$

where

$$V_t^a(S_1,S_2,N) = 1 + \begin{cases} \max\begin{cases} \overline{q_2}V_{t-1}(S_1^{++},S_2,N^-) + q_2V_{t-1}(S_1^{++},S_2^{+},N^-) \\ \overline{q_2}V_{t-1}(S_1,S_2,N^-) + q_2V_{t-1}(S_1,S_2^{+},N^-) \\ \overline{q_2}\phi_{t-1}(S_1,S_2,N^-) + q_2\phi_{t-1}(S_1,S_2^{+},N^-) \end{cases} & S_1 \le D_1 \\ \overline{q_2}V_{t-1}(S_1^{+},S_2,N^-) + q_2V_{t-1}(S_1^{+},S_2^{+},N^-) & S_1 > D_1 \end{cases} \tag{4.16}$$

$$V_t^b(S_1,S_2,N) = \begin{cases} \max\begin{cases} V_{t-1}(S_1^{++},S_2,N) \\ V_{t-1}(S_1,S_2,N) \\ \phi_{t-1}(S_1,S_2,N) \end{cases} & S_1 \le D_1 \\ V_{t-1}(S_1^{+},S_2,N) & S_1 > D_1 \end{cases} \tag{4.17}$$

$$\phi_t(S_1, S_2, N) = \begin{cases} \overline{f_1^{(S_1)}} \left[\overline{q_1} V_t(S_1, S_2, N^+) + q_1 V_t(S_1^+, S_2, N^+)\right] + f_1^{(S_1)} V_t(S_1, S_2, N) & 0 \le N < C \\ V_t(S_1, S_2, C) & N = C \end{cases} \tag{4.18}$$

and

$$\begin{aligned} s^+ &= \begin{cases} s+1 & \text{if } s < D_i + T_i^{(D_i)} \\ 1 & \text{if } s = D_i + T_i^{(D_i)} \end{cases}, \\ s^{++} &= D_i + T_i^{(D_i)} - T_i^{(s-1)} + 2, \\ N^+ &= N+1, N^- = N-1. \end{aligned} \tag{4.19}$$

Equation (4.15) shows the state transition of machine M_2 before the decision is made on M_1 (i.e., the state transition during time $[t, t_+)$ in Fig. 4.6). The superscripts "a" and "b" represent cases a and b, respectively. Note that, case a only occurs when M_2 is not under maintenance ($S_2 \le D_2$), M_2 is not starved ($N \ge 1$), the action is default (a_2=D), and no random failure occurs (with a probability $\overline{f_2^{(S_2)}}$). While case b always exists no matter what is the state of M_2 and what action is taken.

Equation (4.16) (resp., Equation (4.17)) represents the transition of the system states after the decision is made on M_1 (i.e., the state transition during time $[t_+, t+1)$ in Fig. 4.6), in case a (resp., case b).

In Equation (4.19), s^+ means the transition from s when the machine degrades, or when the remaining maintenance time is decreased by 1. s^{++} represents the transition when maintenance is performed (i.e., the transition from degradation state s to remaining maintenance time $T_i^{(s-1)} - 1$). N^+ (resp., N^-) represents the case where the buffer level increases (resp., decreases) by 1.

Theorem 4.2. *$V_t(S_1, S_2, N)$ has the following properties:*

(1) $0 \le V_t(S_1, S_2, N^+) - V_t(S_1, S_2, N) \le 1$;

(2) $V_t(S_1^+, S_2, N) \le V_t(S_1, S_2, N)$ for $S_1 \le D_1$;

(3) $V_t(S_1^+, S_2, N) \geq V_t(S_1, S_2, N)$ *for* $S_1 > D_1$*;*

(4) $V_t(S_1, S_2^+, N) \leq V_t(S_1, S_2, N)$ *for* $S_2 \leq D_2$*;*

(5) $V_t(S_1, S_2^+, N) \geq V_t(S_1, S_2, N)$ *for* $S_2 > D_2$*.*

Proof. See Appendix C. □

Theorem 4.2 shows that, the model developed with an MDP approach guarantees the following properties of the maximal rewards $V_t(S_1, S_2, N)$. First, when machines M_1 and M_2 have the same state, the system with a higher buffer level results in a larger reward (Property (1)). Second, when M_1 (or M_2) is not under maintenance, the machine with a less degraded state results in a larger reward (Properties (2) and (4)). Moreover, when M_1 (or M_2) is under maintenance, the machine with a shorter remaining maintenance time results in a larger reward (Properties (3) and (5)).

Note that, by combining Equations (4.15) to (4.18), one can obtain the Bellman Equation, as

$$\begin{aligned}
&V_t(S_1, S_2, N) \\
&= \max_{(a_1^a, a_1^b, a_2)\in\mathcal{A}} \Bigg\{ \sum_{(S_1', S_2', N')} \Pr\{(S_1', S_2', N') | \big((S_1, S_2, N), (a_1^a, a_1^b, a_2)\big)\} \cdot \\
&\qquad \Big[PR\big((S_1, S_2, N), (a_1^a, a_1^b, a_2), (S_1', S_2', N')\big) + V_{t-1}(S_1', S_2', N') \Big] \Bigg\} \\
&= \Psi_t((S_1, S_2, N), \mathcal{A})
\end{aligned} \tag{4.20}$$

where $\mathcal{A}$ is the action space, $\Pr\{(S_1', S_2', N') | \big((S_1, S_2, N), (a_1^a, a_1^b, a_2)\big)\}$ is the transition probability from state (S_1, S_2, N) to state (S_1', S_2', N') when action (a_1^a, a_1^b, a_2) is taken, and $PR\big((S_1, S_2, N), (a_1^a, a_1^b, a_2), (S_1', S_2', N')\big)$ is the immediate reward (either 1 or 0) associated with the transition.

The next lemma shows that, removing the action "stopping both machines" from the action space will not affect the value of $V_t(S_1, S_2, N)$.

Table 4.5: Possible actions associated with different (S_1, S_2, N)

(S_1, S_2, N)	(a_1^a, a_1^b, a_2)
$S_1 \leq D_1,\ S_2 \leq D_2,\ N = 0$	(M,M,M),(S,S,M),(D,D,M),(M,M,D),(D,D,D)
$S_1 \leq D_1,\ S_2 \leq D_2,\ 0 < N < C$	(M,M,M),(S,S,M),(D,D,M),(M,M,S),(D,D,S), (M,M,D),(M,S,D),(M,D,D),(S,M,D),(S,S,D), (S,D,D),(D,M,D),(D,S,D),(D,D,D)
$S_1 \leq D_1,\ S_2 \leq D_2,\ N = C$	(M,M,M),(D,D,M),(M,M,S),(M,M,D),(M,D,D), (S,M,D),(S,D,D),(D,M,D),(D,D,D)
$S_1 \leq D_1,\ S_2 > D_2, N < C$	(M,M,D),(S,S,D),(D,D,D)
$S_1 \leq D_1,\ S_2 > D_2, N = C$	(M,M,D),(D,D,D)
$S_1 > D_1,\ S_2 \leq D_2,\ N = 0$	(D,D,M),(D,D,D)
$S_1 > D_1,\ S_2 \leq D_2,\ N > 0$	(D,D,M),(D,D,S),(D,D,D)
$S_1 > D_1,\ S_2 > D_2$	(D,D,D)

Lemma 4.1. *Action (a_1^a, a_1^b, a_2) =(S, S, S) can be removed from the action space without loss of optimality.*

Proof. See Appendix C. □

Then all the possible actions at every state (S_1, S_2, N) are summarized in Table 4.5, which shows that, for any (S_1, S_2, N), the maximum number of actions is 14 (i.e., $|\mathcal{A}| = 14$, when $S_1 \leq D_1,\ S_2 \leq D_2,\ 0 < N < C$). Moreover, if the maintenance times on all degradation states are the same, then the action space can be further reduced, as stated in the following corollary.

Corollary 4.1. *If $T_i^{(d_i+1)} = T_i^{(d_i)}\ \forall i, d_i$, then the actions space can be reduced to $\mathcal{A}' = \big\{(M,M,M),(M,M,D),(D,M,M),(D,M,D),(D,D,M),(D,D,D)\big\}$ without loss of optimality.*

Proof. See Appendix C. □

Let $V(S_1, S_2, N)$ be the maximal average reward when the initial state is (S_1, S_2, N). Mathematically,

$$V(S_1, S_2, N) = \limsup_{T\to\infty} \left[\sum_{t=0}^{T-1} V_t(S_1, S_2, N)/T \right] \tag{4.21}$$

$V(S_1, S_2, N)$ is independent of the initial state (S_1, S_2, N), as stated in the next lemma.

Lemma 4.2. *$V(S_1, S_2, N)$ is constant $\forall (S_1, S_2, N)$.*

Proof. We consider a policy that only CM is performed on both machines (although such a policy may not be optimal). Under such a policy, every state (S_1, S_2, N) is accessible from another state. Then, the states of the system satisfy the weak accessibility (WA) condition [78]. According to [78], if WA condition holds, then the average reward $V(S_1, S_2, N)$ is independent of the initial state (S_1, S_2, N). $\square$

Let PR_{mdp} be the maximal average reward that is calculated by the MDP approach. Then $PR_{mdp} = V(S_1, S_2, N) \ \forall \ (S_1, S_2, N)$, and it can be calculated by using a relative value iteration approach [78]. In the next section, the system production rate under the control limit policy (PR_{cl}) and that under the MDP policy (PR_{mdp}) will be compared through case studies.

4.5 Case Studies

4.5.1 Case study I: structure of the optimal policy

We consider the maintenance policy in a 2M1B system, where the buffer capacity is $C = 10$. The two machines are identical, and their parameters are: $D = 4$, $f^{(1:4)} = [0.02\ 0.05\ 0.1\ 0.2]$, $q = 0.002$, and $T^{(1:4)} = [25\ 30\ 40\ 60]$.

By solving the MDP problem in Section 4.4, one can obtain the optimal policy. The system production rate under the optimal policy is calculated as $PR_{mdp} = 0.9078$. It is noticed that, if we remove all the actions that contain "stop", or $a_1^a \neq a_2^b$, the system production rate is $\tilde{PR}_{mdp} = 0.9078$. Since there is no difference between PR_{mdp} and $\tilde{PR}_{mdp}$, to simplify the analysis, we only discuss the remaining four actions, i.e., (M,M,M),(M,M,D),(D,D,M) and (D,D,D).

Figure 4.7 shows the optimal policy on the downstream machine M_2. In these figures, the optimal policy for "PM" and "default" are represented in black and white, respectively. The vertical axes represent the state of M_2, which is set to be $S_2 \leq D_2$, because otherwise, M_2 is under maintenance and the only available decision on it is "default". The horizontal axes represent the state of M_1. For the figures in the left column, machine M_1 is in a degradation state (i.e., $S_1 \leq D_1$), while the figures in the right column are where M_1 is under maintenance ($S_1 > D_1$). Different rows in Fig. 4.7 represent different real-time buffer levels. These buffer levels are low ($N = 0, 1$), medium ($N = 5$), and high ($N = 9, 10$). The horizontal line is the threshold degradation state for M_2, that is calculated from Section 4.3.1. The control limit policy is to always perform PM if and only if the state of M_2 reaches the threshold state.

Based on Fig. 4.7, the optimal policy differs from the control limit policy in the following two ways.

On the one hand, there are areas where PM is not needed on machine M_2 even if M_2 reaches its threshold degradation state. In other words, maintenance can be postponed to the time when the machine degradation state is worse than the threshold state. We call such policy a *"postponement"* policy. The postponement policy is optimal if M_1 is in a degradation state that is more severe than d_2^*, because $S_1 > d_2^*$ indicates that M_1 will suffer a maintenance time which is longer than $T^{(d_2^*-1)}$, and during the maintenance of M_1, the system can tolerate a similar length of downtime of M_2. For the same reason, if M_1 is already under maintenance, and its remaining maintenance time is long (e.g., longer than $T^{(d_2^*-1)}$), then the maintenance on M_2 can also be postponed.

On the other hand, there is also area where the PM should be performed before the machine degrades to its threshold state. We denote it as an "*advancement*" policy, because the PM is performed in advance compared to the control limit policy.

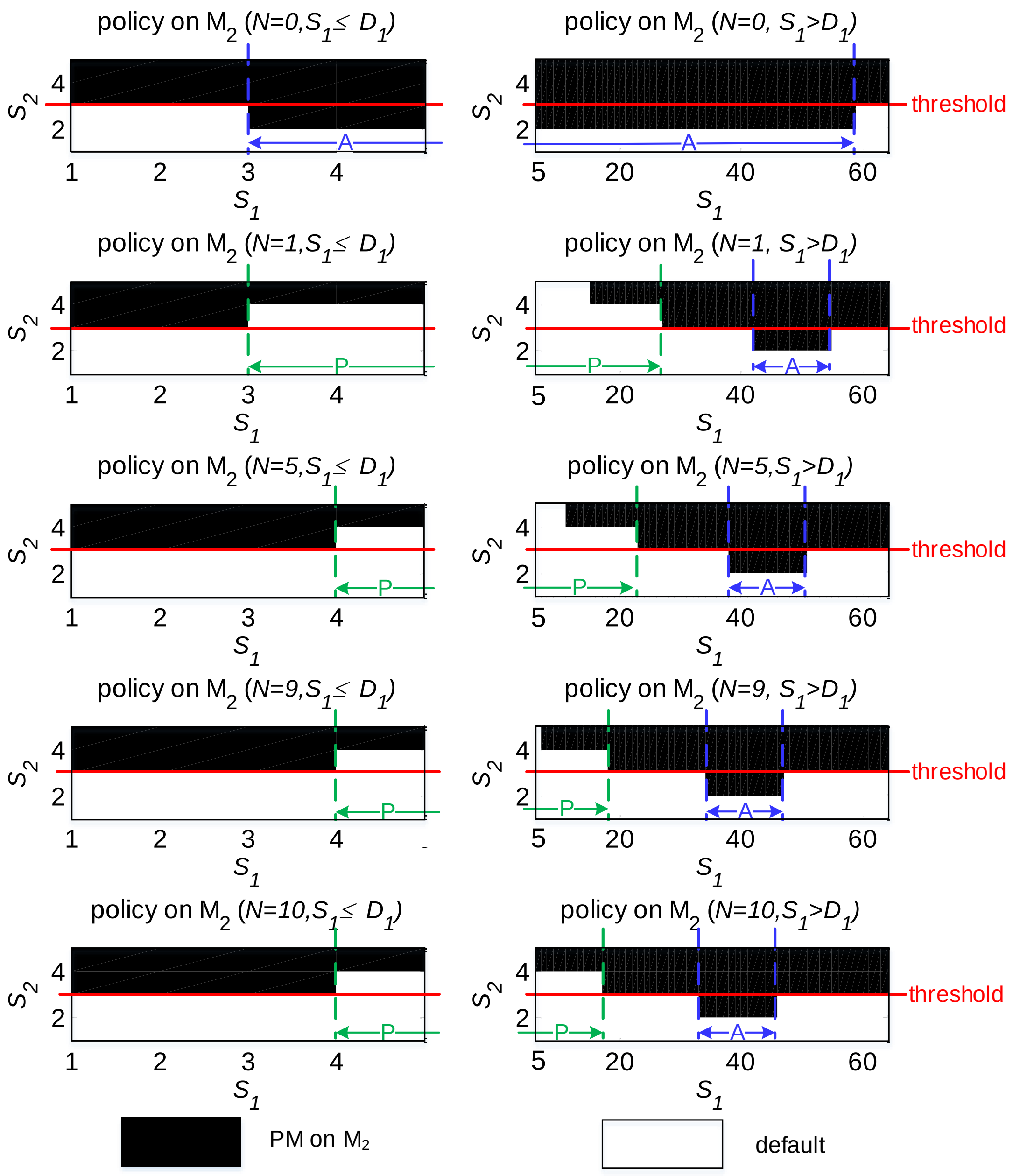


Figure 4.7: Optimal policy for M_2

As the remaining maintenance time on M_1 becomes shorter, the downtime of M_2 that the system can tolerate also becomes shorter. Therefore, it is preferred to performing maintenance on M_2 before it degrades to its threshold state, in order to prevent M_2 from a longer downtime. However, if the maintenance on M_1 is close to completion, the "advancement" policy is not optimal because stopping M_2 for maintenance will block M_1 later after M_1 resumes working.

The area where the "postponement" policy (resp. "advancement" policy) is optimal is marked with "P" (resp. "A") in Fig. 4.7. It can be seen that these areas are also affected by the real-time buffer level N. Based on the analysis in Chapter III, the lower N is, the longer time that M_2 can be down for maintenance without making M_1 blocked. Therefore, if N is lower, then there are more opportunities for "postponement" policies on M_2, and the area associated with the "advancement" policy is allocated where the remaining maintenance time on M_1 is smaller. Especially, if the buffer is empty ($N = 0$), then M_2 is starved. In this case, maintenance on M_2 can be carried out immediately instead of being postponed, and therefore, there is no "postponement" area.

Similarly, the structure of the optimal policy on the upstream machine M_1 can be analyzed, as shown in Fig. 4.8. There exist areas where the optimal policy is the "postponement" policy (when the degradation state of M_2 is severe or the remaining maintenance time on M_2 is long) or the "advancement" policy (when the remaining maintenance time on M_2 is short but not close to completion). The allocation of these areas is affected by the buffer level N: if N is higher, then there are more opportunities for the "postponement" policy, and the area associated with the "advancement" policy is allocated when S_2 is larger. Moreover, if the buffer is full, then M_1 is blocked, and PM on M_1 does not need to be postponed.

Figure 4.9 shows the optimal policy on M_1 and M_2 jointly, when both machines are in degradation states (i.e., $S_1 \leq D_1$ and $S_2 \leq D_2$). However, in most of the cases

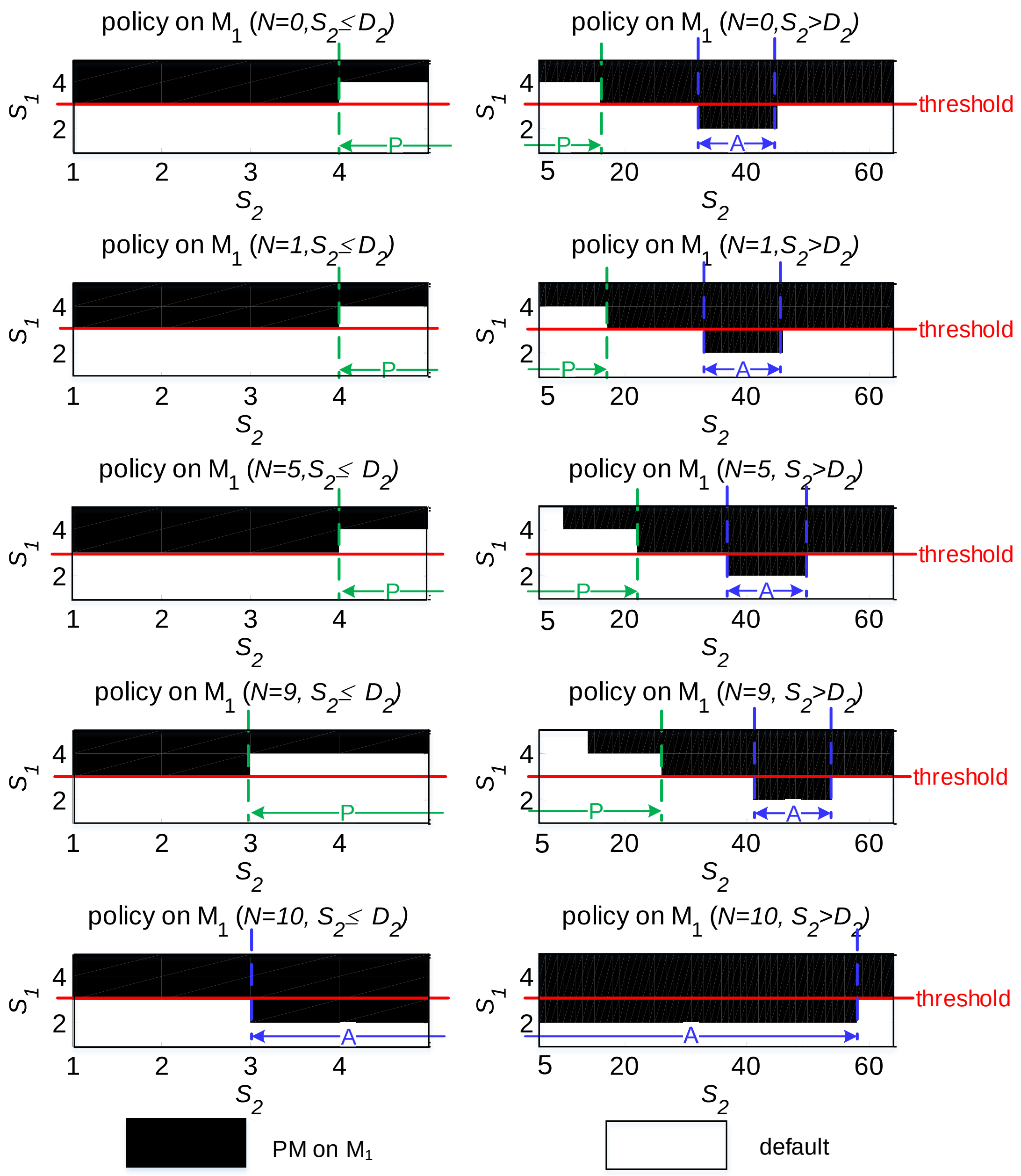


Figure 4.8: Optimal policy for M_1

(i.e., $3 \leq N \leq 8$), the action "PM on both machines" only occurs when both machines are in the same degradation state that exceeds their threshold degradation state (i.e., $S_1 = S_2 \geq d_i^*$, $i = 1, 2$), so these two machines will have *synchronized* maintenance periods. Moreover, if the buffer level is low but not empty (i.e., $N = 1, 2$), the maintenance on M_2 can be postponed; if the buffer level is high but not full (i.e., $N = 9$), the maintenance on M_1 can be postponed. The "advancement" policy on M_1 is optimal only when the buffer is full and M_1 is blocked, and that on M_2 is optimal when the buffer is empty and M_2 is starved.

4.5.2 Case study II: comparison of the MDP policy and the control limit policy

Next, we compare the system production rates under the control limit policy and the MDP policy. The results are compared in 2M1B and 3M2B systems consisting of identical machines. The parameters of the machine are: $D = 4$, $f^{(1:4)} = [0.02\ 0.05\ 0.1\ 0.15]$, $q = 0.01$, and $T^{(1:4)} = [8\ 10\ 15\ 20]$. In both systems, the total buffer capacity varies among 2, 4, ..., 20. The calculated system production rates are shown in Table 4.6.

Table 4.6: Comparison of baseline and optimal policies

two-machine-one-buffer (2M1B) system					
buffer capacity	2	4	6	8	10
PR_{cl}	0.8677	0.8781	0.8847	0.8905	0.8959
PR_{mdp}	0.8861	0.8941	0.8977	0.9003	0.9027
increment(%)	2.12	1.82	1.47	1.10	0.76

three-machine-two-buffer (3M2B) system					
buffer capacity	2	4	6	8	10
PR_{cl}	0.7966	0.8273	0.8393	0.8479	0.8551
PR_{mdp}	0.8396	0.8637	0.8727	0.8778	0.8814
increment(%)	5.40	4.40	3.98	3.53	3.08

Table 4.6 illustrates that, in both systems, as the buffer capacity increases, the steady-state system production rates under both policies increase, because a larger

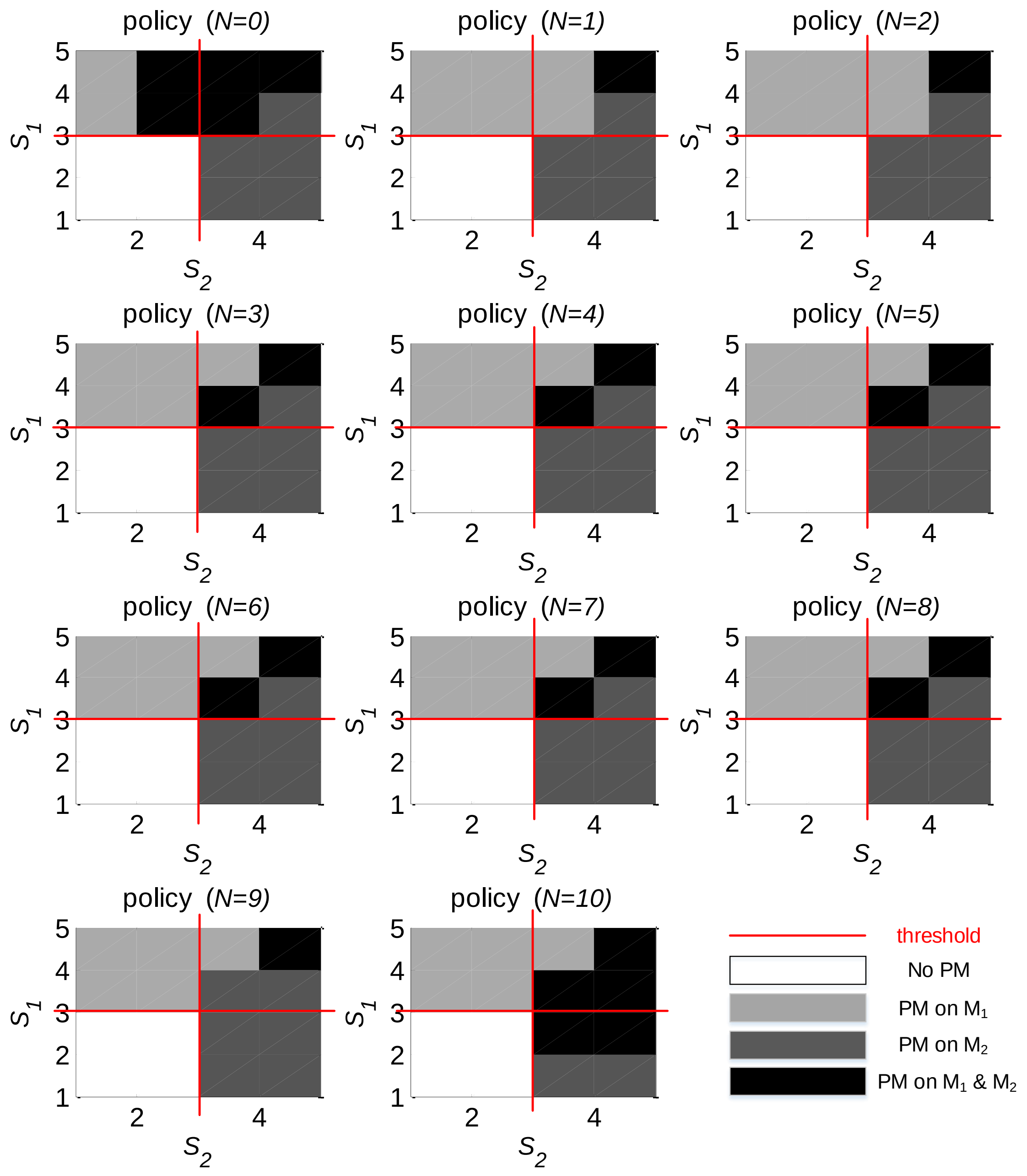


Figure 4.9: Optimal policy under different buffer levels when $S_1 < D_1$ and $S_2 < D_2$

buffer can reduce the chance of making the upstream machine blocked and the downstream machine starved. Moreover, the difference of the MDP policy and the control limit policy decreases as buffer capacity increases. This is also reasonable because if the buffer level is infinity, then the upstream and the downstream machines work independently, so that the control limit policy is optimal. Moreover, comparing the increment in the 2M1B and 3M2B systems, when the system has more machines, the interdependence between these machines and buffers will become more complicated, and the MDP approach will become more advantageous.

