Analysis and Design of Manufacturing Systems with Multiple-Loop Structures

by

Zhenyu Zhang

B.S., Shanghai Jiao Tong University, China (2000)
M.S., Shanghai Jiao Tong University, China (2002)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author .....
Department of Mechanical Engineering
May 10, 2006

Certified by .....
Stanley B. Gershwin
Senior Research Scientist, Department of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Lallit Anand
Chairman, Department Committee on Graduate Students

To Mom, Dad, and Grandparents

and

To Mr. and Mrs. Kwok, Wo Kong and Tzu-Wen

Nothing in the world can take the place of Persistence.

Talent will not; nothing is more common than unsuccessful men with talent.

Genius will not; unrewarded genius is almost a proverb.

Education will not; the world is full of educated failure.

Keep Believing.

Keep Trying.

Persistence and Determination alone are omnipotent.

— Calvin Coolidge

Analysis and Design of Manufacturing Systems with Multiple-Loop Structures

by

Zhenyu Zhang

Submitted to the Department of Mechanical Engineering
on May 10, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

Kanban means card or token. A kanban-controlled production system is one where the flow of material is controlled by the presence or absence of a kanban, and where kanbans travel in the system according to certain rules.

The study of kanban-controlled production systems can be traced back to the Toyota Production System in the 1950s. The classic kanban-controlled system was designed to realize Just-In-Time (JIT) production. Kanban-controlled production systems, though pervasively used in industry and studied for decades, are not well understood quantitatively yet.

The essence of kanban-controlled production systems is to use single or multiple closed loops to provide information flow feedback using kanbans. By doing this, the systems keep tight controls over inventory levels, while providing satisfactory production rates. The goal of this research is to study the behavior of the class of manufacturing systems with multiple closed loop structures and explore the applications in design and operational control of production systems using multiple-kanban loops. To do so, stochastic mathematical models and efficient analytical methods for evaluating the performance of systems with complex structures are required.

In this thesis, we present an assembly/disassembly network model which integrates the control information flows with material flows. Blocking and starvation properties due to machine failures in a system are analyzed by establishing an efficient underlying graph model of the system. Based on the mathematical model and blocking and starvation properties, efficient and accurate algorithms are developed for evaluating the performance of systems with arbitrary multiple-loop structures. We study the behavior of multiple-loop structures and develop intuition for optimal design and operational control using multiple-kanban loops. Some practical guidelines for the design and control of production systems using multiple-kanban loops are provided at the end.

Thesis Supervisor: Stanley B. Gershwin
Title: Senior Research Scientist, Department of Mechanical Engineering

Doctoral Thesis Committee

Stanley B. Gershwin (chair)

Senior Research Scientist, Department of Mechanical Engineering

Stephen C. Graves

Abraham J. Siegel Professor of Management Science, Professor of Mechanical Engineering and Engineering Systems

David E. Hardt

Professor of Mechanical Engineering and Engineering Systems

Acknowledgments

I would like to take this opportunity to express my acknowledgement to the people and institutions that have supported me, in many forms, during my life at MIT.

First and foremost, I would like to thank my advisor Dr. Stanley B. Gershwin for his inspiration, guidance, patience, and encouragement. Stan, you are a fantastic advisor, mentor, and friend. Without you, I could not have achieved so much along this journey. It is a privilege to have the other members in my thesis committee, Professor Stephen C. Graves and Professor David E. Hardt, who provided me with their valuable suggestions.

Thanks also go to my colleagues, Irvin Schick, Joogyoon Kim, Youngjae Jang, Chiwon Kim, Ketty Tanizar, Seok Ho Chang, Alain Patchong, Gianpietro Bertuglia, Andrea Poffe, Rongling Yang, Philippe Frescal. Their thoughtful comments and good cheer have been greatly appreciated. In addition, I would express my gratitude to Karuna Mohindra, Leslie Regan, Joan Kravit, Professor Lallit Anand, and Professor Ain Sonin in the Mechanical Engineering Department. Their help were of tremendous value to my study at MIT. My thanks are also extended to the faculty members in Shanghai Jiao Tong University (SJTU). Special thanks to Professor Chengtao Wang and Professor Jianguo Cai.

Out of academia, most of all, I would thank all my family for their never-ending encouragement and love. I also feel very grateful to Mr. and Mrs. Kwok, for their invaluable support, advice and encouragement that make me always achieve higher in my life.

There are numerous friends I wish to thank. All of you make my journey at MIT an enjoyable and unforgettable one. To friends at MIT, Andy Wang, Wei Mao, Qi Wang, Yuan(Arthur) Cao, Xiaoyu Shi, Bin Chen, Chen Ma, Yong Zhao, Ting Zhu, Yuhua Hu, Miao Ye, Huan Zhang, Lin Han, Minliang Zhao, Xixi Chen, etc. To friends at MIT Sloan, Jun He, Mengjin Tian, Jingan Liu, etc. To all fellows in MIT Chinese Student and Scholar Association (CSSA), especially in Career and Academic Development (CAD) department. To my friends in Greater Boston area and SJTU

fellows — You know who you are. Special thanks to Yakun Zhou, Zhang Chen, Ji Bian, Ye Zhou, Hongxing Chang, Jingjing Zheng, and Hui Liu. I would thank Min Qin, Mei Chen, Jie (Janet) Lin, Jing Lv, Hao Zhu, Xinyu Chen, Conghe Wang, for hosting me during my travel around North America. Thanks also to my students at SMA Classes 2005 and 2006. Specials thanks to Yang Ni and Jun Rao for your host during my short stay in Singapore.

I want to give special thanks to Sudhendu Rai (Xerox Corporation) and Henghai (Helen) Liu (Johnson & Johnson) for the great summer intern opportunities they offered me. I am very proud of the achievements that we have made. I also thank my friends in Intel, Guoquan Liu, Tao (Mike) Zhang, for their warm reception during Stan's trip to Shanghai in 2005.

Last but certainly not least, special thanks to my unique Cao Zhang for her affection, inspiration, and support throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Summary of Kanban Systems

Kanban means card or token. A *kanban-controlled production system* is one where the flow of material is controlled by the presence or absence of a kanban, and where kanbans travel in the system according to certain rules.

The study of kanban-controlled systems can be traced back to the Toyota Production System in the 1950s. The classic kanban-controlled system was designed to realize Just-In-Time (JIT) production, keeping a tight control over the levels of individual buffers, while providing a satisfactory production rate (Figure 1-1).

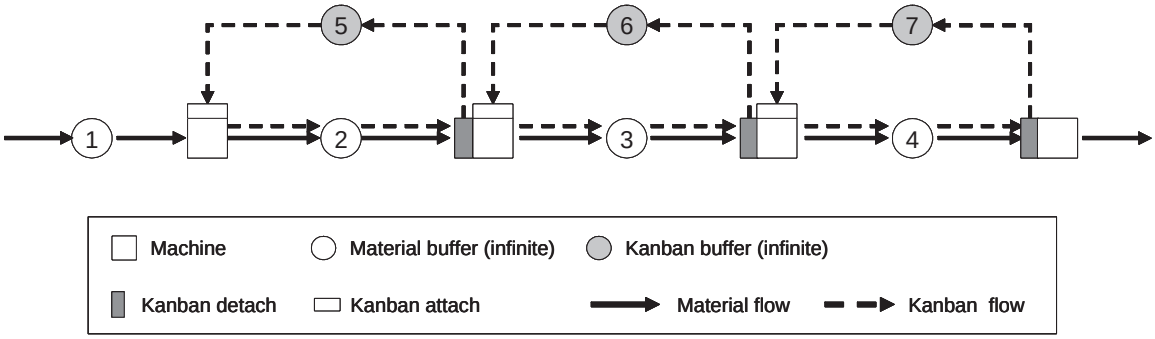


Figure 1-1: Classic kanban-controlled system

From the perspective of control, feedback is implemented at each processing stage by circulating kanbans from its downstream buffer to the upstream of the stage. The circulation routes of kanbans form one closed loop per stage. Each stage has one control parameter: the number of kanbans. In the classic kanban-controlled system, a constant number of kanbans is imposed to limit the level of the buffer inventory in each closed loop. An infinite buffer controlled by a closed loop is equivalent to a finite buffer since the maximal inventory level can have in the infinite buffer is limited by the number of kanbans. The size of the finite buffer is equal to the number of kanbans in the loop.

There are several variations of kanban control widely used in industry, such as CONWIP control and hybrid control (Figure 1-2). Unlike the classic kanban-controlled system which uses kanbans to regulate the levels of individual buffers, the other two systems in Figure 1-2 implement a control strategy which limits the sum of the buffer levels within the large closed loop. Feedback is implemented from the last stage to the first stage.

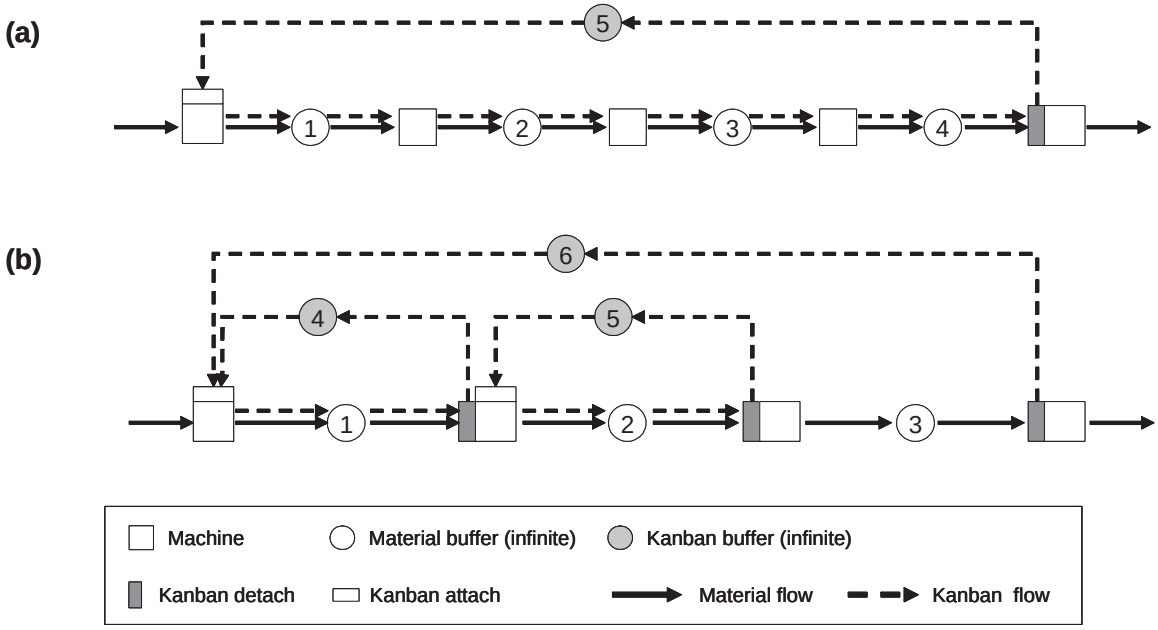


Figure 1-2: Variations of kanban systems
(a) CONWIP-controlled system; (b) Hybrid-controlled system

In Figure 1-1 and Figure 1-2, detaching and attaching kanbans with workpieces

are separate operations before the work pieces proceed to machine operations. Notice that the operations of detaching and attaching kanbans are instantaneous compared to machine operations. Therefore, we integrate the kanban detach, kanban attach, and machine operation as one single operation (Figure 1-3). In addition, when kanbans are attached to workpieces, an integrated flow is used to represent two separate kanban and material flows in Figure 1-1 and Figure 1-2. In the remainder of this thesis, we use rectangles to represent integrated operations, and arrows to represent integrated flows.

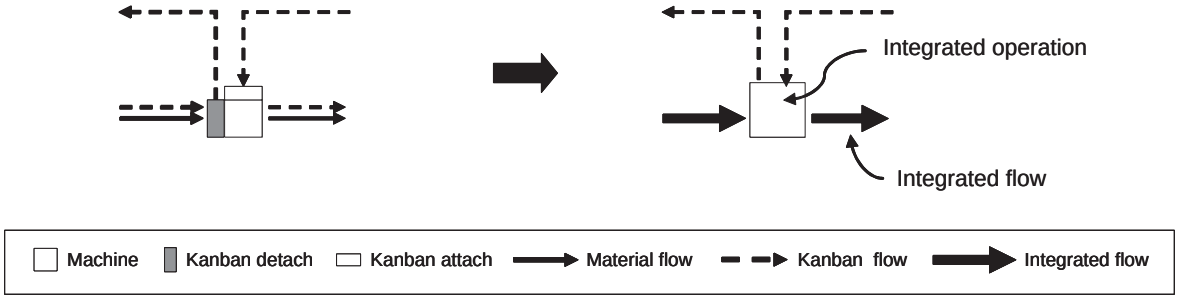


Figure 1-3: Integrated operation including kanban detach, kanban attach, and machine operation

1.1.2 Essence of Kanban Control

Consider the system in Figure 1-1. Once a part enters a closed loop, a kanban card is attached to it. The kanban card is detached from the part when it leaves the closed loop and proceeds to the next stage. The number of kanbans within the closed loop is constant. We define it as the *invariant* of the loop. Similarly, when we look at the CONWIP loop in Figure 1-2(a), kanban cards are attached to the parts at the first stage of the production line while they are detached from the parts at the last stage. The total number of kanbans circulated within the CONWIP loop gives the loop invariant I:

$$I = b(1,t) + b(2,t) + b(3,t) + b(4,t) + b(5,t) \tag{1.1}$$

in which $b(i,t)$ is the level of buffer B_i at time t.

The invariant imposes an upper limit of the buffer levels within the closed loop. For example, the total number of parts W allowed in the large CONWIP loop is constrained by:

$$W = b(1,t) + b(2,t) + b(3,t) + b(4,t) \leq I \tag{1.2}$$

More generally, systems using kanban controls can be represented as a set of systems with multiple-loop structures. Each closed loop has a loop invariant. In the remainder of this thesis, material and kanban buffers are assumed to be finite because it is impossible to have infinite buffers in real world. A finite buffer is equivalent to an infinite buffer controlled by a classic kanban loop.

1.1.3 Importance of Multiple-Loop Structures

To control a given production system, a variety of kanban control methods can be used. Classic kanban control, CONWIP control and hybrid control are compared by Bonvik (1996), Bonvik et al. (1997), and Bonvik et al. (2000). The hybrid control method is demonstrated to have the best inventory control performance among these three control methods. Therefore, to study the design of control structures is valuable for developing insights into operational control.

After we determine the control structure, the design parameters of the closed loop, such as the number of kanbans, are also related to the system's performance and cost. Consider a production line with pallets in Figure 1-4. CONWIP control is implemented by circulating pallets instead of kanban cards. Raw parts are loaded on pallets at machine M_1 and unloaded from pallets at machine M_{10}.

In the system, all machines are identical with failure rate $p = 0.01$, repair rate $r = 0.1$, and processing rate $\mu = 1.0$. All the material buffer sizes are 20. We vary the number of pallets Q and the size of pallet buffer B to generate five cases. The parameters and performance measures in terms of production rate and total inventory level are summarized in Table 1.1. The performance measures are plotted in Figure 1-5.

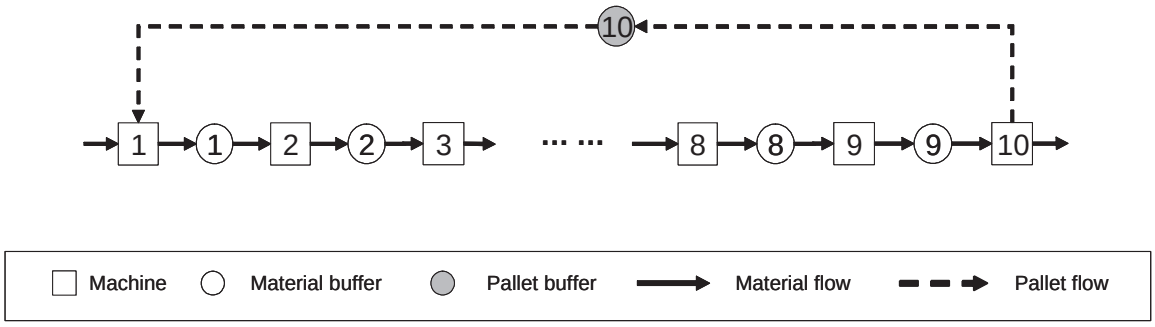


Figure 1-4: CONWIP control of a production line with pallets

Case Number	Q	B	Production rate P	Total inventory level T_{inv}
1	200	200	0.800288	90.1366
2	50	70	0.786535	66.41147
3	60	50	0.763461	53.30389
4	30	30	0.715328	38.18662
5	20	15	0.646548	20.80521

Table 1.1: Design parameters and performance measures of a CONWIP-controlled production line

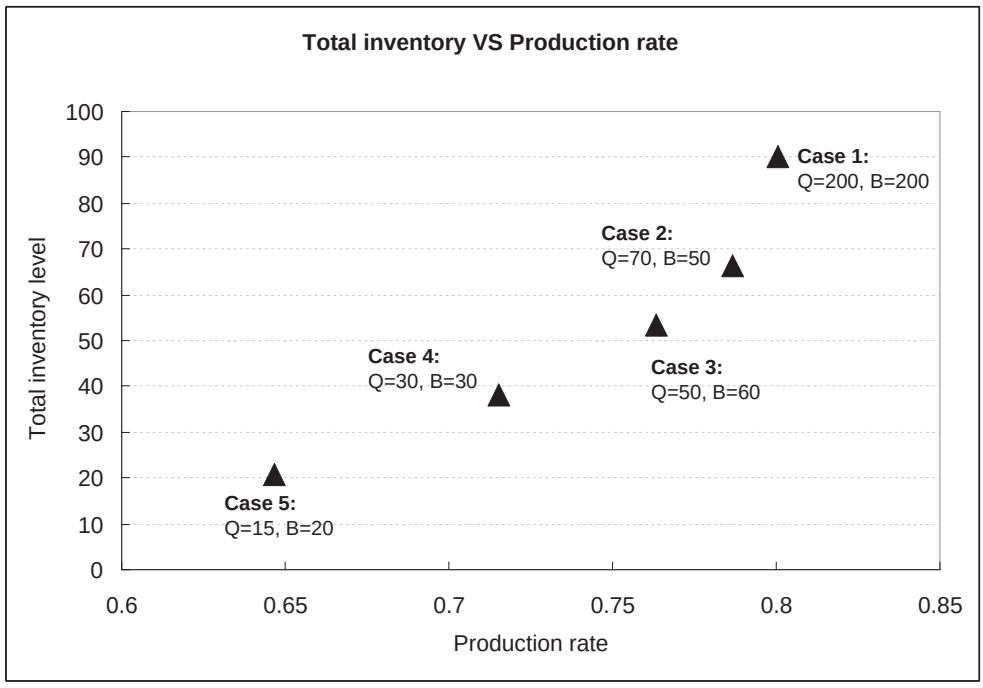


Figure 1-5: Production rate and total inventory level of CONWIP control while varying number of pallets Q and size of pallet buffer B

As these pallets cost money and take up space, the optimal selection of design parameters, such as the number of pallets and the storage buffer space of the pallets, has a significant dollar impact in profit. We formulate the profit Y as a function of production rate P, total inventory level T_{inv}, number of pallets Q, and size of pallet buffer B:

$$Y = C_P P - C_T T_{inv} - C_Q Q - C_B B \tag{1.3}$$

where C_P is margin per unit production rate; C_T, C_Q, and C_B are cost coefficients of inventory, pallet, and pallet buffer, respectively.

We perform a set of scenario analyses by varying the margin and cost coefficients. The profits of five cases in six scenarios are listed in Table 1.2. We observe that, when pallet cost or pallet buffer cost is high (Scenarios 3 or 6), the optimal solution is Case 5, which has the smallest number of pallets and smallest pallet buffer.

Scenario Number	Coefficients				Profit of 5 cases					Optimal Case
	C_P	C_T	C_Q	C_B	Case 1	Case 2	Case 3	Case 4	Case 5	
1	1000	1	0	1	510.15	650.12	660.16	647.14	610.74	3
2	1000	1	0	2	310.15	580.12	610.16	617.14	595.74	4
3	1000	1	0	10	-1289.85	20.12	210.16	377.14	475.74	5
4	1000	1	1	1	310.15	600.12	600.16	617.14	590.74	4
5	1000	1	2	1	110.15	550.12	540.16	587.14	570.74	4
6	1000	1	10	1	-1489.85	150.12	60.16	347.14	410.74	5

Table 1.2: Profits of five cases in six scenarios

In summary, the performance and cost of multiple-kanban production systems depend not only on control structures, but also on parameters such as number of kanbans. Therefore, an exhaustive study of the behavior of multiple-loop structures is needed. This study is challenging but highly valuable. It will provide a theoretical basis and practical guidelines for factory design and operational control using multiple-loop structures.

1.2 Background and Previous Work

In recent years there has been a large amount of literature on the analysis and design of kanban systems. The methods can be categorized into simulation and analytical methods. As the analysis and design of kanban systems usually involve evaluating a large number of variations with different structures and parameters, analytical methods are much more promising in terms of computational efficiency. Another advantage of analytical methods is their effectiveness in investigating the properties of multiple-loop structures and developing intuition for system design and control.

In this section, we review the development of manufacturing systems engineering with focus on the analytical work. Key issues and difficulties in analyzing systems with multiple-loop structures are explained. The review helps further understand the motivation of this research.

1.2.1 Fundamental Models and Techniques

A large number of models and methods have been developed to address the design and operations of manufacturing systems. An extensive survey of the literature of manufacturing systems engineering models up to 1991 appeared in Dallery and Gershwin (1992). More recent surveys can be found in Gershwin (1994), Altiok (1997), and Helber (1999). A review focused on MIT work and closely related research was presented by Gershwin (2003).

Initially, the research of this area started from modeling two-machine transfer lines with unreliable machines and finite buffers using Markov chains (Buzacott and Shanthikumar 1993). When Markov chains are used to model the stochastic behavior inherent in larger manufacturing systems, the scale and complexity of these systems often result in a huge state space and a large number of transition equations.

Decomposition was invented as an approximation technique to evaluate complex manufacturing systems by breaking them down into a set of two-machine lines (building blocks). These building blocks can be evaluated analytically by using methods in Gershwin (1994). Gershwin (1987) was one of the first authors to analyze finite-

buffer production lines by developing an approximate decomposition method. Mascolo et al. (1991) and Gershwin (1991) extended the decomposition method to analyze tree structured assembly/disassembly networks.

Consider the decomposition of a long production line in Figure 1-6. Each buffer in the original system has a corresponding two-machine line (building block). The buffer of this building block has the same size as the original buffer. In each building block, its upstream and downstream machines are pseudo-machines which approximate the behavior observed in the original buffer. Each pseudo-machine is assigned one failure mode. The building blocks are evaluated iteratively and the failure rate and repair rate of each failure mode are updated till convergence.