4.6 Summary

In this chapter, a discrete-time Markov chain model is developed to investigate the condition-based maintenance policy by incorporating the degradation states or remaining maintenance time of machines, and the contents in the intermediate buffers. First, a control limit policy is denoted as a baseline policy, where the maintenance is only carried out on one machine when its degradation state reaches a threshold value. The control limit policy is evaluated in multi-stage manufacturing systems. Then, an optimal policy is developed by using a Markov Decision Process approach. The baseline policy and the optimal policy are compared by numerical case studies, in 2M1B and 3M2B systems. The result shows that the difference between the optimal policy and the baseline policy increases as the number of machines increases, but it diminishes as the buffer capacity increases.

Future extension of this work is to efficiently investigate heuristic maintenance policies. Although the MDP approach offers an optimal solution, practically it may not be efficient when the number of machines in the system is large. Heuristics for large and complex systems still need to be developed, based on some approximation techniques, such as Approximate Dynamic Programming (ADP).

CHAPTER V

Manufacturing System Design for Resilience

5.1 Introduction

As manufacturing systems age, there are possibilities for increased degradation and failures of systems, subsystems, and components that may cause significant performance and yield losses, increase system life-cycle cost, or create new failure modes. In todays global market, manufacturing enterprises face more frequent and unpredictable changes and disruptions, both externally (e.g., loss of suppliers, operational environment) and internally (e.g., machine breakdowns, machine degradation). These disruptive events may severely impact the system performance.

Therefore, there is a need to improve the system capability of remaining productive during adverse conditions and recovering quickly from faults/failures to normal conditions with minimum performance loss. Such a capability can be viewed as the resilience of a manufacturing system against disruption. Resilience is regarded as an important property for today's manufacturing systems in terms of achieving a sustainable success [79].

In this chapter, we consider a disruptive event that occurs on one machine and causes the machine nonproductive for a certain period of time. After the disruption ends, the machine resumes to its normal working condition and the system recovery begins, and eventually, the system will recover to its original steady state. Figure 5.1

shows the evolution of the system throughput when an unexpected disruption occurs. To understand how the system throughout is affected by the disruption, several research questions need to be addressed: (1) What is the production loss caused by the disruption? (2) After the disruption, how long will it take for the system to recover to its steady state? (3) What is the total time that the system is underproducing? In order to answer these three questions, in this chapter, we study three resilience metrics: production loss (PL), throughput recovery time (TRT_ϵ) and total underproduction time (TUT_ϵ). Similarly as Chapters II to IV, these resilience metrics are evaluated based on the analysis of downtime propagation. However, Chapters II to IV studied the downtime propagation from system operations perspective, while this chapter investigates the problem from system design perspective.

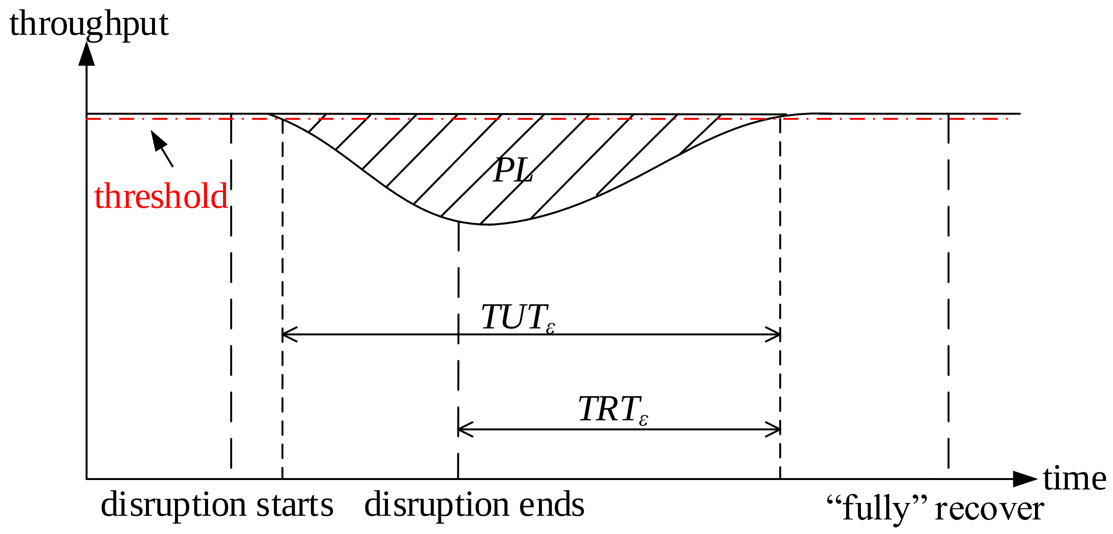


Figure 5.1: The impact of a disruptive event on the system throughput

A resilient system can return to its stable state by incorporating adaptation [80]. Hence, it should be designed with a certain degree of redundancy or flexibility. In this chapter, we consider two types of polices to mitigate the impact of disruptions: (1) the use of built-in system redundancy, and (2) the use of built-in system flexibility. To be more specific, the first policy is to increase the speed of the other machines in the system when disruption occurs. North American automotive factories operate typically at utilization levels of 60% to 70%, so if needed, there exists an opportunity to increase the speed of machines [81]. The systems that are not run at full capacity or operation speed are regarded to have redundancy. The second policy is to take advantages of the built-in flexibility of the system. Specifically, we consider in this chapter the capability of system reconfiguration. Reconfigurable manufacturing system (RMS), suggested by Koren et al. [82], is a system that can rapidly and cost-effectively adjust its production resources in response to unpredictable market changes and intrinsic system events [83, 84]. Such a system has the ability to reallocate its tasks and rebalance itself when its capacity changes [85, 86]. In this chapter, we study how the system throughput will be affected when the system is designed with such redundancy and flexibility.

In this chapter, we study resilience for a system with a RMS configuration (or serial-parallel configuration, i.e., multiple stages connected by crossovers) [83]. Figure 5.2 [85] is the schematic layout of such a system, which consists of multiple parallel CNC machines or reconfigurable machines in each stage. The reasons for studying such a system are two-fold. First, this system is the easiest to be reconfigured [85]. Second, the built-in buffers in the system can delay the propagation of the disruption, and mitigate its negative impact on the system throughput.

The objective of this chapter is to define a set of resilience metrics, develop new modeling and analytical tools to evaluate system resilience, and pursue resilience-based design optimization including redundancy and flexibility allocation, in order to mitigate the impact of disruptions in manufacturing systems.

Resilience is a concept originally from the field of ecology [87]. In recent years, this concept is extended from ecology to other fields such as psychology, economics, organizational science, and engineering [88]. The difference between engineering resilience and ecological resilience is that an engineering system has only one equilibrium state, while an ecological system may have more than one equilibrium state [87]. Zhang et al. [89] defined resilience as the system property on how the system can still function to a desired level when the system suffers from a partial damage, and they distinguished the concept of resilience from other concepts such as reliability, robustness, and fault-tolerance.

For manufacturing community, most of these studies deal with supply chain disruptions at the enterprise level. Christopher and Peck [90] discussed the design principles for resilient supply chains, including "keeping several options open" and "re-examing the efficiency vs redundancy tradeoff". Sheffi and Rice [91] discussed how resilience could be enhanced by investing redundancy and flexibility into the supply chain. Klibi et al. [92] reviewed the papers on supply chain design under uncertainties, and emphasized the importance of resilience for the design. There have been efforts in modeling and analysis of the performance of supply chains under disruptions. Hu et al. [93] studied the optimal capacity control policy in serial manufacturing networks under a disruptive event that could be known in advance. Tomlin [94] investigated the optimal strategy for a system with two suppliers: one unreliable and the other reliable but expensive. Hishamuddin et al. [95] developed a real-time recovery model for a single stage production-inventory system under disruption. DeCroix [96] investigated the optimal ordering policy in an assembly system where one or more of the items was subject to random supply disruptions. However, in these studies, a manufacturing system is usually simplified as a node in the manufacturing network, and the detailed information inside the manufacturing system is usually ignored. Few studies have investigated the resilience of manufacturing systems with regard to instrinsic machine-level disruptions.

The performance of multi-stage manufacturing systems with parallel machines have been studied for decades. For a two-stage system, there exists an exact analytical solution for ther performance analysis. For example, Tan and Gershwin [49] developed a general two-stage continuous-flow model where both stages could have parallel machines. Alexandros and Chrissoleon [97] investigated the behavior of a two-stage system with non-identical parallel machines via an exact Markovian analysis. Liu et al. [98] used Matrix-Analytic method to analyze a two-stage parallel system where machines had three states (working, down, idle). For multi-stage serial-parallel systems, Burman [99] developed a method to aggregate the parallel machines in one stage into one virtual machine, so that the system could be analyzed as a serial line. Such aggregation method is popular in dealing with multi-stage parallel systems [100, 101]. However, the aggregation method is developed to approximate the steady-state behavior of the system, while its transient characteristics may be lost. In order to analyze the transient behavior of systems under the disruption, more analytical work is still needed.

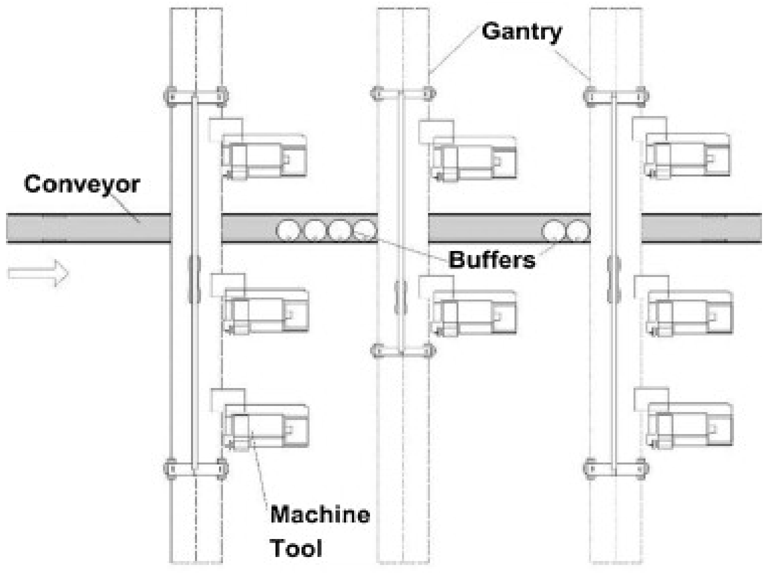


Figure 5.2: Schematic layout of a reconfigurable manufacturing system

The remaining of the chapter is organized as follows. In Section 5.2, we introduce the model and assumptions, as well as different policies that are available because of the built-in redundancy or flexibility of the system. In Sections 5.3 and 5.4, the resilience metrics are evaluated in a two-stage-one-buffer (2S1B) system and a multi-stage system, respectively. Section 5.5 formulates a design optimization problem. In

Section 5.6, a numerical case study is conducted to study the optimal design problem and investigate how the system resilience metrics are affected by different design factors. The summaries and future work are given in Section 5.7.

5.2 Problem Statement

5.2.1 Model and assumptions

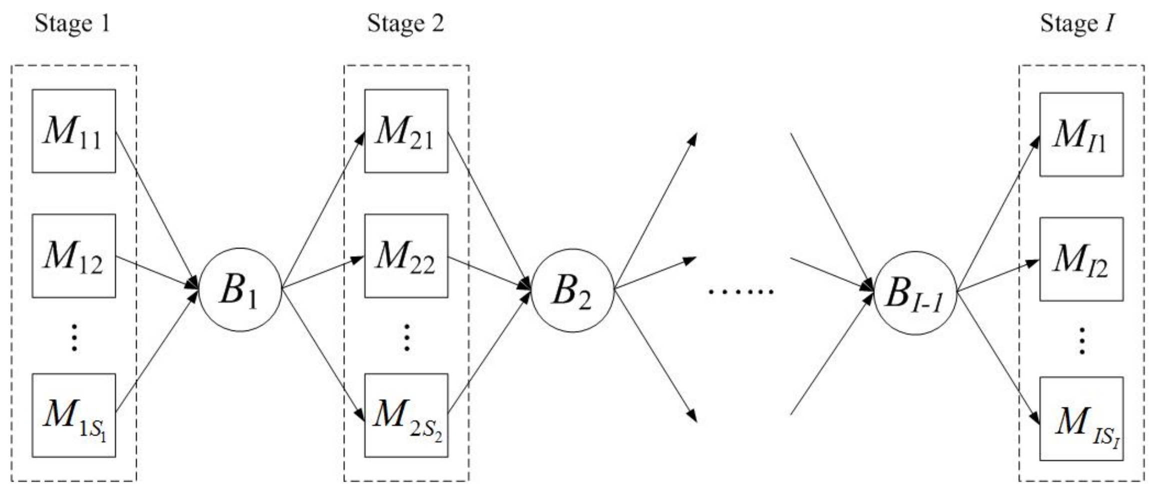


Figure 5.3: An I-stage reconfigurable manufacturing system

We consider an I-stage serial-parallel manufacturing system as shown in Fig. 5.3. There are S_i machines in stage i, with a cycle time T_i ($i = 1, 2, ..., I$). B_i ($i = 1, 2, ..., I-1$) is the intermediate buffer between stage i and stage $i+1$. The assumptions for the system are:

(1) In normal operation, all machines have an identical cycle time (i.e., $T_i = T\ \forall i$) and synchronous operations. All stages have the same number of machines (i.e., $S_i = S$ $\forall i$).

(2) In every cycle, each machine in stage i is "up" with probability p_i, and "down" with probability $1 - p_i$.

(3) If the number of "up" machines in stage S_i is larger than the number of parts in buffer B_{i-1}, then the excessive machines will be starved; if the number of "up" ma-

chines in stage i is larger than the available space in buffer B_i (after the non-starved machines in stage $i+1$ have taken some parts out from B_i), the excessive machines will be blocked. Machines in stage 1 are never starved and machines in stage I are never blocked.

(4) If one machine is blocked or starved at the beginning of the cycle, then it will be blocked or starved during the entire cycle.

(5) Only one disruptive event is considered, and when it occurs, the system is in the steady state. The disruption occurs in stage i_D, and lasts for t_D.

(6) The system required production rate is constant and equal to the system production rate in the steady state.

Remark 5.1. Assumption (1) states that, before and after the disruption, the system is homogeneous (all machines have an identical cycle time) [5]. This assumption is made for simplifying the steady-state analysis. If the machine cycle time is changed during the disruption, the system becomes inhomogeneous. In this chapter, we will also investigate the performance of general and inhomogeneous systems (see Sections 5.3.2 and 5.4.2). Assumption (4) guarantees that the machines in the same stage have synchronous cycles.

5.2.2 Description of the process

A resilient system is designed with redundancy or flexibility so that it can mitigate the effects of disruption. If the system is designed with these capabilities, then associated actions are available to deal with unexpected disruption. Figure 5.4 shows the system throughput profile when the system is designed with different capabilities.

If the system is designed with redundancy, then during the disruption, the speed of machines in stage i_D can be increased, or equivalently, their cycle time can be reduced. This is denoted as policy A. It is assumed that the time to change machine speed is negligible. Let Δ be the maximal percentage of the cycle time reduction.

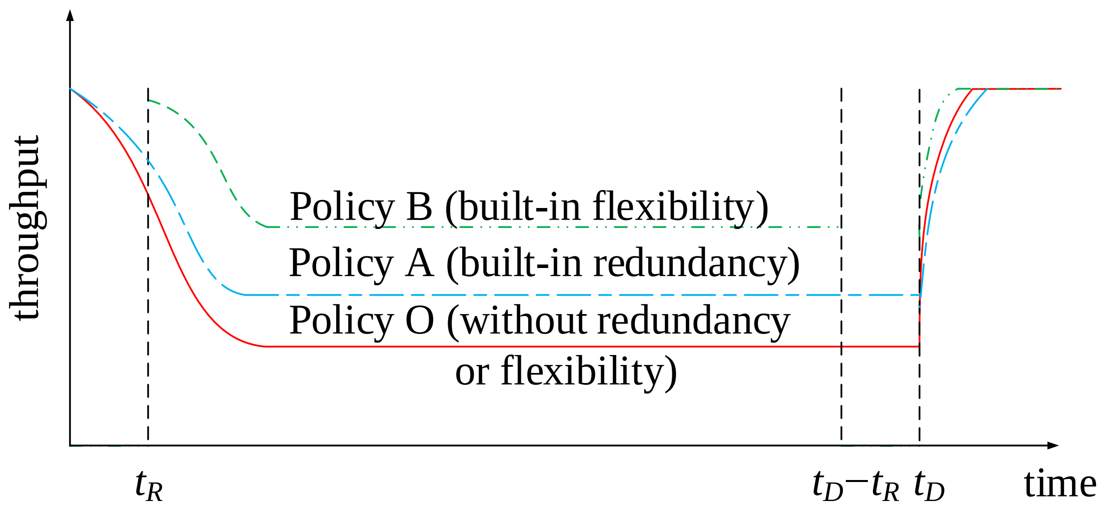


Figure 5.4: Comparison of system production rate under policies O, A, or B

Then, during the disruption, the cycle time of stage i can be calculated as

$$T_i^A = \begin{cases} \max\{T(1-\Delta), (S-1)T/S\} & i = i_D \\ T & i \neq i_D \end{cases}$$

Note that, it is unnecessary to make the cycle time smaller than $(S-1)T/S$, because otherwise the stage will overproduce.

If the system is designed with flexibility, then when the disruption occurs, the system can be reconfigured; and before the disruption ends, the system is reconfigured back to its original operation. This policy is denoted as policy B. Although reconfiguration can help mitigate the effect of disruption, the reconfiguration process itself is usually not instant and requires the production system to be shut down. We assume both reconfigurations take t_R units of time, and during each reconfiguration period, tasks are reallocated and the completed operations on the work-in-process parts can be adjusted for the system after this reconfiguration. Under the disruption, the productive period is $t_R < k \leq t_D - t_R$, and let T_i^B be the cycle time of stage i during this period. It is assumed that, the system can be perfectly reconfigured so that the isolated production rates for every stage are the same (i.e., $T_{i_D}^B/(S-1) = T_i^B/S$ $\forall i \neq i_D$). Therefore, T_i^B can be calculated as

$$T_i^B = \begin{cases} (S-1)IT/(SI-1) & i = i_D \\ SIT/(SI-1) & i \neq i_D \end{cases}$$

Specially, we also denote policy O for the system designed with no redundancy or flexibility, as regard it as a baseline policy. Policy O means no action can be taken during the disruption, and it can be regarded as a special case of policy A where $\Delta = 0$.

To summarize, if we denote $t_{prep}^X = t_R \cdot \delta_{XB}$ as the preparation time for conducting policy X ($X = O, A, B$), then the system is productive during periods $t_{prep}^X < k \leq t_D - t_{prep}^X$ and $k > t_D$. Let $t_{i,avail}^X$ be the time that machines in stage i are available (i.e., when the required operations on the previous parts are completed, and the stage is ready to work on the next parts) under policy X, and it can be calculated as

$$t_{i,avail}^X = \begin{cases} t_{prep}^X + hT_i & h = 1, ..., \lfloor (t_D - t_{prep}^X)/T_i^X \rfloor \\ t_D + hT & h = 1, 2, ... \end{cases} \tag{5.1}$$

Let PL^X, TRT_ϵ^X and TUT_ϵ^X be the three resilience metrics under policy X ($X = 0, A, B$). In Section 5.3, we evaluate them in two-stage-one-buffer (2S1B) systems, and in Section 5.4, we extend the analysis to multi-stage systems.

5.3 Resilience Metrics in Two-Stage-One-Buffer Systems

5.3.1 Analysis of steady-state performance of 2S1B systems

First, we perform an exact analysis for a two-stage system, where there are S_1 upstream machines and S_2 downstream machines. Since there is only one buffer, we denote it as B, its capacity as C and its inventory level at time k as $N(k)$.

In our model, since all the machines have an identical cycle time and work synchronously, the system state will only change at time $k = 0, T, 2T, ...$ Therefore, in this section, we only consider the system behavior at these time points. Similarly with Chapter III, we let $\pi_n^S = \lim_{h\to+\infty} Pr\{N(hT) = n\}$ be the probability that the buffer level is n in the steady state. Then steady-state distribution of the buffer level, $\boldsymbol{\pi}^S = [\pi_0^S \ \pi_1^S \ \cdots \ \pi_C^S]^T$, can be calculated as

$$\begin{aligned}\boldsymbol{\pi}^S = &\begin{bmatrix} r_1(0) & 0 & \cdots & 0 & 0 \\ r_1(1) & r_1(0) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_1(C-1) & r_1(C-2) & \cdots & r_1(0) & 0 \\ \sum\limits_{j=C}^{S_1} r_1(j) & \sum\limits_{j=C-1}^{S_1} r_1(j) & \cdots & \sum\limits_{j=1}^{S_1} r_1(j) & \sum\limits_{j=C}^{S_1} r_1(j) \end{bmatrix} \\ &\cdot \begin{bmatrix} \sum\limits_{j=0}^{S_2} r_2(j) & \sum\limits_{j=1}^{S_2} r_2(j) & \cdots & \sum\limits_{j=C-1}^{S_2} r_2(j) & \sum\limits_{j=C}^{S_2} r_2(j) \\ 0 & r_2(0) & \cdots & r_2(C-2) & r_2(C-1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & r_2(0) & r_2(1) \\ 0 & 0 & \cdots & 0 & r_2(0) \end{bmatrix} \boldsymbol{\pi}^S \\ :=&\mathbf{P}(\mathbf{r}_1, \mathbf{r}_2, C)\boldsymbol{\pi}^S \end{aligned} \tag{5.2}$$

where $\mathbf{r}_i = [r_i(0) \ r_i(1) \ \cdots \ r_i(S_i)]^T (i = 1, 2)$ and $\mathbf{P}(\mathbf{r}_1, \mathbf{r}_2, C)$ is the $(C+1)\times(C+1)$ matrix of the transition probabilities. Let P_{mn} be the element at the $(n+1)^{th}$ column and the $(m+1)^{th}$ row of $\mathbf{P}(\mathbf{r}_1, \mathbf{r}_2, C)$, and it is easy to get

$$P_{mn} = \begin{cases} \sum\limits_{l=0}^{n-1} r_2(l) r_1(l+m-n) + r_1(m) \sum\limits_{l=n}^{S_2} r_2(l) & \text{if } m < C \\ \sum\limits_{h=C}^{n+S_1} \left(\sum\limits_{l=0}^{n-1} r_2(l) r_1(l+h-n) + r_1(h) \sum\limits_{l=n}^{S_2} r_2(l) \right) & \text{if } m = C \end{cases} \tag{5.3}$$

Moreover, in the two-stage system, the upstream machines are never starved and the downstream machines are never blocked. Therefore,

$$r_i(j) = \begin{cases} \binom{S_i}{j} p_i^j (1-p_i)^{S_i - j} & \text{if } 0 \le j \le C \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, 2) \tag{5.4}$$

Note that when $S_1 = S_2 = 1$, the system is the two-machine-one-buffer system in Chapter III.

From Equation (5.2), the *steady-state* distribution of the buffer $\boldsymbol{\pi}^S$ can be calculated as

$$\boldsymbol{\pi}^S = \mathbf{\Pi}(\mathbf{P}) := \boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C) \tag{5.5}$$

where $\mathbf{\Pi}(\cdot)$ is defined in Definition 4.1.

Let $pr(j)$ be the probability that the number of parts completed by stage 2 is j; $cr(j)$ be the probability that the number of parts entering stage 1 is j; $st(j)$ be the probability that the number of starved machines in stage 2 is j; and $bl(j)$ be the probability that the number of blocked machines in stage 1 is j. The analytical expressions of these metrics in the steady state are summarized in Lemma 5.1.

Lemma 5.1. *The steady-state $pr^S(j)$, $cr^S(j)$, $st^S(j)$, and $bl^S(j)$ can be calculated as follows:*

(1)

$$\mathbf{pr}^S = [pr(0) \quad \cdots \quad pr(S_2)]^T = \mathbf{p}(\mathbf{r}_2, C)\boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C) \tag{5.6}$$

where $p_{mn} = \sum_{l=0}^{n} \delta_{lm} r_2(m) + \delta_{nm} \sum_{l=n+1}^{S_2} r_2(l)$;

(2)

$$\mathbf{cr}^S = [cr(0) \quad \cdots \quad cr(S_1)]^T = \mathbf{c}(\mathbf{r}_1, \mathbf{r}_2, C)\boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C) \tag{5.7}$$

where $c_{m,C-n} = r_1(m) \sum_{l=m-n}^{S_2} r_2(l) + r_2(m-n) \sum_{l=m+1}^{S_1} r_1(l)$;

(3)

$$\mathbf{st}^S = [st(0) \quad \cdots \quad st(S_2)]^T = \mathbf{s}(\mathbf{r}_2, C)\boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C) \tag{5.8}$$

where $s_{mn} = r_2(n+m)$;

(4)

$$\mathbf{bl}^S = [bl(0) \quad \cdots \quad bl(S_1)]^T = \mathbf{b}(\mathbf{r}_1, \mathbf{r}_2, C)\boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C), \tag{5.9}$$

where $b_{m,C-n} = \sum_{l=0}^{S_2} r_1(n+m+l)r_2(l)$.

Proof. The elements can be easily obtained by taking condition on the number of machines that are available in either stage. □

Based on these metrics, one can easily calculate the steady-state production rate as $PR^S = [0 \quad \cdots \quad S_2]\mathbf{pr}^S$, consumption rate as $CR^S = [0 \quad \cdots \quad S_1]\mathbf{cr}^S$, expected number of starved machines as $ST^S = [0 \quad \cdots \quad S_2]\mathbf{st}^S$, expected number of blocked machines as $bl^S = [0 \quad \cdots \quad S_1]\mathbf{bl}^S$, and work-in-process as $WIP^S = [0 \quad \cdots \quad C]\boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C)$.

5.3.2 Analysis of transient performance of 2S1B systems

Then we analyze the transient behavior of the 2S1B system. The methodology developed in this section can be generally used to evaluate the resilience metrics for the system under any policy X ($X = O, A, B$). Therefore, in the analysis, the superscript "X" is omitted for convenience.

As mentioned in Section 5.2.2, the system behavior needs to be investigated in two periods. The first period $t^X_{prep} < k \leq t_D - t^X_{prep}$ is the time when policy X is conducted, and in the second period $k > t_D$, the system is back to its normal operation.

The transient system dynamics in both periods can be obtained by applying Equation (5.2) at time k, as

$$\boldsymbol{\pi}(k) = \mathbf{P}(\mathbf{r}_1(k), \mathbf{r}_2(k), C)\boldsymbol{\pi}(k-1) \tag{5.10}$$

where

$$r_i(j,k) = \begin{cases} \delta_{j0} & \text{if } k \neq t_{i,avail} \\ \binom{S_i - 1}{j} p_i^j (1-p_i)^{S_i-1-j} & \text{if } i = i_D,\ k \leq t_D \text{ and } k = t_{i,avail} \\ \binom{S_i}{j} p_i^j (1-p_i)^{S_i-j} & \text{otherwise} \end{cases} \tag{5.11}$$

and the system production rate at time k can be calculated as

$$PR(k) = [0 \cdots S_2]\mathbf{p}(\mathbf{r}_2(k), C)\boldsymbol{\pi}(k-1) \tag{5.12}$$

Based on the transient system production rate $PR(k)$, three resilience metrics can be evaluated.

Production Loss (PL): The first resilience metric is the total production loss before the system returns to its steady state. It is defined as

$$PL = \sum_{k=1}^{\infty} \left[PR^S/T - PR(k) \right]. \tag{5.13}$$

However, Equation (5.13) requires to calculate $PR(k)$ for every k until k is sufficiently large. Next, we develop a method to calculate PL without obtaining $PR(k)$ for all k's. From Equation (5.13), the production loss during the disruption is $\sum_{k=1}^{t_D} \left[PR^S/T - PR(k) \right]$. Similarly to Chapter III, the production loss after the disruption ends can be calculated by using a delayed renewal process.

We define $v_{mn} := \min\{h : N((h+s)T) = m, h > 0 | N(sT) = n\}$ and $t_{mn}^{(l)} := \sum_{h=1}^{v_{mn}} \delta_{N(hT),l}$ for $l = 0, 1, ..., S_2 - 1$. In other words, v_{mn} is the time duration from state n to the first time the system reaches state m, and $t_{mn}^{(l)}$ is total time that the buffer level is l during v_{mn}. Denote $V_{mn} = \mathbb{E}[v_{mn}]$ and $T_{mn}^{(l)} = \mathbb{E}[t_{mn}^{(l)}]$, and they can be obtained from the following theorem.

Theorem 5.1. *Denote* $\mathbf{V}_n = [V_{0n} \quad \cdots \quad V_{Cn}]$, $\mathbf{T}_n = [T_{0n} \quad \cdots \quad T_{Cn}]$ *($n = 0, ..., C$),* *and* $\mathbf{V} = [\mathbf{V}_0 \quad \cdots \quad \mathbf{V}_C]$, $\mathbf{T}^{(l)} = [\mathbf{T}_0^{(l)} \quad \cdots \quad \mathbf{T}_C^{(l)}]$ *(*$l = 0, ..., S_2 - 1$*). Then,* $\mathbf{V} = (\mathbf{I} - \mathbf{W}^T)^{-1} \cdot \mathbf{1}^T$ *and* $\mathbf{T}^{(l)} = (\mathbf{I} - \mathbf{W}^T)^{-1}[\delta_{0l} \cdot \mathbf{1} \quad \cdots \quad \delta_{Cl} \cdot \mathbf{1}]$*, where* $\mathbf{1} = [1 \quad \cdots \quad 1]$*,*

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{00} & \cdots & \mathbf{W}_{0C} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_{C0} & \cdots & \mathbf{W}_{CC} \end{bmatrix}, \textit{ and } \mathbf{W}_{mn} = P_{mn} \begin{bmatrix} 1 - \delta_{0m} & & \\ & \ddots & \\ & & 1 - \delta_{Cm} \end{bmatrix}.$$

Proof. See Appendix D. □

Then we calculate the production loss from state n to state m. When the system buffer level is l, the expected production rate can be calculated as $\sum_{h=0}^{S_2} hp_{hl}$, where p_{hl} is calculated in Lemma 5.1. Note that, when the buffer level is greater or equal to the number of machines in the second stage (i.e., $l \geq S_2$), no downstream machines will be starved, and thus $p_{hl} = r_2(h)$. The expected production loss from state n to state m can thus be calculated as

$$PL_{mn} = V_{mn}PR^S - \sum_{l=0}^{S_2-1} \sum_{h=0}^{S_2} T_{mn}^{(l)} \cdot h \cdot p_{hl} - (V_{mn} - \sum_{l=0}^{S_2-1} T_{mn}^{(l)}) \sum_{h=0}^{S_2} h \cdot r_2(h) \quad (5.14)$$

Finally, the expected production loss caused by the disruption can be calculated as

$$PL = \sum_{k=1}^{t_D} \left[PR^S/T - PR(k)\right] + \boldsymbol{\pi}^T(t_D) \begin{bmatrix} PL_{00} & \cdots & PL_{0C} \\ \vdots & \ddots & \vdots \\ PL_{C0} & \cdots & PL_{CC} \end{bmatrix}^T \boldsymbol{\pi}^S(\mathbf{r}_1, \mathbf{r}_2, C) \quad (5.15)$$

where the first item is the production loss during the disruption and the second item is the production loss after the machine resumes working. Numerical result shows that the production losses calculated from Equations (5.13) and (5.15) are equal.

Throughput recovery time (TRT_ϵ**):** The second resilience metric TRT_ϵ is the elapse time after t_D that system production rate returns to $(1 - \epsilon)PR^S$, and never drops below that value afterwards. Mathematically,

$$TRT_\epsilon = \max\{h \mid h \geq t_D/T, PR(hT) < (1-\epsilon)PR^S\})T - t_D \tag{5.16}$$

Total underproduction time (TUT_ϵ): The third resilience metric TUT_ϵ is the total time when the system is underproducing. Note that, the required production rate is PR^S in every T units of time. In other words, for any h, the required production units in time interval $(hT+1, hT+T]$ is PR^S. Under any policy, if the actual number of the production units in $(hT+1, hT+T]$ is smaller than PR^S, the system is regarded to underproduce during the entire T units of time. Therefore, TUT_ϵ is defined as

$$TUT_\epsilon = \sum_{h=0}^{\infty} \mathbb{1}\{\sum_{l=1}^{T} PR(hT+l) < (1-\epsilon)PR^S\}T \tag{5.17}$$

5.4 Resilience Metrics in Multi-Stage Systems

As the number of stages increases, the number of states increases dramatically, making the exact solution difficult to obtain. In Chapter III, we have developed a decomposition method to study the transient behavior of manufacturing system where there is only one machine in each stage. In this chapter, we extend the analysis to the multi-stage systems with parallel machines in each stage.

5.4.1 Analysis of the steady-state performance of multi-stage systems

For the 2S1B system studied in Section 5.3, the upstream machines are never starved and the downstream machines are never blocked. In order to analyze the decomposed 2S1B system, we need to find the non-starved and the non-blocked parts of the stages. We denote S_i^{NS} (resp., S_i^{NB}) as the virtual stage that represents the non-starved (resp., non-blocked) part of stage i, and it can be characterized by $\mathbf{r}_i^{NS} = [r_i^{NS}(0) \quad \cdots \quad r_i^{NS}(S_i)]$ (resp., $\mathbf{r}_i^{NB} = [r_i^{NB}(0) \quad \cdots \quad r_i^{NB}(S_i)]$), where $r_i^{NS}(j)$ (resp.,

$r_i^{NB}(j)$) is the probability that there are j machines in stage i that are up and not starved (resp., up and not blocked). Then, viewed from buffer B_i ($i = 1, ..., I-1$), the system can be represented as a two-stage system, $S_i^{NS} - B_i - S_{i+1}^{NB}$. Hence, the entire system can be decomposed into $I-1$ 2S1B systems (i.e., $S_i^{NS} - B_i - S_{i+1}^{NB}$ for $i = 1, ..., I-1$), where each 2S1B system can be analyzed by the approach developed in Section 5.3. The remaining challenge is to find $\mathbf{r}_i^{NS}$ and $\mathbf{r}_i^{NB}$.

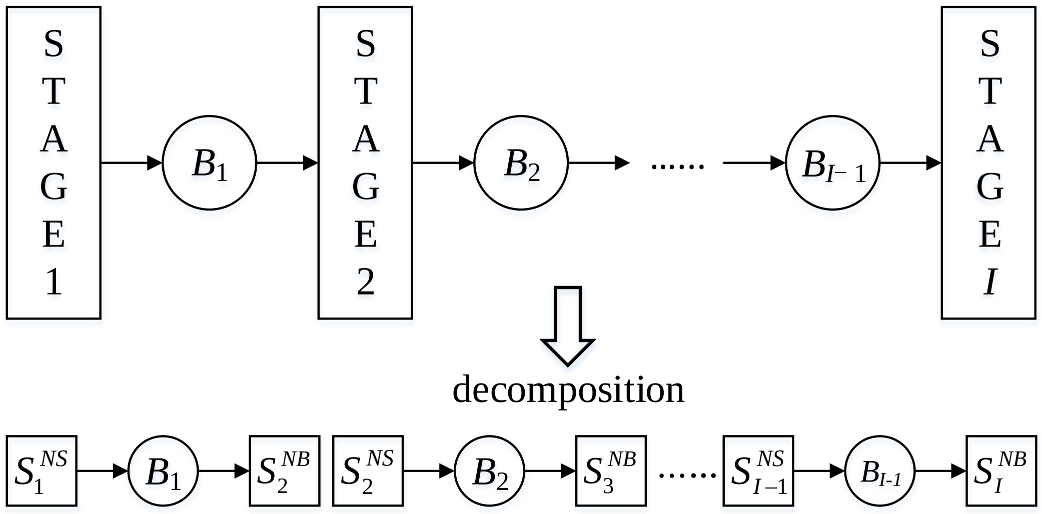


Figure 5.5: Decomposition of the I-stage system

First, because the first stage is never starved and the last stage is never blocked, we have $\mathbf{r}_1^{NS} = \mathbf{r}_1$ and $\mathbf{r}_I^{NB} = \mathbf{r}_I$.Then we consider the other stages. By definition, $r_i^{NS}(j)$ ($i = 2, ..., I; j = 0, ..., S_i$) is the probability that there are j machine in stage i that are available and not starved. Since the starvation of stage S_i comes from buffer B_{i-1}, we consider the two-stage system $S_{i-1}^{NS} - B_i - S_i^{NB}$. By Equation (5.5), the stationary distribution of B_{i-1} can be written as $\boldsymbol{\pi}^S(\mathbf{r}_{i-1}, \mathbf{r}_i, C_{i-1})$. By Lemma 5.1, $\mathbf{p}(\mathbf{r}_i, C_{i-1})$ transforms the distribution of the buffer level into the distribution of the production rate of stage i with the starvation taken into consideration. Therefore, we have

$$\mathbf{r}_i^{NS} = \mathbf{p}(\mathbf{r}_i, C_{i-1})\boldsymbol{\pi}^S(\mathbf{r}_{i-1}^{NS}, \mathbf{r}_i^{NB}, C_{i-1}) \quad (i = 2, ..., I-1) \tag{5.18}$$

Similarly, we can obtain $\mathbf{r}_i^{NB}$ $(i = 1, .., I-1)$, the vector characterizing the distribution of the number of the non-blocked machines in stage i, which can also be illustrated as the distribution of the consumption rate of the 2S1B system $S_i^{NS} - B_i - S_{i+1}^{NB}$ considering the blockage (but not starvation) of stage i. Hence,

$$\mathbf{r}_i^{NB} = \mathbf{c}(\mathbf{r}_i, \mathbf{r}_{i+1}^{NB}, C_i)\boldsymbol{\pi}^S(\mathbf{r}_i^{NS}, \mathbf{r}_{i+1}^{NB}, C_i) \quad (i = 2, ..., I-1) \tag{5.19}$$

Moreover, $\mathbf{r}_i^{NB}$ and $\mathbf{r}_i^{NS}$ in Equations (5.18) and (5.19) can be obtained by using the following recursive procedure, whose convergence is given in Theorem5.2.

Recursive Procedure 5.1. We solve the following equations recursively for $a = 0, 1, 2, ...$

Step 1. For $i = 2, ..., I-1$:

$$\mathbf{r}_i^{NS}(a+1) = \mathbf{p}(\mathbf{r}_i, C_{i-1})\boldsymbol{\pi}^S(\mathbf{r}_{i-1}^{NS}(a+1), \mathbf{r}_i^{NB}(a), C_{i-1}); \tag{5.20}$$

Step 2. For $i = I-1, ..., 2$:

$$\mathbf{r}_i^{NB}(a+1) = \mathbf{c}(\mathbf{r}_i, \mathbf{r}_{i+1}^{NB}(a+1), C_i)\boldsymbol{\pi}^S(\mathbf{r}_i^{NS}(a+1), \mathbf{r}_{i+1}^{NB}(a+1), C_i); \tag{5.21}$$

with the initial condition

$$\mathbf{r}_i^{NB}(0) = \mathbf{r}_i \quad (i = 2, ..., I-1); \tag{5.22}$$

and the boundary conditions

$$\mathbf{r}_1^{NS}(a) = \mathbf{r}_1^{NS} \text{ and } \mathbf{r}_I^{NB}(a) = \mathbf{r}_I^{NB} \quad (\forall a). \tag{5.23}$$

Theorem 5.2. *Recursive Procedure 5.1 is convergent, i.e.*

$$\lim_{a\to\infty} \mathbf{r}_i^{NS}(a) = \mathbf{r}_i^{NS}, \textit{and} \lim_{a\to\infty} \mathbf{r}_i^{NB}(a) = \mathbf{r}_i^{NB} \quad (\forall i) \tag{5.24}$$

Proof. See Appendix D. □

Now, we are able to calculate the steady-state production rate of the system, which is the production rate of the last 2S1B subsystem, $S_{I-1}^{NS} - B_{I-1} - S_I^{NB}$, as

$$PR^S = [0 \quad 1 \quad \cdots \quad S_I]\mathbf{p}(\mathbf{r}_I, C_{I-1})\boldsymbol{\pi}^S(\mathbf{r}_{I-1}^{NS}, \mathbf{r}_I, C_{I-1}) \tag{5.25}$$

5.4.2 Analysis of the transient performance of multi-stage systems

The dynamics of a multi-stage serial-parallel system is shown in Equations (5.18) and (5.19). However, instead of finding the steady-state $\mathbf{r}_i^{NS}$ and $\mathbf{r}_i^{NB}$, we need to calculate them at each time unit k; which are denoted as $\mathbf{r}_i^{NS}(k)$ and $\mathbf{r}_i^{NB}(k)$, respectively. Based on Assumption (3), the blockage of one machine may depend on the states of its downstream machines, while the starvation of that machine does not depend on the states of its upstream machines. Therefore, the following backward recursive procedure is developed to recursively calculate the transient reliabilities at each time unit k.

Recursive Procedure 5.2. We solve the following equations recursively for $k = 1, 2, ...$

Step 1. For $i = I-1, ..., 1$: $\boldsymbol{\pi}_i(0) = \boldsymbol{\pi}^S(\mathbf{r}_i^{NS}, \mathbf{r}_{i+1}^{NB}, C_i)$, where $\mathbf{r}_i^{NS}$ and $\mathbf{r}_{i+1}^{NB}$ can be obtained from Recursive Procedure 5.1.

Step 2. Then we solve Steps 2a to 2c recursively for $k = 1, 2, ...$

Step 2a. For $i = 2, ..., I$: $\mathbf{r}_i^{NS}(k) = \mathbf{p}(\mathbf{r}_i(k), C_{i-1})\boldsymbol{\pi}_{i-1}(k-1)$;

Step 2b. For $i = I-1, ..., 1$: $\mathbf{r}_i^{NB}(k) = \mathbf{c}(\mathbf{r}_i(k), \mathbf{r}_{i+1}^{NB}(k), C_i)\boldsymbol{\pi}_i(k-1)$ with the boundary condition $\mathbf{r}_I^{NB} = \mathbf{r}_I$;

Step 2c. For $i = 1, ..., I-1$: $\boldsymbol{\pi}_i(k) = \mathbf{P}(\mathbf{r}_i^{NS}(k), \mathbf{r}_{i+1}^{NB}{}_i(k), C_i)\boldsymbol{\pi}_i(k-1)$ with the boundary condition $\mathbf{r}_1^{NS} = \mathbf{r}_1$.

Similarly as Equation (5.25), the production rate at k ($k = 1, 2, ...$) can be calculated as

$$PR(k) = [0 \;\; 1 \;\; \cdots \;\; S_I]\mathbf{p}(\mathbf{r}_I, C_{I-1})\boldsymbol{\pi}(k-1) \tag{5.26}$$

Once all $PR(k)$'s are obtained, one can use Equations (5.13), (5.16) and (5.17) to evaluate PL, TRT_ϵ, and TUT_ϵ, respectively.

5.5 Manufacturing System Design for Resilience

In Sections 5.3 and 5.4, three resilience metrics are evaluated in multi-stage manufacturing systems. Next, we formulate an optimization problem to illustrate how these resilience metrics can be utilized for system design. More specifically, one can decide the optimal degree of redundancy and flexibility to be allocated in the system so that the system is both resilient and cost-effective. These decisions include the system configuration, the capabilities of machines, and the capacities of buffers, etc.

In the design optimization problem, there are two kinds of cost to be considered: one is the investment cost on the equipment such as machines and buffers, and the second is the operating cost. Furthermore, the operating cost consists of the cost associated with the regular production (e.g., energy consumption, labor cost, etc.), and the cost associated with the disruption-induced performance loss (e.g., it incurs an overtime cost if one uses overtime to compensate for the lost production).

We consider the problem of designing a manufacturing system with a design life of Y years. Let c_{inv} be the total investment cost, $c_{op}(y)$ be the operating cost in year y, and ω be the discount factor. The objective of the design optimization problem is to minimize the total cost over the system designed life cycle, as

$$\min c_{inv} + \sum_{y=1}^{Y} \omega^{y-1} \cdot c_{op}(y) \tag{5.27}$$

where

$$c_{op}(y) = \kappa \cdot DEM(y)/PR^S + \sum_D \Big[f_D(y) \cdot \big(\alpha \cdot PL_D + \beta \cdot TRT_{\epsilon,D} + \gamma \cdot TUT_{\epsilon,D} \big) \Big] \tag{5.28}$$

In Equation (5.28), κ is the unit time cost associated with the normal operation, $DEM(y)$ is the demand in year y, PR^S is the system production rate when it is in normal operation. Moreover, $f_D(y)$ is the frequency of the disruption event D in year y, which can be practically predicted or estimated by using prognostics and health management (PHM) tools [102]. PL_D, $TRT_{\epsilon,D}$ and $TUT_{\epsilon,D}$ are the corresponding resilience metrics under the disruptive event D; and α, β, γ are the cost parameters associated with these resilience metrics.

5.6 Case Studies

5.6.1 System description

We consider the design optimization problem of a 6-machine system, in order to satisfy a demand of 300,000 units/year. The manufacturing process consists of a sequence of 30 operations, each of which taking 10 s. Therefore, the total processing time to complete one part is 300 s. Two alternative configurations are used for comparison, as shown in Fig. 5.6.

In each configuration, one can choose three types of machines: dedicated machine (machine with neither redundancy nor flexibility), machine with redundancy, and machine with flexibility (reconfigurability). Additionally, one can choose the capacities of the buffers in the system. Table 5.1 shows the investment cost on machines and buffers in both configurations. The reliability of all machines is assumed to be 0.95.

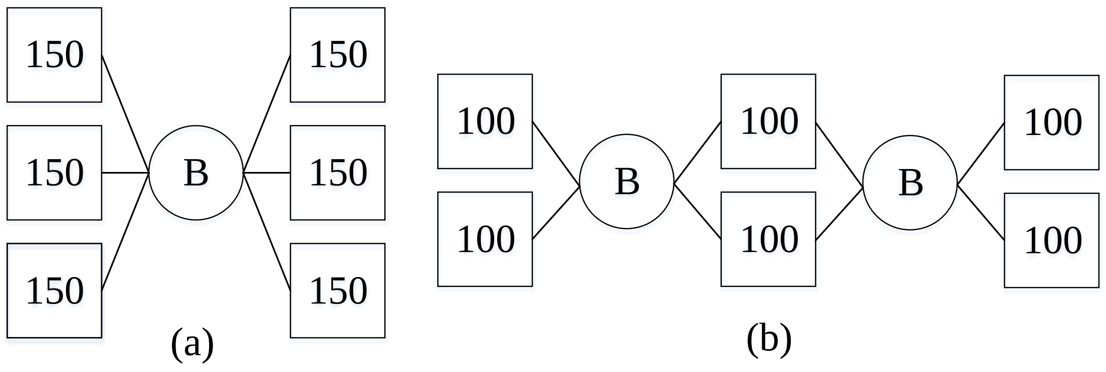


Figure 5.6: Configurations of the six-machine systems

Note that, the cost of machines in configuration (a) is higher than those in configuration (b) because there are more tasks assigned on each machine in configuration (a) than configuration (b).

Table 5.1: Investment cost of machines and buffers

configuration	(a)	(b)
dedicated machine ($/machine)	120,000	90,000
machine with redundancy ($/machine)	150,000	120,000
machine with flexibility ($/machine)	170,000	160,000
buffer ($/capacity)	10,000	

If the system is designed with redundancy or flexibility, then the machine cycle time can be changed during the disruption. If the system has built-in redundancy, then under policy A, the machine cycle time can be reduced. It is assumed that under policy A, the cycle time of each machine can be reduced by at most 20% (i.e., $\Delta = 0.2$), and under policy O, $\Delta = 0$. If the system is designed with flexibility (reconfigurability), then it is assumed that each reconfiguration period is 5 min (300 s), and the system can be perfectly reconfigured under policy B. Figure 5.6 shows the machine cycle times under different policies when the disruption occurs in the first stage. For example, in configuration (a), the machine cycle time is 150 s under policy O, and 120 s under policy A, which is a 20% reduction. Under policy B, the cycle time of the first stage (120 s) is 2/3 of that of the second stage (180 s). It indicates that, during the first reconfiguration period, 3 operations should be reallocated from the second stage to the first stage. A similar calculation can be applied to configuration (b).

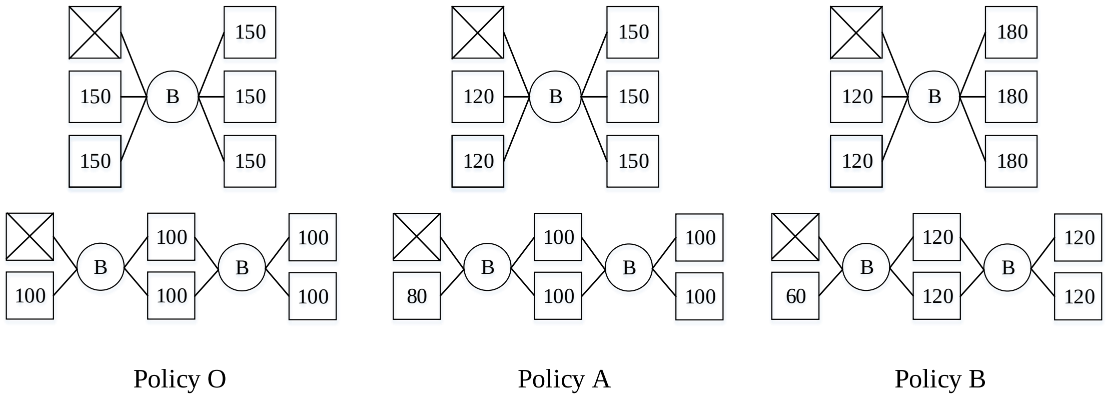


Figure 5.7: Machine cycle times under different policies

Table 5.2 shows the operating cost of the system, including the cost when the system is under normal operation (i.e., when no disruptive event occurs), and the additional cost associated with the resilience metrics under disruptions.