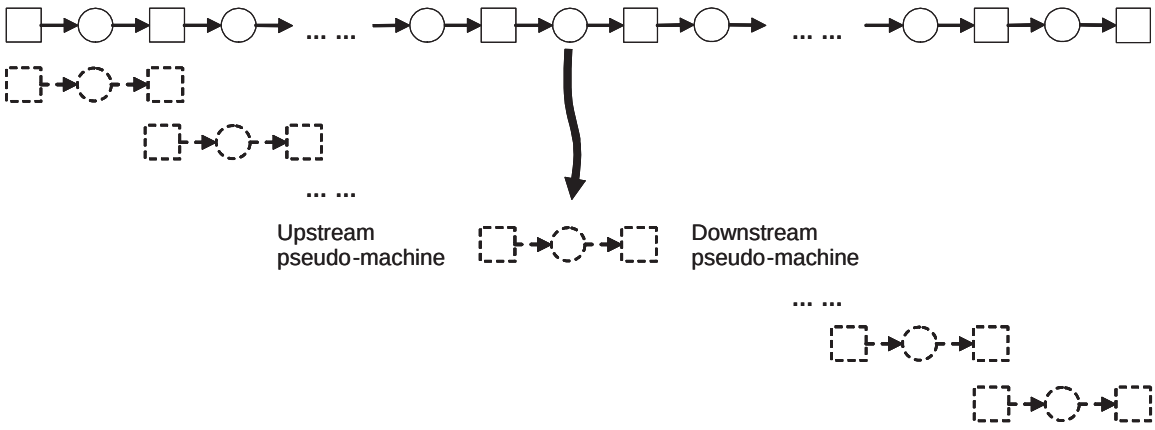


Figure 1-6: Long production line decomposition

1.2.2 Decomposition Using Multiple-Failure-Mode Model

A new decomposition method was presented by Tolio and Matta (1998). This method models the two-machine lines (building blocks) by assigning multiple failure modes to the pseudo-machines, instead of using single failure mode for each pseudo-machine.

In this model, the downstream pseudo-machine is assigned all the failure modes in the original system that can cause the original buffer to be full. The failure modes in the original system that can cause the original buffer to be empty belong to the upstream pseudo-machine.

For example, the decomposition of a six-machine production line using Tolio's multiple-failure-mode model is shown in Figure 1-7. When the tandem line is decomposed into a set of building blocks, the building block corresponding to buffer B_2 approximates the behavior observed by a local observer at B_2. The failure modes of machines M_1 and M_2 are assigned to the upstream pseudo-machine $M^u(2)$ as they can cause B_2 to be empty; while the failure modes of M_3, M_4, M_5, and M_6 are assigned to the downstream pseudo machine $M^d(2)$ as they can cause B_2 to be full.

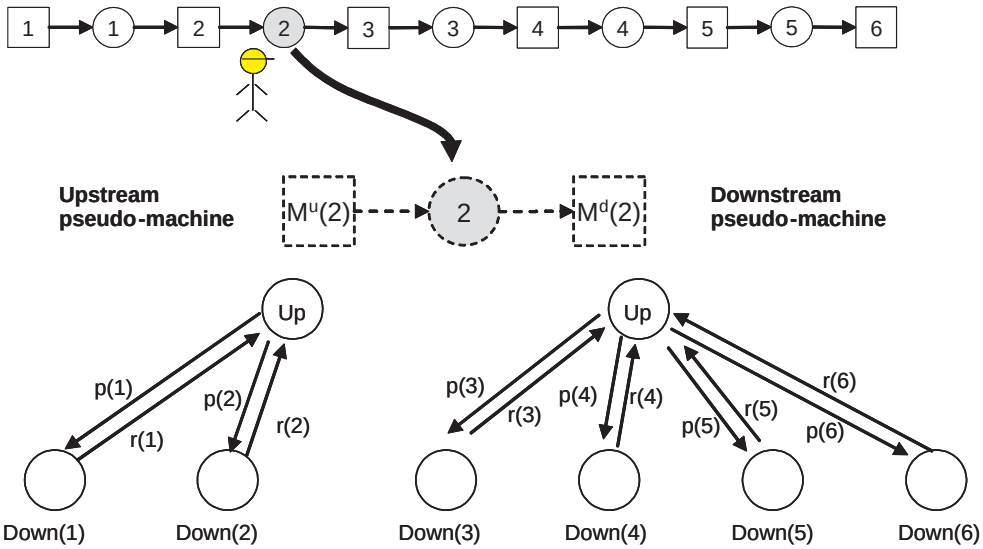


Figure 1-7: Decompose a six-machine production line using multiple-failure-mode model

Up to this point, the systems we discussed are acyclic systems. In other words, there is no closed loop in these systems.

1.2.3 Systems with Closed Loops

In the studies of systems with closed loops, Frein et al. (1996), Werner (2001) and Gershwin and Werner (2006) developed an efficient method to evaluate large single-loop systems using the decomposition method based on multiple-failure-mode model. Levantesi (2001) extended it to evaluate small multiple-loop systems. However, this method is not able to provide satisfactory speed and reliability while evaluating large-scale multiple-loop systems. Levantesi's method demonstrated the feasibility

of his approach. But it had a very inefficient method for analyzing the propagation of blocking and starvation.

In addition, a systematic understanding of the behavior of multiple-loop structures has not been developed yet. It is important as we need it for developing methods for optimal design and control using multiple closed loops. Key issues include how to choose control structures, and determine kanban quantities. In the literature, Monden (1983) presents the Toyota approach for determining the number of kanbans for each stage. This method, however, does not fully consider the randomness due to machine failures and depends on subjective parameters, such as safety factor. Several methods are presented in Hopp and Spearman (1996), Hopp and Roof (1998), Ryan et al. (2000), and Ryan and Choobineh (2003) to determine WIP level for CONWIP-controlled systems. These methods have the disadvantages that the manufacturing systems for control are simple production lines, and the methods are limited to specified control structure — CONWIP. There are some studies on the variations of kanban control stuctures in Gaury et al. (2000, 2001). However, these approaches are simulation-based and the number of variations is limited.

In summary, the studies in the literature have the following two limitations:

- The manufacturing systems for control have relatively simple structures, such as production lines or simple assembly systems. In fact, manufacturing systems are much more complicated in real factories.

- The control structures used are classic methods, such as single-stage kanban, CONWIP. Multiple-loop control structures might have better performance than the classic ones.

One of the reasons for these limitations is that there are no efficient methods for evaluating complex manufacturing systems with multiple-loop structures. Therefore, an efficient evaluation method is desired such that the behavior of multiple-loop control can be explored to help design and control complex manufacturing systems.

1.3 Research Goal and Contributions

This thesis is intended to investigate the behavior of multiple-loop structures, to develop mathematical models and efficient analytical methods to evaluate system performance, and to help design factories. Specifically, the contributions of this thesis include:

- A unified assmebly/disassembly network model to represent multiple-loop structures which integrates information flows with material flows. (Refer to Chapter 2.)
- A systematic method to analyze blocking and starvation propagation based on graph theory. (Refer to Chapters 3 and 4.)
- An efficient algorithm to evaluate large-scale assembly/disassembly systems with arbitrary topologies. (Refer to Chapters 5 and 6.)
- Numerical experiments that demonstrate the algorithm is accurate, reliable, and fast. (Refer to Chapter 7.)
- Characteristic behavior analysis of systems with multiple-loop structures. (Refer to Chapter 8.)
- Insights for optimal design and control of production lines and assembly systems. (Refer to Chapter 8.)
- Practical guidelines for the design and operational control of multiple-kanban production systems. (Refer to Chapter 8.)

1.4 Thesis Outline

We introduce assembly/disassembly networks with multiple-loop structures in Chapter 2. This model is used to integrate information flows with material flows. To analyze the machine failure and blocking and starvation propagation in systems with

complex topologies, in Chapter 3, we develop a graph model. In Chapter 4, the analysis of blocking and starvation is discussed based on the graph model. The decomposition method is provided in Chapter 5. In Chapter 6, we present an efficient algorithm for evaluating assembly/disassembly networks with arbitrary topologies. Chapter 7 discusses the performance of the algorithm in terms of accuracy, reliability and speed. In Chapters 8, we present the behavior analysis and design insights on production control using multiple-loop structures. Finally, a summary of the contributions of this research, and a brief description of future research directions are presented in a conclusion in Chapter 9.

Chapter 2

Assembly/Disassembly Networks with Multiple-Loop Structures

A set of production control policies using multiple closed-loop structures, such as classic kanban, CONWIP, hybrid control, have been invented and implemented. Kanban cards perform as media which convey information flows to control material flows. In this chapter, we develop a model to provide an integrated view of material flows and information flows.

First, in Section 2.1, assembly/disassembly networks with multiple-loop structures are presented as a unified model to integrate information flows with material flows. Two important properties — assembly/disassembly operations, and loop invariants — are discussed. In Section 2.2, we present an approach to evaluating system performance. The challenges in evaluation are discussed in Section 2.3.

2.1 A Unified Model

Consider the system in Figure 2-1. The machines in the system are only allowed to perform single part, assembly, or disassembly operations. (We do not include merges or splits of flows.) Material flow is disassembled into two sub-flows at machine M_B and these two sub-flows are reassembled at machine M_E. The processes in the lower sub-flow requires pallets which are circulated from machine M_D to machine M_C. A

CONWIP control loop is implemented between machine M_G and M_A. Demand tokens are generated by machine M_H for base stock control at machines M_A and M_F. Four closed loops are formed in the system.

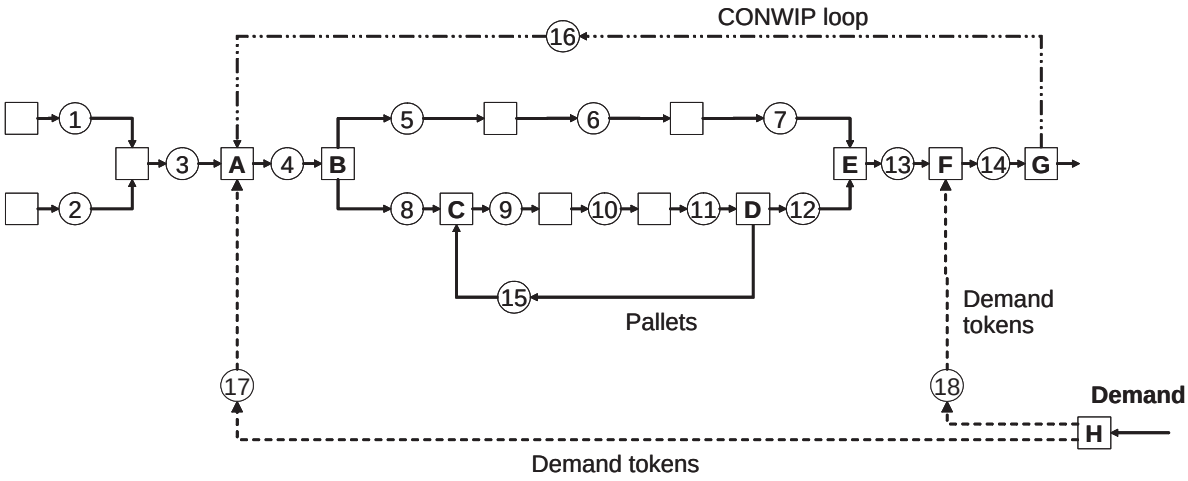


Figure 2-1: An assembly/disassembly network with four closed loops

Four loops can be identified in the network:

- Loop 1: $B_5 \rightarrow B_6 \rightarrow B_7 \rightarrow B_{12} \rightarrow B_{11} \rightarrow B_{10} \rightarrow B_9 \rightarrow B_8$;
- Loop 2: $B_9 \rightarrow B_{10} \rightarrow B_{11} \rightarrow B_{15}$;
- Loop 3: $B_4 \rightarrow B_8 \rightarrow B_9 \rightarrow B_{10} \rightarrow B_{11} \rightarrow B_{12} \rightarrow B_{13} \rightarrow B_{14} \rightarrow B_{16}$;
- Loop 4: $B_4 \rightarrow B_8 \rightarrow B_9 \rightarrow B_{10} \rightarrow B_{11} \rightarrow B_{12} \rightarrow B_{13} \rightarrow B_{18} \rightarrow B_{17}$.

2.1.1 Assembly/Disassembly Operations

Traditionally, assembly/disassembly is used to describe a process involving two or more real workpieces. However, this need not to be the case. Various meanings of assembly/disassembly operations are represented in the unified graph of Figure 2-1.

Sometimes, a workpiece is assembled to a pallet or fixture when it enters a system. After going though a set of processes, the workpiece is disassembled from the pallet. Therefore, assembly/disassembly takes place between a real workpiece and a fixture or pallet. For example, machines M_B and M_E in Figure 2-1 perform disassembly and

assembly of real parts. Machines M_C and M_D disassemble and assemble real parts with pallets.

When we consider kanban control, the attach and detach operations between kanbans and workpieces are actually assembly/disassembly operations. By imagining the kanban as a medium which conveys control information, the assembly/disassembly operations take place between information flows and material flows. For example, at machine M_G, kanbans are disassembled from real workpieces; while at machine M_A, kanbans are assembled with real workpieces. The kanbans take the information from downstream machine M_G to upstream machine M_A such that new workpieces are authorized to be dispatched from buffer B_3.

More interestingly, Gershwin (2000) shows that the demand information can be embedded into a production line with material flow by using virtual assembly/disassembly machines. In Figure 2-1, machine M_H is a demand machine which generates demand token flows by disassembly. The demand tokens are assembled with real parts at machines M_A and M_F. The stock level between M_A and M_F is controlled.

Assembly/disassembly machines in assembly/disassembly networks have the following property:

Equality of Flows When an assembly/disassembly machine completes one operation, each of its upstream buffers is decreased by one workpiece, while each of its downstream buffers is increased by one workpiece. In any given time frame, the amount of material coming from each upstream buffer of an assembly/disassembly machine is equal to the amount of material going into each downstream buffer of the machine.

Here we illustrate one example of the Extended Kanban Control Systems (EKCS) described in Dallery and Liberopoulos (2000). The EKCS is modeled as an assembly/disassembly network in Figure 2-2. Kanbans are circulated between machines and demand tokens are used to implement base stock control. A machine in the system could assemble kanbans (e.g. demand tokens) with real workpieces, and disassemble

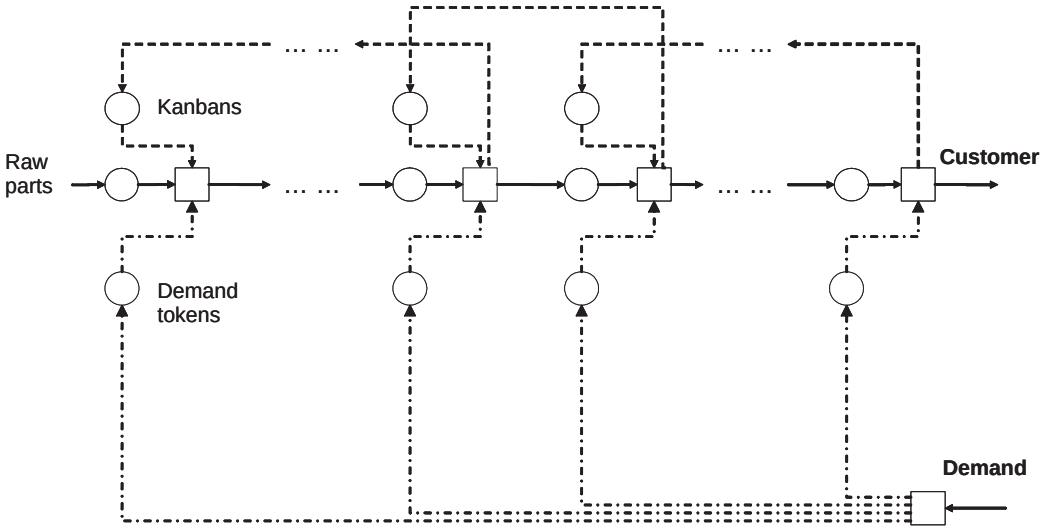


Figure 2-2: An extended kanban control system (EKCS)

kanbans from real workpieces.

2.1.2 Loop Invariants

In Chapter 1, we briefly introduce the loop invariant by illustrating a CONWIP loop example. Here, we provide a general definition of a loop invariant.

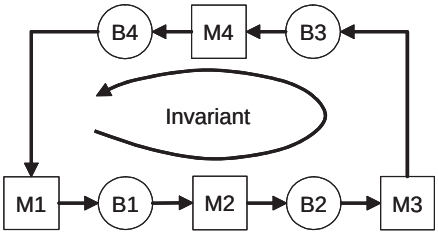
In Figure 2-3(a), the total number of workpieces travelling in the closed loop is a constant of time. The constant number of workpieces is the loop invariant:

$$I = b(1,t) + b(2,t) + b(3,t) + b(4,t) \tag{2.1}$$

in which $b(i,t), i = 1, ..., 4$ are the buffer levels of $B_i, i = 1, ..., 4$ at time t.

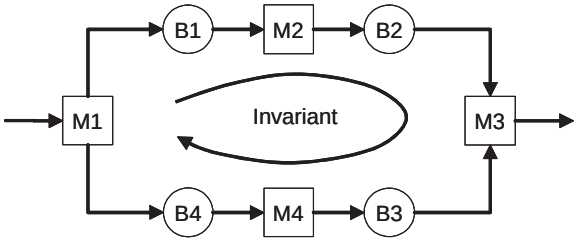
In the above loop, the flow follows a single direction. In other words, there is no direction change within the closed loop and the flow is unidirectional. When the flow in the closed loop is not unidirectional, we can also define the invariant but in a more complex way.

Consider the variation of single loop in Figure 2-3(b), which has flow direction changes at M_1 and M_3. Although there is no part circulated in this assembly/disassembly system, an invariant still can be found. Observe the disassembly at



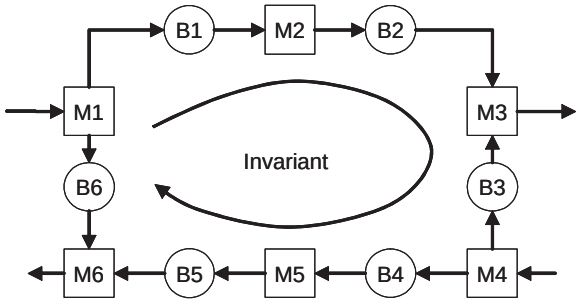
Invariant = b(1,t) + b(2,t) + b(3,t) + b(4,t)

(a)



Invariant = b(1,t) + b(2,t) - b(3,t) - b(4,t)

(b)



Invariant = b(1,t) + b(2,t) - b(3,t) + b(4,t) + b(5,t) - b(6,t)

(c)

Figure 2-3: Loop invariants of single-loop systems

machine M_1. According to the property of 'equality of flows', whenever one sub-part goes into the upper branch, there must be the other sub-part going into the lower branch. Similar things happen to the assembly at machine M_3. Therefore, the difference between the sum of the buffer levels of the upper branch and that of the lower branch is a constant of time. Usually, the constant is zero in disassembly/reassembly systems. However, it can be non-zero value which depends on the initial condition of the distribution of buffer levels. For the loop in Figure 2-3(b), an invariant I can be expressed as follows:

$$I = b(1,t) + b(2,t) - b(3,t) - b(4,t) \tag{2.2}$$

A more complicated variation of a single loop is shown in Figure 2-3(c). The flow direction changes four times in the loop. Loop orientation, as an abstract concept, is introduced for the purpose of defining the invariant. An invariant can be then obtained by adding the levels of buffers whose direction agree with the loop orientation and subtracting the levels of all the rest. For example, assume the loop orientation is clockwise, the invariant of the loop in Figure 2-3(c) is given by

$$I = b(1,t) + b(2,t) - b(3,t) + b(4,t) + b(5,t) - b(6,t) \tag{2.3}$$

A general definition of loop invariant is summarized:

- Step 1: Define the loop orientation
- Step 2: Given a buffer in the loop
  - If the flow though the buffer agrees with the loop orientation, add the buffer level to the invariant.
  - If the flow though the buffer does not agree with the loop orientation, subtract the buffer level from the invariant.

Incidentally, the number of direction changes in a closed loop is always an even number.

Recall the four loops in the assembly/disassembly network of Figure 2-1. Their loop invariants are defined as follows:

$$
\begin{aligned}
I_1 &= b(5,t) + b(6,t) + b(7,t) - b(12,t) - b(11,t) - b(10,t) - b(9,t) - b(8,t) \\
I_2 &= b(9,t) + b(10,t) + b(11,t) + b(15,t) \\
I_3 &= b(4,t) + (8,t) + b(9,t) + b(10,t) + b(11,t) + b(12,t) + b(13,t) + b(14,t) + b(16,t) \\
I_4 &= b(4,t) + b(8,t) + b(9,t) + b(10,t) + b(11,t) + b(12,t) + b(13,t) - b(18,t) + b(17,t)
\end{aligned}
\tag{2.4}
$$

Loop invariants, as the basic property of closed loops, constrain the distribution of buffer levels and result in complicated behavior of blocking and starvation propagation. At this point, we only introduce the concept of loop invariants. Several key issues of the complicated behavior of blocking and starvation are discussed in Section 2.3. Detailed analysis is presented in Chapters 4 and 5.

In summary, *assembly/disassembly networks* are defined as a class of systems composed of finite buffers and unreliable machines which can perform single-stage operation, assembly/disassembly, but *NOT* flow merge or split and *NOT* reentrant flow. The definition of assembly/disassembly operations is extended to include both material flows and information flows. Kanban, CONWIP, hybrid and other information-based control methods can be modeled as assembly/disassembly networks with multiple-loop structures.

2.2 Approach to Evaluation

Given an assembly/disassembly network with unreliable machines and finite buffers, we are interested to know the system performance in terms of production rate and average buffer levels. In this section, we presented an approach for evaluating assembly/disassembly networks with arbitrary topologies. The detailed procedures of this evaluation approach are presented in Section 6.2.

2.2.1 Decomposition

Markov chain processes are used to model stochastic behavior of machine failures and repairs in manufacturing systems. However, the state space is huge when system is large-scale. Decomposition has been developed as an approximation technique to evaluate large-scale complex manufacturing systems. Decomposition algorithms have been developed for production lines and tree-structured systems in Gershwin (1987), Mascolo et al. (1991), and Gershwin (1991). The multiple-failure-mode model developed by Tolio and Matta (1998) improves the efficiency of long line decomposition, and enables the development of decomposition algorithms of closed-loop systems in Frein et al. (1996), Gershwin and Werner (2006) and Levantesi (2001).

All of the above decomposition techniques break down manufacturing systems into a set of two-machine one-buffer lines (building blocks), which can be evaluated analytically. For each buffer in the original system, a two-machine line is designed to approximate the behavior of flow observed in the buffer. The upstream pseudo-machine of the two-machine line approximates the failures propagated from the upstream portion of the system, while the downstream pseudo-machine approximates the failures propagated from downstream.

A rule for failure modes assignment is developed based on the multiple-failure-mode method in Tolio and Matta (1998):

Failure Modes Assignment Rule Given a building block corresponding to one buffer in the original system, failure modes in the original system which can cause the original buffer to be full are assigned to the downstream pseudo-machine; while the upstream pseudo-machine collects the failure modes in the original system which can cause the buffer to be empty.

When the original buffer is empty, its immediate downstream machine in the original system is starved. When the original buffer is full, its immediate upstream machine in the original system is blocked. Therefore, before we decompose the system into building blocks, we should first analyze the properties of blocking and starvation

propagation.

2.2.2 Blocking and Starvation Analysis

The phenomena of blocking and starvation result from the propagation of flow disruption due to unreliable machines and finite buffers. For a given machine in the system, suppose one or more of its immediate downstream buffers is full. The machine is forced to stop processing parts. This situation is defined as *blocking*. Similarly, the situation of *starvation* is defined when the machine is forced to stop if one or more of its immediate upstream buffers is empty.