Table 5.2: Operating cost of the system

configuration	(a)	(b)
normal operation cost ($/h)	300	260
additional operating cost associated with PL ($/unit)	8	
additional operating cost associated with $TRT_{0.02}$ ($/h)	5	
additional operating cost associated with $TUT_{0.02}$ ($/h)	5	

Additionally, it is assumed that 10 possible disruptive events may occur, whose durations and frequencies can be found in Table 5.3. Some of the disruptive events (i.e., Events 1, 2, 5, 6, 8) are more likely to occur in configuration (a) because each machine in configuration (a) needs to perform more operations. For simplicity, we assume that the disruption profile is the same in different years, and it only depends on the system configurations, regardless of the types of the machines in the system. Moreover, we assumed that, all of the ten disruptive events can occur on any of the six machines, and with an equal probability. For example, the frequency of event 1 occurring on any specific machine in configuration (a) is 0.002 times/hour.

Table 5.3: Disruptive events

disruptive events	duration (h)	frequency (times/h)	
		configuration (a)	configuration (b)
Event 1	0.25	0.012	0.01
Event 2	0.5	0.025	0.02
Event 3	0.75	0.04	0.04
Event 4	1	0.05	0.05
Event 5	1.25	0.12	0.1
Event 6	1.5	0.12	0.1
Event 7	1.75	0.05	0.05
Event 8	2	0.025	0.02
Event 9	2.25	0.01	0.01
Event 10	2.5	0.01	0.01

Next, we study the resilience metrics and the design optimization problem in these two configurations.

5.6.2 Resilience metrics

In Sections 5.3 and 5.4, we have developed methods to evaluate the three resilience metrics (i.e., production loss PL, throughput recovery time TRT_ϵ, and total underproduction time TUT_ϵ) for manufacturing systems designed with different capabilities. In this section, we numerically investigate how these resilience metrics are affected by different factors, including system configuration, duration and location of the disruption. The total buffer capacity in this section is set to be 20.

5.6.2.1 Transient production rate

First, we assume that the disruption lasts for half an hour (1800 s), and the total buffer capacity is 20. Based on Recursive Procedures 5.2, the evolution of the system production rate under policy X ($X = O, A, B$) can be calculated, as plotted in Fig. 5.8. It shows that, if the disruption occurs in the end-of-line stage, then during the machine downtime, all $PR^X(k)$'s $(X = O, A, B)$ drop to around $(S-1)PR^S/S$, but under policies A and B, the production frequency (i.e., the frequency that the batch of products is produced) is greater than that under policy O. It is because under policy A or B, the cycle time of the last stage is reduced. If the disruption occurs in a non-end-of-line stage, $PR^O(k)$ and $PR^A(k)$ also decrease to around $(S-1)PR^S/S$ during the disruption, but periodically $PR^A(k)$ returns to around PR^S; while under policy B, $PR^B(k)$ is always close to PR^S, but the production frequency is reduced.

Based on the production rates, one can use Equations (5.13), (5.16) and (5.17) to obtain the resilience metrics of the system with policy X (PL^X, TRT^X_ϵ, TUT^X_ϵ; $X = O, A, B$). These resilience metrics will be studied in Sections 5.6.2.2 to 5.6.2.4, where the durations of the disruptions are set as $t_D = 900n$ $(n = 1, ..., 10)$ s, corresponding to durations of the 10 disruptive events in Table 5.3.

5.6.2.2 Production loss

The first resilience metric PL^X $(X = O, A, B)$ is shown in Fig. 5.9. PL^O and PL^A in configuration (a) are larger than that in configuration (b), while PL^B's in these two configurations are similar. Because the system can be perfectly reconfigured, when $t_R < k \le t_D - t_R$, about 1/6 of the required production is lost in both configurations. Moreover, when the duration of the disruption is short, PL^B mainly comes from the reconfiguration periods, and is greater than PL^O and PL^A. However, as the duration of disruption increases, PL^B becomes smaller than PL^O and PL^A, which makes reconfiguration increasingly beneficial.

Figure 5.9 also illustrates that, production losses not only depend on the system configuration and the disruption duration, but also depend on where the disruption occurs. In configuration (b), if the disruption occurs in the middle stage of the system, the PL^O is larger than if the disruption occurs in the first and last stages of the system. The reason is that, the machines in the middle of the systems are more

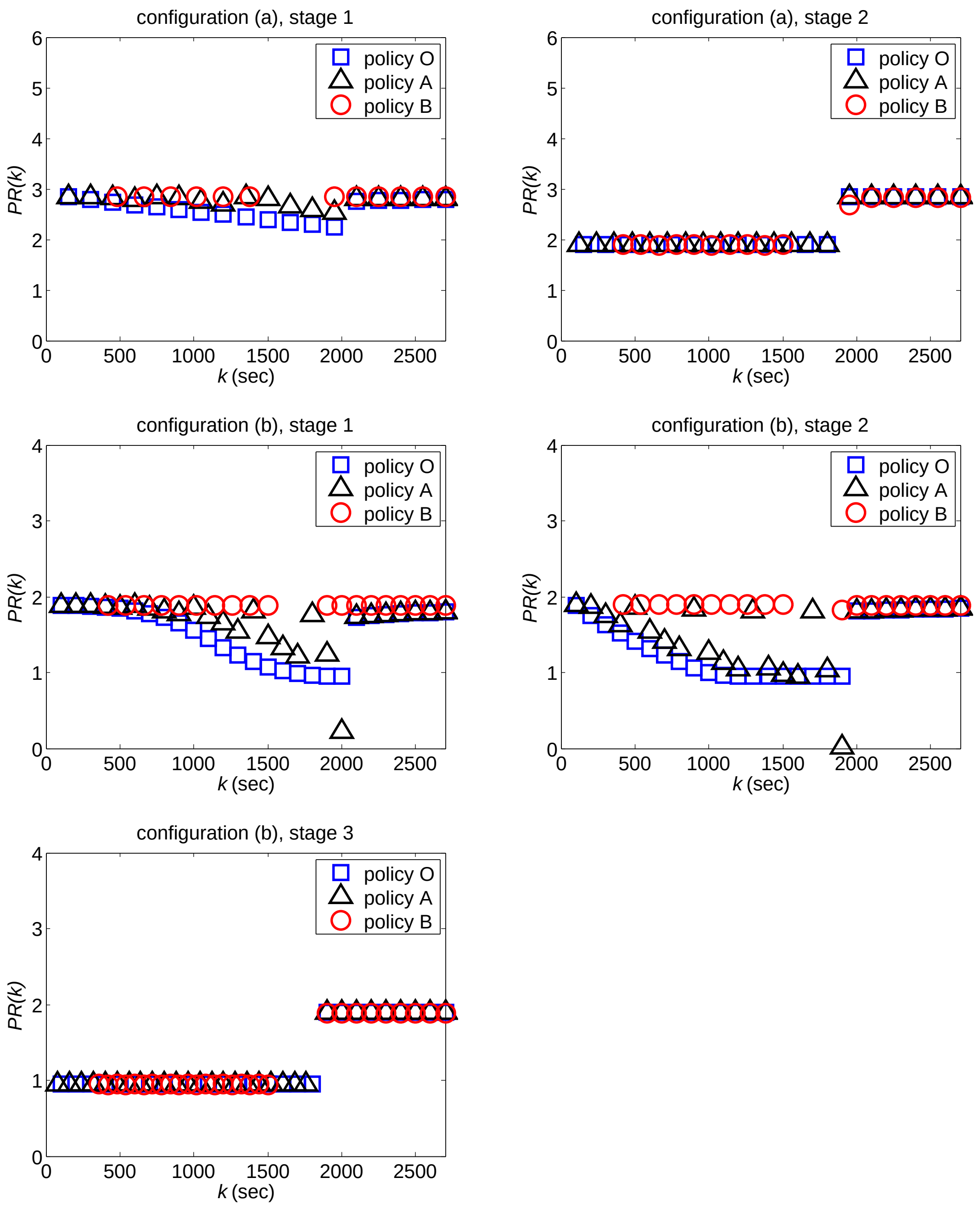


Figure 5.8: Evolution of the production rates ($PR^O(k)$, $PR^A(k)$ and $PR^B(k)$) in both configurations

likely to be starved or blocked than the machines in the other stages, so the downtime on them has a more severe impact on the system throughput.

5.6.2.3 Throughput recovery time

The second resilience metric $TRT^{X}_{0.02}$ $(X = O, A, B)$ is shown in Fig. 5.10. If the disruption occurs in an upstream stage, then its $TRT^{O}_{0.02}$ will increase with the duration of disruption t_D, and it will reach a constant value if t_D is long enough. The reason is that, the shorter the disruption is, the easier its effect can be mitigated by the contents in the buffers. Moreover, $TRT^{O}_{0.02}$ decreases as i_D increases. This is because it will take a longer time for the resume of an upstream machine to propagate to the end-of-line than a downstream machine. Moreover, $TRT^{A}_{0.02}$ is shorter than or equal to $TRT^{O}_{0.02}$, and when the disruption is long enough, $TRT^{B}_{0.02}$ is equal to or close to zero, indicating a quick throughput recovery after the disruption.

5.6.2.4 Total underproduction time

The third resilience metric $TUT^{X}_{0.02}$, $(X = O, A, B)$ is shown in Fig. 5.11. $TUT^{O}_{0.02}$ decreases as i_D increases, indicating that, the downtime propagates to the last stage faster than the recovery does. Moreover, $TUT^{A}_{0.02}$ is smaller than $TUT^{O}_{0.02}$, and when the duration of the disruption is long enough, $TUT^{B}_{0.02}$ is the smallest of the three.

5.6.3 Optimal design

In this case study, we consider 30 different candidate designs, as shown in Table 5.4. These designs have different configurations, consisting of machines with different capabilities, and buffers with different capacities.

Based on the analysis in Section 5.5, one can calculate the investment cost and the operating cost for all these 30 different designs, as plotted in Fig. 5.12. Note that, one may not necessarily use the built-in capabilities when disruption occurs.

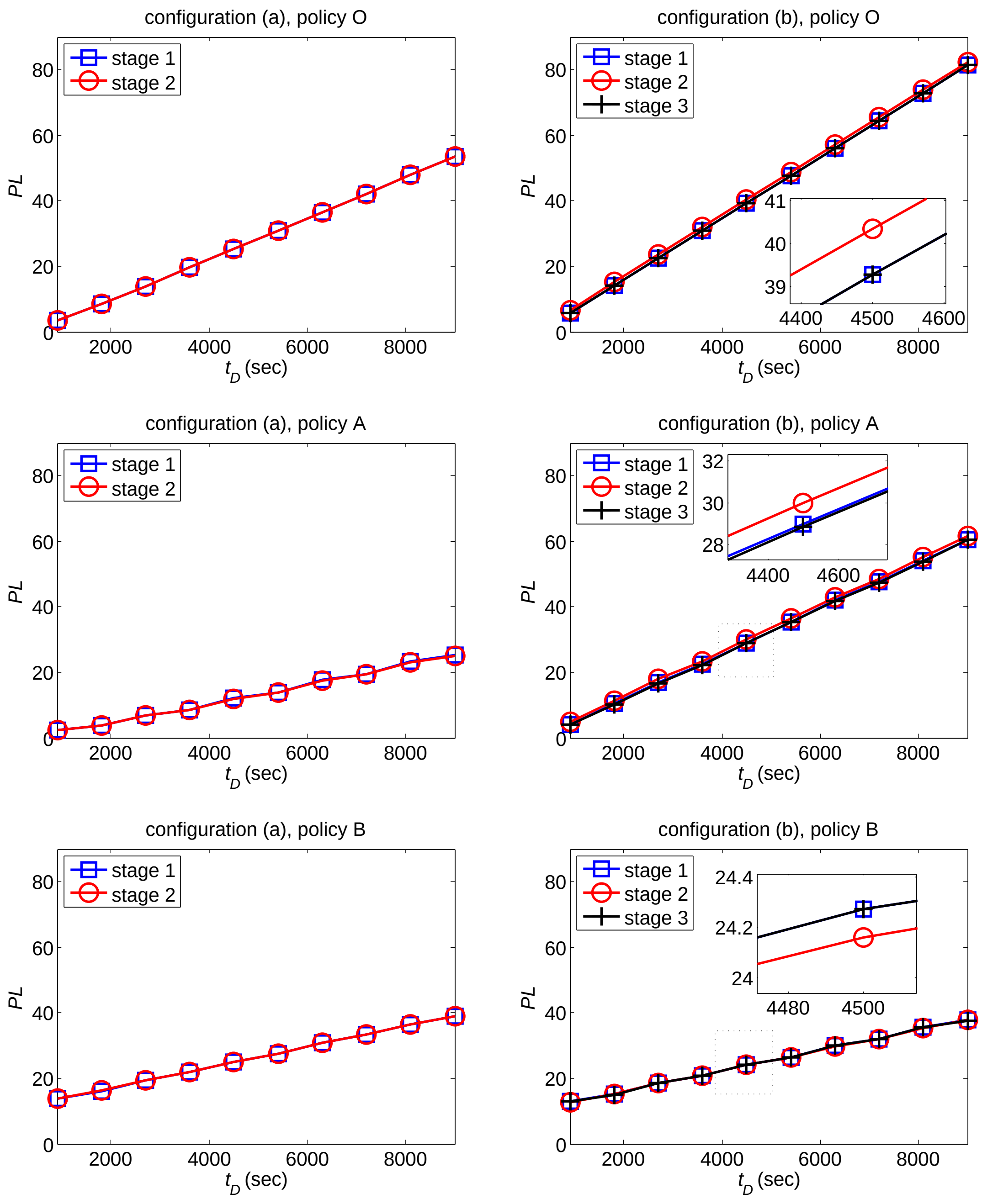


Figure 5.9: Production losses (PL^O, PL^A and PL^B) under different duration of disruptions

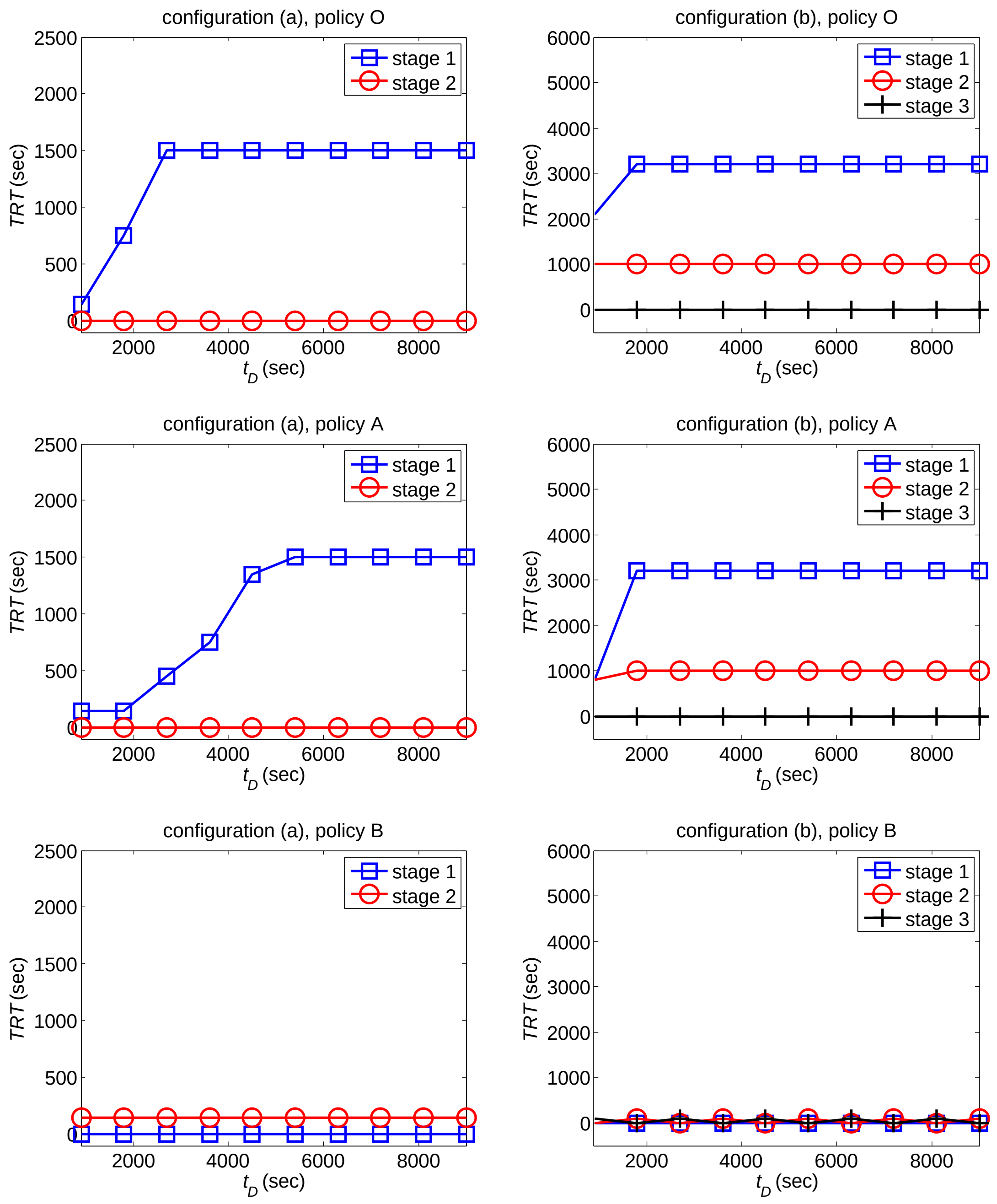


Figure 5.10: Throughput recovery times ($TRT^{O}_{0.02}$, $TRT^{A}_{0.02}$ and $TRT^{B}_{0.02}$) under different duration of disruptions

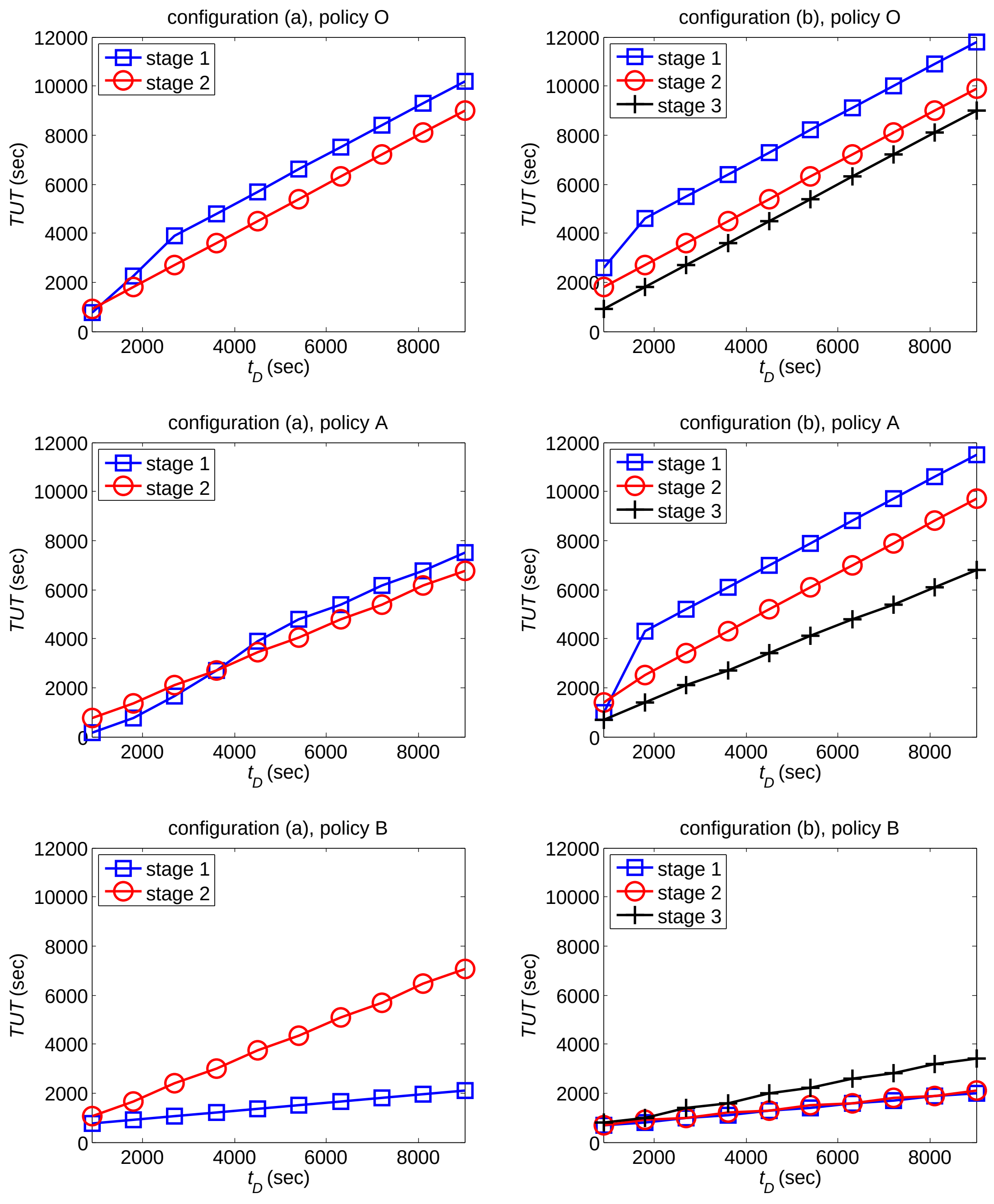


Figure 5.11: Total underproduction times ($TUT^O_{0.02}$, $TUT^A_{0.02}$ and $TUT^B_{0.02}$) under different duration of disruptions

Table 5.4: Indecies for different designs

design no.	configuration	machine type	buffer capacity
1-5	(a)	dedicated machine	10, 20, 30, 40, 50
6-10	(a)	machine with redundancy	10, 20, 30, 40, 50
11-15	(a)	machine with flexibility	10, 20, 30, 40, 50
16-20	(b)	dedicated machine	10, 20, 30, 40, 50
21-25	(b)	machine with redundancy	10, 20, 30, 40, 50
26-30	(b)	machine with flexibility	10, 20, 30, 40, 50

For example, under disruptive event 1 (which takes 15 min), reconfiguration is not desired to be performed because of the two 5-min reconfiguration periods.

For different system design life Y's, the optimal design may be different. However, since the objective function Equation (5.27) has a weighted sum format, all possible optimal solution should be on the convex hull of the Pareto set [103]. It is shown in Fig. 5.12 that all possible optimal designs numbers are 16, 21, 26, 27, and 10. Among these five designs, design no. 16 (configuration (b), dedicated machines, buffer capacity 10) has the smallest investment cost but the largest operating cost, while design no. 10 (configuration (a), machine with redundancy, buffer capacity 50) has the largest investment cost but the smallest operating cost. Design no. 10 is more resilient than design no. 16 with regard to the configuration, the type of machines, and the buffer capacity.

Figure 5.13 shows the total cost of the five designs under different system design lifes. It can be seen that, as the system design life Y increases, a system design with a smaller operating cost is more preferable. Table 5.5 summarizes the relationship between the system design life and the optimal design no. It can be seen that under different Y's, all of the five designs can be optimal, and these five designs are all the possible optimal designs.

Table 5.5: System design life and optimal design

system design life Y	1	2	3, 4, 5, 6	8, 9, 10	> 10
optimal design no.	16	21	26	27	10

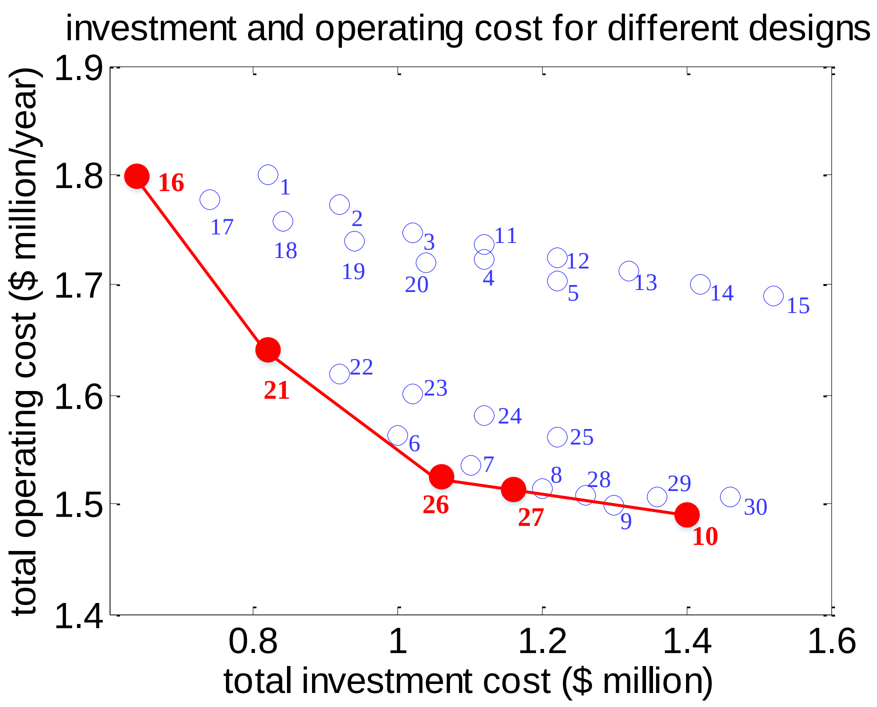


Figure 5.12: Cost analysis for different designs

5.7 Summary

In this chapter, we analyzed three important resilience metrics for manufacturing systems under disruptions: production loss (PL), throughput recovery time (TRT_ϵ), and total underproduction time (TUT_ϵ). By using a Bernoulli reliability model, we performed an exact analysis to calculate the resilience metrics in two-stage systems, and developed an approximation method based on system decomposition to study general inhomogeneous multi-stage manufacturing systems. Numerical studies are conducted to investigate how the system resilience metrics are affected by built-in redundancy or flexibility, and how these resilience metrics can be integrated into the system design stage.