To obtain the blocking and starvation properties, we are interested to know the *'Machine failure – Buffer level' relationship* in the limiting propagation state: given a buffer, which machine failures in the system can cause the buffer to be full, and thus block the immediate upstream machines of the buffer? Which machine failures in the system can cause the buffer to be empty, and thus starve the immediate downstream machines of the buffer?

2.2.3 Summary of Evaluation Approach

We summarize the evaluation method for an assembly/disassembly network with arbitrary topology as follows:

- Phase I: Analyze blocking and starvation properties
- Phase II: Decomposition
  - Decompose the system into two-machine one-buffer building blocks;
  - Set up each building block according to blocking and starvation properties;
  - Iteratively evaluate building blocks using multiple-failure-mode model.

2.3 Challenges

When there is no closed loop in assembly/disassembly networks, the properties of blocking and starvation are straightforward and the decomposed building blocks can be easily set up (Gershwin 1991, 1994). However, when there exist closed loops, the invariants of these loops result in complicated blocking and starvation behavior.

2.3.1 Thresholds in Buffers

In systems without closed-loop structures, such as production lines or tree-structured systems, given a single machine failure which lasts for a sufficient amount of time and all other machines are operational, the buffer levels are eventually either full or empty. For systems with closed loops, loop invariants can cause some buffers to be partially filled.

Consider the unidirectional single loop in Figure 2-3(a). Suppose all buffer sizes are 10 and the loop invariant is 22. If M_3 is failed for a sufficient amount of time and all other machines are operational, the buffer levels in the limiting state are:

$$\begin{aligned} b(1,\infty) &= 10; \\ b(2,\infty) &= 10; \\ b(3,\infty) &= 0; \\ b(4,\infty) &= 2; \end{aligned}$$

so that the invariant is satisfied:

$$I = b(1,\infty) + b(2,\infty) + b(3,\infty) + b(4,\infty) = 22$$

B_2 and B_3 are full while B_4 is empty. B_1 is partially filled in order to satisfy the invariant equation. The partially filled level is defined as a *threshold* in the buffer. According to the failure assignment rule described in Section 2.2, in a building block,

the failure modes of M_3 can not be assigned to either upstream pseudo-machine or downstream pseudo-machine. When we model building blocks using Markov chain processes, the methods developed in Gershwin (1994) and Tolio et al. (2002) can not be used because the transitions between the states in which the buffer level is above the threshold have different behavior than those between the states in which the buffer level is under the threshold. Therefore, a new method is needed to deal the thresholds in buffers. Maggio et al. (2006) presents a method for evaluating small loops with thresholds. A more efficient method for evaluating large loops is developed by Gershwin and Werner (2006). It eliminates thresholds by introducing perfectly reliable machines. This method is presented in Section 5.2.

To find the buffer levels in unidirectional loops is still simple. However, when there exist flow direction changes in loops, determining the buffer levels is much more complicated. Consider the single-loop system in Figure 2-3(c). The flow direction changes four times. Suppose the loop invariant is 28 and all buffer sizes are 10. If machine M_3 goes down for a sufficient amount of time and all other machines are operational, the buffer levels in the limiting state are:

$$\begin{aligned} b(1,\infty) &= 10; \\ b(2,\infty) &= 10; \\ b(3,\infty) &= 10; \\ b(4,\infty) &= 8; \\ b(5,\infty) &= 10; \\ b(6,\infty) &= 0; \end{aligned}$$

so that the invariant is satisfied:

$$I = b(1,\infty) + b(2,\infty) - b(3,\infty) + b(4,\infty) + b(5,\infty) - b(6,\infty) = 28$$

The buffer levels depend not only on the buffer sizes and loop invariant, but also

on the direction of flows. The buffer levels can not be easily determined unless the loop is carefully analyzed.

2.3.2 Coupling among Multiple Loops

When two loops have one or more common buffer, these two loops are coupled together. Consider loop 1 and loop 3 in Figure 2-1.

- Loop 1: $B_5 \rightarrow B_6 \rightarrow B_7 \rightarrow B_{12} \rightarrow B_{11} \rightarrow B_{10} \rightarrow B_9 \rightarrow B_8$;
- Loop 3: $B_4 \rightarrow B_8 \rightarrow B_9 \rightarrow B_{10} \rightarrow B_{11} \rightarrow B_{12} \rightarrow B_{13} \rightarrow B_{14} \rightarrow B_{16}$;.

Loop 1 and loop 3 are coupled as they have common buffers B_9, B_{10}, and B_{11}.

As there exist four loops in the system, all the buffer levels in the system should satisfy the invariant equations (2.4). Therefore, the distribution of buffer levels is constrained by the loop invariants and coupling structures.

2.3.3 Routing Buffer of Failure Propagation

In addition to the 'Machine failure – Buffer level' relationship described in Section 2.2.2, we need to know the routing buffer through which the machine failure is propagated. This information is essential for updating the parameters of pseudo-machines while performing iterative iterations during decomposition.

Consider the single loop system in Figure 2-4. All the buffer sizes are 10. The loop invariant is 5. For the building block corresponding to buffer B_1, a single machine failure at machine M_E can cause B_1 to be full. To satisfy the invariant condition, the buffer levels in the limiting state are

$$
\begin{aligned}
b(1,\infty) &= 10; \\
b(2,\infty) &= 10; \\
b(3,\infty) &= 10; \\
b(4,\infty) &= 5; \\
b(5,\infty) &= 10;
\end{aligned}
$$

such that the invariant is satisfied:

$$I = b(2,\infty) + b(3,\infty) - b(5,\infty) - b(4,\infty) = 5 \tag{2.5}$$

There exist two paths which connect B_1 and M_E:

- Path 1: $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow M_E$;
- Path 2: $B_1 \rightarrow B_4 \rightarrow B_5 \rightarrow M_E$;

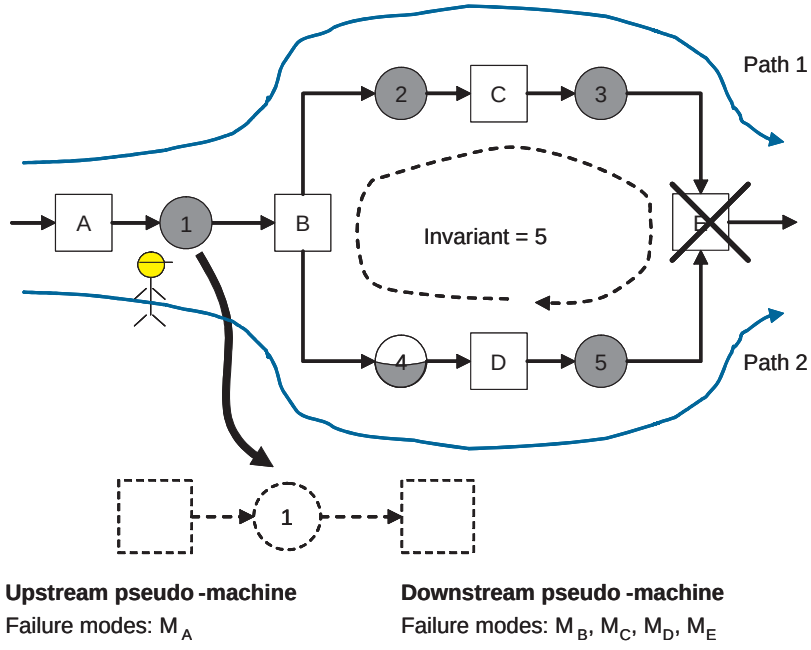


Figure 2-4: Routing buffer of the failure propagation from machine M_E to buffer B_1

The failure at M_E is propagated via Path 1 but not Path 2. Therefore, B_2 is defined as the *routing buffer* of the failure propagation from M_4 to B_1. The importance and more details of routing buffers are presented in Section 5.3.2.

To summarize, in this section, several key issues are briefly discussed, including thresholds in buffers, coupling effects among multiple loops, and routing buffer of failure propagation. These issues are very common in assembly/disassembly networks with complex multiple-loop structures. Given the complexity of blocking and starvation, efficient methods for analyzing blocking and starvation are desired. The detailed analysis and related methods are presented in Chapters 4 and 5. For the purpose of analyzing blocking and starvation of assembly/disassembly networks with complex multiple-loop structures, a graph model is developed in Chapter 3 as a framework for blocking and starvation analysis.

Chapter 3

Graph Model of Assembly/Disassembly Networks

In the previous chapter, we define assembly/disassembly networks as a class of manufacturing systems composed of finite buffers and unreliable single-part or assembly/disassembly machines. The structure or topology of assembly/disassembly networks can vary from a simple production line to a large-scale assembly/disassembly shop which contains over hundred machines and implements multiple kanban loops for production control.

In the literature (Dallery and Gershwin 1992, Buzacott and Shanthikumar 1993, Gershwin 1994, Altiok 1997), models of manufacturing systems are usually described by machine and buffer indices, and their upstream and downstream context. This representation scheme is sufficient to describe the systems with relatively simple topologies. However, it becomes difficult to represent large-scale complex systems with multiple-loop structures using this way. When information flows are integrated with the material flows in the system to implement closed loop control, the complexity of the system topology is further increased. Therefore, a new method is required to efficiently represent manufacturing systems. From a graphical perspective, in any manufacturing system, machines and buffers are organized as a connected directed graph, i.e. a digraph. Graph theory, therefore, is introduced as a generic framework to represent assembly/disassembly networks.

This chapter presents, in the manufacturing system context, how to apply graph theory to develop a framework for the analysis and design of assembly/disassembly networks with arbitrary topologies. Section 3.1 introduces the graph theory basics which are directly related to the modeling of assembly/disassembly networks. The graph model is presented in Section 3.2. In Section 3.3, we discuss two important conditions of the graph model. Linear equation systems of connected flow networks are presented in Section 3.4. The notations introduced in this chapter are needed for the analysis in this thesis.

3.1 Graph Theory Basics

In this section, we introduce concepts in graph theory which are closely related to this thesis. The terminologies of graph theory vary from one book to another. The terminology used in this thesis is taken from Swamy (1981). For readers who are interested in further topics in graph theory and its applications, references can be found in Carre (1979), Swamy (1981), and Boffey (1982).

3.1.1 Some Essential Concepts

The central concept of a graph is that it is a set of entities called *vertices* or *nodes*, which are interrelated via certain correspondences called *edges* or *links*. A graph is often thought of as a set of points (representing vertices) with interconnecting lines (representing edges) and is frequently pictured in this way. Each edge is identified with a pair of vertices. If the edges of a graph are directed (with directions shown by arrows), in which case they are called *arcs*, then the graph is called a *directed* or an *oriented graph*. Otherwise, the graph is called an *undirected graph*. Since manufacturing systems are networks with directed flows, we restrict our discussions to directed graphs.

Digraphs, Vertices and Arcs

Formally, a directed graph or, for short, a *digraph* $G = (V, A)$ consists of two sets: a finite set V of elements called *vertices* and a finite set A of elements called *arcs*. We use the symbols v_1, v_2, ... to represent the vertices and the symbols a_1, a_2, ... to represent the arcs of a digraph. Some important concepts are:

- Each arc is associated with an ordered pair of vertices. If $a_l = (v_i, v_j)$, then v_i and v_j are called the *end vertices* of arc a_l. v_i is the *initial vertex* while v_j is the *terminal vertex* of a_l.

- Two vertices are *adjacent* if they are the end vertices of the same arc.

- An arc is said to be *incident on* its end vertices. An arc is said to be *incident out of* its initial vertex and *incident into* its terminal vertex.

- An arc is called a *self-loop* at vertex v_i, if v_i is the initial as well as the terminal vertex of the arc.

- The vertex set V can be decomposed into three subsets, X, Y and I. The vertices in X are called *sources*, which only have outgoing arcs, while the vertices in Y are called *sinks*, which only have incoming arcs. Vertices in I, which are neither sources nor sinks, are called *intermediate vertices*.

In the pictorial representation of a digraph, a vertex is represented by a circle or rectangle and an arc is represented by an arrowhead line segment which is drawn from the initial to the terminal vertex.

For example, if

$$\begin{aligned} V &= \{v_1, v_2, v_3, v_3, v_4, v_5\}; \\ A &= \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}; \end{aligned}$$

such that

$$
\begin{aligned}
a_1 &= (v_1, v_2);\\
a_2 &= (v_1, v_3);\\
a_3 &= (v_3, v_2);\\
a_4 &= (v_2, v_4);\\
a_5 &= (v_3, v_4);\\
a_6 &= (v_3, v_5);\\
a_7 &= (v_5, v_4).
\end{aligned}
$$

then the digraph $G = (V, A)$ is represented in Figure 3-1(a). In this digraph, v_1 is a source, v_4 is a sink, and v_2, v_3, v_5 are intermediate vertices.

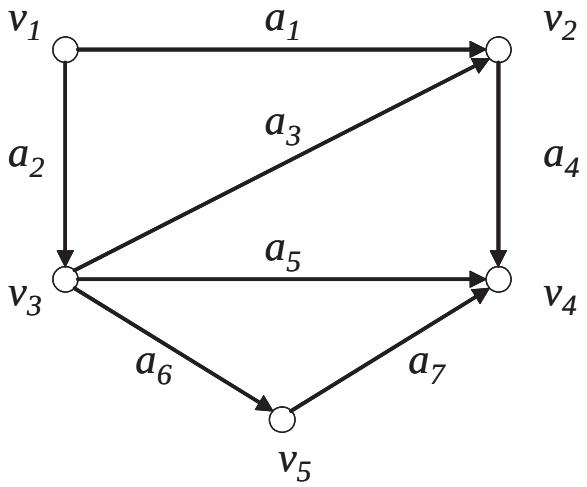


Figure 3-1: A digraph

Walks, Trails, Paths, and Circuits

Here, we define the concepts of walks, trails, paths, and circuits. The directions of the arcs are not important in the definitions.

A *walk* in a digraph $G = (V, A)$ is a finite alternating sequence of vertices and arcs $v_1, a_1, v_2, a_2, ..., v_{k-1}, a_{k-1}, v_k$ beginning and ending with vertices such that v_{i-1}

and v_i are the end vertices of the arc $a_{i-1}, 1 \leq i \leq k$. Vertices v_{i-1} and v_i need not be the initial and terminal vertices of a_{i-1}, respectively. Vertices v_1 and v_k are called end vertices of the walk.

A walk is a *trail* if all its arcs are distinct. A trail is *open* if the end vertices of the corresponding walk are distinct; otherwise, it is *closed*. For example, in Figure 3-1, the sequence $v_1, a_1, v_2, a_3, v_3, a_5, v_4, a_7, v_5, a_6, v_3$ is an open trail, whereas the sequence $v_1, a_1, v_2, a_3, v_3, a_5, v_4, a_7, v_5, a_6, v_3, a_2, v_1$ is a closed trail.

An open trail is a *path* if all its vertices are distinct. Two distinct vertices v_i and v_j are said to be *connected* if there exists a path from v_i to v_j in G.

A closed trail is a *circuit* if all its vertices except the end vertices are distinct. For example, in Figure 3-1, the sequence v_1, a_1, v_2, a_3, v_3 is a path, whereas the sequence $v_1, a_1, v_2, a_3, v_3, a_2, v_1$ is a circuit.

A digraph G is said to be *connected* if there exists a path between every pair of vertices in G.

Subgraphs, Trees, and Fundamental Circuits

Consider a digraph $G = (V, A)$. $G' = (V', A')$ is a *subgraph* of G if V' and A' are, respectively, subsets of V and A such that an arc (v_i, v_j) is in A' only if v_i and v_j are in V_j.

A *tree* of a digraph G is a connected subgraph of G, whose underlying undirected graph is acyclic. A *spanning tree* T of digraph G is a tree of G having all the vertices of G. The arcs of a spanning tree are called the *branches* of T. The arcs which are not in T are called *chords* or *links*. Consider digraph G shown in Figure 3-2(a). Digraph T of Figure 3-2(b) is a spanning tree of G.

A connected digraph G with n vertices and m arcs has following properties:

- The numbers of arcs and vertices satisfy $m \geq n - 1$
- A spanning tree T of G has n vertices and $n - 1$ arcs, or branches.
- For any pair of vertices in T, there exists a unique path in T between them. The path between v_i and v_j is denoted by $p(i, j)$.

Consider a spanning tree T of a connected digraph G. Denote the chords of T by $c_1, c_2, ..., c_{m-n+1}$, where m is the number of arcs in G, and n is the number of vertices in G. The union of T and c_i, $T \cup c_i$, contains exactly one circuit C_i. This circuit consists of chord c_i and a unique path in T between the end vertices of c_i. Circuit C_i is called the *fundamental circuit* of G with respect to chord c_i of spanning tree T.

In a digraph G, the set of all the $m-n+1$ fundamental circuits $C_1, C_2, ..., C_{m-n+1}$ of G with respect to the chords of spanning tree T is known as *the set of fundamental circuits* of G with respect to T. Each fundamental circuit C_i contains exactly one chord, namely, chord c_i. Further, chord c_i is not presented in any other fundamental circuit with respect to T.

Consider the digraph G in Figure 3-2(a), a set of fundamental circuits with respect to spanning tree T of Figure 3-2(b) is shown in Figure 3-2(c).

3.1.2 Matrices of a Graph

Incidence Matrix Φ

Given a digraph G with n vertices and m arcs and having no self-loops, there corresponds a $n \times m$ matrix called the *all-vertex incidence matrix* of G. The all-vertex incidence matrix $\Phi = [\phi_{ij}]$ has n rows, one for each vertex, and m columns, one for each arc. The element ϕ_{ij} of Φ is defined as follows:

$$\phi_{ij} = \begin{cases} 1 & \text{if the } j\text{th arc is incident out of the } i\text{th vertex;} \\ -1 & \text{if the } j\text{th arc is incident into the } i\text{th vertex;} \\ 0 & \text{if the } j\text{th arc is not incident on the } i\text{th vertex.} \end{cases} \tag{3.1}$$

A row of Φ will be referred to as an *incidence vector* of G. Consider digraph G in Figure 3-2(a). Its all-vertex incidence matrix is given by:

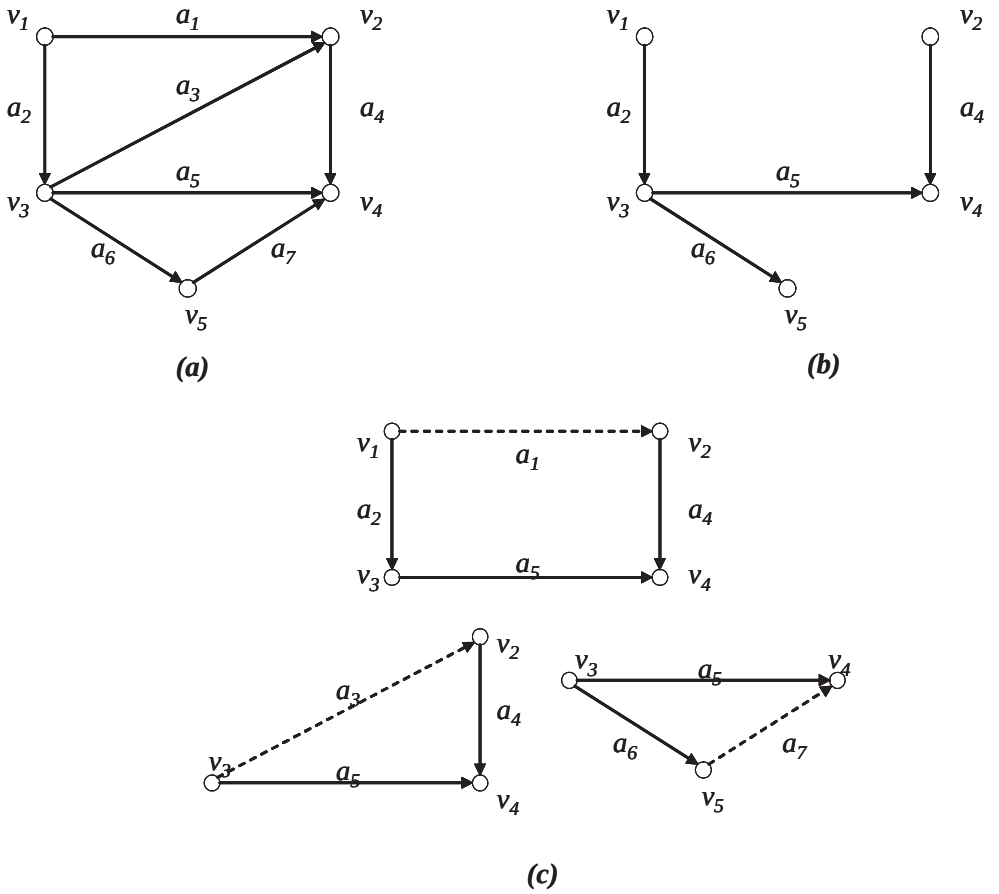


Figure 3-2: A spanning tree and a set of fundamental circuits of a digraph G. (a) Digraph G. (b) Spanning tree T of G. (c) Set of fundamental circuits of G with respect to T. (Chords are indicated by dashed lines)

$$\Phi = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{array} \\ \left[\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right] \end{array}$$

In all-vertex incidence matrix Φ, each column contains exactly two nonzero entries, one $+1$ and one -1. Given any $n \times m$ matrix that satisfies this condition, the corresponding digraph G can be uniquely determined and pictured. If $n \leq m + 1$, digraph G is connected.

Notice that, in all-vertex incidence matrix Φ of any connected digraph G with n vertices, the rows of Φ are linearly dependent and there exists a nonsingular submatrix of order $n - 1$. An $(n - 1)$-row submatrix Φ_f of Φ will be referred to as an *incidence matrix* of G. The vertex which corresponds to the row of Φ which is not in Φ_f will be called the *reference vertex* of Φ_f. Thus, for arbitrary n-vertex connected digraph,

$$\text{rank}(\Phi_f) = \text{rank}(\Phi) = n - 1 \tag{3.2}$$

Path Matrix $\Gamma(T, s)$

In a spanning tree T with reference vertex V_s, the *path matrix* $\Gamma(T, s) = [\gamma_{ij}(T, s)]$ is an $(n - 1) \times m$ matrix defined as follows:

$$\gamma_{ij}(T, s) = \begin{cases} 1 & \text{if the } j\text{th buffer is in the unique path } p(s, i) \text{ and connected} \\ & \text{to the } s\text{th machine via its upstream machine } u(j); \\ -1 & \text{if the } j\text{th buffer is in the unique path } p(s, i) \text{ and connected} \\ & \text{to the } s\text{th machine via its downstream machine } d(j); \\ 0 & \text{if the } j\text{th buffer is not in the unique path } p(s, i) \end{cases}$$

Consider digraph G in Figure 3-2(b). Suppose the reference vertex is v_3. Then

we have:

$$\Gamma(T,3) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccc} a_2 & a_4 & a_5 & a_6 \end{array} \\ \left[\begin{array}{cccc} -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

The rank of path matrix $\Gamma(T,s)$ is $n-1$. All its row vectors $\mathbf{\Gamma}_{i\star}(T,s), i=1,...,n;$ $i \neq s$ are linearly independent.

Circuit Matrix Ψ

In a closed loop, a circuit can be traversed in one of two directions, clockwise or anticlockwise. The direction we choose for traversing a circuit defines its orientation at each arc. The *circuit matrix* $\Psi = [\psi_{ij}]$ of a digraph G with m arcs has m columns and as many rows as the number of circuits in G. The entry ψ_{ij} is defined as follows:

$$\psi_{ij} = \begin{cases} 1 & \text{if the } j\text{th arc is in the } i\text{th circuit and its orientation} \\ & \text{agrees with the circuit orientation;} \\ -1 & \text{if the } j\text{th arc is in the } i\text{th circuit and its orientation} \\ & \text{does not agree with the circuit orientation;} \\ 0 & \text{if the } j\text{th arc is not in the } i\text{th circuit.} \end{cases} \tag{3.3}$$

A row of Ψ will be referred to as a *circuit vector* of G.

Recall that, for any spanning tree T of a connected digraph G having n vertices and m arcs, there exists $m-n+1$ fundamental circuits with respect to chords $c_1, c_2, ..., c_{m-n+1}$ of T. A matrix composed of the vectors corresponding to these $m-n+1$ fundamental circuits is known as the *fundamental circuit matrix* of G with respect to spanning tree T. In the remainder of this thesis, $\Psi = [\psi_{ij}]$ is always referred to as a fundamental circuit matrix.

In fundamental circuit matrix Ψ, we choose the orientation of a fundamental circuit to agree with that of the defining chord. Then the entry which corresponds to circuit C_i and chord c_i is equal to 1.

For example, the fundamental circuit matrix Ψ of Figure 3-2(c) is

$$\Psi = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{array} \\ \left[\begin{array}{ccccccc} 1 & -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{array}\right] \end{array}$$

in which ψ_{11}, ψ_{23}, and ψ_{37} are equal to 1.

For arbitrary connected digraph G with n vertices and m arcs,

$$\text{rank}(\Psi) = m - n + 1 \tag{3.4}$$

3.2 Graph Model

In this section, we develop a graph model for analyzing blocking and starvation property in assembly/disassembly networks with arbitrary topologies. Three important issues are covered:

- How to map a network to a digraph;
- How to model flows in the network;
- How to represent machine failures, and blocking and starvation.

3.2.1 Underlying Digraph

The manufacturing systems we consider can be represented as flow networks with interconnected machines and buffers. For any pair of machines in an assembly/disassembly network, a buffer is used to model the connection if there exists a flow between the pair of machines. By convention, a machine is allowed to handle multiple parts by

assembly/disassembly, whereas a buffer is only allowed to process a single incoming flow from its upstream machine and generate a single outgoing flow toward its downstream machine. In such a way, a buffer could be defined exactly as an arc in graph theory,

$$B_j = (u(j), d(j)) \tag{3.5}$$

where $u(j)$ and $d(j)$ denote the upstream and downstream machines of buffer B_j, respectively.

Machines in the network are mapped to vertices in the digraph. The initial vertex of an arc is the upstream machine of the corresponding buffer, while the terminal one is the downstream machine. Loops in assembly/disassembly networks are defined in the same way as circuits in digraphs. Each loop corresponds to a circuit vector of the digraph.

In summary, any assembly/disassembly network Ω has a unique underlying connected digraph G (Figure 3-3). The vertex set V and arc set A correspond to the set of machines M and the set of buffers B, respectively. The set of circuits C is identical to the set of loops L. The concepts and notations of machine-vertex, buffer-arc, and loop-circuit are interchangeable. Network $(M, B, L)_\Omega$ is equivalent to digraph $(V, A, C)_G$.

$$(M, B, L)_\Omega \equiv (V, A, C)_G \tag{3.6}$$

To depict the upstream and downstream context in G,

- For a machine M_i, we define $U(M_i)$ as the set of its upstream buffers, and $D(M_i)$ as the set of its downstream buffers.

- For an buffer B_j, its upstream machine is defined as $u(j)$, while its downstream machine is denoted as $d(j)$.

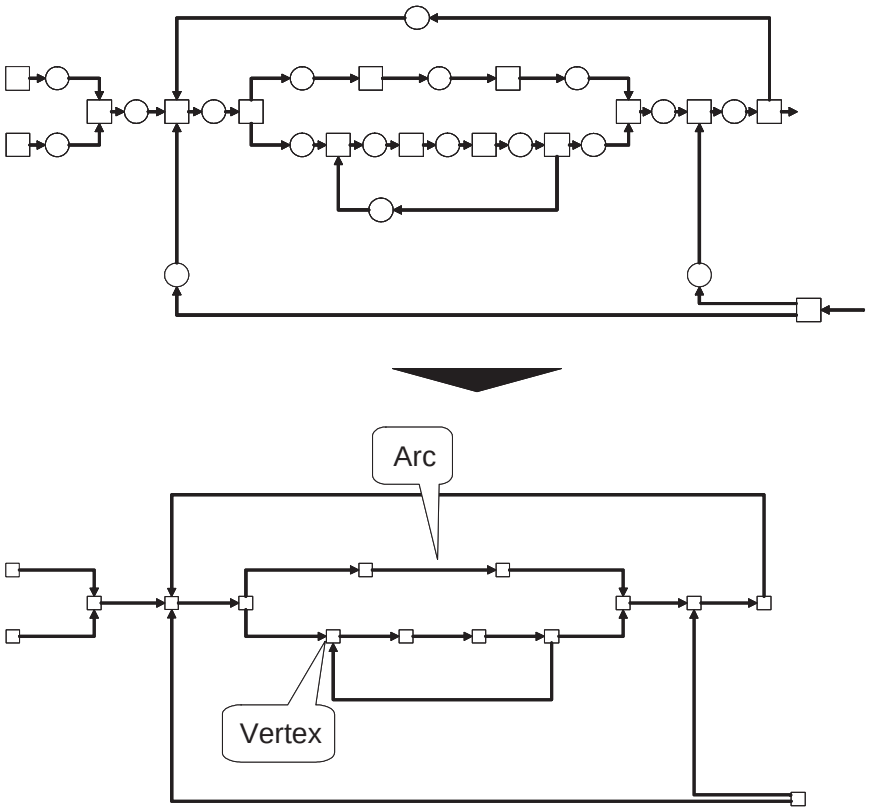


Figure 3-3: Mapping of an assembly/disassembly network to its underlying digraph

For example, in Figure 3-4, a 6-machine 7-buffer assembly/disassembly network is illustrated by drawing its underlying digraph. The all-vertex incidence matrix Φ of the network is:

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

Choose a spanning tree $T = \{a_1, a_2, a_3, a_4, a_5\}$ of G, the corresponding chords are a_6 and a_7. The set of fundamental circuits or loops can be written as

$$\Psi = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

3.2.2 Buffer Size and Level

In an assembly/disassembly network Ω, the *size* of buffer B_j is defined as N_j, the maximal amount of material that can be temporarily stored in the buffer. The *level* of buffer B_j is $b(j,t)$, the amount of material stored in B_j at time point t. It is subject to the *buffer size constraint*

$$0 \leq b(j,t) \leq N_j \qquad \forall B_j \in B \tag{3.7}$$

3.2.3 Flow

The flows through machine M_i at time t are denoted by

- $f_v^+(i,t)$: instantaneous incoming flow rate transported from upstream buffer B_v of M_i at time t;

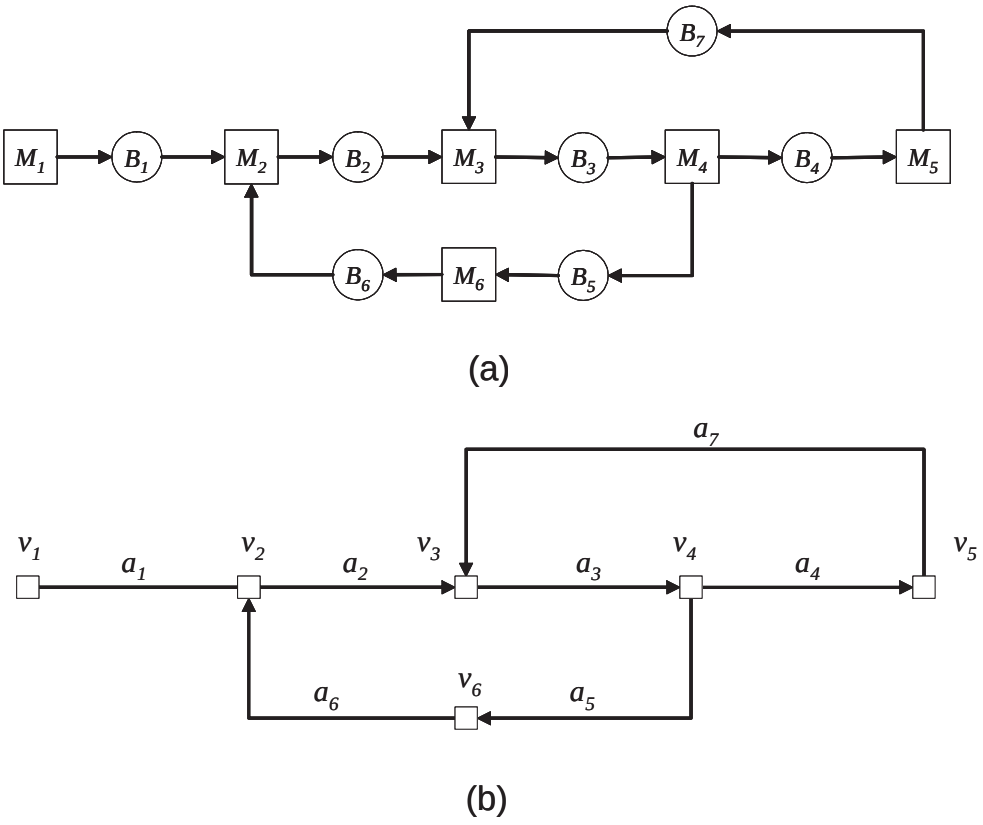


Figure 3-4: An assembly/disassembly network and its underlying digraph. (a) A 6-machine 7-buffer assembly/disassembly network. (b) The underlying digraph

- $f_w^-(i,t)$: instantaneous outgoing flow rate transported toward downstream buffer B_w of M_i at time t.

Given a machine M_i in an assembly/disassembly network, the rates at which material flows are transported into M_i from its upstream buffers $U(M_i)$ are equal to the rates at which material flows are transported out of machine M_i to its downstream buffers $D(M_i)$. Therefore, the instantaneous flow rate through M_i at time t is given by

$$f(i,t) = f_v^-(i,t) = f_w^+(i,t) \geq 0 \qquad \forall M_i \in M, v \in U(M_i), w \in D(M_i) \tag{3.8}$$

For continuous time, we integrate the instantaneous flow rate through M_i over a period of time. The amount of material transported through M_i during the time period (t_1, t_2) is defined as the cumulative flow of M_i:

$$q(i)\Big|_{t_1}^{t_2} = \int_{t_1}^{t_2} f(i,t)dt \qquad M_i \in M \tag{3.9}$$

If the time period is $(0, t)$, we can further simplify the notation as

$$q(i,t) = \int_0^t f(i,t)dt \qquad M_i \in M \tag{3.10}$$

For simplicity, we use $q(i,t)$ in the remainder of this thesis.

Since $f(i,t) \geq 0$, $q(i,t)$ is non-decreasing.

$$\frac{dq(i,t)}{dt} = f(i,t) \geq 0 \qquad M_i \in M \tag{3.11}$$

Notice that when $t = 0$:

$$q(i,t) = q(i,0) = \int_0^0 f(i,t)dt = 0 \qquad M_i \in M \tag{3.12}$$

3.2.4 Machine Failure, Blocking and Starvation

When a machine M_s is down at time $t = 0$ and stays down for all $t > 0$, the flow transported through the machine is zero:

$$f(s,t) = 0 \qquad t \geq 0 \tag{3.13}$$

$$q(s,t) = \int_0^t f(s,t)dt = 0 \qquad t \geq 0 \tag{3.14}$$

As a result, the upstream buffers of M_s tend to be filled, while the downstream buffers of M_s tend to be depleted.

We also formulate the phenomena of blocking and starvation in graph model. For a given machine M_i in the system, suppose one or more of its downstream buffers is full:

$$b(w,t) = N_w \qquad w \in D(M_i) \tag{3.15}$$

M_i is said to be *blocked* at time t by buffer B_w and thus there is no flow through M_i, i.e. $f(i,t) = 0$.

Similarly, if one or more of its upstream buffers is empty, M_i is *starved* at time t by B_v :

$$b(v,t) = 0 \qquad v \in U(M_i) \tag{3.16}$$

The starvation is propagated through buffer B_v so that the instantaneous flow rate through M_i is forced to be zero, i.e. $f(i,t) = 0$.

3.3 Properties

To describe the properties of the graph model, two conditions are specified to illustrate the relationships among buffer sizes, buffer levels, and cumulative flows in assembly/disassembly networks.

3.3.1 Conservation Condition

Suppose at time $t = 0$, the initial inventory level of buffer B_j is $b(j, 0)$. Recall that the amount of material transported through M_i during the time period $(0, t)$ is denoted by $q(i, t)$. The resultant inventory level of buffer B_j at time t is given by

$$b(j,t) = b(j,0) + q(u(j),t) - q(d(j),t) \tag{3.17}$$

in which $u(j)$ and $d(j)$ are, respectively, the upstream and downstream machines of B_j.

This equation is called the *conservation condition* which requires that, during a time period, the increase or decrease of the buffer level is equal to the net cumulative flow of material entering or leaving the buffer.

By using the corresponding column vector $[\phi_{1j}, \phi_{2j}, \ldots, \phi_{nj}]^T$ of incidence matrix Φ, we can rewrite the conservation condition for each buffer as follows:

$$b(j,t) = b(j,0) + [\phi_{1j}, \phi_{2j}, \ldots, \phi_{nj}] \begin{bmatrix} q(1,t) \\ q(2,t) \\ \vdots \\ q(n,t) \end{bmatrix} \tag{3.18}$$

$$\forall B_j \in B$$

Define the *level vector* of all the buffers in the network

$$\mathbf{b}(t) = \begin{bmatrix} b(1,t) \\ b(2,t) \\ \vdots \\ b(m,t) \end{bmatrix}$$

Define the *cumulative flow vector* during $(0,t)$ of machine set M

$$\mathbf{q}(t) = \begin{bmatrix} q(1,t) \\ q(2,t) \\ \vdots \\ q(n,t) \end{bmatrix}$$

The equations of the conservation condition can be concisely written:

$$\mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) \qquad t \geq 0 \tag{3.19}$$

3.3.2 Invariance Condition

When there exist loops or circuits in assembly/disassembly network Ω with n machines and m buffers, we can obtain another set of equations called the *invariance condition*. Fundamental circuit matrix Ψ is used to depict the condition.

The *invariant* of each fundamental loop L_k is defined by

$$I_k = [\psi_{k1}, \psi_{k2}, \ldots, \psi_{km}] \begin{bmatrix} b(1,t) \\ b(2,t) \\ \vdots \\ b(m,t) \end{bmatrix} \tag{3.20}$$

$$k = 1, 2, ..., n - m + 1$$

The above equation is the invariance condition of loop L_k. The invariants of a set of fundamental loops can be organized into the invariant vector $\mathbf{I}$:

$$\mathbf{I} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{m-n+1} \end{bmatrix}$$

The invariance condition with respect to the set of fundamental loops can be written:

$$\Psi \mathbf{b}(t) = \mathbf{I} \qquad t \geq 0 \tag{3.21}$$

For example, consider the two-loop system in Figure 3-4. Two loops are specified in loop matrix Ψ:

$$\Psi = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Suppose the invariants of L_1, L_2 are 25 and 15 respectively. Then we can write

$$\Psi \mathbf{b}(t) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b(1,t) \\ b(2,t) \\ \vdots \\ b(7,t) \end{bmatrix} = \mathbf{I} = \begin{bmatrix} 25 \\ 15 \end{bmatrix}$$

or

$$\begin{aligned} I_1 &= b(2,t) + b(3,t) + b(5,t) + b(6,t) = 25 \\ I_2 &= b(3,t) + b(4,t) + b(7,t) = 15 \end{aligned}$$

3.4 Connected Flow Networks

3.4.1 Dynamic Connected Flow Network

The graph model of an assembly/disassembly network is summarized as a *dynamic connected flow network*. For an arbitrary assembly/disassembly network Ω with n

machines and m buffers, we have the following dynamic system:

$$\begin{aligned} &\text{For all} \quad t \geq 0, \\ &\mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) && (3.22) \\ &\Psi \mathbf{b}(t) = \mathbf{I} && (3.23) \\ &0 \leq \mathbf{b}(t) \leq \mathbf{N} && (3.24) \\ &\mathbf{q}(t) \quad \text{non-decreasing} && (3.25) \\ &\mathbf{q}(0) = 0 && (3.26) \end{aligned}$$

where $\mathbf{N}$ is defined as the *buffer size vector*:

$$\mathbf{N} = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_m \end{bmatrix}$$

In the linear system, (3.22) is the conservation condition; (3.23) is the invariance condition; (3.24) is the buffer size constraint; (3.25) is the non-decreasing property described in (3.11); and (3.26) is the initial condition presented in (3.12).

3.4.2 Static Connected Flow Network

When we ignore time, we can define the following *static connected flow network*:

$$\begin{aligned} &\mathbf{b} = \mathbf{b}^0 + \Phi^T \mathbf{q} && (3.27) \\ &\Psi \mathbf{b} = \mathbf{I} && (3.28) \\ &0 \leq \mathbf{b} \leq \mathbf{N} && (3.29) \\ &\mathbf{q} \geq 0 && (3.30) \end{aligned}$$

where $\mathbf{b}^0$ is the initial buffer level vector.

3.4.3 Feasibility of Buffer Sizes and Loop Invariants

Notice that the formulations of dynamic connected flow networks and static connected flow networks both contain invariance conditions and buffer size constraints:

$$\Psi \mathbf{b} = \mathbf{I} \tag{3.31}$$

$$0 \leq \mathbf{b} \leq \mathbf{N} \tag{3.32}$$

Since loop invariant vector $\mathbf{I}$ and buffer size vector $\mathbf{N}$ are given parameters and buffer level vector $\mathbf{b}$ is an unknown, we discuss the feasibility of $\mathbf{b}$ in relation to the given parameters $\mathbf{N}$ and $\mathbf{I}$.

Consider the two-loop system in Figure 3-4. We can write the invariance condition:

$$\Psi \mathbf{b}(t) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b(1,t) \\ b(2,t) \\ \vdots \\ b(7,t) \end{bmatrix} = \mathbf{I} = \begin{bmatrix} I_1 \\ I_1 \end{bmatrix}$$

or

$$\begin{aligned} I_1 &= b(2,t) + b(3,t) + b(5,t) + b(6,t) \\ I_2 &= b(3,t) + b(4,t) + b(7,t) \end{aligned}$$

Suppose all the buffer sizes are 10. We illustrate several examples in which there does not exist a feasible solution.

- When $I_1 = -10$, we can not find a feasible solution since all buffer levels are non-negative. Therefore, in this example, I_1 must be greater than 0.

- When $I_1 = 80$, the loop invariant is larger than the total buffer size of loop L_1 and there does not exist a feasible solution. Therefore, loop invariant I_1 must be less than the total buffer size in L_1, which is 40.

- When $I_1 = 2, I_2 = 25$, the invariant of L_2 is non-negative and is less than the total buffer size in L_1. I_2 is also nonnegative and is less than the total buffer size in L_2, which is 30. However, a feasible solution is still not available. This is because L_1 and L_2 are coupled at B_3.

  According to loop L_1, the maximal number of parts allowed in B_3 is 2. According to loop L_2, the minimal number of parts that must be allocated to B_3 is 5, when B_6 and B_7 are full. Therefore, the minimal number of parts we need to put in B_3 according to L_2 is greater than the maximal number of parts allowed in B_3 according to L_1. These two constraints cannot both be satisfied.

  Therefore, to ensure that a feasible solution exists, when $I_1 = 2$, the upper bound of the selection of I_2 is 22. When $I_2 = 25$, the lower bound of the selection of I_1 is 5. The selection of two invariants is correlated.

In the above example, there are only two coupled loops and the flow direction of each loop does not change. When the loops have flow direction changes and multiple closed loops are coupled, the selection of loop invariant vector is very complicated for a given buffer size vector.

Summarily, loop invariants and buffer sizes should be carefully selected such that there exists a feasible solution of $\mathbf{b}(t)$ or $\mathbf{b}$. When we design a manufacturing network with multiple closed loops, we must verify the feasibility of buffer sizes and loop invariants. For simplicity, in the remainder of this thesis, we assume that $\mathbf{N}$ and $\mathbf{I}$ are selected such that a feasible solution exists.

3.5 Conclusion

The graph model of assembly/disassembly networks shares many common features with other network models. However, machine failures, blocking and starvation prop-

agation, and the loop invariant condition in the manufacturing system context differentiate this graph model from them. In Chapter 4, we apply this graph model to analyze blocking and starvation properties in complex assembly/disassembly networks.

Chapter 4

Blocking and Starvation Analysis

When we evaluate the performance of a manufacturing system by decomposition, the blocking and starvation properties of the system provide essential information for setting up the parameters of the pseudo-machines in a set of two-machine one-buffer lines (building blocks) after decomposition. A detailed decomposition method which requires information about the propagation of blocking and starvation is presented in Chapter 5. To obtain the blocking and starvation properties, we must know the limiting propagation state: Given a buffer, which machine failures in the system can cause this buffer to be full or empty?

This chapter presents a rigorous formulation for blocking and starvation analysis in Section 4.1.2. An efficient algorithm based on induction is developed to analyze the blocking and starvation propagation in complex assembly/disassembly networks in Section 4.6. The result is the 'Machine failure – Buffer level' matrix which is used in the decomposition of Chapter 5. The notation used in this chapter is defined in the graph model of Chapter 3.

4.1 Formulation

4.1.1 Assumption

For a given buffer in a manufacturing system, in order to find all failure modes which can cause it to be full or empty, we must understand the maximal effect of a single machine failure on the propagation of blocking and starvation. Therefore, we make the *single machine failure* assumption as follows:

Single Machine Failure In an assembly/disassembly network, once there is a failure at machine M_s, none of the other machines is allowed to fail. The failure of M_s persists for a sufficient amount of time such that the levels of all the buffers reach a limiting state.

We only make this assumption in this chapter for the purpose of finding the maximal possible propagation of failures. Then we use that information in Chapter 5 when we consider random failures and repairs.

In this assumption, the *limiting propagation state* is the state of the buffers and machines when there is a single machine failure at M_s and the propagation of blocking and starvation has reached the maximal effect after a sufficient amount of time such that

- The buffer levels can no longer change with time.
- There is no flow through any machine in the system:

$$\forall M_i \in M \qquad f(i, \infty) = 0 \tag{4.1}$$

- Except for the failed machine, each machine can only be in one of the three states: blocked, starved or simultaneously blocked and starved. Blocking and starvation of a machine are defined as follows:

$$M_i \in M, i \neq s \quad \text{is} \begin{cases} \text{blocked} & \text{if} \quad \exists w \in D(M_i) \qquad b(w, \infty) = N_w \\ \text{starved} & \text{if} \quad \exists v \in U(M_i) \qquad b(v, \infty) = 0 \end{cases} \tag{4.2}$$

In this chapter, M_s is the name of the failed machine. In the limiting propagation state due to the single machine failure at M_s, we can identify which buffers the failure of machine M_s causes to be full or empty.

Suppose $M_i, i \neq s$ is not in the limiting propagation state at time t. Then all its upstream buffers are not empty and all its downstream buffers are not full.

$$\forall w \in D(M_i) \qquad b(w, t) < N_w \tag{4.3}$$

$$\forall v \in U(M_i) \qquad b(v, t) > 0 \tag{4.4}$$

Since machine M_i is operational, the instantaneous flow rate $f(i, t)$ through M_i at time t is positive. Then cumulative flow $q(i, t)$ is strictly increasing at time t. This condition appears in the following problem statement.