The results show that: (1) system resilience can be improved by building redundancy (e.g., increasing the degree of cycle time reduction, and increasing the buffer capacities) or flexibility (i.e., the option for system reconfiguration) in the system; (2) the system with a more parallel configuration is more resilient; (3) reconfigurability

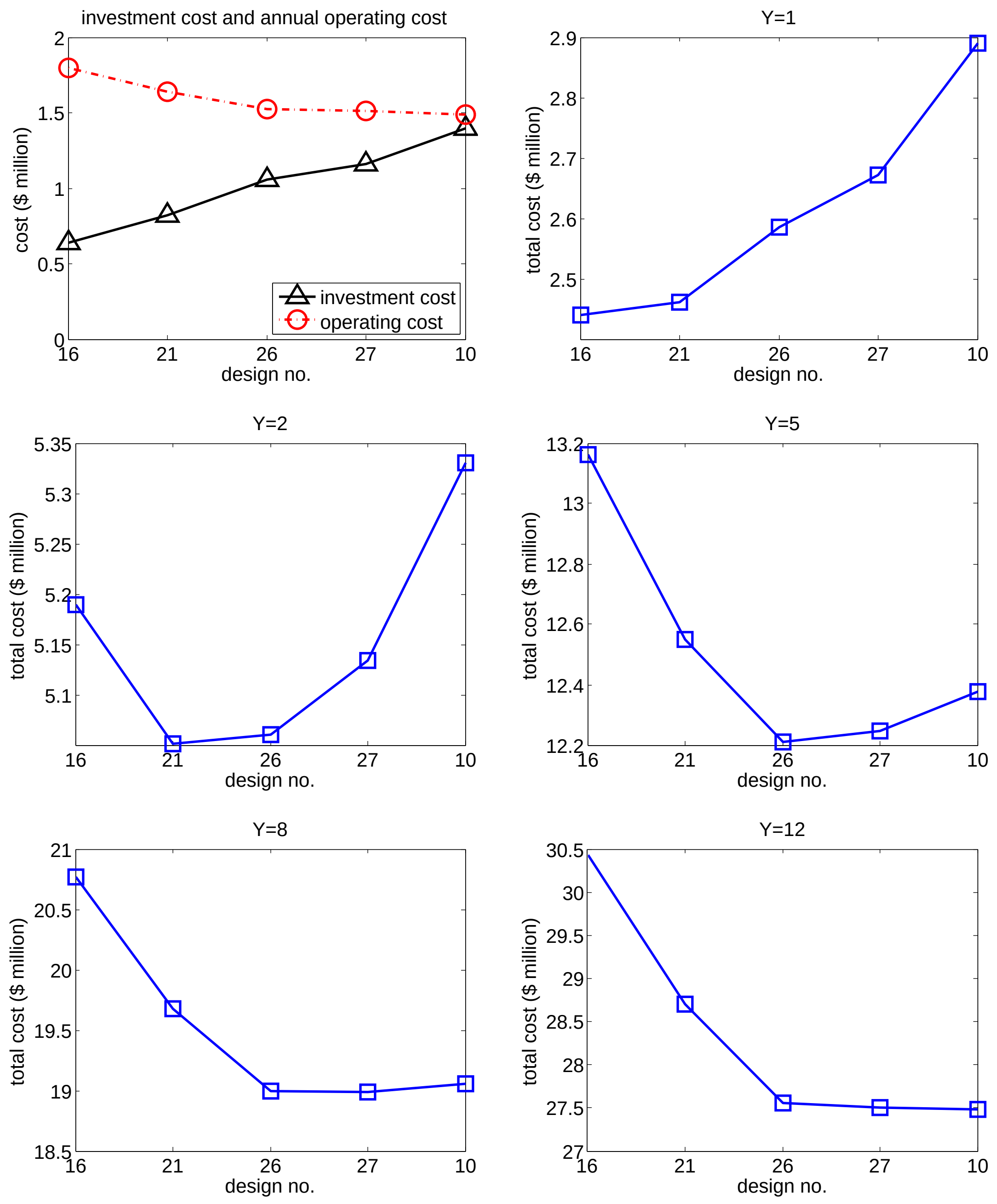


Figure 5.13: Total cost for the five designs

is very advantageous when the duration of disruption is relatively long; and (4) the optimal design of the system depends on the investment costs, the operating costs, as well as the system design life.

The resilience analysis considering the propagation of unexpected disruptive events and the capability of recovery provides a useful guidance for manufacturing system design. It also provides a knowledge base for better planning and controlling of manufacturing systems to mitigate the risks from the disruptions.

The model and methodology that we have developed in this chapter can be extended to study other resilience-related problems. For example, it can be easily implemented to study the resilience metrics for systems that only have the capability for partial reconfiguration, or have both built-in redundancy and flexibility. Another extension is to develop optimal real-time control policies for systems under disruptions. If the system is designed with redundancy or flexibility, then given the real-time buffer levels and machine reliabilities, should these capabilities be used? The third extension is to integrate the degradation of machines into the problem. In this case, how to jointly develop the resilient control policy and maintenance policy, so that the system has the best resilience performance in the long run? These problems will also be investigated in the future.

CHAPTER VI

Conclusions and Future Work

6.1 Conclusions

This research investigates the real-time maintenance policies for complex manufacturing systems by incorporating real-time production information.

The major achievements of this dissertation can be summarized as follows.

In Chapter II, we have proposed the concept of passive maintenance opportunity window (PMOW), which is the time that one machine is starved or blocked due to the occurrence of downtime on the other machines in the system. A deterministic model has been developed to predict real-time PMOW on the bottleneck machine in manufacturing systems with serial and non-serial structures. It is found that, such PMOWs exist if the duration of the machine downtime exceeds a threshold value. Moreover, a dynamic PMOW updating algorithm has been developed when multiple failures occur in the system. Simulation and real plant case studies validate the effectiveness of the proposed algorithm.

In Chapter III, we have introduced the concept of active maintenance opportunity window (AMOW), which is the time that one machine can be actively shut down for preventive maintenance while the production requirement is still satisfied. We have developed a Bernoulli model to analytically estimate AMOWs in a stochastic two-machine-one-buffer production line. A recursive procedure based on system decomposition has been developed to estimate AMOWs in manufacturing systems with complex structures. Also, a heuristic approach has been proposed to estimate AMOWs in balanced serial lines. Numerical case studies demonstrate that, the AMOW-based maintenance policies is able to provide more time for maintenance compared with the baseline policy where maintenance is carried out after the system production requirement has been satisfied.

In Chapter IV, we have incorporated machine degradation into the maintenance decision-making. A system decomposition method has been utilized to evaluate the performance of multi-stage manufacturing systems under the control limit policy. A Markov decision process approach has been developed to investigate the optimal policy in two-machine-one-buffer systems. These two policies are compared by numerical case studies, which show that the optimal policy depends on not only the machine health conditions, but also buffer levels. However, as the buffer capacity increases, the machines become less coupled, and the control limit policy approaches the optimal policy.

In Chapter V, we have analyzed the resilience performance of the system, which is the capability of a system to tolerate and recover from a downtime. We propose production loss (PL), throughput recovery time (TRT), and total underproduction time (TUT) as three measures to analyze the resilience of multi-stage serial-parallel systems. Based on these resilience metrics, a system design optimization problem is formulated. Numerical case studies have been conducted to compare the designs with different levels of built-in redundancy and flexibility. We find that: (1) a system with more parallel machines in each stage is more resilient; (2) system resilience can be improved by built-in redundancy, including the capability of speed increase and buffer capacities; and (3) reconfiguration is advantageous when the downtime is long; and (4) the optimal system design depends on the system resilience capability and its corresponding investment cost, as well as the system design life. These results provide significant managerial insights to the design of resilient manufacturing systems.

6.2 Contributions

The model-based approach in this dissertation aims to provide more efficient and effective maintenance strategies for complex manufacturing systems.

The scientific contributions of the research are summarized as follows.

1. An equivalent pseudo serial line method has been developed to study the propagation of downtime in a manufacturing system with complex structures.

2. System decomposition methods have been developed to study the transient behavior of complex manufacturing systems with Bernoulli machines. Particularly, in this work, such analysis is applied to systems including: (1) serial line, assembly system, disassembly system, and closed-loop system, with one machine in every stage; (2) systems with serial-parallel configurations, with non-degrading machines; and (3) systems with degrading machines.

3. A novel model, integrating the real-time machine states (including its health condition when the machine is up or the remaining maintenance time when the machine is under maintenance) and the real-time buffer level, has been developed to investigate the optimal maintenance policy.

4. Resilience metrics for manufacturing systems are proposed. These resilience metrics can be used as tools for system designers in evaluating the resilience of system with different configurations, and determining the optimal level of redundancy/flexibility to be built in the system.

6.3 Future Work

Some possible directions for future research are summarized below.

1. To develop group maintenance policies by integrating PMOW and AMOW.

AMOW and PMOW depend on each other in different ways: when random failure occurs in the system, the real-time AMOW can still be calculated by estimating the transient production rate under both the downtime and AMOW-based PM; when one machine is down for AMOW-based PM, the real-time AMOWs and PMOWs on other machines can also be updated. Therefore, if resources (e.g., maintenance crews, spare parts) are sufficient, one can consider to perform maintenance on different machines simultaneously. How to effectively perform such group maintenance policies can be further studied.

2. To extend the transient behavior analysis to systems with continuous materials. In Chapters III to V, the system is built by a discrete-time Markov chain. Such model is effective in modeling system that produce discrete manufacturing products such as automobiles. However, in continuous manufacturing systems such as chemical manufacturing systems, this model cannot be directly used, because the buffer levels are not discretely changed. Therefore, for these continuous manufacturing systems, other models (such as continuous flow models) need to be built, and their transient behavior needs to be further investigated.

3. To develop heuristic maintenance policy for large production systems with degrading machines. In Chapter IV, the MDP policy is only investigated in two-machine-one-buffer and three-machine-two-buffer systems. The curse of dimensionality arises if the number of machine increases. Therefore, we need to develop heuristics polices for large systems. The heuristics need to be more computationally effective, and more easily implemented.

4. To further develop dynamical maintenance policies in manufacturing systems by integrating more information. Such information includes the availability of maintenance resources, the skill levels of maintenance crews, etc. These factors will place more constraints as well as challenges on the maintenance decision-making problems.

5. To investigate the joint maintenance and resilient system control strategies.

Maintenance actions can improve system resilience in a long run because it increases the machine reliability. However, in a short time, it is a disruption to the system throughput. When maintenance is performed, resilience control actions can be taken simultaneously. The system maintenance strategies and resilience capabilities need to be jointly considered in the system design stage.

APPENDICES

APPENDIX A

Proof for Chapter II

Proof of Theorem 2.1

Proof. We consider a machine M_i on the downstream of the bottleneck machine in 2.1, i.e., $i = b+1, \ldots, I$, and its corresponding machine M_i^r in the equivalent pseudo serial line (Fig. 2.2). Based on the above analysis, in the equivalent pseudo line, the time to consume is

$$T_i^{cons,r} = \sum_{j=b+1}^{i} N_{j-1}^r(0) \cdot T_b = \sum_{j=b}^{i-1} \left(C_j - N_j(0)\right) T_b \tag{A.1}$$

which is also the time to consume all the empty space between machines M_i and M_b in the original line.

Moreover, the time to resume in the equivalent pseudo serial line can be calculated as

$$T_i^{res,r} = \sum_{j=b+1}^{i} T_j^r = 0 \tag{A.2}$$

which is equal to that in the original line (when the travelling time between buffers is negligible, M_b will run immediately after M_i resumes running).

Moreover, for the critical downtime of machine in Fig. 2.2, we have

$$DT_i^{*,r} = T_i^{cons,r} - T_i^{res,r} = \sum_{j=b+1}^{i} N_{j-1}^r(0) \cdot T_b - \sum_{j=b+1}^{i} T_j^r = \sum_{j=b}^{i-1} \left(C_j - N_j(0)\right) T_b = DT_i^* \tag{A.3}$$

which agrees with the result in Equation (2.1) when machine M_i is on the downstream of M_b. Therefore, the effect of a failure on machine M_i which is on the downstream of M_b can be evaluated in the EPSL. $\square$

Proof of Theorem 2.2

Proof. If $DT_i \leq DT_i^*$, $PMOW_i = \varnothing$ and $|PMOW_i| = 0$, the result simply holds. Otherwise, we prove it by mathmatical induction.

Step 1. Define $j^* = \underset{k=1,\ldots,j}{\arg\min} DT_{i,(k)}^*$. Then $j_1^* = \underset{k=1,\ldots,j_1}{\arg\min} DT_{i,(k)}^* = j_1$, $PMOW_{(j_1)} = \left[T_{i,(j_1)}^{cons}, DT_i + T_{i,(j_1^*)}^{res}\right)$ and $|PMOW_{(j_1)}| = DT_i - DT_{i,(j_1^*)} > 0$.

Step 2. Assume that there exist $j_m \in 1,\ldots,K_i$ and $j_m^* = \underset{k=1,\ldots,j_m}{\arg\min} DT_{i,(k)}^*$, such that $PMOW_{(j_m)} = \left[T_{i,(j_1)}^{cons}, DT_i + T_{i,(j_1^*)}^{res}\right)$ and $|PMOW_{(j_1)}| = DT_i - DT_{i,(j_1)}^*$. Then there are two possible cases.

Case 1 $DT_i \leq DT_{i,(k)}^*$ for all $k = j_{m+1},\ldots,K_i$:

In this case, $K_i^* = j_m^*$ and $PMOW_i = PMOW_{(j_m^*)}$. Proved.

Case 2 $\exists k \in j_{m+1,\ldots,K_i}$, such that $DT_i > DT_{i,(k)}^*$:

In this case, we let $j_{m+1} = \min_{k=j_m+1,\ldots,K_i} \left\{k : DT_i > DT_{i,(k)}^*\right\}$ and $j_{m+1}^* = \underset{k=1,\ldots,j_m}{DT_{i,(k)}^*}$. From Equation (2.9),

$$PMOW_{(j_{m+1})} = PMOW_{(j_m)} \bigcup \left[DT_i + T_{i,(j_m^*)}^{res} - T_{i,(j_m^*)}^{cons} + T_{i,(j_{m+1})}^{cons}, DT_i + T_{i,(j_{m+1})}^{res}\right).$$

Case 2 can be further divided into the following two cases.

Case 2a $DT_{i,(j_{m+1})^*} \geq DT^*_{i,(j_{m+1})^*}$: Then $j^*_{m+1} = j^*_m$ and

$$\left[DT_i + T^{res}_{i,(j^*_m)} - T^{cons}_{i,(j^*_m)} + T^{cons}_{i,(j_{m+1})}, DT_i + T^{res}_{i,(j_{m+1})}\right) = \varnothing,$$

Therefore, $PMOW_{(j_{m+1})} = PMOW_{(j^*_{m+1})}$, and $\left|PMOW_{(j_{m+1})}\right| = DT_i - DT^*_{i,(j_{m+1})}$.

Case 2b $DT_{i,(j_{m+1})^*} < DT^*_{i,(j_{m+1})^*}$: Then $j^*_{m+1} = j_{m+1}$, $T^{res}_{i,(j_{m+1})} > T^{res}_{i,(j^*_m)} - T^{cons}_{i,(j^*_m)} + T^{cons}_{i,(j_{m+1})}$, and

$$\begin{aligned}\left|PMOW_{(j_{m+1})}\right| &= \left|PMOW_{(j_m)}\right| + \left|\left[DT_i + T^{res}_{i,(j^*_m)} - T^{cons}_{i,(j^*_m)} + T^{cons}_{i,(j_{m+1})}, DT_i + T^{res}_{i,(j_{m+1})}\right)\right| \\ &= DT_i - DT^*_{i,(j^*_{m+1})}\end{aligned}$$

These results prove that $PMOW_{(j_{m+1})} \subseteq \left[T^{cons}_{i,(j_1)}, DT_i + T^{res}_{i,(j^*_{m+1})}\right)$ and $\left|PMOW_{(j_{m+1})}\right| = DT_i - DT^*_{i,(j^*_{m+1})}$.

Step 3. Based on Steps 1 and 2,

$$|PMOW_{(j)}| = DT_i - DT^*_{i,(j^*)} \quad \forall j$$

Let $j_M = \max_{k=1,\ldots,K_i} \left\{k : DT_i > DT^*_{i,(k)}\right\}$ and $j^*_M = \underset{k=1,\ldots,j_M}{\arg\min} DT^*_{i,(k)} = K^*_i$. Then $DT^*_{i,j^*_M} = DT^*_i M$, and

$$|PMOW_{(i)}| = |PMOW_{(j_M)}| = DT_i - DT^*_{i,(j^*_M)} = DT_i - DT^*_{i,(K^*_i)} = DT_i - DT^*_i$$

Summarizing the results from all cases, we have $|PMOW_i| = \max(0, DT_i - DT^*_i)$.

□

Proof of Proposition 2.1

Proof. If $|PMOW_i| = 0$, there is no downtime on M_b, and the proposition is true.

Otherwise, $|PMOW_i| = DT_i - DT^*_i > 0$. We consider a virtual line $l_{i,0}$, where

$T^{cons}_{i,(0)} = T_0$ and $T^{res}_{i,(0)} = T_0 - DT^*_i$, such that $DT^{res}_{i,(0)} = DT^*_i$ and $PMOW_{(0)} = [T_0, T_0 + |PMOW_i|)$.

Then based on Theorem 2.2, $PMOW_{(k+1)} = PMOW_{(k)}$ if $DT^*_{i,(k+1)} \geq DT^*_{i,(k)}$. Since $DT^*_{i,(0)} = DT^*_i = \min\limits_{k=1,\ldots,K_i} DT^*_{i,(k)}$, $PMOW_{(k)} = PMOW_{(0)}$ for all $k = 1, \ldots, K_i$.

Therefore, there are no additional idle durations on M_b in lines $l_{i,(1)}, \ldots, l_{i,(K_i)}$. $\square$

APPENDIX B

Proof for Chapter III

Proof of Lemma 3.2

Proof. We consider a process starting from state n at time 0 and arriving at state m at time k_m. Denoting $PL_{mn}(k_m)$ as the expected production losses during time interval $[0, k_m]$. We also assume that during time interval $[0, k_m]$, the system has entered state m l times, i.e., at time $k_{m_1}, k_{m_2}, \ldots, k_{m_l}$, respectively. Then,

$$\begin{aligned} PL_{mn}(k_m) &= PL_{mn}(k_m) - PL_{mn}(k_{m-1}) + \cdots + PL_{mn}(k_{m_2}) - PL_{mn}(k_{m_1}) + PL_{mn}(k_{m_1}) \\ &= (l-1)PL_{mn} + PL_{mn}(k_{m_1}) \end{aligned} \tag{B.1}$$

where PL_{mm} is the expected production losses between two successive events of entering into state m, and from Lemma 3.1, $PL_{mm} = (T_{mm} - \pi_0 V_{mm})p_2 = 0$. Therefore,

$$PL_{mn}(k_m) = PL_{mn}(k_{m_1}) = (1-\pi_0)V_{mn}p_2 - (S_{mn} - T_{mn})p_2 = (T_{mn} - \pi_0 V_{mn})p_2 \tag{B.2}$$

where $(T_{mn} - \pi_0 V_{mn})p_2$ is the actual units produced and $T_{mn}p_2$ is the required number of production. □

Proof of Theorem 3.1

Proof. By substituting Equations (3.6) and (3.7) into Equation (B.2) , the production losses form state n to state m, PL_{mn}, can be obtained as

If $p_1 \neq p_2$:

$$\begin{aligned} PL_{mn} &= \frac{(m-n)(1-p^+/p^-) - \left((p^+/p^-)^{C+1-m} - (p^+/p^-)^{C+1-n}\right)}{(1-p^+/p^-)(1-p_1/p_2(p^+/p^-)^C)} \\ &= \frac{f(n)-f(m)}{(1-p^+/p^-)(1-p_1/p_2(p^+/p^-)^C)} \end{aligned} \tag{B.3}$$

where $f(x) := -x(1-p^+/p^-) + (p^+/p^-)^{C+1-x} + j(1-p^+/p^-) - (p^+/p^-)^{C+1-j}$. Then,

$$f'(x) = -(1-p^+/p^-) - \log(p^+/p^-)(p^+/p^-)^{C+1-x}, \tag{B.4}$$

and

$$f''(x) = \log^2(p^+/p^-)(p^+/p^-)^{C+1-x} > 0. \tag{B.5}$$

which indicates that $f'(x)$ is increasing in x when $x \in [0, C]$.

Let $h(x) = 1 - x + x\log x$ $(x > 0)$, then $h''(x) = 1/x > 0$. Thus $h(x)$ is convex and $h_{\min}(x) = h(1) = 0$ ($h(x)$ reaches its minimum when $h'(x) = \log x = 0$). Since $p^+/p^- \neq 1$, we have

$$f'(C) = -h(p^+/p^-) < 0. \tag{B.6}$$

Based on Equations (B.5) and (B.6), $f'(x) < 0$ when $x \in [0, C]$, and thus $f(x)$ is monotonically decreasing.

From Equation (3.9a),

$$PL_n = \sum_{m=0}^{C} PL_{mn}\pi_m = \frac{\sum_{m=0}^{C} f(n)\pi_m}{p_2(1-p^+/p^-)(1-p_1/p_2(p^+/p^-)^C)}. \tag{B.7}$$

PL_n is a convex and decreasing function of n because $f'(n) < 0$ and $f''(n) > 0$.

Similarly, if $p_1 = p_2 = p$:

$$PL_{mn} = \frac{(C-n)(C+1-n)-(C-m)(C+1-m)}{2(C+1-p)} = \frac{g(n)-g(m)}{2(C+1-p)} \tag{B.8}$$

where $g(x) := (C-n)(C+1-n)$.

When $0 \leq x \leq C$, $g'(x) := 2x-(2C+1) < 0$, and $g''(x) = 2x \geq 0$. Therefore,

$$PL_n = \sum_{m=0}^{C} PL_{mn}\pi_m = \frac{\sum_{m=0}^{C} g(n)\pi_m - \sum_{m=0}^{C} g(m)\pi_m}{2(C+1-p)} \tag{B.9}$$

is convex and decreasing.

Then we prove $-C < PL_n < C$. We only need to prove that $PL_0 < C$ and $PL_C > -C$.

If $p_1 \neq p_2$:

$$PL_0 = \frac{p_1(1+(p^+/p^-)^{C+1}(1-(p^+/p^-)^C)/(1-p^+/p^-)-2C(p^+/p^-)^C)}{p_2(1-p_1/p_2(p^+/p^-)^C)^2}$$

$PL_0 < C \Leftrightarrow$

$$\left(1+(p^+/p^-)^{C+1}\right)\left(1-(p^+/p^-)^C\right)/(1-p^+/p^-) < C(p_2/p_1+p_1/p_2(p^+/p^-)^{2C}) \tag{B.10}$$

$$\begin{aligned}\text{LHS of (B.10)} &= \sum_{n=0}^{C-1}\left((p^+/p^-)^n + (p^+/p^-)^{2C-n}\right)\\ &< \sum_{n=0}^{C-1}\left((p^+/p^-)^0 + (p^+/p^-)^{2C}\right)\\ &< C(p_2/p_1 + p_1/p_2(p^+/p^-)^{2C}) = \text{RHS of (B.10)}\end{aligned}$$

Moreover,

$$PL_C = \frac{p_1(1 + (p^+/p^-)(1 - (p^+/p^-)^C)/(1 - p^+/p^-) - (p_1C + p^+)(p^+/p^-)^C) + p^+ - p_2C}{p_2(1 - p_1/p_2(p^+/p^-)^C)^2}$$

$$PL_C > -C \Leftrightarrow$$

$$(1 + (1 - p_2)/(1 - p_1)) \left(1 - (p^+/p^-)^C\right)/(1-p^+/p^-)+p_1/p_2C(p^+/p^-)^{2C} > 3C(p^+/p^-)^C \tag{B.11}$$

$$\begin{aligned}\text{LHS of (B.11)} &= \sum_{n=0}^{C-1} \left((p^+/p^-)^n + (1 - p_2)/(1 - p_1)(p^+/p^-)^{C-n} + p_1/p_2(p^+/p^-)^{2C}\right) \\ &> \sum_{n=0}^{C-1} \left(3(p^+/p^-)^C\right) = \text{RHS of (B.11)}.\end{aligned}$$

Therefore, $PL_0 < C$ and $PL_C > -C$ are proved when $p_1 \neq p_2$.