4.1.2 Problem Statement

We use $b(j, \infty | M_s)$ to denote the level of buffer B_j in the limiting propagation state due to the single machine failure at M_s. Suppose M_s is failed at time $t = 0$, the buffer level vector in the limiting propagation state due to the single machine failure at M_s satisfies:

$$LT : \mathbf{b}(\infty|M_s) = \lim_{t\to\infty} \quad \mathbf{b}(t) \tag{4.5}$$

$$\text{subject to} \quad \mathbf{b}(t) = \mathbf{b}(0) + \Phi^T\mathbf{q}(t) \tag{4.6}$$

$$\Psi\mathbf{b}(t) = \mathbf{I} \tag{4.7}$$

$$0 \leq \mathbf{b}(t) \leq \mathbf{N} \tag{4.8}$$

$$\mathbf{q}(t) \quad \text{non-decreasing} \tag{4.9}$$

$$\mathbf{q}(0) = 0 \tag{4.10}$$

$$q(s,t) = 0 \tag{4.11}$$

$$\forall M_i, i \neq s \tag{4.12}$$

$$s.t. \quad b(w,t) < N_w, \forall w \in D(M_i) \quad \text{and}$$

$$b(v,t) > 0, \forall v \in U(M_i),$$

$$q(i,t) \text{ is strictly increasing.}$$

The objective of the blocking and starvation analysis is to determine the buffer levels in the limiting propagation state. That is, we must find $\mathbf{b}(\infty|M_s)$. Constraints (4.6)–(4.11) are the dynamic connected flow network formulation described in Section 3.4.1. Constraint (4.12) is the condition discussed at the end of Section 4.1.1.

4.1.3 The Uniqueness of Blocking and Starvation

In the decomposition method, the buffer levels in the limiting propagation state are used to assign the failure modes for the pseudo-machines in each building block. In order to ensure that the assignment of failure modes is possible and unique, we must prove that $\mathbf{b}(\infty|M_s)$ exists, is independent of the initial buffer level vector $\mathbf{b}(0)$, and is unique.

Theorem: The Uniqueness of Blocking and Starvation In a dynamic connected flow network with a single machine failure, when $t \to \infty$, the system is in a limiting propagation state and the cumulative flows of all the machines $q(i,\infty), i =$

$1, ..., n$ are finite and are simultaneously maximized. There exists a corresponding limiting buffer level vector $\mathbf{b}(\infty|M_s)$ which is independent of the initial buffer level vector and is unique.

The proof of the theorem is given in Appendix A. This theorem provides the necessary and sufficient conditions to obtain the deterministic blocking and starvation properties due to a single machine failure. According to the statement of cumulative flows in the theorem (Section A.4.5), we can write the equivalent Vector Optimization Problem (VOP) as follows:

$$
\begin{aligned}
VOP: \text{maximize} \quad & q(1), q(2), ..., q(n) && (4.13)\\
\text{subject to} \quad & \mathbf{b} = \mathbf{b}^0 + \Phi^T \mathbf{q} && (4.14)\\
& \Psi \mathbf{b} = \mathbf{I} && (4.15)\\
& 0 \leq \mathbf{b} \leq \mathbf{N} && (4.16)\\
& q(s) = 0 && (4.17)\\
& \mathbf{q} \geq 0 && (4.18)
\end{aligned}
$$

From Lemma A.4.3, the multiple objectives $q(1), ..., q(n)$ can be simultaneously maximized. Constraints (4.14)–(4.16) and (4.18) are the static connected flow network formulation in Section 3.4.2.

4.2 Definition of Terminology

Here we define the terminology used in this chapter.

4.2.1 Blocking and Starvation

When the system is in the limiting propagation state due to the single machine failure at M_s, the buffer level vector is $\mathbf{b}(\infty|M_s)$. From the perspective of buffer B_j:

- If $b(j, \infty|M_s) = N_j$, the immediate upstream machine $M_{u(j)}$ of B_j is *blocked* by M_s *via* B_j;
- If $b(j, \infty|M_s) = 0$, the immediate downstream machine $M_{d(j)}$ of B_j is *starved* by M_s *via* B_j.

When we look at machine $M_i, i \neq s$:

- If $\exists w \in D(M_i)$ such that $b(w, \infty|M_s) = N_w$, then M_i is *blocked* by M_s *via* B_w.
- If $\exists v \in U(M_i)$ such that $b(v, \infty|M_s) = 0$, then M_i is *starved* by M_s *via* B_v.

Buffers B_w, B_v are called *routing buffers* of failure propagation. Detailed definitions are presented in Section 5.3.2. Notice that a machine can be blocked via multiple downstream buffers or starved via multiple upstream buffers.

4.2.2 Domains of Blocking and Starvation

Under the single machine failure assumption at M_s, the set of all buffers B_w which have $b(w, \infty|M_s) = N_w$ are in the *domain of blocking* of M_s, while the set of all buffers B_v which have $b(v, \infty|M_s) = 0$ are in the *domain of starvation* of M_s.

Consider a long production line with n machines in Figure 4-1(a). Given a single machine failure at M_s, the buffers in its downstream part $\{B_s, ..., B_{n-1}\}$ are in the domain of starvation of M_s, while the buffers in its upstream part $\{B_1, ..., B_{s-1}\}$ are in the domain of blocking of M_s.

4.2.3 Ranges of Blocking and Starvation

Under the single machine failure assumption, the *range of blocking* of buffer B_j is the set of machines that could cause $b(j, \infty|M_s) = N_j$ and thus block the immediate upstream machine of B_j. Similarly, the *range of starvation* of buffer B_j is the set of machines that could cause $b(j, \infty|M_s) = 0$ and thus starve the immediate downstream machine of B_j.

Observe the level of buffer B_s in Figure 4-1(b). The machines in the range of blocking of B_s are the entire downstream part of the line $\{M_{s+1}, ..., M_n\}$, while the machines in the range of starvation of B_s are the entire upstream part of the line $\{M_1, ..., M_s\}$.

4.2.4 'Machine Failure – Buffer Level' Matrix

We define matrix Θ to neatly represent the blocking and starvation relationships between machine failures and buffer levels under the single machine failure assumption. Consider the five-machine production line in Figure 4-2 in which all the buffer sizes are 10. The 'Machine failure – Buffer level' matrix of this production line is:

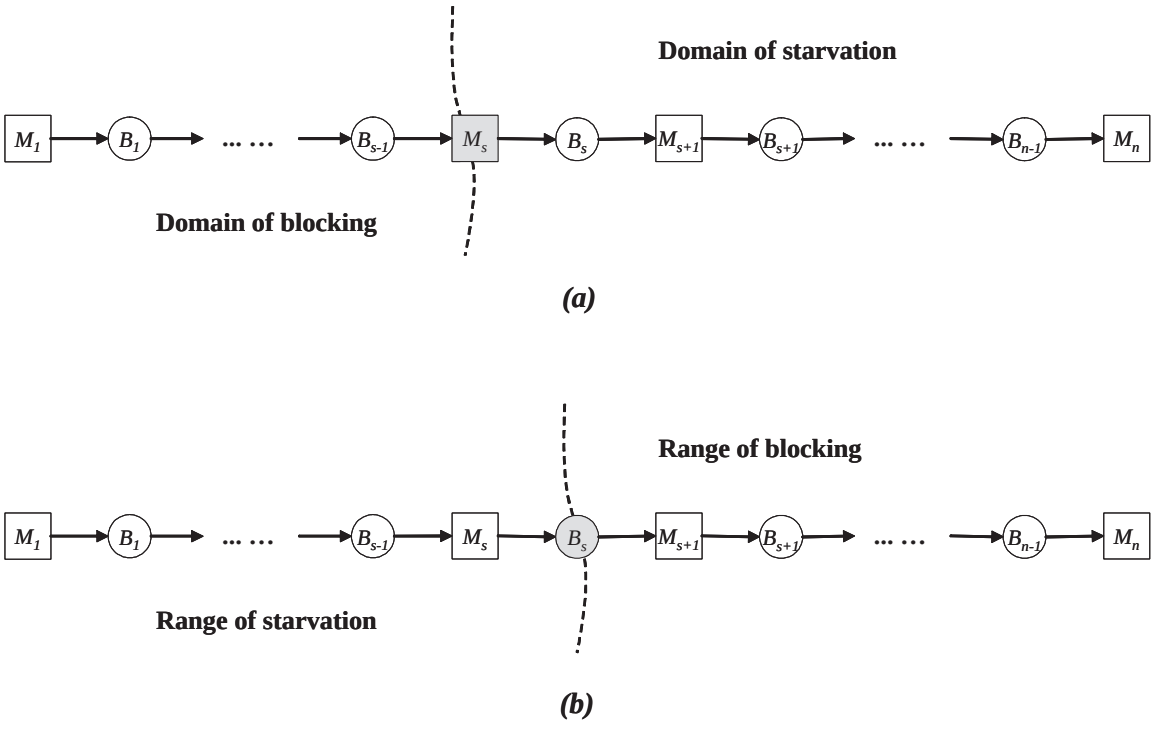


Figure 4-1: Domains and ranges of blocking and starvation

$$\Theta = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \end{array} \begin{array}{c} \begin{array}{cccc} B_1 & B_2 & B_3 & B_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 \\ 10 & 10 & 10 & 0 \\ 10 & 10 & 10 & 10 \end{bmatrix} \end{array}$$

Figure 4-2: A five-machine production line

In the matrix, the vertical dimension represents failures of different machines, while the horizontal one records the levels of different buffers in the limiting propagation state. Entry θ_{ij} reflects the level of B_j in the limiting propagation state due to the single machine failure at M_i:

$$\theta_{ij} = b(j, \infty | M_i) \qquad M_i \in M, B_j \in B \tag{4.19}$$

By looking into row i in matrix Θ, we can identify buffer level vector $\mathbf{\Theta}_{i\star}$ due to the single machine failure at M_i. The domains of blocking and starvation can be easily determined. For example, in row i, B_j is in the domain of blocking of M_i if $\theta_{ij} = N_j$, while buffer B_j is in the domain of starvation of M_i if $\theta_{ij} = 0$.

In the matrix, column j represents the levels of B_j due to the failures of different machines. In each column, the ranges of blocking and starvation can be separated. For example, in column j, if $\theta_{ij} = N_j$, then M_i is in the range of blocking; and if $\theta_{ij} = 0$, then M_i is in the range of starvation.

When there exist loops in the network, the buffer levels due to single machine failure could be partially filled because of the loop invariant condition. The partially filled level is called *threshold*. We briefly discuss it in Section 2.3.1. More details are presented in Section 5.2. See also Maggio et al. (2006) and Gershwin and Werner (2006).

The 'Machine failure – Buffer level' matrix efficiently records the results of blocking and starvation analysis. It contains information needed for decomposition evaluation. The details of decomposition are presented in Section 5.3.

4.3 Analysis

The most critical part of the blocking and starvation analysis is to obtain the buffer level vectors in the limiting propagation state. Recall the equivalent vector optimization problem (VOP) in Section 4.1.3. We could determine the buffer level vectors $\mathbf{b}(\infty|M_i), i = 1, ..., n$ by solving this problem. However, directly solving the multi-objective problem is very difficult and inefficient, especially when the system is large-scale and very complex. Therefore, we are seeking an efficient and intuitive solution approach for this VOP.

In this section, we present a way to efficiently analyze the blocking and starvation propagation of two types of systems: tree-structured networks and single-loop systems. Complex assembly/disassembly networks with multiple-loop structures are discussed in the following section.

4.3.1 Tree-Structured Networks

To analyze the propagation of blocking and starvation in a production line is straightforward. We can easily determine the blocking and starvation properties of any machine-buffer pair by examining the spatial relationship between them, i.e. upstream or downstream. However, for tree-structured networks, we are not always able to define the upstream/downstream relationship for any pair of machine and buffer. Consider the tree-structured network in Figure 4-3. Machine M_1 and buffer B_5 are both at the upstream of machine M_4, but we can not define the spatial relationship for this pair.

A tree-structured network is a connected digraph such that, given any machine M_i and buffer B_j in the network, a unique path can be identified to connect them. Recall the definition of path matrix $\Gamma(T, s) = [p_{ij}(T, s)]$ in Chapter 3. A closer observation

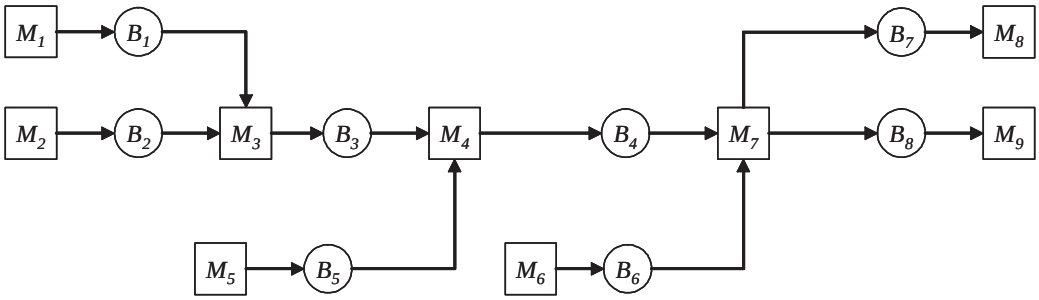


Figure 4-3: A tree-structured network

discloses the *connectivity rule* to determine the blocking and starvation properties. The connectivity rule is only defined for tree-structured networks.

Connectivity Rule Given a buffer B_j in the system,

- If the failed machine M_s is connected to B_j via B_j 's downstream machine $M_{d(j)}$, i.e. $\gamma_{u(j),j}(T,s) = -1$, then $b(j,\infty|M_s) = N_j$.
- If the failed machine M_s is connected to B_j via B_j's upstream machine $M_{u(j)}$, i.e. $\gamma_{d(j),j}(T,s) = 1$, then $b(j,\infty|M_s) = 0$.

By using this rule, we can determine the buffer level vector with respect to a specific single machine failure and thus construct the 'Machine failure – Buffer level' matrix.

The machines which are connected to a buffer via its downstream machine are in the range of blocking of the buffer, while the machines which are connected to a buffer via its upstream machine are in the range of starvation of the buffer.

4.3.2 Single-Loop Systems

The properties of blocking and starvation in single-loop systems are studied in Gershwin and Werner (2006). Given a single machine failure in the system, we could fill up the upstream buffers of the failed machine contiguously until the summation of buffer levels is equal to the loop invariant. The final buffer levels in the system can be full, empty, or partially filled.

In the example shown in Figure 4-4, all the buffer sizes are 10. The invariant of the closed loop is $I_1 = 25$. If the single machine failure is at machine M_3, the parts will fill up the upstream buffers of M_3 contiguously until the invariant condition is satisfied. For the buffers which are not in the closed loop, the connectivity rule is used to determine their levels. Therefore, the buffer level vector in the limiting propagation state due to the single machine failure at M_3 is:

$$\mathbf{b}(\infty|M_3) = \begin{bmatrix} 10 & 10 & 0 & 0 & 5 & 10 \end{bmatrix}^T$$

This buffer level vector will be the row corresponding to M_3 in 'Machine failure – Buffer level' matrix. The whole matrix is:

$$\Theta = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array} \begin{array}{c} \begin{array}{cccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 \end{array} \\ \begin{bmatrix} 0 & 0 & 5 & 0 & 10 & 10 \\ 10 & 0 & 5 & 0 & 10 & 10 \\ 10 & 10 & 0 & 0 & 5 & 10 \\ 10 & 10 & 10 & 0 & 0 & 5 \\ 10 & 10 & 10 & 10 & 0 & 5 \\ 10 & 5 & 10 & 0 & 10 & 0 \end{bmatrix} \end{array}$$

From the row corresponding to M_3, we observe that B_1, B_2 and B_6 are in the domain of blocking due to the single machine failure at M_3, while B_3 and B_4 are in the domain of starvation. B_5 is partially filled so that it is neither in the domain of blocking nor in the domain of starvation. The partially filled buffer creates a threshold.

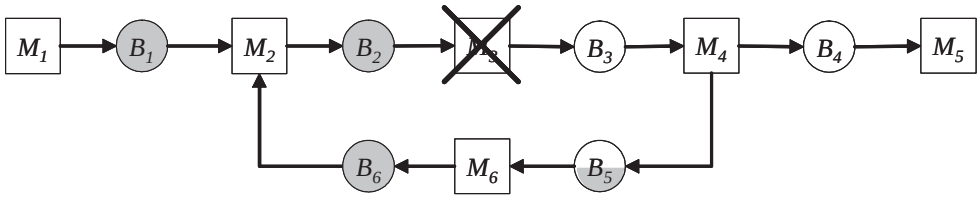


Figure 4-4: A single-loop system, Ω_1

The column corresponding to B_5 indicates that M_1, M_2 and M_6 are in the range of blocking of B_5 while M_4 and M_5 are in the range of starvation of B_5. M_3 is not in the range of blocking or starvation of B_5 since a single machine failure at M_3 causes a threshold in B_5.

4.4 Complex Multiple-Loop Structures

In production lines, tree-structured networks and single-loop systems, the blocking and starvation properties can be easily derived. Some simple algorithms, such as the connectivity rule, can be formulated to determine the buffer level vectors in the limiting propagation state. However, when there exists multiple closed loops which are coupled, i.e., have common buffers in the loops, the blocking and starvation analysis is more complicated.

4.4.1 A Case of Two Coupled Loops

Consider the two-loop system in Figure 4-5. Compared to the system in Figure 4-4, M_5 and M_3 are linked to form a new loop composed of B_3, B_4 and B_7. In the new system, two loops are coupled and the common buffer is B_3. Denote the original single-loop system by Ω_1 and the new two-loop system by Ω_2. Suppose all the buffer sizes are 10 and the invariant of the first loop L_1 is 25. We do not specify I_2, the invariant of the second loop L_2. We describe the limiting propagation state buffer levels as a function of I_2, for $0 < I_2 < 30$.

In system Ω_2, given a single machine failure at M_s, we can solve the buffer level vector in the limiting propagation state due to M_s from the optimization problem as follows:

$$\begin{aligned}
\text{maximize} \quad & q(1), q(2), ..., q(6) && (4.20)\\
\text{subject to} \quad & \mathbf{b}_{\Omega_2} = \mathbf{b}^0_{\Omega_2} + \Phi^T(\Omega_2)\mathbf{q} && (4.21)\\
& \Psi(\Omega_2)\mathbf{b}_{\Omega_2} = \mathbf{I}(\Omega_2) && (4.22)\\
& 0 \leq \mathbf{b}_{\Omega_2} \leq \mathbf{N} && (4.23)\\
& q(s) = 0 && (4.24)\\
& \mathbf{q} \geq 0 && (4.25)
\end{aligned}$$

in which

$$\mathbf{q} = \begin{bmatrix} q(1) \\ q(2) \\ \vdots \\ q(6) \end{bmatrix}, \quad \mathbf{b}_{\Omega_2} = \begin{bmatrix} b(1) \\ b(2) \\ \vdots \\ b(7) \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} 10 \\ 10 \\ \vdots \\ 10 \end{bmatrix}$$

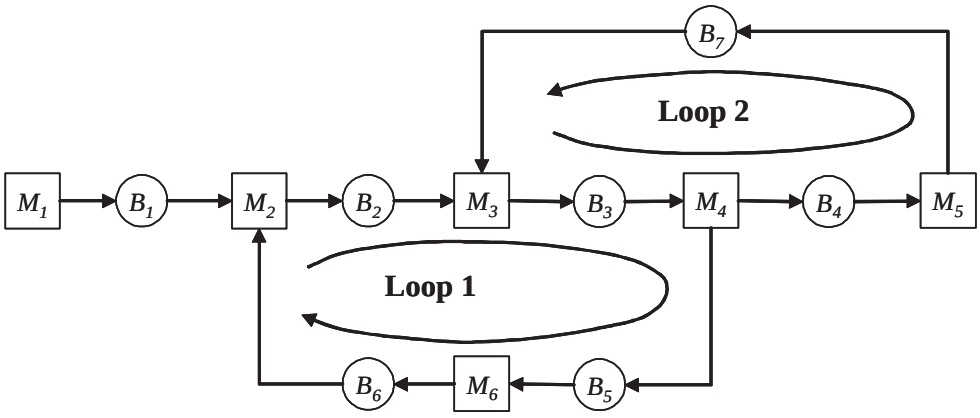


Figure 4-5: A system with two coupled loops, Ω_2

$$\Phi^T(\Omega_2) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}$$

$$\Psi(\Omega_2) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{I}(\Omega_2) = \begin{bmatrix} 25 \\ I_2 \end{bmatrix}$$

Assume a single machine failure at M_3. We compare the limiting propagation state buffer level vector of system Ω_1 with that of system Ω_2 (Figure 4-6). From the single-loop example, we know that, in system Ω_1, the limiting propagation state buffer level vector is

$$\mathbf{b}(\infty|M_3)_{\Omega_1} = [10 \quad 10 \quad 0 \quad 0 \quad 5 \quad 10]^T \tag{4.26}$$

In system Ω_1, the material in L_1 is distributed in the contiguous buffers which are upstream of M_3. However, in system Ω_2, the limiting propagation state buffer level vector will vary according to I_2 and the material in L_1 could be distributed in a non-contiguous manner.

- When $I_2 = 15$, then $\mathbf{b}(\infty|M_3)_{\Omega_2} = [10 \quad 10 \quad 0 \quad 5 \quad 5 \quad 10 \quad 10]^T$. The levels of B_2, B_3, B_5, B_6 remain the same as those in $\mathbf{b}(\infty|M_3)_{\Omega_1}$. The flow in L_1 is still contiguously distributed. The levels of B_7 and B_4 are equal to 10 and 5, respectively, in order to satisfy the invariant condition of L_2. B_3 is empty in this scenario.

- When $I_2 = 25$, then $\mathbf{b}(\infty|M_3)_{\Omega_2} = [10 \quad 10 \quad 5 \quad 10 \quad 10 \quad 0 \quad 10]^T$. B_3 is no longer empty.