If $p_1 = p_2 = p$:

$$PL_0 = \frac{C(C + 1)(2C + 1)}{6(C + 1 - p)^2} < \frac{C(C + 1)(2C + 1)}{6C^2} = \frac{C}{3} + \frac{1}{2} + \frac{1}{6C} \leq \frac{C}{3} + \frac{C}{2} + \frac{C^2}{6C} = C \tag{B.12}$$

and

$$PL_C = \frac{-C^3 - 3(1 - p)C^2 + (3p - 2)C}{6(C + 1 - p)^2} > -C \Leftrightarrow$$

$$-C^2 - 3(1 - p)C + (3p - 2) > -6(C + 1 - p)^2 \tag{B.13}$$

Let $\phi(p) := -C^2-3(1-p)C+(3p-2)+6(C+1-p)^2$. Then $\phi'(p) = 12p-9(C+1) < 0$, and thus $\phi(p)$ is decreasing in p. Since $0 < p < 1$, $\phi_{\min}(p) > \phi(1) = 5C^2 + 1 > 0$, which proves Inequality (B.13).

□

Proof of Theorem 3.2

Proof. Consider two systems A and B where the prior-PM buffer levels satisfying $N_A(0) < N_B(0)$. The lower and upper bounds of buffer level for system A are $N^L(N_A(0))$ and $N^U(N_A(0))$, respectively. Then based on the definition of AMOW,

$$PL\left(N_A(0), N^L(N_A(0))\right) \leq 0 < PL\left(N_A(0), N^L(N_A(0)) - 1\right) \tag{B.14}$$

and

$$PL\left(N_A(0), N^U(N_A(0))\right) \leq 0 < PL\left(N_A(0), N^U(N_A(0)) + 1\right) \tag{B.15}$$

Let N_B^1 be the buffer level if M_1 in system B is stopped for $AMOW_1(N_A(0))$. Then

$$N_B^1 = N^L(N_A(0)) - N_A(0) + N_B(0) > N^L(N_A(0)) \tag{B.16}$$

and from Equations (3.2), (3.5) and (3.10),

$$\begin{aligned} & PL\left(N_B(0), N_B^1\right) - PL\left(N_A(0), N^L(N_A(0))\right) \\ = & \begin{cases} PL_{N_B^1} - PL_{N^L(N_A(0))} & \text{if } N^L(N_A(0)) \geq 0 \\ PL_{N_B^1} - PL_0 + N^L(N_A(0)) & \text{if } N^L(N_A(0)) < 0 \leq N_B^1 \\ N^L(N_A(0)) - N_B^1 & \text{if } N_B^1 < 0 \end{cases} \end{aligned} \tag{B.17}$$

By Theorem 3.1, PL_m is decreasing in m. It can be concluded from Equation (B.17) that $PL(N_B(0), N_B^1) < PL(N_A(0)), N^L(N_A(0))) \leq 0$. Therefore, the lower bound of the post-PM buffer level in system B satisfies $N^L(N_B(0)) \leq N_B^1$, and thus $AMOW_1(N_B(0)) \geq AMOW_1(N_A(0))$.

Similarly, let N_B^2 be the buffer level if M_2 in system B is stopped for $AMOW_2(N_A(0))$. Then

$$N_B^2 = N^U(N_A(0)) - N_A(0) + N_B(0) > N^U(N_A(0)) \tag{B.18}$$

and

$$PL(N_B(0), N_B^2) - PL(N_A(0)), N^U(N_A(0))) = PL_{\min(N_B^2, C)} - PL_{\min(N^U(N_A(0)), C)} \leq 0. \tag{B.19}$$

Therefore, $N^U(N_B(0)) \geq N_B^2$, and $AMOW_2(N_B(0)) \geq AMOW_2(N_A(0))$.

Specially, when $N^U(N_A(0)) \geq C$, $PL_{\min(N_B^2, C)} - PL_{\min(N^U(N_A(0)), C)} = 0$. In this case

$$PL(N_B(0), N_B^2 + 1) = PL(N_A(0)), N^U(N_A(0)) + 1) > 0. \tag{B.20}$$

which implies that $N^U(N_B(0)) < N_B^2 + 1$.

Therefore, when $N^U(N_A(0)) \geq C$, $N^U(N_A(0)) = N_B^2$ and $AMOW_2(N_B(0)) = AMOW_2(N_A(0))$. $\square$

Proof of Proposition 3.1

Proof. Since $p_1 = p_2 = p \in (0, 1)$, we have

$$N^S = \frac{C(C+1)}{2(C+1-p)} > \frac{C}{2}$$

and then $\lceil N^S \rceil \geq \lceil (C+1)/2 \rceil$.

When $n = (C+1)/2$, the numerator of Equation (3.9b) becomes

$$-\frac{(C^3 - C)}{4} - 3(1-p)(C^2 + C + \frac{3}{4}) < 0$$

which indicates that $PL_{\lceil (C+1)/2 \rceil} \leq PL_{(C+1)/2} < 0$.

Therefore, $N^L \leq \lceil (C+1)/2 \rceil \leq \lceil N^S \rceil$. $\square$

Proof of Proposition 3.2

Proof. If $n < 0$,

$$PL(N(0), n) = -\pi_0 N(0) - (1-\pi_0)n + PL_0 \geq -\pi_0 C + 1 - \pi_0 + PL_0. \tag{B.21}$$

Then we prove $-\pi_0 C + 1 - \pi_0 + PL_0 > 0$ under the following two cases:

Case 1 $p_1 = p_2 = p$:

$$-\pi_0 C + 1 - \pi_0 + PL_0 = \frac{Cp}{C+1-p} + PL_0 > 0. \tag{B.22}$$

Case 2 $p_1 > p_2$:

$$\begin{aligned} -\pi_0 C + 1 - \pi_0 + PL_0 > -\pi_0 C + 1 - \pi_0 =& \frac{1 - p_1/p_2(p^+/p^-)^C - (C+1)(1-p_1/p_2)}{1 - p_1/p_2(p^+/p^-)^C} \\ :=& \frac{\theta(C)}{1 - p_1/p_2(p^+/p^-)^C} \end{aligned} \tag{B.23}$$

Since $p_1/p_2 > 1$, we have $p^+/p^- = p_1/p_2 \cdot (1-p_2)/(1-p_1) > 1$, and then $1 - p_1/p_2(p^+/p^-)^C < 0$. For $\theta(C)$, by taking second derivative on the capacity C, we have

$$\theta''(C) = p_1/p_2 \cdot (p^+/p^-)^C \cdot \log^2(p^+/p^-) > 0 \tag{B.24}$$

So $\theta(C)$ is convex on C.

Moreover, $\theta(1) = 2p_1/p_2 - (1 + p_1/p_2 \cdot (p^+/p^-)) < 2p_1/p_2 - (1 + p_1^2/p_2^2) < 0$, and $\lim_{C\to\infty} \theta(C) < 0$. By Jensen's inequality, $\theta(C) < 0$ for all $C \geq 1$, which shows that the nominator is also negative, resulting $-\pi_0 C + 1 - \pi_0 + PL_0 > 0$. □

Proof of Theorem 3.3

Proof. Assume that we have two I-machine-$(I-1)$-buffer lines A and B where the corresponding machines always have the same "up" or "down" states. We prove that if $\mathbf{N}_B(k) \geq \mathbf{N}_A(k)$, then $\mathbf{N}_B(k+1) \geq \mathbf{N}_A(k+1)$. If this statement is not correct, then there exists one sample path (based on the up/down states of machines), and a buffer B_i $(i = 1, \ldots, I)$ such that $N_{B,i}(k+1) < N_{A,i}(k+1)$.

Note that, in one cycle, the buffer level goes up or down by at most one unit. Therefore, $N_{B,i}(k+1) \geq N_{A,i}(k+1)$ and $N_{B,i}(k+1) < N_{A,i}(k+1)$ implies one of the following two situations occurs:

Case 1 $N_{A,i}(k+1) = N_{A,i}(k) + 1$: In this case, at time k, machine M_i is up, and machine M_{i+1} is either down or blocked. If M_{i+1} is down in system A, then it is also down in system B; if M_{i+1} is blocked in system A, it will also be blocked in system B, since $N_{B,j}(k+1) \geq N_{A,j}(k+1)$ for all the downstream buffer B_j $(j = i+1, \ldots, I-1)$. In both situations, $N_{B,i}(k+1) = \min(C_i, N_{B,i}(k)+1) \geq N_{A,i}(k+1)$. Contradiction.

Case 2 $N_{B,i}(k+1) = N_{B,i}(k) - 1$: This indicates that in system B, M_i is down, and M_{i+1} is up and unblocked. However, since $N_{B,j}(k+1) \geq N_{A,j}(k+1)$ for all the downstream buffer B_j $(j = i+1, \ldots, I-1)$, M_{i+1} is unblocked in system B implies that it is also unblocked in system A. Therefore, $N_{A,i}(k+1) = \max(0, N_{A,i}(k) - 1) \leq N_{B,i}(k+1)$. Contradiction.

Summarizing the two cases, if $\mathbf{N}_B(k) \geq \mathbf{N}_A(k)$, then $\mathbf{N}_B(k+1) \geq \mathbf{N}_A(k+1)$. Therefore, if M_i $(i = 1, \ldots, I)$ is down for the time in system A and B, the transient production rate of system B will always be higher than that of system A, indicating that $AMOW_i(\mathbf{N}_B(0)) \geq AMOW_i(\mathbf{N}_A(0))$. $\square$

APPENDIX C

Proof for Chapter IV

Proof of Theorem 4.1

Proof. We prove it by contradiction.

(A) When $d_0 \leq d^*$:

If $d_1 \neq d^*$, then from Equation (4.4), we have

$$\begin{aligned}
&\sum_{d'=d_0}^{d_1-1} 1/q - PR_{d^*}\left(\sum_{d'=d_0}^{d_1-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d_1-1)}\right) \\
&> \sum_{d'=d_0}^{d^*-1} 1/q - PR_{d^*}\left(\sum_{d'=d_0}^{d^*-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d^*-1)}\right) \\
\Longrightarrow &\sum_{d'=1}^{d_1-1} 1/q - PR_{d^*}\left(\sum_{d'=1}^{d_1-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d_1-1)}\right) \\
&> \sum_{d'=1}^{d^*-1} 1/q - PR_{d^*}\left(\sum_{d'=1}^{d^*-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d^*-1)}\right) = 0
\end{aligned}$$

Therefore,

$$PR_{d_1} = \frac{\sum_{d'=1}^{d_1-1} 1/q}{\sum_{d'=1}^{d^*-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d_1-1)}} > PR_{d^*}$$

which contradicts with Equation (4.3).

(B) When $d_0 > d^*$:

Assume $d_1 \neq d_0$. Then,

$$\sum_{d'=d_0}^{d_1-1} 1/q - PR_{d^*}\left(\sum_{d'=d_0}^{d_1-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d_1-1)}\right) > -PR_{d^*} \cdot T^{(d_0-1)}$$

We have

$$PR_{d^*} < \frac{\sum\limits_{d'=d_0}^{d_1-1} 1/q}{\sum\limits_{d'=d_0}^{d_1-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d_1-1)} - T^{(d_0-1)}} = \frac{\sum\limits_{d'=d_0}^{d_1-1} 1/q}{\sum\limits_{d'=d_0}^{d_1-1} \left(1/(\overline{f^{(d')}} \cdot q) + T^{(d')} - T^{(d'-1)}\right)}$$

$$< \frac{\sum\limits_{d'=d^*}^{d_1-d_0+d^*-1} 1/q}{\sum\limits_{d'=d^*}^{d_1-d_0+d^*-1} \left(1/(\overline{f^{(d')}} \cdot q) + T^{(d')} - T^{(d'-1)}\right)} \quad \text{(by Conditions 5.1 and 5.2)}$$

Therefore,

$$PR_{d^*} < \frac{\sum\limits_{d'=1}^{d^*-1} 1/q + \sum\limits_{d'=d^*}^{d_1-d_0+d^*-1} 1/q}{\sum\limits_{d'=1}^{d^*-1} 1/(\overline{f^{(d')}} \cdot q) + T^{(d^*-1)} + \sum\limits_{d'=d^*}^{d_1-d_0+d^*-1} \left(1/(\overline{f^{(d')}} \cdot q) + T^{(d')} - T^{(d'-1)}\right)}$$

$$= \frac{\sum\limits_{d'=1}^{d_1-d_0+d^*-1} 1/q}{\sum\limits_{d'=1}^{d_1-d_0+d^*-1} 1/(\overline{f^{(d')}} \cdot q) - T^{(d_1-d_0+d^*-1)}} = PR_{d_1-d_0+d^*}$$

which contradicts with Equation (4.3).

Based on (A) and (B), $d_1 = d_0$ when $d_0 \leq d^*$; and $d_1 = d^*$ when $d_0 > d^*$.

□

Proof of Theorem 4.2

Proof. We use induction to prove these properties.

Step 1. When $t = 0$, $V_0(S_1, S_2, N) = 0$ for any (S_1, S_2, N). All five properties hold.

Step 2. Assume that properties (1) to (5) hold for $t = 0, ..., t'$. Then we prove that these five properties also hold for $t = t' + 1$.

(A) Proof of Property (1)

(1) First, we prove that $\phi_{t'}(S_1, S_2, N^+) \geq \phi_{t'}(S_1, S_2, N)$ for $0 \leq N \leq C - 1$, and $\phi_{t'}(S_1, S_2, N^+) - \phi_{t'}(S_1, S_2, N) \leq 1$ for $0 \leq N < C - 1$.

$$\begin{aligned}
\phi_{t'}(S_1, S_2, N) =& \overline{f_1^{(S_1)}}\Big[\overline{q_1}V_{t'}(S_1, S_2, N^+) + q_1 V_{t'}(S_1^+, S_2, N^+)\Big] + f_1^{(S_1)}V_{t'}(S_1, S_2, N) \\
\leq& \overline{f_1^{(S_1)}}\Big[\overline{q_1}V_{t'-1}(S_1, S_2, (N^+)^+) + q_1 V_{t'}(S_1^+, S_2, (N^+)^+)\Big] \\
&+ f_1^{(S_1)}V_{t'}(S_1, S_2, N^+) \\
=& \phi_{t'}(S_1, S_2, N^+)
\end{aligned}$$

$$\begin{aligned}
\phi_{t'}(S_1, S_2, N) \geq& \overline{f_1^{(S_1)}}\Big[\overline{q_1}\big(V_{t'}(S_1, S_2, (N^+)^+) - 1\big) + q_1\big(V_{t'}(S_1^+, S_2, (N^+)^+) - 1\big)\Big] \\
&+ f_1^{(S_1)}\big(V_{t'}(S_1, S_2, N^+) - 1\big) \\
=& \phi_{t'}(S_1, S_2, N^+) - 1
\end{aligned}$$

and when $N = C - 1$:

$$\begin{aligned}
\phi_{t'}(S_1, S_2, C - 1) =& \overline{f_1^{(S_1)}}\Big[\overline{q_1}V_{t'}(S_1, S_2, C) + q_1 V_{t'}(S_1^+, S_2, C)\Big] \\
&+ f_1^{(S_1)}V_{t'}(S_1, S_2, C - 1) \\
\leq& \overline{f_1^{(S_1)}}V_{t'}(S_1, S_2, C) + f_1^{(S_1)}V_{t'}(S_1, S_2, C) = \phi_{t'}(S_1, S_2, C)
\end{aligned}$$

(2) Then, we prove $0 \leq V^b_{t'+1}(S_1, S_2, N^+) - V^b_{t'+1}(S_1, S_2, N) \leq 1$.

Case 1 $S_1 \leq D_1$:

$$V^b_{t'+1}(S_1, S_2, N) = \max \begin{cases} V_{t'}(S_1^{++}, S_2, N) \\ V_{t'}(S_1, S_2, N) \\ \phi_{t'}(S_1, S_2, N) \end{cases}$$

$$\leq \max \begin{cases} V_{t'}(S_1^{++}, S_2, N^+) \\ V_{t'}(S_1, S_2, N^+) \\ \phi_{t'}(S_1, S_2, N^+) \end{cases} = V^b_{t'+1}(S_1, S_2, N^+)$$

$$V^b_{t'+1}(S_1, S_2, N) \geq \max \begin{cases} V_{t'}(S_1^{++}, S_2, N^+) - 1 \\ V_{t'}(S_1, S_2, N^+) - 1 \\ \phi_{t'}(S_1, S_2, N^+) - 1 \end{cases} = V^b_{t'+1}(S_1, S_2, N^+) - 1$$

where the "$\geq$" holds when $N = C-1$ because $V^b_{t'+1}(S_1, S_2, C-1) \geq V_{t'}(S_1, S_2, C-1) \geq V_{t'}(S_1, S_2, C) - 1 = \phi_{t'}(S_1, S_2, C) - 1$.

Case 2 $S_1 > D_1$:

$$V^b_{t'+1}(S_1, S_2, N) = V_{t'}(S_1^+, S_2, N) \leq V_{t'}(S_1^+, S_2, N^+) = V^b_{t'+1}(S_1, S_2, N^+)$$

$$V^b_{t'+1}(S_1, S_2, N) \geq V_{t'}(S_1^+, S_2, N^+) - 1 = V^b_{t'+1}(S_1, S_2, N^+) - 1$$

(3) Next, we prove $0 \leq V^a_{t'+1}(S_1, S_2, N^+) - V^a_{t'+1}(S_1, S_2, N) \leq 1$.

Case 1 $S_1 \leq D_1$:

$$\begin{aligned}
V^a_{t'+1}(S_1,S_2,N) &= 1 + \max \begin{cases} \overline{q_2} V_{t'}(S_1^{++}, S_2, N^-) + q_2 V_{t'}(S_1^{++}, S_2^+, N^-) \\ \overline{q_2} V_{t'}(S_1, S_2, N^-) + q_2 V_{t'}(S_1, S_2^+, N^-) \\ \overline{q_2} \phi_{t'}(S_1, S_2, N^-) + q_2 \phi_{t'}(S_1, S_2^+, N^-) \end{cases} \\
&\leq 1 + \max \begin{cases} \overline{q_2} V_{t'}(S_1^{++}, S_2, N) + q_2 V_{t'}(S_1^{++}, S_2^+, N) \\ \overline{q_2} V_{t'}(S_1, S_2, N) + q_2 V_{t'}(S_1, S_2^+, N) \\ \overline{q_2} \phi_{t'}(S_1, S_2, N) + q_2 \phi_{t'}(S_1, S_2^+, N) \end{cases} \\
&= V^a_{t'+1}(S_1, S_2, N^+)
\end{aligned}$$

$$\begin{aligned}
V^a_{t'+1}(S_1,S_2,N) &\geq 1 + \max \begin{cases} \overline{q_2}\big(V_{t'}(S_1^{++}, S_2, N) - 1\big) + q_2\big(V_{t'}(S_1^{++}, S_2^+, N) - 1\big) \\ \overline{q_2}\big(V_{t'}(S_1, S_2, N) - 1\big) + q_2\big(V_{t'}(S_1, S_2^+, N) - 1\big) \\ \overline{q_2}\big(\phi_{t'}(S_1, S_2, N) - 1\big) + q_2\big(\phi_{t'}(S_1, S_2^+, N) - 1\big) \end{cases} \\
&= V^a_{t'+1}(S_1, S_2, N^+) - 1
\end{aligned}$$

Case 2 $S_1 > D_1$:

$$\begin{aligned}
V^a_{t'+1}(S_1,S_2,N) &= 1 + \overline{q_2} V_{t'}(S_1^+, S_2, N^-) + q_2 V_{t'}(S_1^+, S_2^+, N^-) \\
&\leq 1 + \overline{q_2} V_{t'}(S_1^+, S_2, N) + q_2 V_{t'}(S_1^+, S_2^+, N) = V^a_{t'+1}(S_1, S_2, N^+)
\end{aligned}$$

$$\begin{aligned}
V^a_{t'+1}(S_1,S_2,N) &\geq 1 + \overline{q_2}\big(V_{t'}(S_1^+, S_2, N) - 1\big) + q_2\big(V_{t'}(S_1^+, S_2^+, N) - 1\big) \\
&= V^a_{t'+1}(S_1, S_2, N^+) - 1
\end{aligned}$$

(4) Finally, from the definition of $V_t(S_1, S_2, N)$ (Equation (4.15)), $0 \leq V_{t'+1}(S_1, S_2, N^+) - V_{t'+1}(S_1, S_2, N) \leq 1$.

(B) Proof of Properties (2) & (3)

Case 1 $S_1 \leq D_1$:

For $N < C$:

$$\begin{aligned}
\phi_{t'}(S_1^+, S_2, N) =& \overline{f_1^{(S_1^+)}}\Big[\overline{q_1} V_{t'}(S_1^+, S_2, N^+) + q_1 V_{t'}((S_1^+)^+, S_2, N^+)\Big] \\
& + f_1^{(S_1^+)} V_{t'}(S_1^+, S_2, N) \\
\leq& \overline{f_1^{(S_1^+)}} V_{t'}(S_1^+, S_2, N^+) + f_1^{(S_1^+)} V_{t'}(S_1^+, S_2, N) \\
\leq& \overline{f_1^{(S_1)}} V_{t'}(S_1^+, S_2, N^+) + f_1^{(S_1)} V_{t'}(S_1^+, S_2, N) \\
\leq& \overline{f_1^{(S_1)}}\Big[\overline{q_1} V_{t'}(S_1, S_2, N^+) + q_1 V_{t'}(S_1^+, S_2, N^+)\Big] + f_1^{(S_1)} V_{t'}(S_1, S_2, N) \\
=& \phi_{t'}(S_1, S_2, N)
\end{aligned}$$

where the second "$\leq$" holds because $V_{t'}(S_1^+, S_2, N^+) \geq V_{t'}(S_1^+, S_2, N)$ and $\overline{f_1^{(S_1)}} \geq \overline{f_1^{(S_1^+)}}$.

When $N = C$:

$$\phi_{t'}(S_1^+, S_2, C) = V_{t'}(S_1^+, S_2, C) \leq V_{t'}(S_1, S_2, C) = \phi_{t'}(S_1, S_2, C)$$

Then:

$$\begin{aligned}
V_{t'+1}^b(S_1^+, S_2, N) &= \max \begin{cases} V_{t'}((S_1^+)^{++}, S_2, N) \\ V_{t'}(S_1^+, S_2, N) \\ \phi_{t'}(S_1^+, S_2, N) \end{cases} \\
&\leq \max \begin{cases} V_{t'}(S_1^{++}, S_2, N) \\ V_{t'}(S_1, S_2, N) \\ \phi_{t'}(S_1, S_2, N) \end{cases} = V_{t'+1}^b(S_1, S_2, N)
\end{aligned}$$

and

$$\begin{aligned}
V^a_{t'+1}(S_1^+, S_2, N) &= 1 + \max \begin{cases} \overline{q_2} V_{t'}((S_1^+)^{++}, S_2, N^-) + q_2 V_{t'}((S_1^+)^{++}, S_2^+, N^-) \\ \overline{q_2} V_{t'}((S_1^+, S_2, N^-) + q_2 V_{t'}(S_1^+, S_2^+, N^-) \\ \overline{q_2} \phi_{t'}((S_1^+, S_2, N^-) + q_2 \phi_{t'}(S_1^+, S_2^+, N^-) \end{cases} \\
&\leq 1 + \max \begin{cases} \overline{q_2} V_{t'}(S_1^{++}, S_2, N^-) + q_2 V_{t'}(S_1^{++}, S_2^+, N^-) \\ \overline{q_2} V_{t'}(S_1, S_2, N^-) + q_2 V_{t'}(S_1, S_2^+, N^-) \\ \overline{q_2} \phi_{t'}(S_1, S_2, N^-) + q_2 \phi_{t'}(S_1, S_2^+, N^-) \end{cases} \\
&= V^a_{t'+1}(S_1, S_2, N)
\end{aligned}$$

Therefore, in this case, $V^b_{t'+1}(S_1^+, S_2, N) \leq V^b_{t'+1}(S_1, S_2, N)$ and $V^a_{t'+1}(S_1^+, S_2, N) \leq V^a_{t'+1}(S_1, S_2, N)$. Property (2) follows from Equation (4.15).

Case 2 $S_1 > D_1$:

$$V^b_{t'+1}(S_1^+, S_2, N) = V_{t'}((S_1^+)^+, S_2, N) \geq V_{t'}(S_1^+, S_2, N) = V^b_{t'+1}(S_1, S_2, N)$$

and

$$\begin{aligned}
V^a_{t'+1}(S_1^+, S_2, N) &= 1 + \overline{q_2} V_{t'}((S_1^+)^+, S_2, N^-) + q_2 V_{t'}((S_1^+)^+, S_2^+, N^-) \\
&\geq 1 + \overline{q_2} V_{t'}(S_1^+, S_2, N^-) + q_2 V_{t'}(S_1^+, S_2^+, N^-) = V^a_{t'+1}(S_1, S_2, N)
\end{aligned}$$

Therefore, in this case, $V^b_{t'+1}(S_1^+, S_2, N) \geq V^b_{t'+1}(S_1, S_2, N)$ and $V^a_{t'+1}(S_1^+, S_2, N) \geq V^a_{t'+1}(S_1, S_2, N)$. Property (3) holds.