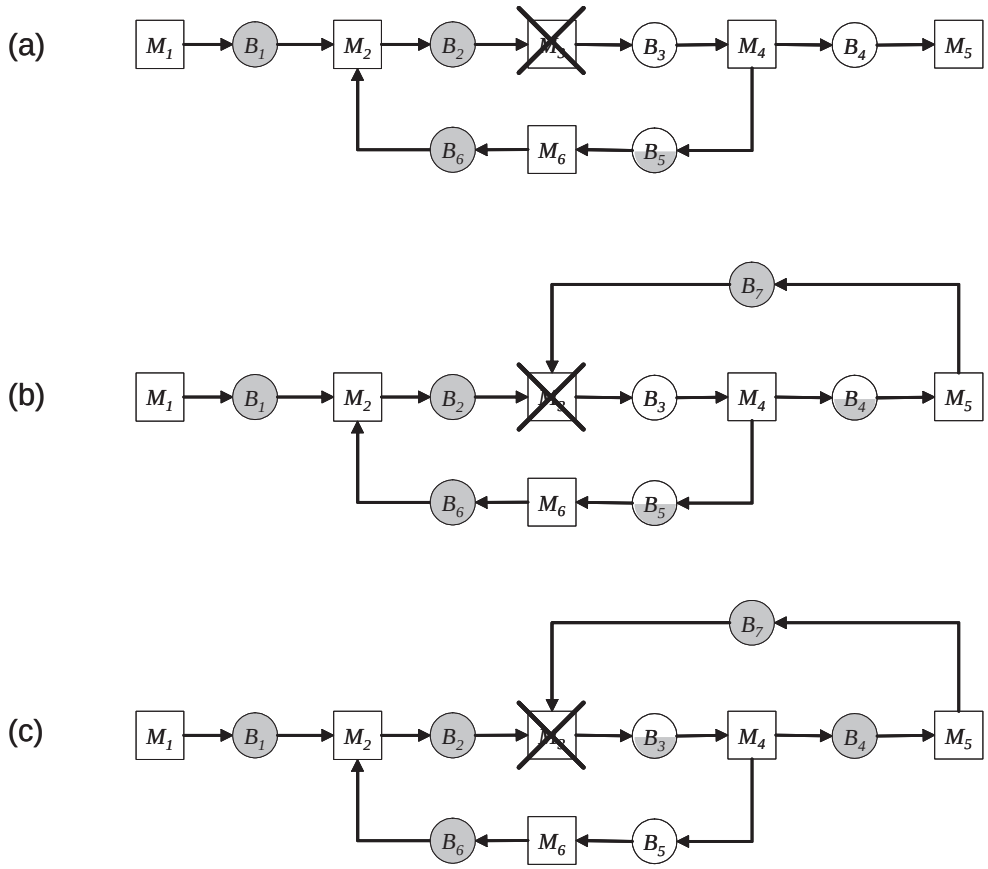


Figure 4-6: Buffer levels in the limiting propagation state due to the single machine failure at M_3. (a) Single-loop system Ω_1. (b) Two-loop system Ω_2 ($I_2 = 15$). (c) Two-loop system Ω_2 ($I_2 = 25$).

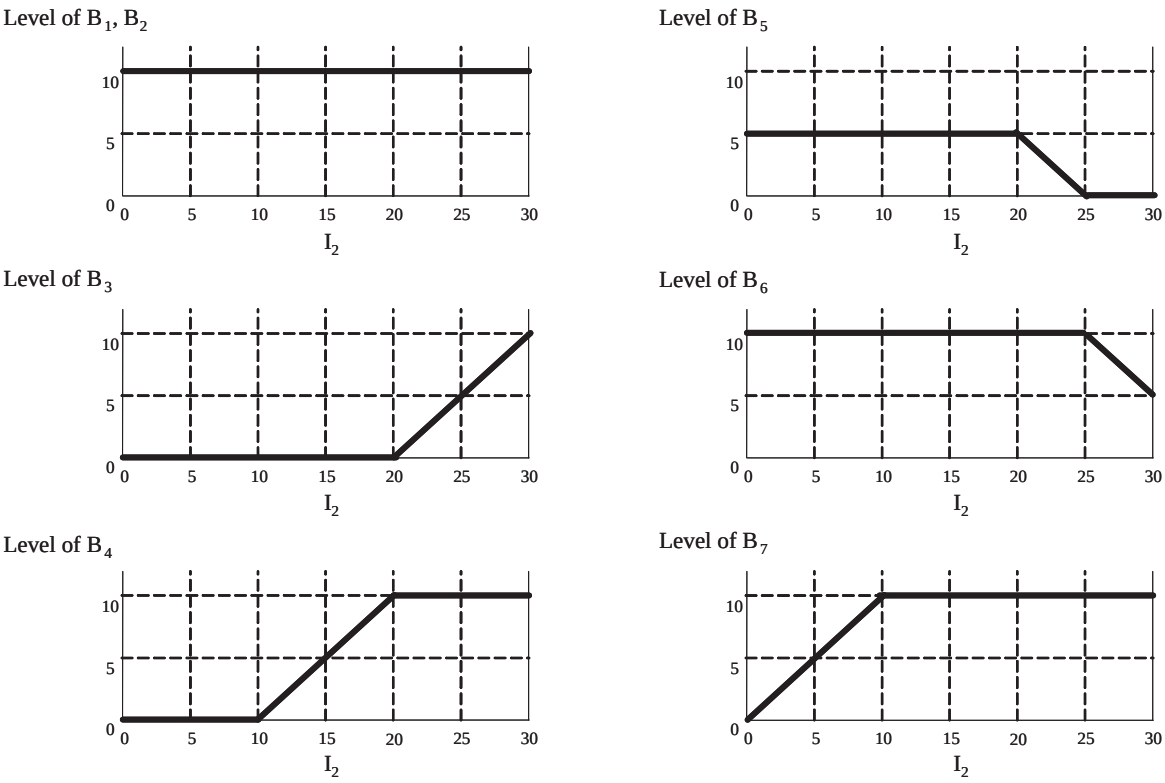


Figure 4-7: Buffer levels in the limiting propagation state due to a single machine failure at M_3

In Figure 4-7, we vary I_2 and plot the buffer levels in the limiting propagation state due to the single machine failure at M_3. In the graph, we observe that

- When $I_2 \leq 10$, the levels of the buffers within Ω_1 remain the same as those in the single-loop case.

$$\forall B_j \in \Omega_1 \qquad b(j, \infty | M_3)_{\Omega_2} = b(j, \infty | M_3)_{\Omega_1} \tag{4.27}$$

- When $10 < I_2 \leq 20$, the space in B_7 is not enough to hold I_2 parts. After B_7 is filled up, buffer B_4 contains $(I_2 - 10)$ parts.

- When $I_2 > 20$, the space in B_7 and B_4 is not enough to hold I_2 parts. After B_7 and B_4 are filled up, the remaining $(I_2 - 20)$ parts will go to B_3 in order to satisfy the invariant condition of L_2. Since the invariant condition of L_1 should also be satisfied, the sum of the levels of B_5 and B_6 decreases by $(I_2 - 20)$.

Now, let us assume a single machine failure at M_4 (Figure 4-8) and plot the buffer levels in the limiting propagation state (see Figure 4-9). Without the second loop, $\mathbf{b}(\infty | M_4)_{\Omega_1} = [10 \quad 10 \quad 10 \quad 0 \quad 0 \quad 5]^T$. In the graph, we observe that

- When $10 \leq I_2 \leq 20$, the levels of the buffers within Ω_1 remain the same as those in the single-loop case.

- When $I_2 < 10$, the level of B_3 is restricted to I_2. Therefore, in the limiting propagation state, the sum of the levels of B_5 and B_6 decreases by $(10 - I_2)$ in order to satisfy the invariant condition of L_1.

- When $I_2 > 20$, B_3 holds 10 parts, however, the space in B_7 is not enough to hold the remaining $(I_2 - 10)$ parts. After B_7 is filled up, $(I_2 - 20)$ parts will be in B_4.

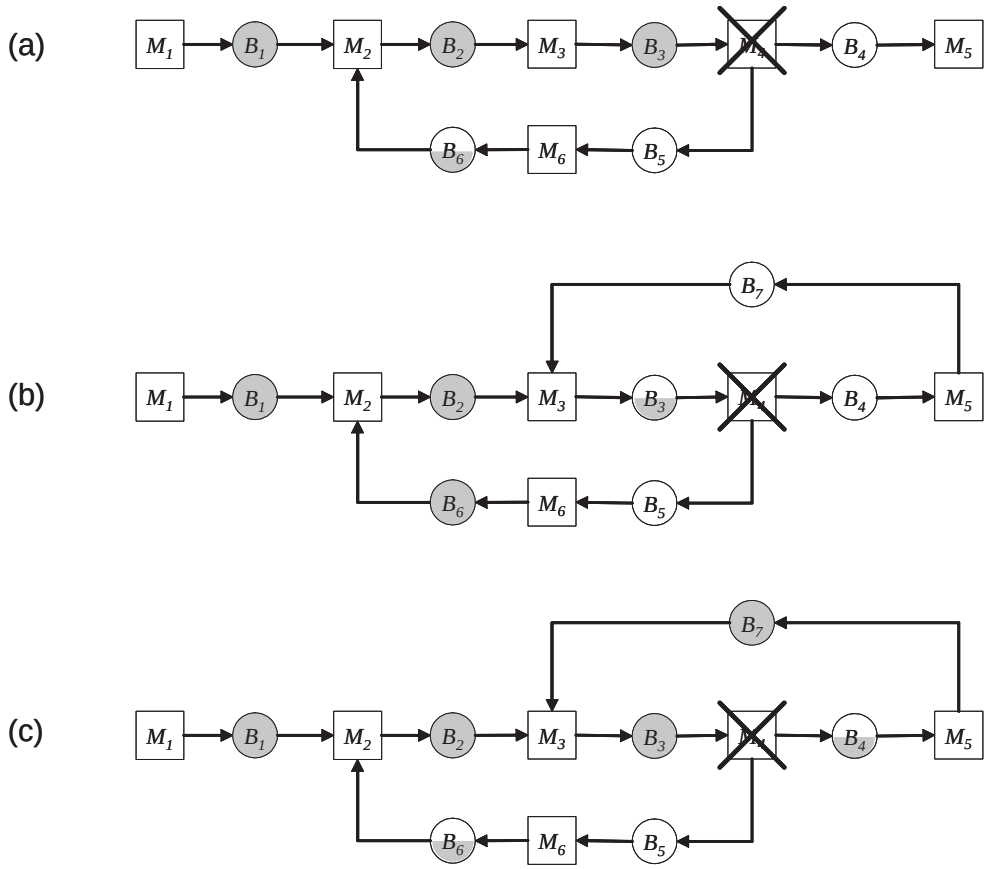


Figure 4-8: Buffer levels in the limiting propagation state due to the single machine failure at M_4. (a) Single-loop system Ω_1. (b) Two-loop system Ω_2 ($I_2 = 5$). (c) Two-loop system Ω_2 ($I_2 = 25$).

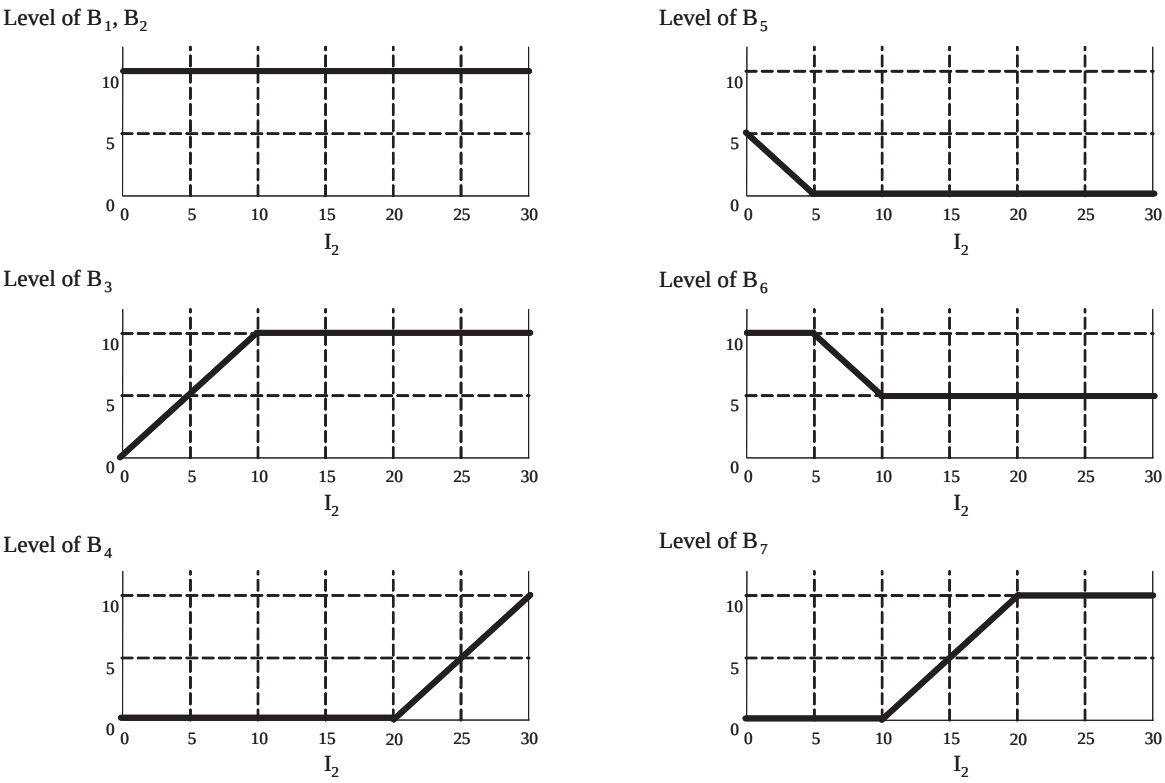


Figure 4-9: Buffer levels in the limiting propagation state due to a single machine failure at M_4

4.4.2 Coupling Effect

In the above case, the dynamics of the limiting propagation state buffer levels in system Ω_2 are affected by the coupling effect between two loops. Generally speaking, when two loops L_i and L_j are coupled together, in their corresponding circuit vectors $\mathbf{\Psi_{i\star}}$ and $\mathbf{\Psi_{j\star}}$, we can identify at least one column k which gives

$$\psi_{ik} = \psi_{jk} = 1 \qquad B_k \in B \tag{4.28}$$

In system Ω_2,

$$\Psi(\Omega_2) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

we have $\psi_{13} = \psi_{23} = 1$ and therefore L_1 and L_2 are coupled at B_3.

When multiple loops are coupled together, the material in the loops might be distributed noncontiguously in order to satisfy an invariant condition. Hence, it is impossible to develop a simple descriptive rule for determining the buffer level vector in limiting propagation state.

4.5 Induction Method

In this section, we present an induction method to efficiently find the buffer level vector in the limiting propagation state for assembly/disassembly networks with arbitrary topologies.

4.5.1 Intuition

Consider digraph G in Figure 4-10(a). Figure 4-10(b) illustrates a spanning tree T of G and the set of chords of T: c_1, c_2, c_3, and c_4. A set of fundamental loops with respect to T is shown in Figure 4-10(c).

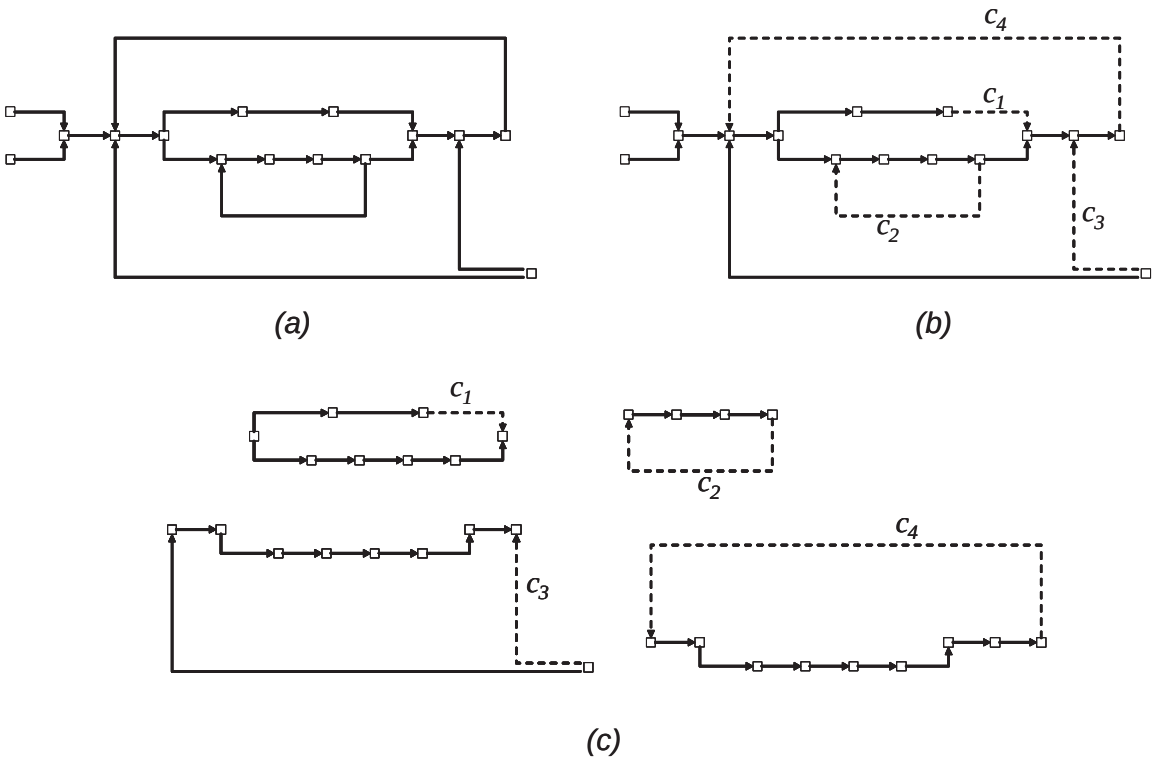


Figure 4-10: A spanning tree and a set of chords of a digraph G. (a) Digraph G. (b) Spanning tree T and the set of chords of T. (c) Set of fundamental circuits of G with respect to T. (Chords are indicated by dashed lines.)

In general, given any connected digraph G with n vertices and m arcs, a spanning tree T of G can be selected. A set of $m - n + 1$ chords of T can be identified. The connected digraph G can be constructed by adding chords $c_1, c_2, ..., c_{m-n+1}$ to spanning tree T:

$$G = T \cup c_1 \cup c_2 \cup ... \cup c_{m-n+1} \tag{4.29}$$

Since each chord c_i corresponds to a fundamental loop L_i of G (i.e., since $L_i = T \cup c_i$), we can also write:

$$G = T \cup L_1 \cup L_2 \cup ... \cup L_{m-n+1} \tag{4.30}$$

Recall that for any assembly/disassembly network Ω with n machines and m buffers, we can establish a unique underlying connected digraph G. The system can be constructed by selecting a tree-structured network corresponding to a spanning tree T of G and adding a set of buffers corresponding to chords $c_1, c_2, ..., c_{m-n+1}$ of T. Notice that directly solving limiting propagation state buffer level vectors of a tree-structured network is easy. Therefore, we develop the induction method as follows:

- Construct the 'Machine failure – Buffer level' matrix for the tree-structured network;
- Add a buffer to form a new loop. Thus the matrix will be expanded by one column, which corresponds to the added buffer. The induction step is to derive limiting propagation state buffer level vectors in the expanded matrix using those in the previous matrix.
- Repeat until all the chord buffers have been added.

4.5.2 Formulation

To formulate the induction method, we specify the

- *Induction sequence*: to select the tree-structured network and determine the sequence in which to add buffers in each induction step;

- *Induction operator*: to derive the buffer level vectors based on the results of the previous step.

Induction Sequence

For an assembly/disassembly network $\Omega = (M, B, L)$, the first step is to select a spanning tree T as a subsystem of Ω. The selection of T is not arbitrary because the set of fundamental loops L is specified by its components. Consider the two-loop system in Figure 4-11(a) and the set of fundamental loops $L = \{L_1, L_2\}$ in Figure 4-11(b).

Then the fundamental loop matrix Ψ is:

$$\Psi = \begin{array}{c} \\ L_1 \\ L_2 \end{array} \begin{array}{c} \begin{array}{ccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 \end{array} \\ \left[\begin{array}{ccccccc} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \end{array}$$

The spanning tree T in Figure 4-11(c) is appropriate because the set of fundamental loops $L = \{L_1, L_2\}$ can be constructed by adding chord buffers B_6 and B_7 respectively. However, the spanning tree T in Figure 4-11(d) is not appropriate because loop L_2 can not be constructed by adding a single buffer on spanning tree T.

Therefore, given a set of fundamental loops L, i.e. a loop matrix Ψ, we need to determine the set of chords and the spanning tree T corresponding to the set of fundamental loops in the underlying digraph G.

In loop matrix Ψ, each row vector $\mathbf{\Psi}_{k\star}$ of Ψ corresponds to a fundamental loop L_k. Each fundamental loop has a defining chord. For a fundamental L_k, a buffer could be its defining chord if the buffer is present only in this loop. In other words, in row k of matrix Ψ, the entry corresponding to this buffer is either $+1$ or -1, while all the other entries in the column corresponding to this buffer are 0.

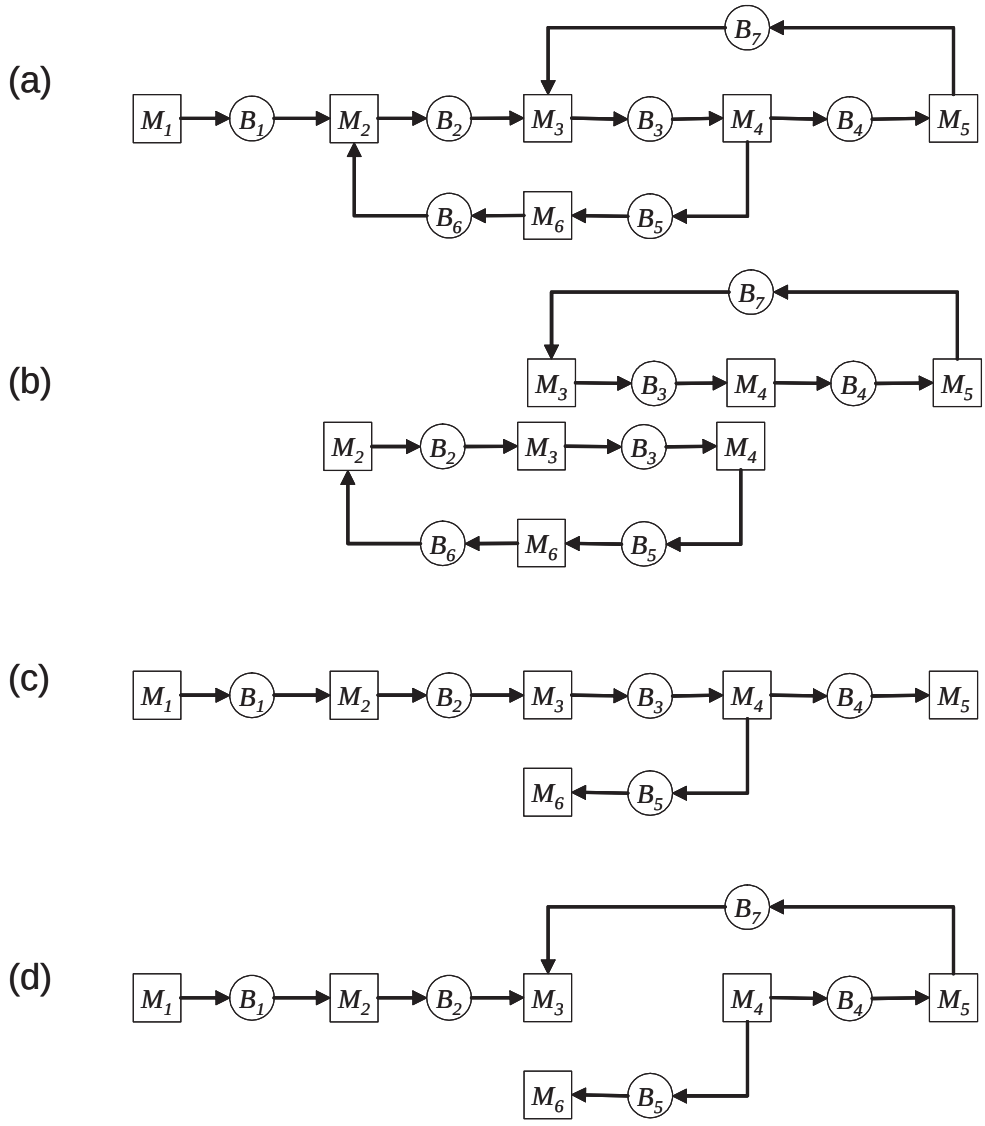


Figure 4-11: Selection of a spanning tree. (a) A Two-loop system Ω. (b) A set of fundamental loops L of Ω. (c) A valid spanning tree T with respect to L. (d) An invalid spanning tree T with respect to L.

That is, in row k corresponding to L_k, we can identify at least one column c_k which gives

$$|\psi_{k,c_k}| = 1 \tag{4.31}$$

$$\psi_{i,c_k} = 0, \qquad i = 1, ..., m-n+1, i \neq k \tag{4.32}$$

or we can write

$$|\psi_{k,c_k}| = \sum_{i=1}^{m-n+1} |\psi_{i,c_k}| = 1 \tag{4.33}$$

In the loop matrix of the two-loop system in Figure 4-11(a), B_2, B_5, or B_6 can be the defining chord of L_1; B_4 or B_7 can be the defining chord of L_2.

Notice that buffer B_{c_k} corresponds to chord c_k of loop L_k. Therefore, among the set of buffers B, a subset $B_c = \{B_{c_1}, B_{c_2}, ..., B_{c_{m-n+1}}\}$ corresponding to the set of fundamental loops $L = \{L_1, L_2, ..., L_{m-n+1}\}$ could be identified. The set of machines M and the remaining $n-1$ buffers compose a spanning tree T which is equivalent to the tree-structured network:

$$T \equiv (M, B - B_c)_\Omega \tag{4.34}$$

For example, in Figure 4-11(b), if the chords are B_6 and B_7, then the remaining buffers and the set of machines compose the spanning tree in Figure 4-11(c).

$$T \equiv (M, B_1, B_2, B_3, B_4, B_5)_\Omega \tag{4.35}$$

Denote the spanning tree T as Ω_0 and define a set of subsystems $\Omega_k, k = 1, 2, ..., m-n+1$ of network Ω:

$$\Omega_k = \begin{cases} (M, B - B_c)_\Omega & k = 0 \\ \Omega_{k-1} \cup B_{c_k} & k = 1, ..., m-n+1 \end{cases} \tag{4.36}$$

The subsystems $\Omega_0, \Omega_1, ..., \Omega_{m-n+1}$ specify the induction sequence to construct the system starting from a tree-structured network. Notice that Ω_0 is a spanning tree and Ω_{m-n+1} is system Ω.

In kth induction step, buffer B_{c_k} is added and the invariant condition with respect to loop L_k is satisfied.

Finally, we reorder the set of buffers such that

$$\begin{aligned} &B'_1, B'_2, ..., B'_{n-1} \in \Omega_0 \\ &B'_{n-1+k} = B_{c_k} \qquad k = 1, 2, ..., m-n+1 \end{aligned} \tag{4.37}$$

For example, the induction sequence of the network of Figure 4-10(a) is illustrated in Figure 4-12.

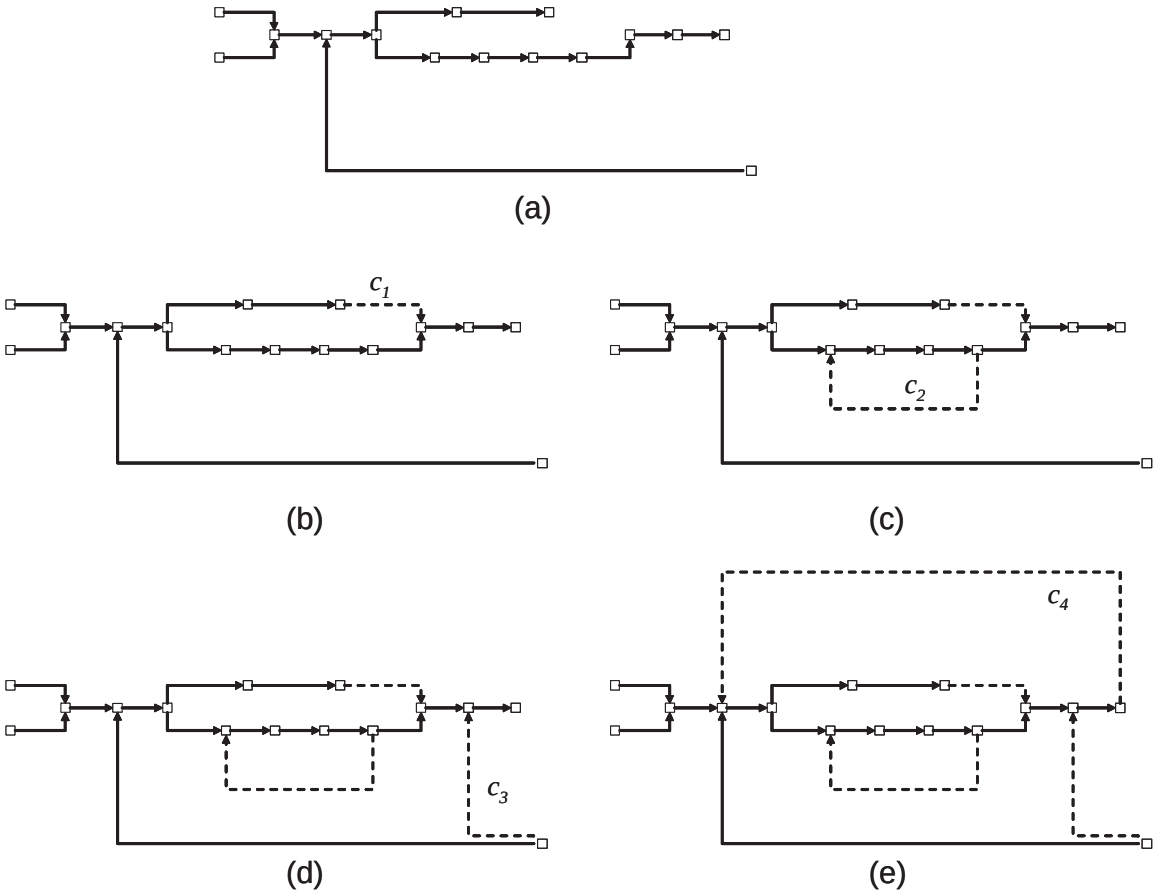


Figure 4-12: Induction sequence of a four-loop network

From this point, we drop the primes of the reordered buffers and just use $B_j, j =$

$1, ..., m$ for an easier presentation.

Induction Operation

During induction, suppose we have constructed the 'Machine failure – Buffer level' matrix of subsystem Ω_{k-1}. Denote the matrix by $\Theta(\Omega_{k-1})$. Entry $\theta_{ij}(\Omega_{k-1})$ is

$$\begin{gathered}\theta_{ij}(\Omega_{k-1}) = b(j, \infty | M_i)_{\Omega_{k-1}} \\ i = 1, ..., n; j = 1, ..., n-2+k\end{gathered} \tag{4.38}$$

The buffer level vector in the limiting propagation state due to the single machine failure at M_i is a row vector of $\Theta(\Omega_{k-1})$ given by:

$$\begin{aligned}\mathbf{\Theta}_{i\star}(\Omega_{k-1}) = [&\theta_{i1}(\Omega_{k-1}) \quad \theta_{i2}(\Omega_{k-1}) \\ &\cdots \quad \theta_{i,n-2+k}(\Omega_{k-1})]\end{aligned} \tag{4.39}$$

When we add a new buffer B_{n-1+k} to subsystem Ω_{k-1}, the new system Ω_k will have one more loop L_k whose chord is c_k. The 'Machine failure – Buffer level' matrix $\Theta(\Omega_k)$ will have one more column which corresponds to buffer B_{n-1+k}, originally B_{c_k}. Therefore, each buffer level vector has one more component:

$$\begin{aligned}\mathbf{\Theta}_{i\star}(\Omega_k) = [&\theta_{i1}(\Omega_k) \quad \theta_{i2}(\Omega_k) \quad \cdots \\ &\theta_{i,n-2+k}(\Omega_k) \quad \theta_{i,n-1+k}(\Omega_k)]\end{aligned} \tag{4.40}$$

To derive buffer level vector $\mathbf{\Theta}_{i\star}(\Omega_k)$ during induction, we should not only derive the value of $\theta_{i,n-1+k}(\Omega_k)$, but also determine the values of other components $\theta_{ij}(\Omega_k), j = 1, ..., (n-2+k)$. Depending on the system structure and the invariant of new added loop, the value of the components of $\mathbf{\Theta}_{i\star}(\Omega_{k-1})$ could either remain the same or change.

Generally speaking, given a single machine failure at M_i in system Ω_k, when invariant I_k is within a certain range, we have

$$\forall B_j \in \Omega_{k-1} \qquad \theta_{ij}(\Omega_k) = \theta_{ij}(\Omega_{k-1}) \tag{4.41}$$

This range is called *conservative range* of I_k due to the single machine failure at M_i in subsystem Ω_k. Within this range, all the components of $\mathbf{\Theta}_{i\star}(\Omega_{k-1})$ are unchanged in $\mathbf{\Theta}_{i\star}(\Omega_k)$. Different single machine failures in different subsystems have different conservative ranges. When invariant I_k is not within the conservative range, some components of buffer level vector $\mathbf{\Theta}_{i\star}(\Omega_{k-1})$ are not conserved. They will change in order to satisfy the constraints of problem LT presented in Section 4.1.2.

From the perspective of induction, the buffer level vectors of current subsystem Ω_k should be derived based on those of previous subsystem Ω_{k-1}. In addition, the invariant I_k should be an input of the induction operation since it determines whether all the components of $\mathbf{\Theta}_{i\star}(\Omega_{k-1})$ are conservative or not. Define Λ as the *induction operator*. Then we have

$$\mathbf{\Theta}_{i\star}(\Omega_k) = \Lambda(\mathbf{\Theta}_{i\star}(\Omega_{k-1}), I_k) \qquad i = 1, 2, ..., n \tag{4.42}$$

in which I_k is the invariant of loop L_k, whose chord is buffer B_{n-1+k}.

Although all-vertex incidence matrix $\Phi(\Omega_k)$ and loop matrix $\Psi(\Omega_k)$ are not specified explicitly in the induction operator, they are part of induction operation such that the conservation and invariant conditions are satisfied.

4.5.3 Solution Technique

Insight

The focus of the induction method is to design the induction operator Λ to derive the buffer level vectors correctly. Recall the limiting problem LT in Section 4.1.2 and compare the formulation of subsystem Ω_k with that of subsystem Ω_{k-1}. In the kth step of induction, B_{n-1+k} is added to form L_k. Therefore, in the formulation of Ω_k, we have additional equations and inequalities as follows:

$$\begin{aligned} b(n-1+k,t) = b(n-1+k,0) + q(u(n-1+k),t) \\ -q(d(n-1+k),t) \end{aligned} \tag{4.43}$$

$$[\psi_{k1}, \psi_{k2}, \ldots, \psi_{k,n-1+k}] \begin{bmatrix} b(1,t) \\ b(2,t) \\ \vdots \\ b(n-1+k,t) \end{bmatrix} = I_k \tag{4.44}$$

$$0 \leq b(n-1+k,t) \leq N_{n-1+k} \tag{4.45}$$

in which $u(n-1+k)$ is the upstream machine of B_{n-1+k} while $d(n-1+k)$ is the downstream machine of B_{n-1+k}. N_{n-1+k} is the size of B_{n-1+k}.

In the limiting propagation state due to the single machine failure at M_i in Ω_k, we conserve the value of components $\theta_{i1}(\Omega_{k-1}), ..., \theta_{i(n-2+k)}(\Omega_{k-1})$

$$\lim_{t\to\infty} b(j,t) = b(j,\infty) = \theta_{ij}(\Omega_{k-1}) \qquad j = 1, ..., n-2+k \tag{4.46}$$

and relax the capacity constraint (4.45). Substitute (4.46) into (4.44) and solve the buffer level of B_{n-1+k} in the limiting propagation state as follows:

$$[\psi_{k1}, \psi_{k2}, \ldots, \psi_{k,n-1+k}] \begin{bmatrix} \theta_{i1}(\Omega_{k-1}) \\ \theta_{i2}(\Omega_{k-1}) \\ \vdots \\ \theta_{i,n-2+k}(\Omega_{k-1}) \\ b(n-1+k,\infty) \end{bmatrix} = I_k \tag{4.47}$$

$$\lim_{t\to\infty} b(n-1+k,t) = b(n-1+k,\infty) = \frac{I_k - \sum_{s=1}^{n-2+k} \psi_{ks}\theta_{is}(\Omega_{k-1})}{\psi_{k,n-1+k}} \tag{4.48}$$

If $b(n-1+k,\infty)$ of (4.48) satisfies (4.45), then I_k is within the conservative range due to the single machine failure at M_i in Ω_k. Therefore, the solution of system Ω_k is

$$\theta_{ij}(\Omega_k) = \begin{cases} \theta_{ij}(\Omega_{k-1}) & \\ & j = 1, ..., n-2+k \\ \dfrac{I_k - \sum_{s=1}^{n-2+k} \psi_{ks}\theta_{is}(\Omega_{k-1})}{\psi_{k,n-1+k}} & \\ & j = n-1+k \end{cases} \tag{4.49}$$

If I_k is not within the conservative range, i.e. does not satisfy (4.45), we must adjust Θ.

- When $b(n-1+k,\infty) < 0$, there is a backlog in B_{n-1+k}. The downstream machine of B_{n-1+k} has overdrafted $-b(n-1+k,\infty)$ parts from B_{n-1+k}. The upstream machine of B_{n-1+k} is in its limiting propagation state — not able to send parts to B_{n-1+k} — due to failure, starvation, or blocking. (4.43) and (4.44) are satisfied but not for (4.45). By operating the downstream machine of B_{n-1+k} in reverse for $-b(n-1+k,\infty)$ steps, the level of B_{n-1+k} will be zero and the overdraft is eliminated. That is, (4.45) is satisfied. (4.43) and (4.44) still hold due to conservation and invariance conditions presented in Section 3.3. Since B_{n-1+k} is empty, the downstream machine of B_{n-1+k} is starved. In addition, the reverse operation does not affect the state of the upstream machine of B_{n-1+k}. Therefore, buffer B_{n-1+k} and its upstream and downstream machines are in the limiting propagation state.

- When $b(n-1+k,\infty) > N_{n-1+k}$, there is an excess in B_{n-1+k}. The downstream machine of B_{n-1+k} has overflowed B_{n-1+k} with $b(n-1+k,\infty) - N_{n-1+k}$ parts. The downstream machine of B_{n-1+k} is in its limiting propagation state — not able to take parts from B_{n-1+k} — due to failure, starvation, or blocking. (4.43) and (4.44) are satisfied but not for (4.45). By operating the upstream machine of B_{n-1+k} in reverse for $b(n-1+k,\infty) - N_{n-1+k}$ steps, the level of B_{n-1+k} will be N_{n-1+k} and the overflow is eliminated. That is, (4.45) is satisfied. (4.43) and (4.44) still hold due to conservation and invariance conditions presented in Section 3.3. Since B_{n-1+k} is full, the upstream machine of B_{n-1+k} is blocked.

In addition, the reverse operation does not affect the state of the downstream machine of B_{n-1+k}. Therefore, buffer B_{n-1+k} and its upstream and downstream machines are in the limiting propagation state.

If we can develop a rule for operating the system in reverse after adding the chord buffer to eliminate the overdraft or overflow, we therefore could make the use of the results in (4.49).

Reverse Operation Policy

We summarize the procedures of the reverse operation policy here and then present the detailed steps. First, we ignore the buffer size constraint (4.45) and calculate the nominal buffer levels by using the limiting buffer levels obtained in the previous induction step. Then we identify the buffers whose nominal levels violate the buffer size constraints. That is, these buffers have either overflows or overdrafts. We select the corresponding machines and perform reverse operations to eliminate the overflows or overdrafts. Each reverse operation might cause overflows or overdrafts in other buffers. Therefore, we must perform the reverse operations iteratively until there is no overflow and overdraft anywhere in the system. Then the buffer levels and machines are in the limiting propagation state.

Before presenting the details of the reverse operation policy, we first define some new quantities.

- *Nominal Level*: In the kth induction step, the nominal levels of buffers $B_j, j = 1, 2, ..., n-2+k$ are assigned according to (4.46).

$$\tilde{b}(j) = \theta_{ij}(\Omega_{k-1}) \qquad j = 1, ..., n-2+k \tag{4.50}$$

The nominal level of buffer B_{n-1+k} is calculated according to (4.44) while ignoring the buffer size constraint:

$$\tilde{b}(n-1+k) = \frac{I_k - \sum_{s=1}^{n-2+k} \psi_{ks}\theta_{is}(\Omega_{k-1})}{\psi_{k,n-1+k}} \tag{4.51}$$

- *Overflow and Overdraft*: Given a buffer B_j in subsystem Ω_k,
  - When $0 \leq \tilde{b}(j) \leq N_j$, B_j has neither overflow nor overdraft;
  - When $\tilde{b}(j) > N_j$, B_j has overflow which is $\tilde{b}(j) - N_j$;
  - When $\tilde{b}(j) < 0$, B_j has overdraft which is $-\tilde{b}(j)$.

  Therefore, overflow and overdraft of buffer B_j are defined as follows:

$$\delta_j^+ = \max\{\ \tilde{b}(j) - N_j, 0\ \} \tag{4.52}$$

$$\delta_j^- = \max\{\ -\tilde{b}(j), 0\ \} \tag{4.53}$$

In Figure 4-13, we plot the overflow and overdraft as functions of buffer level.

Assume in the kth induction step, the single machine failure happens at M_i in subsystem Ω_k. We reuse the solution of the buffer levels in the $(k-1)$th induction step

$$\tilde{b}(j) = \theta_{ij}(\Omega_{k-1}) \qquad j = 1, ..., n-2+k \tag{4.54}$$

and calculate the nominal level of buffer B_{n-1+k} using (4.51). If there exists a nonzero overflow δ_{n-1+k}^+ in buffer B_{n-1+k}, identify the upstream machine $M_{u(n-1+k)}$ of B_{n-1+k} and operate it in reverse to process δ_{n-1+k}^+ parts such that the levels of upstream and downstream buffers will be updated as follows:

- For the upstream buffers of $M_{u(n-1+k)}$:

$$\tilde{b}(v) = \tilde{b}(v) + \delta_{n-1+k}^+ \qquad \forall v \in U(M_{u(n-1+k)}) \tag{4.55}$$

- For the downstream buffers of $M_{u(n-1+k)}$:

$$\tilde{b}(w) = \tilde{b}(w) - \delta^{+}_{n-1+k} \qquad \forall w \in D(M_{u(n-1+k)}) \tag{4.56}$$

Notice that buffer B_{n-1+k} is one of the downstream buffers of $M_{u(n-1+k)}$, i.e. $(n-1+k) \in D(M_{u(n-1+k)})$. Therefore,

$$\tilde{b}(n-1+k) = \tilde{b}(n-1+k) - \delta^{+}_{n-1+k} \tag{4.57}$$

Substituting (4.52) into (4.57), we have

$$\begin{aligned} \tilde{b}(n-1+k) &= \tilde{b}(n-1+k) - \max\{\ \tilde{b}(n-1+k) - N_{n-1+k}, 0\ \} \\ &= \tilde{b}(n-1+k) - (\ \tilde{b}(n-1+k) - N_{n-1+k}\ ) \\ &= N_{n-1+k} \end{aligned} \tag{4.58}$$

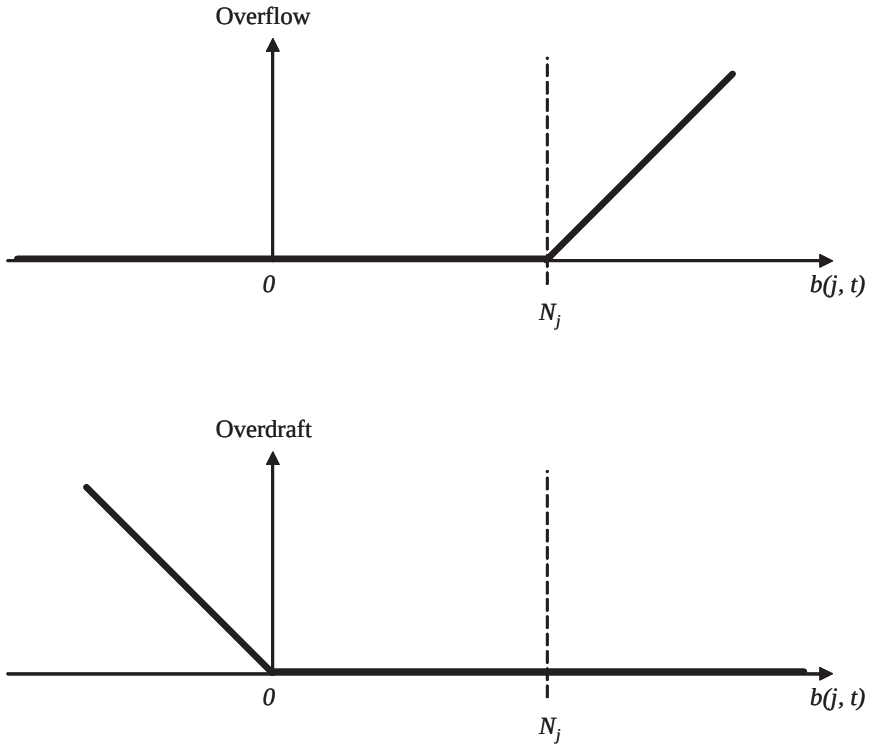


Figure 4-13: Overflow and overdraft

The updated level of buffer B_{n-1+k} becomes N_{n-1+k}. B_{n-1+k} is full and therefore $M_{u(n-1+k)}$ is blocked. The reverse operation does not affect the state of B_{n-1+k}'s downstream machine $M_{d(n-1+k)}$. Therefore, B_{n-1+k}, $M_{u(n-1+k)}$, and $M_{d(n-1+k)}$ are in the limiting propagation state according to Section 4.1.1.

On the other hand, if there exists a nonzero overdraft δ^-_{n-1+k} in buffer B_{n-1+k}, identify the downstream machine $M_{d(n-1+k)}$ of B_{n-1+k} and operate it in reverse to process δ^-_{n-1+k} parts such that the levels of upstream and downstream buffers will be updated as follows:

- For the upstream buffers of $M_{d(n-1+k)}$:

$$\tilde{b}(v) = \tilde{b}(v) + \delta^-_{n-1+k} \qquad \forall v \in U(M_{d(n-1+k)}) \tag{4.59}$$

- For the downstream buffers of $M_{d(n-1+k)}$:

$$\tilde{b}(w) = \tilde{b}(w) - \delta^-_{n-1+k} \qquad \forall w \in D(M_{d(n-1+k)}) \tag{4.60}$$

Notice that buffer B_{n-1+k} is one of the upstream buffers of $M_{d(n-1+k)}$, i.e. $(n-1+k) \in U(M_{d(n-1+k)})$. Therefore,

$$\tilde{b}(n-1+k) = \tilde{b}(n-1+k) + \delta^-_{n-1+k} \tag{4.61}$$

Substituting (4.53) into (4.61), we have

$$\begin{aligned} \tilde{b}(n-1+k) &= \tilde{b}(n-1+k) - \max\{\ -\tilde{b}(n-1+k), 0\ \} \\ &= \tilde{b}(n-1+k) - (\ -\tilde{b}(n-1+k)) \\ &= 0 \end{aligned} \tag{4.62}$$

After the update, the overdraft in buffer B_{n-1+k} is balanced. B_{n-1+k} becomes empty such that its downstream machine $M_{u(n-1+k)}$ is starved. The reverse operation does not affect the state of B_{n-1+k}'s upstream machine $M_{u(n-1+k)}$. Therefore, B_{n-1+k}, $M_{u(n-1+k)}$, and $M_{d(n-1+k)}$ are in the limiting propagation state.