(C) Proof of Properties (4) & (5)

First we consider the case where $S_2 \leq D_2$. When $N < C$:

$$\begin{aligned}
\phi_{t'}(S_1, S_2^+, N) &= \overline{f_1^{(S_1)}}\Big[\overline{q_1} V_{t'}(S_1, S_2^+, N^+) + q_1 V_{t'}(S_1^+, S_2^+, N^+)\Big] + f_1^{(S_1)} V_{t'}(S_1, S_2^+, N) \\
&\leq \overline{f_1^{(S_1)}}\Big[\overline{q_1} V_{t'}(S_1, S_2, N^+) + q_1 V_{t'}(S_1^+, S_2, N^+)\Big] + f_1^{(S_1)} V_{t'}(S_1, S_2, N) \\
&= \phi_{t'}(S_1, S_2, N)
\end{aligned}$$

and when $N = C$:

$$\phi_{t'}(S_1, S_2^+, C) = V_{t'}(S_1, S_2^+, C) \leq V_{t'}(S_1, S_2, C) = \phi_{t'}(S_1, S_2, C)$$

Case 1 $S_1 \leq D_1$:

$$\begin{aligned}
V_{t'+1}^b(S_1, S_2^+, N) &= \max \begin{cases} V_{t'}(S_1^{++}, S_2^+, N) \\ V_{t'}(S_1, S_2^+, N) \\ \phi_{t'}(S_1, S_2^+, N) \end{cases} \\
&\leq \max \begin{cases} V_{t'}(S_1^{++}, S_2, N) \\ V_{t'}(S_1, S_2, N) \\ \phi_{t'}(S_1, S_2, N) \end{cases} = V_{t'+1}^b(S_1, S_2, N)
\end{aligned}$$

$$
\begin{aligned}
V^a_{t'+1}(S_1, S_2^+, N) &= 1 + \max \begin{cases} \overline{q_2} V_{t'}(S_1^{++}, S_2^+, N^-) + q_2 V_{t'}(S_1^{++}, (S_2^+)^+, N^-) \\ \overline{q_2} V_{t'}(S_1, S_2, N^-) + q_2 V_{t'}(S_1, (S_2^+)^+, N^-) \\ \overline{q_2} \phi_{t'}(S_1, S_2^+, N^-) + q_2 \phi_{t'}(S_1, (S_2^+)^+, N^-) \end{cases} \\
&\leq 1 + V^b_{t'+1}(S_1, S_2^+, N^-) \\
&\leq 1 + \max \begin{cases} \overline{q_2} V_{t'}(S_1^{++}, S_2, N^-) + q_2 V_{t'}(S_1^{++}, S_2^+, N^-) \\ \overline{q_2} V_{t'}(S_1, S_2, N^-) + q_2 V_{t'}(S_1, S_2^+, N^-) \\ \overline{q_2} \phi_{t'}(S_1, S_2, N^-) + q_2 \phi_{t'}(S_1, S_2^+, N^-) \end{cases} \\
&= V^a_{t'+1}(S_1, S_2, N)
\end{aligned}
\tag{C.1}
$$

Case 2 $S_1 > D_1$:

$$V^b_{t'+1}(S_1, S_2^+, N) = V_{t'}(S_1^+, S_2^+, N) \leq V_{t'}(S_1^+, S_2, N) = V^b_{t'+1}(S_1, S_2, N)$$

$$
\begin{aligned}
V^a_{t'+1}(S_1, S_2^+, N) &= 1 + \overline{q_2} V_{t'}(S_1^+, S_2^+, N^-) + q_2 V_{t'}(S_1^+, (S_2^+)^+, N^-) \\
&\leq 1 + V^b_{t'+1}(S_1, S_2^+, N^-) \\
&\leq 1 + \overline{q_2} V_{t'}(S_1^+, S_2, N^-) + q_2 V_{t'}(S_1^+, S_2^+, N^-) = V^a_{t'+1}(S_1, S_2, N)
\end{aligned}
\tag{C.2}
$$

Moreover, from (C.1) and (C.2), in both cases, when $N > 0$:

$$V^b_{t'+1}(S_1, S_2^+, N) \leq 1 + V^b_{t'+1}(S_1, S_2^+, N^-) \leq V^a_{t'+1}(S_1, S_2, N)$$

Therefore,

$$
\begin{aligned}
&\overline{f_2^{(S_2^+)}} V^a_{t'}(S_1, S_2^+, N) + f_2^{(S_2^+)} V^b_{t'}(S_1, S_2^+, N) \\
\leq &\overline{f_2^{(S_2^+)}} V^a_{t'}(S_1, S_2, N) + f_2^{(S_2^+)} V^b_{t'}(S_1, S_2^+, N) \\
\leq &\overline{f_2^{(S_2)}} V^a_{t'}(S_1, S_2, N) + f_2^{(S_2)} V^b_{t'}(S_1, S_2^+, N)
\end{aligned}
$$

$$\leq \overline{f_2^{(S_2)}} V^a_{t'}(S_1, S_2, N) + f_2^{(S_2^+)} V^b_{t'}(S_1, S_2, N)$$

Therefore, when $N > 0$:

$$\begin{aligned}
V_{t'+1}(S_1, S_2^+, N) &= \max \begin{cases} V^b_{t'+1}(S_1, (S_2^+)^{++}, N) \\ V^b_{t'+1}(S_1, S_2^+, N) \\ \overline{f_2^{(S_2^+)}} V^a_{t'+1}(S_1, S_2^+, N) + f_2^{(S_2^+)} V^b_{t'+1}(S_1, S_2^+, N) \end{cases} \\
&\leq \max \begin{cases} V^b_{t'+1}(S_1, S_2^{++}, N) \\ V^b_{t'+1}(S_1, S_2, N) \\ \overline{f_2^{(S_2)}} V^a_{t'+1}(S_1, S_2, N) + f_2^{(S_2)} V^b_{t'+1}(S_1, S_2, N) \end{cases} \\
&= V_{t'+1}(S_1, S_2, N)
\end{aligned}$$

and when $N = 0$:

$$\begin{aligned}
V_{t'+1}(S_1, S_2^+, N) &= \max \begin{cases} V^b_{t'+1}(S_1, (S_2^+)^{++}, N) \\ V^b_{t'+1}(S_1, S_2^+, N) \end{cases} \\
&\leq \max \begin{cases} V^b_{t'+1}(S_1, S_2^{++}, N) \\ V^b_{t'+1}(S_1, S_2, N) \end{cases} \\
&= V_{t'+1}(S_1, S_2, N)
\end{aligned}$$

Property (4) is proved. Then we prove Property (5). If $S_2 > D_2$, then similar as (C.2), we have $V^b_{t'+1}(S_1, S_2^+, N) \geq V^b_{t'+1}(S_1, S_2, N)$, and from Equation (4.15), $V_{t'+1}(S_1, S_2^+, N) = V^b_{t'+1}(S_1, (S_2^+)^+, N) \geq V^b_{t'+1}(S_1, S_2^+, N) = V_{t'+1}(S_1, S_2, N)$.

Therefore, all five properties are proved when $t = t' + 1$. Based on induction, all the five properties hold for any t. $\square$

Proof of Lemma 4.1

Proof. Equation (4.20) can be rewritten as

$$V_t(S_1,S_2,N)=\max\left\{\tilde{V}_t(S_1,S_2,N),V_{t-1}(S_1,S_2,N)\right\}$$

where $\tilde{V}_t(S_1,S_2,N)=\Psi_t((S_1,S_2,N),\mathcal{A}\setminus\{(\text{S,S,S})\})$.

Then we prove that $\tilde{V}_t(S_1,S_2,N)\geq V_{t-1}(S_1,S_2,N)\ \forall t,S_1,S_2,N$ by induction.

Step 1. When $t=1$, it is obvious that $\tilde{V}_1(S_1,S_2,N)\geq 0=V_0(S_1,S_2,N)$.

Step 2. Assume $\tilde{V}_t(S_1,S_2,N)\geq V_{t-1}(S_1,S_2,N)$ for $t=1,...,t'$. Then $V_{t'}(S_1,S_2,N)=\tilde{V}_{t'}(S_1,S_2,N)$. Assume the optimal action when the remaining time is t' is $(a_1^a,a_1^b,a_2)^*$. Then we have

$$\begin{aligned}\tilde{V}_{t'+1}(S_1,S_2,N)\geq&\Big\{\sum_{(S_1',S_2',N')}\Pr\{(S_1',S_2',N')|\big((S_1,S_2,N),(a_1^a,a_1^b,a_2)^*\big)\}\cdot\\&\Big[PR\big((S_1,S_2,N),(a_1^a,a_1^b,a_2)^*,(S_1',S_2',N')\big)+V_{t'}(S_1',S_2',N')\Big]\Big\}\\\geq&\Big\{\sum_{(S_1',S_2',N')}\Pr\{(S_1',S_2',N')|\big((S_1,S_2,N),(a_1^a,a_1^b,a_2)^*\big)\}\cdot\\&\Big[PR\big((S_1,S_2,N),(a_1^a,a_1^b,a_2)^*,(S_1',S_2',N')\big)+V_{t'-1}(S_1',S_2',N')\Big]\Big\}\\=&\tilde{V}_{t'}(S_1,S_2,N)=V_{t'}(S_1,S_2,N)\end{aligned}$$

Based on Steps 1 and 2, $\tilde{V}_t(S_1,S_2,N)\geq V_{t-1}(S_1,S_2,N)\ \forall t,S_1,S_2,N$. Therefore, $V_t(S_1,S_2,N)=\tilde{V}_t(S_1,S_2,N)\ \forall t$. □

Proof of Corollary 4.1

Proof. We use the similar approach as in the proof of Lemma 4.1.

Let $\hat{V}_t(S_1,S_2,N)=\Psi_t((S_1,S_2,N),\mathcal{A}')$. Then we prove that $\hat{V}_t(S_1,S_2,N)\geq V_t(S_1,S_2,N)\ \forall t,S_1,S_2,N$ by induction.

Step 1. When $t = 1$, it is obvious that

$$\Psi_t((S_1, S_2, N), \{(a_1^a, a_1^b, \text{M})\}) = \Psi_t((S_1, S_2, N), \{(a_1^a, a_1^b, \text{S})\}) = 0$$

and

$$\Psi_t((S_1, S_2, N), \{(a_1^a, a_1^b, \text{D})\}) = \overline{f^{(S_2)}}.$$

Therefore, $\Psi_1((S_1, S_2, N), \mathcal{A}') \geq \Psi_1((S_1, S_2, N), \mathcal{A})$ and $\hat{V}_1(S_1, S_2, N) \geq V_1(S_1, S_2, N)$.
Step 2. Assume $\hat{V}_t(S_1, S_2, N) \geq V_t(S_1, S_2, N)$ for $t = 1, ..., t' - 1$. We need to prove that $\hat{V}_{t'}(S_1, S_2, N) \geq V_{t'}(S_1, S_2, N)$, which is to prove, the action "stop" is not optimal on either M_1 or M_2.

First, we show that the action "stop" is not optimal on M_2 (when $S_2 < D_2$ and $N > 0$, otherwise the action "stop" is infeasible). In other words, we need to prove that actions (M,M,S) and (D,D,S) can be removed without loss of optimality. We prove

$$\Psi_{t'}((S_1, S_2, N), (a_1^a, a_1^b, \text{D})) \geq \Psi_{t'}((S_1, S_2, N), \{(a_1^a, a_1^b, \text{S})\}). \tag{C.3}$$

Let $a_i^{t'}$ and $\hat{a}_i^{t'}$ be the actions on machines M_i ($i = 1, 2$) that is associated with the cost function $V_{t'}(S_1, S_2, N)$ and $\hat{V}_{t'}(S_1, S_2, N)$, respectively.
Case 1 $a_1^{t'}$ =MM:
If $a_1^{t'}$ =MM and $a_2^{t'-1}$ =M, then we construct a policy $\hat{a}_1{}^{t'}$ =MM and $\hat{a}_2{}^{t'-1}$ =M, then it is easy to obtain that

$$\begin{aligned}
&\Psi_{t'}((S_1, S_2, N), \{(\text{M,M,D})\}) - \Psi_{t'}((S_1, S_2, N), \{(\text{M,M,S})\}) \\
=&\overline{f^{(S_2)}}\big(1 + q_2 V_{t'-1}(S_1^{++}, S_2^{+}, N^-) + \overline{q_2} V_{t'-1}(S_1^{++}, S_2, N^-) - V_{t'-1}(S_1^{++}, S_2, N)\big) \\
\geq&\overline{f^{(S_2)}} q_2 \big(V_{t'-1}(S_1^{++}, S_2^{+}, N) - V_{t'-1}(S_1^{++}, S_2, N)\big) \text{(from Theorem 4.2, Property 1)} \\
=&0 \quad \text{(Based on } V_{t'-2}(S_1^{++} + 1, (S_2^{+})^{++}, N) = V_{t'-2}(S_1^{++} + 1, S_2^{++}, N))
\end{aligned}$$

If $a_1^{t'}$ =MM and $a_2^{t'-1}$ =D, then we construct a policy $\hat{a}_1{}^{t'}$ =MM and $\hat{a}_2{}^{t'-1}$ =S.

Then the system states when the remaining time is $t' - 2$ are the same. Therefore, $\hat{V}_t(S_1, S_2, N) \geq V_t(S_1, S_2, N)$.

Case 2a $a_1^{t'}$ =DD and $S_1 > D_1$:

This case is similar as Case 1. If $a_2^{t'-1}$ =M or $a_2^{t'-1}$ =D, we can construct a policy $\hat{a_2}^{t'-1}$ =M or $\hat{a_2}^{t'-1}$ =S, respectively, so that (C.3) is satisfied.

Case 2b $a_1^{t'}$ =DD and $S_1 \leq D_1$:

In this case, $a_2^{t'-1}$ may take three different values based on the different transition states at time $t^{'}$ (i.e., M_1 is not working, M_1 is working and not degrading, and M_1 is degrading). Denote $\mathbf{a}_2^{t'-1} = [a_{2,1}^{t'-1}, a_{2,2}^{t'-1}, a_{2,3}^{t'-1}]$ as the actions in these three different situations. We define a function $\varphi(x)$ =M if x =M and $\varphi(x)$ =S if x =D, then the policy can be constructed as $\hat{\mathbf{a}}_2^{t'-1} = [\varphi(a_{2,1}^{t'-1}), \varphi(a_{2,2}^{t'-1}), \varphi(a_{2,3}^{t'-1})]$, so that (C.3) is satisfied.

Similarly, we can prove that the action "stop" is not optimal on M_1. Let $\mathbf{a}_1^{t'} = [a_{1,1}^{t'}, \cdots, a_{1,n_{1,t'}}^{t'}]$ be the policy on M_1 when the remaining time is t', and $\mathbf{a}_1^{t'-1} = [a_{1,1}^{t'-1}, \cdots, a_{1,n_{1,t'-1}}^{t'-1}]$ be the policy when the remaining time is $t' - 1$, then we can construct a policy $\hat{\mathbf{a}}_1^{t'} = [\varphi'(a_{1,1}^{t'}), \cdots, \varphi'(a_{1,n_{1,t'}}^{t'})]$ where $\varphi'(x)$ =D if x =S, and $\varphi'(x) = x$ if $x \neq$S, and $\hat{\mathbf{a}}_1^{t'-1} = [\varphi(a_{1,1}^{t'-1}), \cdots, \varphi(a_{1,n_{1,t'-1}}^{t'-1})]$. By such construction, we have $\hat{V}_t'(S_1, S_2, N) \geq V_t'(S_1, S_2, N)$.

Therefore, "stop" is not optimal on either M_1 or M_2 when $t = t'$. By induction, $\hat{V}_t(S_1, S_2, N) \geq V_t(S_1, S_2, N)$ $\forall t, S_1, S_2, N$. Therefore, $V_t(S_1, S_2, N) = \hat{V}_t(S_1, S_2, N)$ $\forall t$.

$\square$

APPENDIX D

Proof for Chapter V

Proof of Theorem 5.1

By taking one-step conditional expectation, V_{mn}'s and $T^{(l)}_{mn}$'s can be expressed as

$$V_{mn} = 1 + \sum_{h \neq m} P_{hn} V_{mh} \quad (0 \leq n, m \leq C) \tag{D.1}$$

and

$$T^{(l)}_{mn} = \delta_{nl} + \sum_{h \neq m} P_{hn} T^{(l)}_{mh} \quad (0 \leq n, m \leq C, 0 \leq l \leq S_2 - 1) \tag{D.2}$$

In the matrix form, Equations (D.1) and (D.2) can be written as

$$\mathbf{V} = \mathbf{W}^T \mathbf{V} + \mathbf{1}^T \tag{D.3}$$

and

$$\mathbf{T}^{(l)} = \mathbf{W}^T \mathbf{T}^{(l)} + [\delta_{0l} \cdot \mathbf{1} \quad \cdots \quad \delta_{Cl} \cdot \mathbf{1}]^T \tag{D.4}$$

The expressions of $\mathbf{V}$ and $\mathbf{T}^{(l)}$ in Theorem 5.1 result directly from Equations (D.3) and (D.4). $\square$

Proof of Theorem 5.2

First, we state some results that are useful for the proof.

Definition D.1. Let $\mathbf{a} = [a_1 \ \cdots \ a_M]^T$ and $\mathbf{b} = [b_1 \ \cdots \ b_M]^T$ be two $M \times 1$ vectors, where $\sum\limits_{m=1}^{M} a_m = \sum\limits_{m}^{M} b_m = 1$. We define $\mathbf{a} \sqsupset \mathbf{b}$ (or equivalently, $\mathbf{b} \sqsubset \mathbf{a}$) if $\sum\limits_{m=1}^{M'} a_m \geq \sum\limits_{m=1}^{M'} b_m$ (or equivalently, $\sum\limits_{m=M'+1}^{M} a_m \leq \sum\limits_{m=M'+1}^{M} b_m$) for all $M' = 1, ..., M-1$.

Then we have the following lemmas.

Lemma D.1. *For any* $\mathbf{P}(\mathbf{r}_1, \mathbf{r}_2, C)$*, we have* $[P_{0n} \ \cdots \ P_{Cn}]^T \sqsupset [P_{0,n+1} \ \cdots \ P_{C,n+1}]^T$.

Proof. For any $n = 0, ..., C-1$ and $M' = 0, ..., C-1$,

$$\begin{aligned}\sum_{m=0}^{M'} (P_{mn} - P_{m,n+1}) &= \sum_{m=0}^{M'} \left(\sum_{j=m+n-M'}^{S_2} r_1(m) r_2(j) - \sum_{j=m+n+1-M'}^{S_2} r_1(m) r_2(j) \right) \\ &= \sum_{m=0}^{M'} r_1(m) r_2(m+n-M') \geq 0.\end{aligned}$$

$[P_{0n} \ \cdots \ P_{Cn}]^T \sqsupset [P_{0,n+1} \ \cdots \ P_{C,n+1}]^T$ follows immediately by Definition D.1.

□

Lemma D.2. *Let* P^a_{mn} *and* P^b_{mn} *be the element at the* $(n+1)^{th}$ *conlumn and the* $(m+1)^{th}$ *row of matrices* $\mathbf{P}(\mathbf{r}^a_1, \mathbf{r}^a_2, C)$ *and* $\mathbf{P}(\mathbf{r}^b_1, \mathbf{r}^b_2, C)$*, respectively. If* $\mathbf{r}^a_1 \sqsubset \mathbf{r}^b_1$ *and* $\mathbf{r}^a_2 \sqsupset \mathbf{r}^b_2$*, then* $[P^a_{0n} \ \cdots \ P^a_{Cn}]^T \sqsubset [P^b_{0n} \ \cdots \ P^b_{Cn}]^T$ *($\forall n$).*

Proof. For $n = 0, ..., C$ and $M' = 0, ..., C-1$, we have

$$\begin{aligned}
\sum_{m=0}^{M'} P^a_{mn} - \sum_{m=0}^{M'} P^b_{mn} =& \sum_{m=0}^{M'} \sum_{j=m+n-M'}^{S_2} \left(r^a_1(m) r^a_2(j) - r^b_1(m) r^b_2(j)\right) \\
=& \sum_{m=0}^{M'} \sum_{j=m+n-M'}^{S_2} \left(r^a_1(m) - r^b_1(m)\right) r^a_2(j) \\
&+ \sum_{m=0}^{M'} \sum_{j=m+n-M'}^{S_2} r^b_1(m) \left(r^a_2(j) - r^b_2(j)\right) \\
=& \sum_{j=0}^{S_2} \sum_{m=0}^{j+M'-n} r^a_2(j) \left(r^a_1(m) - r^b_1(m)\right) \\
&+ \sum_{m=0}^{M'} \sum_{j=m+n-M'}^{S_2} r^b_1(m) \left(r^a_2(j) - r^b_2(j)\right).
\end{aligned}$$

Let $A_1 := \sum_{j=0}^{S_2} \sum_{m=0}^{j+M'-n} r^a_2(j) \left(r^a_1(m) - r^b_1(m)\right)$ and $B_1 := \sum_{m=0}^{M'} \sum_{j=m+n-M'}^{S_2} \Big[r^b_1(m) \cdot \left(r^a_2(j) - r^b_2(j)\right)\Big]$. If $\mathbf{r}^a_1 \sqsubset \mathbf{r}^b_1$ and $\mathbf{r}^a_2 \sqsupset \mathbf{r}^b_2$, then $A_1 \leq 0$ and $B_1 \leq 0$. Therefore, $A_1 + B_1 \leq 0$. □

Lemma D.3. *If* $\mathbf{r}^a_1 \sqsubset \mathbf{r}^b_1$, $\mathbf{r}^a_2 \sqsupset \mathbf{r}^b_2$ *and* $\boldsymbol{\pi}^a \sqsubset \boldsymbol{\pi}^b$, *then* $\mathbf{P}(\mathbf{r}^a_1, \mathbf{r}^a_2, C)\boldsymbol{\pi}^a \sqsubset \mathbf{P}(\mathbf{r}^b_1, \mathbf{r}^b_2, C)\boldsymbol{\pi}^b$.

Proof. Let $\boldsymbol{\pi}'^a := \mathbf{P}(\mathbf{r}^a_1, \mathbf{r}^a_2, C)\boldsymbol{\pi}^a$ and $\boldsymbol{\pi}'^b := \mathbf{P}(\mathbf{r}^b_1, \mathbf{r}^b_2, C)\boldsymbol{\pi}^b$. Then for $M' = 0, ..., C-1$

$$\begin{aligned}
\sum_{m=0}^{M'} \pi'^a_m - \sum_{m=0}^{M'} \pi'^b_m =& \sum_{n=0}^{C} \sum_{m=0}^{M'} \left((P^a_{mn} - P^b_{mn})\pi^a_n + P^b_{mn}(\pi^a_n - \pi^b_n)\right) \\
=& \sum_{n=0}^{C} \sum_{m=0}^{M'} (P^a_{mn} - P^b_{mn})\pi^a_n + \sum_{m=0}^{M'} \sum_{n=0}^{C-1} \sum_{l=0}^{n} \left((P^b_{mn} - P^b_{m,n+1})(\pi^a_l - \pi^b_l)\right) \\
&+ \sum_{m=0}^{M'} \sum_{l=0}^{C} \left(P^b_{Cm}(\pi^a_l - \pi^b_l)\right) \\
=& \sum_{n=0}^{C} \sum_{m=0}^{M'} (P^a_{mn} - P^b_{mn})\pi^a_n + \sum_{m=0}^{M'} \sum_{n=0}^{C-1} \sum_{l=0}^{n} \left((P^b_{mn} - P^b_{m,n+1})(\pi^a_l - \pi^b_l)\right).
\end{aligned}$$

Let $A_2 := \sum_{n=0}^{C} \sum_{m=0}^{M'} (P^a_{mn} - P^b_{mn})\pi^a_n$ and $B_2 := \sum_{m=0}^{M'} \sum_{n=0}^{C-1} \sum_{l=0}^{n} \left((P^b_{mn} - P^b_{m,n+1})(\pi^a_l - \pi^b_l)\right)$. If $\mathbf{r}^a_1 \sqsubset \mathbf{r}^b_1$, $\mathbf{r}^a_2 \sqsupset \mathbf{r}^b_2$ and $\boldsymbol{\pi}^a \sqsupset \boldsymbol{\pi}^b$, then $A_2 \leq 0$ (by Lemma D.2) and $B_2 \leq 0$ (by

Lemma D.1). Therefore, $A_2 + B_2 \leq 0$. $\square$

Lemma D.4. *If* $\mathbf{r}_1^a \sqsubset \mathbf{r}_1^b$ *and* $\mathbf{r}_2^a \sqsupset \mathbf{r}_2^b$*, then* $\boldsymbol{\pi}^S(\mathbf{r}_1^a, \mathbf{r}_2^a, C) \sqsubset \boldsymbol{\pi}^S(\mathbf{r}_1^b, \mathbf{r}_2^b, C)$*.*

Proof. Let $\boldsymbol{\pi}^a := \boldsymbol{\pi}^S(\mathbf{r}_1^a, \mathbf{r}_2^a, C)$ and $\boldsymbol{\pi}^b := \boldsymbol{\pi}^S(\mathbf{r}_1^b, \mathbf{r}_2^b, C)$. Also,we define the following recursive procedure:

$$\boldsymbol{\pi}^b(a+1) = \mathbf{P}^b \boldsymbol{\pi}^b(a) \quad (a = 0, 1, 2, ...)$$

with the initial condition $\boldsymbol{\pi}^b(0) = \boldsymbol{\pi}^a$. Then, $\boldsymbol{\pi}^b$ can be calculated as $\boldsymbol{\pi}^b = \lim_{a\to\infty} \boldsymbol{\pi}^b(a)$.

We use induction to show that $\boldsymbol{\pi}^b(a+1) \sqsupset \boldsymbol{\pi}^b(a)$.

Step 1. Based on Lemma D.3: $\boldsymbol{\pi}^b(1) = \mathbf{P}^b \boldsymbol{\pi}^b(0) \sqsupset \mathbf{P}^a \boldsymbol{\pi}^b(0) = \boldsymbol{\pi}^b(0)$.

Step 2. When $a \geq 1$, by Lemma D.3: $\boldsymbol{\pi}^b(a+1) = \mathbf{P}^b \boldsymbol{\pi}^b(a) \sqsupset \mathbf{P}^b \boldsymbol{\pi}^b(a-1) = \boldsymbol{\pi}^b(a)$.

Step 3. By Steps 1 and 2, $\boldsymbol{\pi}^b(a+1) \sqsupset \boldsymbol{\pi}^b(a)\ \forall a = 0, 1, ...$

Therefore, $\boldsymbol{\pi}^b = \lim_{a\to\infty} \boldsymbol{\pi}^b(a) \sqsupset \boldsymbol{\pi}^b(0) = \boldsymbol{\pi}^a$. $\square$

Lemma D.5. *If* $\mathbf{r}_2^a \sqsubset \mathbf{r}_2^b$ *and* $\boldsymbol{\pi}^a \sqsubset \boldsymbol{\pi}^b$*, then* $\mathbf{r}_2^a \sqsubset \mathbf{p}(\mathbf{r}_2^a, C)\boldsymbol{\pi}^a \sqsubset \mathbf{p}(\mathbf{r}_2^b, C)\boldsymbol{\pi}^b$*.*

Proof. For $S_2' = 0, ..., S_2 - 1$:

$$\sum_{s=S_2'+1}^{S_2} pr^a(s) = \sum_{s=S_2'+1}^{S_2} r_2^a(s) \sum_{s=S_2'+1}^{S_2} \pi^a(s) \geq \sum_{s=S_2'+1}^{S_2} r_2^b(s) \sum_{s=S_2'+1}^{S_2} \pi^b(s) = \sum_{s=S_2'+1}^{S_2} pr^b(s)$$

and

$$\sum_{s=S_2'+1}^{S_2} pr^a(s) = \sum_{s=S_2'+1}^{S_2} r_2^a(s) \sum_{s=S_2'+1}^{S_2} \pi^a(s) \leq \sum_{s=S_2'+1}^{S_2} r_2^a(s).$$

Therefore, $\mathbf{r}_2^a \sqsubset \mathbf{p}(\mathbf{r}_2^a, C)\boldsymbol{\pi}^a \sqsubset \mathbf{p}(\mathbf{r}_2^b, C)\boldsymbol{\pi}^b$. $\square$

Lemma D.6. *If* $\mathbf{r}_1^a \sqsupset \mathbf{r}_1^b$*,* $\mathbf{r}_2^a \sqsupset \mathbf{r}_2^b$ *and* $\boldsymbol{\pi}^a \sqsubset \boldsymbol{\pi}^b$*, then* $\mathbf{c}(\mathbf{r}_1^a, \mathbf{r}_2^a, C)\boldsymbol{\pi}^a \sqsupset \mathbf{c}(\mathbf{r}_1^b, \mathbf{r}_2^b, C)\boldsymbol{\pi}^b \sqsupset \mathbf{r}_1^b$*.*

Proof. For $S_1' = 0, ..., S_1 - 1$:

$$\begin{aligned}
\sum_{s=S_1'+1}^{S_1} cr^a(s) &= \sum_{l=S_1'+1}^{S_1} r_1^a(l) \sum_{j=0}^{S_2} \sum_{s=0}^{C-S_1'+j} \pi^a(s) r_2^a(j) \\
&\leq \sum_{l=S_1'+1}^{S_1} r_1^b(l) \sum_{j=0}^{S_2} \sum_{s=0}^{C-S_1'+j} \pi^b(s) r_2^a(j) = \sum_{l=S_1'+1}^{S_1} r_1^b(l) \sum_{s=0}^{C} \sum_{j=C-S_1'-s}^{S_2} r_2^a(j) \pi^b(s) \\
&\leq \sum_{l=S_1'+1}^{S_1} r_1^b(l) \sum_{s=0}^{C} \sum_{j=C-S_1'-s}^{S_2} r_2^b(j) \pi^b(s) = \sum_{s=S_1'+1}^{S_1} cr^b(s) \\
&= \sum_{l=S_1'+1}^{S_1} r_1^b(l) \sum_{j=0}^{S_2} \sum_{s=0}^{C-S_1'+j} \pi^b(s) r_2^b(j) \\
&\leq \sum_{l=S_1'+1}^{S_1} r_1^b(l).
\end{aligned}$$

Therefore, $\mathbf{c}(\mathbf{r}_1^a, \mathbf{r}_2^a, C)\boldsymbol{\pi}^a \sqsupset \mathbf{c}(\mathbf{r}_1^b, \mathbf{r}_2^b, C)\boldsymbol{\pi}^b \sqsupset \mathbf{r}_1^b$. □

Then we prove Theorem 5.2, i.e., the convergence of Recursive Procedure 5.1.

We prove by induction that $\mathbf{r}_i^{NB}(a+1) \sqsupset \mathbf{r}_i^{NB}(a)$ for $i = 2, ..., I$ and $a = 0, 1, ...$

Step 1. By Lemma D.6, for $i = 2, ..., I$:

$$\mathbf{r}_i^{NB}(1) = \mathbf{c}(\mathbf{r}_i, \mathbf{r}_{i+1}^{NB}(1), C_i)\boldsymbol{\pi}^S(\mathbf{r}_i^{NS}(1), \mathbf{r}_{i+1}^{NB}(1), C_i) \sqsupset \mathbf{r}_i = \mathbf{r}_i^{NB}(0)$$

Moreover, based on Equation (5.23), $\mathbf{r}_1^{NS}(a+1) \sqsupset \mathbf{r}_1^{NS}(a)$ and $\mathbf{r}_I^{NB}(a+1) \sqsubset \mathbf{r}_I^{NB}(a)$ for all a. Therefore, $\mathbf{r}_i^{NB}(1) \sqsupset \mathbf{r}_i^{NB}(0)$ for $i = 1, ..., I$.

Step 2. Assume we have $\mathbf{r}_i^{NB}(a') \sqsupset \mathbf{r}_i^{NB}(a'-1)$ for $i = 1, ..., I$ and $a' = 1, ..., a$; $\mathbf{r}_i^{NS}(a+1) \sqsubset \mathbf{r}_i^{NS}(a)$ for $i = 1, ..., i_1 - 1 (i_1 \leq I)$; and $\mathbf{r}_i^{NB}(a+1) \sqsupset \mathbf{r}_i^{NB}(a)$ for $i = I, ..., i_2 + 1 (i_2 \geq 1)$.

By Lemmas D.4 and D.5,

$$\begin{aligned}
\mathbf{r}_{i_1}^{NS}(a+1) &= \mathbf{p}(\mathbf{r}_{i_1}, C_{i_1-1})\boldsymbol{\pi}^S(\mathbf{r}_{i_1-1}^{NS}(a+1), \mathbf{r}_{i_1}^{NB}(a), C_{i_1-1}) \\
&\sqsubset \mathbf{p}(\mathbf{r}_{i_1}, C_{i_1-1})\boldsymbol{\pi}^S(\mathbf{r}_{i_1-1}^{NS}(a), \mathbf{r}_{i_1}^{NB}(a-1), C_{i_1-1}) = \mathbf{r}_{i_1}^{NS}(a)
\end{aligned} \tag{D.5}$$

By applying (D.5) recursively for $i = i_1+1, ..., I$, we have $\mathbf{r}_i^{NS}(a+1) \sqsubset \mathbf{r}_i^{NS}(a)$ $\forall i$.

Moreover, by Lemmas D.4 and D.6,

$$\begin{aligned}\mathbf{r}_{i_2}^{NB}(a+1) &= \mathbf{c}(\mathbf{r}_{i_2}, \mathbf{r}_{i_2+1}^{NB}(a+1), C_{i_2})\boldsymbol{\pi}^S(\mathbf{r}_{i_2}^{NS}(a+1), \mathbf{r}_{i_2+1}^{NB}(a+1), C_{i_2}) \\ &\sqsupset \mathbf{c}(\mathbf{r}_{i_2}, \mathbf{r}_{i_2+1}^{NB}(a), C_{i_2})\boldsymbol{\pi}^S(\mathbf{r}_{i_2}^{NS}(a), \mathbf{r}_{i_2+1}^{NB}(a), C_{i_2}) = \mathbf{r}_{i_2}^{NB}(a)\end{aligned} \tag{D.6}$$

Applying (D.6) recursively for $i = i_2 - 1, ..., 1$ leads to $\mathbf{r}_i^{NB}(a+1) \sqsupset \mathbf{r}_i^{NB}(a)\ \forall i$.

Step 3. By induction, $\mathbf{r}_i^{NB}(a+1) \sqsupset \mathbf{r}_i^{NB}(a)\ \forall a, i$.

We define a sequence

$$W(a) := \sum_{s=0}^{S_2} s \cdot r_2^{NB}(s;a) = \sum_{j=0}^{S_2-1} \sum_{s=S_2-j}^{S_2} r_2^{NB}(s;a) \geq \sum_{j=0}^{S_2-1} \sum_{s=S_2-j}^{S_2} r_2^{NB}(s;a+1) = W(a+1).$$

Then $W(a)$ is decreasing in a. Moreover, $W(a) \geq 0$ for all a. Therefore, the sequence $W(a)$ converges, which proves the convergence of Recursive Procedure 5.1.

□

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Jun Ni and Xiaoning Jin. Decision support systems for effective maintenance operations. *CIRP Annals-Manufacturing Technology*, 61(1):411–414, 2012.

[2] SA Spiewak, R Duggirala, and K Barnett. Predictive monitoring and control of the cold extrusion process. *CIRP Annals-Manufacturing Technology*, 49(1):383–386, 2000.

[3] R Keith Mobley. *An introduction to predictive maintenance*. Butterworth-Heinemann, 2002.

[4] Yoram Koren. *The global manufacturing revolution: product-process-business integration and reconfigurable systems*, volume 80. John Wiley & Sons, 2010.

[5] Jingshan Li, Dennis E. Blumenfeld, Ningjian Huang, and Jeffrey M. Alden. Throughput analysis of production systems: recent advances and future topics. *International Journal of Production Research*, 47(14):3823–3851, 2009.

[6] Qing Chang, Jun Ni, Pulak Bandyopadhyay, Stephan Biller, and Guoxian Xiao. Maintenance opportunity planning system. *Journal of manufacturing science and engineering*, 129(3):661–668, 2007.

[7] Qing Chang, Stephan Biller, and Guoxian Xiao. Transient analysis of downtimes and bottleneck dynamics in serial manufacturing systems. *Journal of Manufacturing Science and Engineering*, 132(5):051015, 2010.

[8] Yiqiang Wang, Richard CM Yam, Ming J Zuo, and Peter Tse. A comprehensive reliability allocation method for design of cnc lathes. *Reliability engineering & system safety*, 72(3):247–252, 2001.

[9] Ashraf W Labib. A decision analysis model for maintenance policy selection using a cmms. *Journal of Quality in Maintenance Engineering*, 10(3):191–202, 2004.

[10] Hoang Pham and Hongzhou Wang. Imperfect maintenance. *European Journal of Operational Research*, 94(3):425–438, 1996.

[11] T Chitra. Life based maintenance policy for minimum cost. In *Reliability and Maintainability Symposium, 2003. Annual*, pages 470–474. IEEE, 2003.

[12] Ying Peng, Ming Dong, and Ming Jian Zuo. Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology*, 50(1-4):297–313, 2010.

[13] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006.

[14] Marzio Marseguerra, Enrico Zio, and Luca Podofillini. Condition-based maintenance optimization by means of genetic algorithms and monte carlo simulation. *Reliability Engineering & System Safety*, 77(2):151–165, 2002.

[15] Xiao Liu, Jingrui Li, Khalifa N Al-Khalifa, Abdelmagid S Hamouda, David W Coit, and Elsayed A Elsayed. Condition-based maintenance for continuously monitored degrading systems with multiple failure modes. *IIE Transactions*, 45(4):422–435, 2013.

[16] Richard E Barlow and Frank Proschan. *Mathematical theory of reliability*, volume 17. Siam, 1996.

[17] William Q Meeker and Luis A Escobar. *Statistical methods for reliability data*, volume 314. John Wiley & Sons, 1998.

[18] Toshio Nakagawa. *Maintenance theory of reliability*. Springer, 2006.

[19] D Banjevic, AKS Jardine, V Makis, and M Ennis. A control-limit policy and software for condition-based maintenance optimization. *INFOR-OTTAWA-*, 39(1):32–50, 2001.

[20] Antoine Grall, Christophe Bérenguer, and Laurence Dieulle. A condition-based maintenance policy for stochastically deteriorating systems. *Reliability Engineering & System Safety*, 76(2):167–180, 2002.

[21] Laurence Dieulle, Christophe Bérenguer, Antoine Grall, and Michel Roussignol. Sequential condition-based maintenance scheduling for a deteriorating system. *European Journal of Operational Research*, 150(2):451–461, 2003.

[22] Bora Çekyay and Süleyman Özekici. Condition-based maintenance under markovian deterioration. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[23] Wolfgang Stadje and Dror Zuckerman. A generalized maintenance model for stochastically deteriorating equipment. *European Journal of Operational Research*, 89(2):285–301, 1996.

[24] Dongyan Chen and Kishor S Trivedi. Closed-form analytical results for condition-based maintenance. *Reliability Engineering & System Safety*, 76(1):43–51, 2002.

[25] Dongyan Chen and Kishor S Trivedi. Optimization for condition-based maintenance with semi-markov decision process. *Reliability Engineering & System Safety*, 90(1):25–29, 2005.

[26] Yisha Xiang. Joint optimization of control chart and preventive maintenance policies: A discrete-time markov chain approach. *European Journal of Operational Research*, 229(2):382 – 390, 2013.

[27] LC Thomas. A survey of maintenance and replacement models for maintainability and reliability of multi-item systems. *Reliability Engineering*, 16(4):297–309, 1986.

[28] Danny I Cho and Mahmut Parlar. A survey of maintenance models for multi-unit systems. *European Journal of Operational Research*, 51(1):1–23, 1991.

[29] Rommert Dekker, Ralph E Wildeman, and Frank A van der Duyn Schouten. A review of multi-component maintenance models with economic dependence. *Mathematical Methods of Operations Research*, 45(3):411–435, 1997.

[30] Robin P Nicolai and Rommert Dekker. *Optimal maintenance of multi-component systems: a review.* Springer, 2008.

[31] Hongzhou Wang. A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, 139(3):469–489, 2002.

[32] Fran Barbera, Helmut Schneider, and Ed Watson. A condition based maintenance model for a two-unit series system. *European Journal of Operational Research*, 116(2):281–290, 1999.

[33] Bruno Castanier, Antoine Grall, and Christophe Bérenguer. A condition-based maintenance policy with non-periodic inspections for a two-unit series system. *Reliability Engineering & System Safety*, 87(1):109–120, 2005.

[34] Zhigang Tian and Haitao Liao. Condition based maintenance optimization for multi-component systems using proportional hazards model. *Reliability Engineering & System Safety*, 96(5):581–589, 2011.

[35] FA Van der Duyn Schouten and SG Vanneste. Maintenance optimization of a production system with buffer capacity. *European Journal of Operational Research*, 82(2):323–338, 1995.

[36] Saumil Ambani, Lin Li, and Jun Ni. Condition-based maintenance decision-making for multiple machine systems. *Journal of Manufacturing Science and Engineering*, 131(3):031009, 2009.

[37] Seungchul Lee, Lin Li, and Jun Ni. Markov-based maintenance planning considering repair time and periodic inspection. *Journal of Manufacturing Science and Engineering*, 135(3):031013, 2013.

[38] Yves Dallery and Stanley B Gershwin. Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems*, 12(1-2):3–94, 1992.

[39] John A Buzacott and J George Shanthikumar. Design of manufacturing systems using queueing models. *Queueing Systems*, 12(1-2):135–213, 1992.

[40] HT Papadopoulos and C Heavey. Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research*, 92(1):1–27, 1996.

[41] Manish K Govil and Michael C Fu. Queueing theory in manufacturing: A survey. *Journal of Manufacturing Systems*, 18(3):214–240, 1999.

[42] N Viswanadham and Y Narahari. *Performance modeling of automated manufacturing systems*. Prentice Hall Englewood Cliffs, NJ, 1992.

[43] John A Buzacott and J George Shanthikumar. *Stochastic models of manufacturing systems*, volume 4. Prentice Hall Englewood Cliffs, NJ, 1993.

[44] Stanley B Gershwin. *Manufacturing systems engineering*. PTR Prentice Hall Englewood Cliffs, New Jersey, 1994.

[45] Tayfur Altiok. *Performance analysis of manufacturing systems*. Springer, 1997.

[46] Jingshan Li and Semyon M Meerkov. *Production systems engineering*. Springer, 2008.

[47] Stanley B Gershwin and Oded Berman. Analysis of transfer lines consisting of two unreliable machines with random processing times and finite storage buffers. *AIIE Transactions*, 13(1):2–11, 1981.

[48] Jingshan Li, Dennis E Blumenfeld, and Jeffrey M Alden. Comparisons of two-machine line models in throughput analysis. *International Journal of Production Research*, 44(7):1375–1398, 2006.

[49] Barış Tan and Stanley B Gershwin. Analysis of a general markovian two-stage continuous-flow production system with a finite buffer. *International Journal of Production Economics*, 120(2):327–339, 2009.

[50] Barış Tan and Stanley B Gershwin. Modelling and analysis of markovian continuous flow systems with a finite buffer. *Annals of Operations Research*, 182(1):5–30, 2011.

[51] Stanley B Gershwin. An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research*, 35(2):291–305, 1987.

[52] Yves Dallery, Rene David, and X-L Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, 34(9):943–953, 1989.

[53] Saumil Ambani. *Analytical estimation of throughput distribution for serial manufacturing systems with multi-state machines and its application.* PhD thesis, The University of Michigan, 2011.

[54] Yadati Narahari and Nukala Viswanadham. Transient analysis of manufacturing systems performance. *IEEE Transactions on Robotics & Automation*, 10(2):230–244, 1994.

[55] Debasis Mitra. Stochastic theory of a fluid model of producers and consumers coupled by a buffer. *Advances in Applied Probability*, pages 646–676, 1988.

[56] Semyon M Meerkov and Liang Zhang. Transient behavior of serial production lines with bernoulli machines. *IIE Transactions*, 40(3):297–312, 2008.

[57] Liang Zhang, Chuanfeng Wang, Jorge Arinez, and Stephan Biller. Transient analysis of bernoulli serial lines: performance evaluation and system-theoretic properties. *IIE Transactions*, 45(5):528–543, 2013.

[58] Dieter Armbruster, Simone Göttlich, and Michael Herty. A scalar conservation law with discontinuous flux for supply chains with finite buffers. *SIAM Journal on Applied Mathematics*, 71(4):1070–1087, 2011.

[59] Jianbo Liu, Qing Chang, Guoxian Xiao, and Stephan Biller. The costs of downtime incidents in serial multistage manufacturing systems. *Journal of Manufacturing Science and Engineering*, 134(2):021016, 2012.

[60] Xi Gu, Seungchul Lee, Xinran Liang, Mark Garcellano, Mark Diederichs, and Jun Ni. Hidden maintenance opportunities in discrete and complex production lines. *Expert Systems with Applications*, 40(11):4353–4361, 2013.

[61] C-T Kuo, J-T Lim, and Semyon M Meerkov. Bottlenecks in serial production lines: A system-theoretic approach. *Mathematical Problems in Engineering*, 2(3):233–276, 1996.

[62] Lin Li, Qing Chang, and Jun Ni. Data driven bottleneck detection of manufacturing systems. *International Journal of Production Research*, 47(18):5019–5036, 2009.

[63] `http://www.simul8.com/`.

[64] Dragan Djurdjanovic, Jay Lee, and Jun Ni. Watchdog agentan infotronics-based prognostics approach for product performance degradation assessment and prediction. *Advanced Engineering Informatics*, 17(3):109–125, 2003.

[65] MM Srinivasan and H-S Lee. Production-inventory systems with preventive maintenance. *IIE Transactions*, 28(11):879–890, 1996.

[66] Ki Ling Cheung and Warren H Hausman. Joint determination of preventive maintenance and safety stocks in an unreliable production environment. *Naval Research Logistics (NRL)*, 44(3):257–272, 1997.

[67] Anis Chelbi and Daoud Ait-Kadi. Analysis of a production/inventory system with randomly failing production unit submitted to regular preventive maintenance. *European Journal of Operational Research*, 156(3):712–718, 2004.

[68] EG Kyriakidis and TD Dimitrakos. Optimal preventive maintenance of a production system with an intermediate buffer. *European Journal of Operational Research*, 168(1):86–99, 2006.

[69] Xiaoning Jin, Lin Li, and Jun Ni. Option model for joint production and preventive maintenance system. *International Journal of Production Economics*, 119(2):347–353, 2009.

[70] Jingshan Li and Semyon M Meerkov. Customer demand satisfaction in production systems: A due-time performance approach. *IEEE Transactions on Robotics and Automation*, 17(4):472–482, 2001.

[71] Jorge Arinez, Stephan Biller, Semyon M Meerkov, and Liang Zhang. Quality/quantity improvement in an automotive paint shop: A case study. *IEEE Transactions on Automation Science and Engineering*, 7(4):755–761, 2010.

[72] Cyrus Derman. On optimal replacement rules when changes of state are markovian. *Mathematical Optimization Techniques*, 396, 1963.

[73] Edward PC Kao. Optimal replacement rules when changes of state are semi-markovian. *Operations Research*, 21(6):1231–1249, 1973.

[74] Kut C So. Optimality of control limit policies in replacement models. *Naval Research Logistics (NRL)*, 39(5):685–697, 1992.

[75] Nir Douer and Uri Yechiali. Optimal repair and replacement in markovian systems. *Stochastic Models*, 10(1):253–270, 1994.

[76] Russell D Meller and David S Kim. The impact of preventive maintenance on system cost and buffer size. *European Journal of Operational Research*, 95(3):577–591, 1996.

[77] Richard Barlow and Larry Hunter. Optimum preventive maintenance policies. *Operations Research*, 8(1):90–100, 1960.

[78] Dimitri P Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 2012.

[79] WJ Zhang and CA Van Luttervelt. Toward a resilient manufacturing system. *CIRP Annals-Manufacturing Technology*, 60(1):469–472, 2011.

[80] J Lee, M Ghaffari, and S Elmeligy. Self-maintenance and engineering immune systems: towards smarter machines and manufacturing systems. *Annual Reviews in Control*, 35(1):111–122, 2011.

[81] Zimin Yang, Dragan Djurdjanovic, and Jun Ni. Maintenance scheduling for a manufacturing system of machines with adjustable throughput. *IIE Transactions*, 39(12):1111–1125, 2007.

[82] Yoram Koren, Uwe Heisel, Francesco Jovane, Toshimichi Moriwaki, G Pritschow, G Ulsoy, and H Van Brussel. Reconfigurable manufacturing systems. *CIRP Annals–Manufacturing Technology*, 48(2):527–540, 1999.

[83] Yoram Koren and Moshe Shpitalni. Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 29(4):130–141, 2010.

[84] Yoram Koren. The rapid responsiveness of rms. *International Journal of Production Research*, 51(23-24):6817–6827, 2013.

[85] Wencai Wang and Yoram Koren. Scalability planning for reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 31(2):83–91, 2012.

[86] G Putnik, A Sluga, H ElMaraghy, R Teti, Y Koren, T Tolio, and B Hon. Scalability in manufacturing systems design and operation: State-of-the-art and future developments roadmap. *CIRP Annals-Manufacturing Technology*, 62(2):751–774, 2013.

[87] Crawford S Holling. Resilience and stability of ecological systems. *Annual Review of Ecology and Systematics*, pages 1–23, 1973.

[88] Byeng D Youn, Chao Hu, and Pingfeng Wang. Resilience-driven system design of complex engineered systems. *Journal of Mechanical Design*, 133(10):101011, 2011.

[89] Wen-Jun Zhang and Yingzi Lin. On the principle of design of resilient systems–application to enterprise information systems. *Enterprise Information Systems*, 4(2):99–110, 2010.

[90] Martin Christopher and Helen Peck. Building the resilient supply chain. *The International Journal of Logistics Management*, 15(2):1–14, 2004.

[91] Yossi Sheffi and James B Rice Jr. A supply chain view of the resilient enterprise. *MIT Sloan Management Review*, 47(1), 2005.

[92] Walid Klibi, Alain Martel, and Adel Guitouni. The design of robust value-creating supply chain networks: a critical review. *European Journal of Operational Research*, 203(2):283–293, 2010.

[93] Yao Hu, Jingshan Li, and Lawrence E Holloway. Resilient control for serial manufacturing networks with advance notice of disruptions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(1):98–114, 2013.

[94] Brian Tomlin. On the value of mitigation and contingency strategies for managing supply chain disruption risks. *Management Science*, 52(5):639–657, 2006.

[95] H Hishamuddin, Ruhul A Sarker, and Daryl Essam. A disruption recovery model for a single stage production-inventory system. *European Journal of Operational Research*, 222(3):464–473, 2012.

[96] Gregory A DeCroix. Inventory management for an assembly system subject to supply disruptions. *Management Science*, 59(9):2079–2092, 2013.

[97] Diamantidis C Alexandros and Papadopoulos T Chrissoleon. Exact analysis of a two-workstation one-buffer flow line with parallel unreliable machines. *European Journal of Operational Research*, 197(2):572–580, 2009.

[98] Jialu Liu, Sheng Yang, Aiguo Wu, and S Jack Hu. Multi-state throughput analysis of a two-stage manufacturing system with parallel unreliable machines and a finite buffer. *European Journal of Operational Research*, 219(2):296–304, 2012.

[99] Mitchell H Burman. *New results in flow line analysis.* PhD thesis, Massachusetts Institute of Technology, 1995.

[100] Alain Patchong and Didier Willaeys. Modeling and analysis of an unreliable flow line composed of parallel-machine stages. *IIE Transactions*, 33(7):559–568, 2001.

[101] Chao-Bo Yan and QC Zhao. A unified effective method for aggregating multi-machine stages in production systems. *IEEE Transactions on Automatic Control*, 58(7):1674–1687, 2013.

[102] Jay Lee, Jun Ni, Dragan Djurdjanovic, Hai Qiu, and Haitao Liao. Intelligent prognostics tools and e-maintenance. *Computers in Industry*, 57(6):476–489, 2006.

[103] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.