As we updated the levels of all the upstream and downstream buffers of $M_{u(n-1+k)}$ or $M_{d(n-1+k)}$, there could be some new buffers other than B_{n-1+k} whose update levels have overflow or overdraft. Therefore, we need to update the buffer levels by reverse operation iteratively till there is no overflow and overdraft throughout the system.

In case that, for a given machine M_q, there exist multiple overflows among its upstream buffers or multiple overdrafts among its downstream buffers, we define the *reverse quantity*:

$$\delta_q = \max\{\delta_v^+, \delta_w^-\} \qquad \forall v \in U(M_q), w \in D(M_q) \tag{4.63}$$

we update the buffer levels as follows:

- For the upstream buffers of M_q:

$$\tilde{b}(v) = \tilde{b}(v) + \delta_q \qquad \forall v \in U(M_q) \tag{4.64}$$

- For the downstream buffers of M_q:

$$\tilde{b}(w) = \tilde{b}(w) - \delta_q \qquad \forall w \in D(M_q) \tag{4.65}$$

Induction Operator

Finally, we summarize induction operator Λ. Assume in the kth induction step, the single machine failure happens at M_i in subsystem Ω_k:

1. Assign nominal levels of buffers $B_j, j = 1, ..., n-2+k$ with the buffer level vector $\mathbf{\Theta}_{i\star}(\Omega_{k-1})$ in subsystem Ω_{k-1} using (4.54).

2. Calculate the nominal level, overflow and overdraft of buffer B_{n-1+k} using (4.51), (4.52), and (4.53).

3. Identify a set of machines $\tilde{M}$ which has either overflow among its downstream buffers or overdraft among its upstream buffers.

4. Calculate the reverse quantity δ_q of $M_q \in \tilde{M}$ using (4.63).

5. Update the upstream and downstream buffers of M_q using (4.64) and (4.65).

6. Check the limiting propagation state condition using (4.1) and (4.2).

   - If the system is not in limiting propagation state, repeat steps (3) to (6).
   - If the system is in the limiting propagation state, assign buffer level vector $\mathbf{\Theta}_{i\star}(\Omega_k)$:

$$\theta_{ij}(\Omega_k) = \tilde{b}(j) \qquad j = 1, ..., n-1+k \tag{4.66}$$

4.6 Algorithm

Given an assembly/disassembly network with topology, the algorithm for blocking and starvation analysis is given as follows:

1. Establish the graph model for assembly/disassembly system $\Omega = (M, B, L)$ composed of n machines, m buffers, and $m - n + 1$ loops.

2. Identify a set of subsystems $\Omega_k, k = 0, ..., m - n + 1$, in which Ω_0 is a spanning tree of Ω.

3. For subsystem Ω_0, construct the 'Machine failure – Buffer level' matrix $\Theta(\Omega_0)$ using 'Connectivity rule' specified in Section 4.3.1.

4. Construct matrices $\Theta(\Omega_k), k = 1, ..., m - n + 1$ using the induction method introduced in Section 4.5.3. In the kth induction step, assume the single machine failure at M_i, and derive buffer level vector $\mathbf{\Theta}_{i\star}(\Omega_k)$ using induction operator $\Lambda(\mathbf{\Theta}_{i\star}(\Omega_{k-1}), I_k)$, for each i.

In the $(m - n + 1)$th induction step, after we obtain the 'Machine failure – Buffer level' matrix of Ω_{m-n+1}, the blocking and starvation properties of the whole assembly/disassembly network are determined.

4.7 Example

Recall the two-loop system in Figure 4-5. Let B_6 and B_7 be the chords of two loops, then we can define the subsystems as follows:

$$\Omega_0 = \{M_1, M_2, M_3, M_4, M_5, M_6, B_1, B_2, B_3, B_4, B_5\}$$
$$\Omega_1 = \Omega_0 \cup B_6$$
$$\Omega_2 = \Omega_1 \cup B_7$$

Three subsystems are shown in Figure 4-14. Matrix $\Theta(\Omega_0)$ of the spanning tree is:

$$\Theta(\Omega_0) = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array} \begin{array}{c} \begin{array}{ccccc} B_1 & B_2 & B_3 & B_4 & B_5 \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 \\ 10 & 10 & 10 & 0 & 10 \end{array}\right] \end{array}$$

In the second induction step, we add buffer B_6 to form loop L_1 and assume $I_1 = 25$. When the single machine failure is at M_1, the nominal buffer level $\tilde{b}(6)$ of B_6 is 25 and the overflow δ_6^+ is 15. Conduct the reverse operation at M_6, then we have

$$\tilde{b}(6) = 25 - 15 = 10$$
$$\tilde{b}(5) = 0 + 15 = 15$$

There is a new overflow of 5 in buffer B_5. Repeat the reverse operation at M_4:

$$\tilde{b}(5) = 15 - 5 = 10$$
$$\tilde{b}(4) = 0 - 5 = -5$$
$$\tilde{b}(3) = 0 + 5 = 5$$

The levels of B_3 and B_5 satisfy the capacity constraint, whereas there is a overdraft of 5 in buffer B_4. Operate M_5 in reverse to process five parts such that $\tilde{b}(4) = -5+5 =$

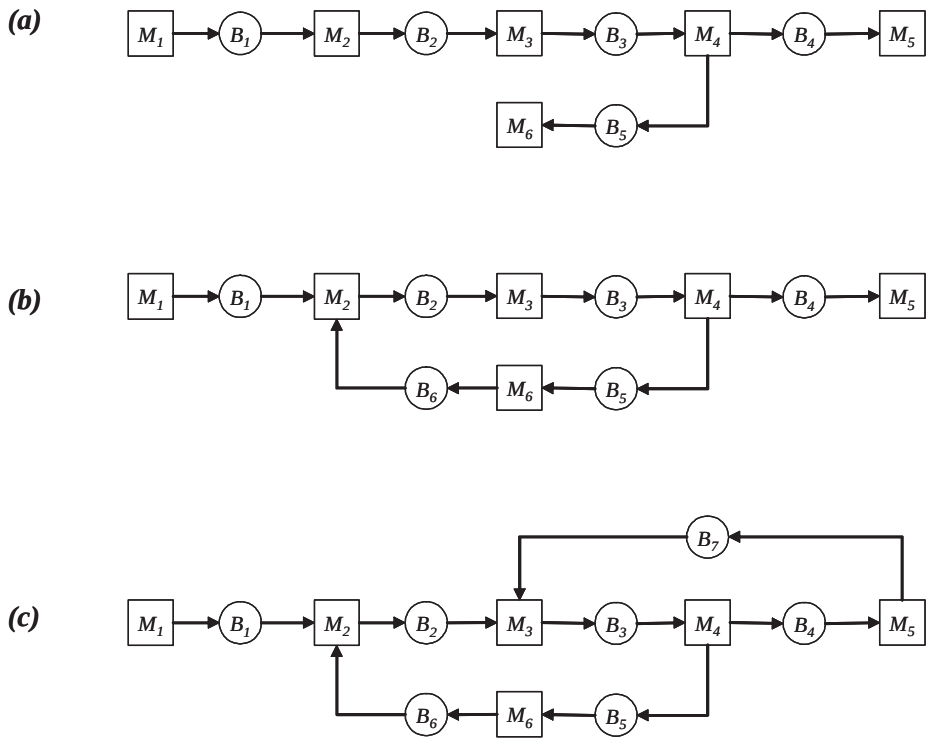


Figure 4-14: Subsystems of a two-loop system. (a) Ω_0: tree-structured network. (b) Ω_1: single-loop system. (c) Ω_2: two-loop system.

0. Finally, there is no overflow or overdraft in the system and the buffer level vector in the limiting propagation state with respect to the single machine failure at M_1 is:

$$\boldsymbol{\Theta}_{1\star}(\Omega_1) = [\ 0 \quad 0 \quad 5 \quad 0 \quad 10 \quad 10\ ]$$

Conduct similar analysis for the single machine failures at $M_2, ..., M_6$ and construct matrix $\Theta(\Omega_1)$:

$$\Theta(\Omega_1) = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array} \begin{array}{c} \begin{array}{cccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 \end{array} \\ \left[\begin{array}{cccccc} 0 & 0 & 5 & 0 & 10 & 10 \\ 10 & 0 & 5 & 0 & 10 & 10 \\ 10 & 10 & 0 & 0 & 5 & 10 \\ 10 & 10 & 10 & 0 & 0 & 5 \\ 10 & 10 & 10 & 10 & 0 & 5 \\ 10 & 5 & 10 & 0 & 10 & 0 \end{array}\right] \end{array}$$

In subsystem Ω_2, let $I_2 = 15$. For each single machine failure, we calculate the nominal buffer level of B_7. Assume the single machine failure at M_5, then we have $\tilde{b}(7) = -5$ and $\delta_7^- = 5$. Operate M_3 in reverse to process 5 parts and then we have

$$\tilde{b}(7) = -5 + 5 = 0$$
$$\tilde{b}(2) = 10 + 5 = 15$$
$$\tilde{b}(3) = 10 - 5 = 5$$

There is a new overflow of 5 in B_2. Repeat the reverse operation at M_2:

$$\tilde{b}(2) = 15 - 5 = 10$$
$$\tilde{b}(1) = 10 + 5 = 15$$
$$\tilde{b}(6) = 5 + 5 = 10$$

The levels of B_2 and B_6 satisfy the capacity constraint, whereas there is a overflow of 5 in buffer B_1. Simply operate M_1 in reverse to process five parts such that $\tilde{b}(1) = 15 - 5 = 10$. Finally, there is no overflow or overdraft in the system and the buffer level vector in the limiting propagation state due to the single machine failure

at M_5 is:

$$\mathbf{\Theta}_{5\star}(\Omega_2) = [\ 10 \quad 10 \quad 5 \quad 10 \quad 0 \quad 10 \quad 0\ ]$$

Use reverse operation policy to solve buffer level vectors $\mathbf{\Theta}_{\mathbf{i}\star}(\Omega_2), i = 1, ..., 6$. Finally, we obtain the 'Machine failure - Buffer level' matrix for the two-loop system:

$$\Theta(\Omega_2) = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 \end{array} \\ \left[\begin{array}{ccccccc} 0 & 0 & 5 & 0 & 10 & 10 & 10 \\ 10 & 0 & 5 & 0 & 10 & 10 & 10 \\ 10 & 10 & 0 & 5 & 5 & 10 & 10 \\ 10 & 10 & 10 & 0 & 0 & 5 & 5 \\ 10 & 10 & 5 & 10 & 0 & 10 & 0 \\ 10 & 5 & 10 & 0 & 10 & 0 & 5 \end{array}\right] \end{array}$$

Three additional examples of analyzing the blocking and starvation properties of large scale complex networks are presented in Appendix B.

4.8 Conclusion

To analyze the blocking and starvation of a complex assembly/disassembly network by directly solving the vector optimization problem is difficult and inefficient. An induction method provides an intuitive perspective on how to build an assembly/disassembly network with arbitrary topology from a spanning tree. A set of subsystems is constructed to specify the induction sequence. To find the buffer level vector in limiting propagation state, the reverse operation policy is invented as the induction operator. Consequently, we are able to efficiently derive the blocking and starvation properties for complex assembly/disassembly networks with arbitrary topologies.

Chapter 5

Decomposition

In this chapter, we present approximate decomposition methods for evaluating assembly/disassembly networks with arbitrary topologies. The blocking and starvation properties derived in the previous chapter provide essential information for decomposition evaluation.

We have briefly reviewed decomposition methods and the multiple-failure-mode model in Sections 1.2.1 and 1.2.2. In this chapter, we go over the decomposition evaluation procedures in Section 5.1. From Section 5.2 to Section 5.5, we present the steps for decomposition evaluation.

Most of this chapter can be found in several papers, including Tolio and Matta (1998), Levantesi et al. (May, 1999), Levantesi et al. (September, 1999), Werner (2001), Levantesi (2001), Tolio et al. (2002), and Gershwin and Werner (2006). However, in these papers, some contents are not consistent and some complicated key issues (e.g. routing buffer of failure propagation, Section 5.3.2; equations for updating remote failure rates, Section 5.4.3; equations for updating processing rates, Section 5.4.4) are not well discussed. Therefore, this chapter serves as an integrated presentation of the above papers with important corrections, extensions, and supplements.

5.1 Introduction

Decomposition is approximation technique developed to evaluate large-scale complex manufacturing systems (Gershwin 1987). We assume there is a local observer in each buffer to observe the buffer level. When the buffer level decreases and becomes empty, it appears to be due to a machine failure propagated from upstream. Likewise, when the buffer level increases gradually and becomes full, it appears to be due to a machine failure propagated from downstream. Therefore, a two-machine line is designed to approximate the behavior of the flow in the buffer. The upstream pseudo-machine of the two-machine line approximates the failures propagated from upstream, while the downstream pseudo-machine approximates the failures propagated from downstream.

5.1.1 Building Blocks

In general, an assembly/disassembly network with n machines and m buffers can be decomposed into a set of m two-machine lines. These two-machine lines are called *building blocks*, which can be evaluated analytically. Denote the building blocks by $BB(j), j = 1, 2, ..., m$. Building block $BB(j)$ corresponds to buffer B_j of the original network. The buffer of building block $BB(j)$ is the same size as buffer B_j. For example, consider the decomposition of a five-machine production line in Figure 5-1. The line is decomposed into four building blocks, $BB(1)$ to $BB(4)$.

Each building block has two pseudo-machines. In building block $BB(j)$, upstream pseudo-machine $M^u(j)$ models the collective upstream failures observed in buffer B_j. $M^u(j)$ appears to be down when the upstream machine of B_j is down or it is starved or blocked via some buffer other than B_j. On the other hand, downstream pseudo-machine $M^d(j)$ models the collective downstream failures observed in buffer B_j. In Figure 5-1, consider building block $BB(3)$. Its upstream pseudo-machine approximates the upstream failures that propagate to B_3, while its downstream pseudo-machine approximates the downstream failures observed in B_3.

Besides structurally decomposing the network into a set of building blocks, we seek the parameters of the pseudo-machines of each building block such that the behavior

of the flow in the buffer of the building block closely matches that of the flow in the buffer of the original network. The pseudo-machines are modeled as machines with single failure mode by Gershwin (1987). In analyzing the continuous processing time model, each failure mode is characterized by the failure rate p and repair rate r. For building block $BB(j)$, we must find the failure, repair and processing rates of upstream and downstream pseudo-machines ($p^u(j)$, $r^u(j)$, $\mu^u(j)$, $p^d(j)$, $r^d(j)$, $\mu^d(j)$) by deriving and solving a set of equations for conditional probabilities. This method is feasible for decomposing production lines and tree-structured networks. However, when it is adapted to closed-loop systems, the invariant condition makes it very difficult to derive the equations for the conditional probabilities.

A newer decomposition method is described in Tolio et al. (2002) and Tolio and Matta (1998). This method models the pseudo-machines as machines with multiple failure modes. Upstream pseudo-machine $M^u(j)$ of building block $BB(j)$ has all the failure modes which can cause B_j to be empty so that it could approximate the upstream failures observed in B_j. To approximate the downstream failures observed in B_j, the failure modes that can cause B_j to be full are assigned to downstream pseudo-machine $M^d(j)$. The building blocks with multiple failure modes can be

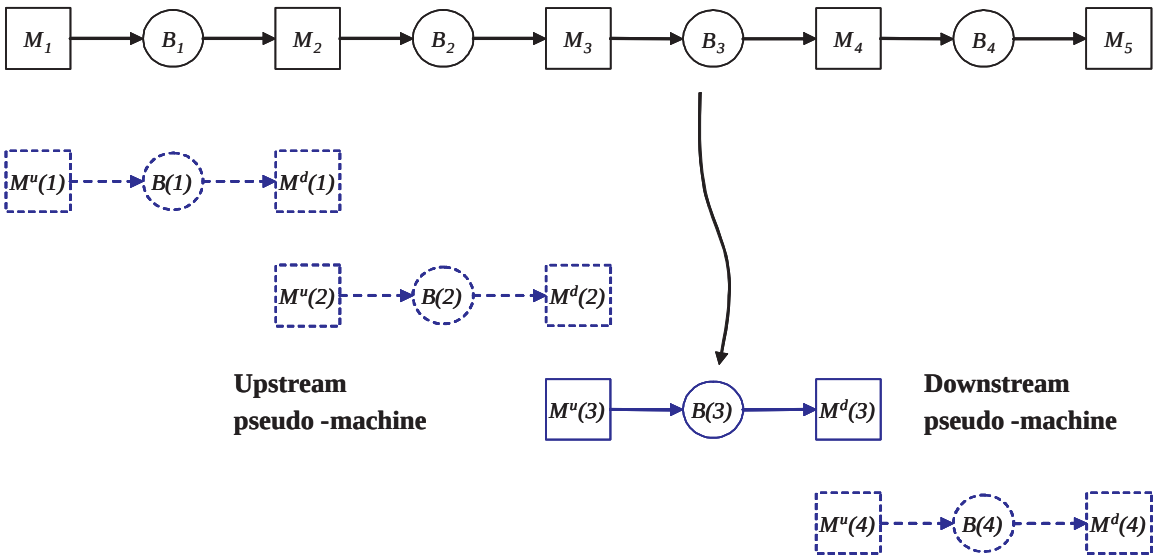


Figure 5-1: Decomposition of a five-machine production line

analytically solved by Tolio's decomposition algorithm (Tolio et al. 2002).

5.1.2 Assignment of Failure Modes

In the multiple-failure-mode method, the assignment of the failure modes is determined according to the blocking and starvation properties. Recall the definition of the ranges of blocking and starvation in Section 4.2. We use symbol $RB(j)$ to represent the set of machines in the range of blocking of B_j and symbol $RS(j)$ to represent the set of machines in the range of starvation of B_j. In building block $BB(j)$, the failure modes of the machines which are in $RS(j)$ are assigned to upstream pseudo-machine $M^u(j)$, whereas the failure modes of the machines which are in $RB(j)$ are assigned to downstream pseudo-machine $M^d(j)$.

In analyzing the blocking and starvation properties of production lines or tree-structured networks, every buffer is either full or empty under the single machine failure assumption. Machines are either in $RB(j)$ or in $RS(j)$. Therefore, in building block $BB(j)$, the failure modes of the real machines of the network are assigned either to upstream pseudo-machine $M^u(j)$ or to downstream pseudo-machine $M^d(j)$.

However, when there exist closed loops in the network, single machine failure could create partially filled buffer levels in limiting propagation state. The partially filled levels are called *buffer thresholds*. This term is first defined in Maggio et al. (2006). The transitions between the states in which the buffer level is above the threshold have different behaviors than those between the states in which the buffer level is under the threshold.

This phenomenon poses difficulties in solving the building block analytically by Tolio's multiple-failure-mode model (Tolio et al. 2002). A new evaluation method is therefore needed. Maggio et al. (2006) presents a method for evaluating small loops with thresholds. A transformation method is introduced by Werner (2001) and Gershwin and Werner (2006) for evaluating large loops. It eliminates thresholds by using perfectly reliable machines. In Section 5.2, we extend the transformation method to eliminate thresholds in complex assembly/disassembly networks by analyzing the 'Machine failure – Buffer level' matrix introduced in Section 4.2.4. The transforma-

tion method eliminates the buffer thresholds such that we can still use the evaluation algorithm developed by Tolio et al. (2002).

5.1.3 Decomposition Evaluation

After we transform an assembly/disassembly network into a new network without buffer thresholds, the new network can be decomposed into a set of building blocks. Each of them corresponds to a buffer in the system after transformation.

To decompose a network without buffer thresholds, we should not only establish a set of building blocks, but also find a way to relate the building blocks to one another in order to update the parameters through iterative evaluation. Section 5.3 discusses the setup of building blocks. The equations used to relate the parameters of the building blocks are derived in Section 5.4. Finally, in Section 5.5, an iterative algorithm is presented to determine the parameters of the building blocks and evaluate the performance in terms of production rate and average buffer levels.

5.2 Elimination of Thresholds

5.2.1 Thresholds

The threshold of a buffer is the partially filled level in the limiting propagation state due to a single machine failure. Assume a single machine failure at M_s. If the limiting propagation state level of buffer B_j satisfies the inequality:

$$0 < b(j, \infty | M_s) < N_j \tag{5.1}$$

then $l(j) = b(j, \infty | M_s)$ is said to be a threshold value of B_j. Imagine the buffer as a pipeline in Figure 5-2. The flow fills the buffer from the downstream end upward until it reaches the threshold. In buffer B_j, if we split the buffer into two parts at the threshold, the upstream part of size $N_j - l(j)$ is empty while the downstream part of size $l(j)$ is full.

A buffer could have multiple threshold values due to different single machine failures. Recall the 'Machine failure – Buffer level' matrix Θ introduced in Chapter 4. Given an assembly/disassembly network Ω with n machines and m buffers, in column j of $\Theta(\Omega)$, we can obtain the limiting propagation state levels of buffer B_j under single machine failures at M_1 through M_n. If there are g_j threshold values in column j, sort these values in descending order so that we have

$$N_j > l_1(j) > l_2(j) > \cdots > l_{g_j}(j) > 0 \tag{5.2}$$

5.2.2 Transformation

At each threshold position, insert a perfectly reliable machine M^* such that the original buffer N_j is divided into $g_j + 1$ sub-buffers $B_{j1}, B_{j2}, ..., B_{j(g_j+1)}$ from the downstream to the upstream (Figure 5-3(a)). The processing rate of the perfectly reliable machine is equal to the maximal processing rate of the machines in the original network.

The sizes of the sub-buffers are given as follows:

$$N'_{jk} = \begin{cases} l_k(j) & k = 1 \\ l_k(j) - l_{k-1}(j) & k = 2, ..., g_j \\ N_j - l_{k-1}(j) & k = g_j + 1 \end{cases} \tag{5.3}$$

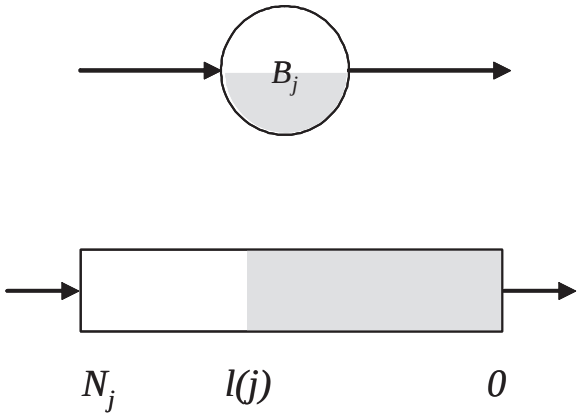


Figure 5-2: Threshold in a buffer

Therefore, buffer B_j and its upstream machine $M_{u(j)}$ and downstream machine $M_{d(j)}$ are transformed into a production line with $g_j + 2$ machines and $g_j + 1$ new buffers $B'_{j1}, B'_{j2}, ..., B'_{j(g_j+1)}$ (Figure 5-3(b)). The first machine and the last machine of the production line are, respectively, the upstream and downstream machines of B_j. The remaining g_j machines are perfectly reliable machines. When a specific single machine failure creates a threshold of $l_k(j)$, the limiting propagation state levels of $B'_{j1}, B'_{j2}, ..., B'_{jk}$ are full while the limiting propagation state levels of $B'_{j(k+1)}, B'_{j(k+2)}, ..., B'_{j(g_j+1)}$ are empty.

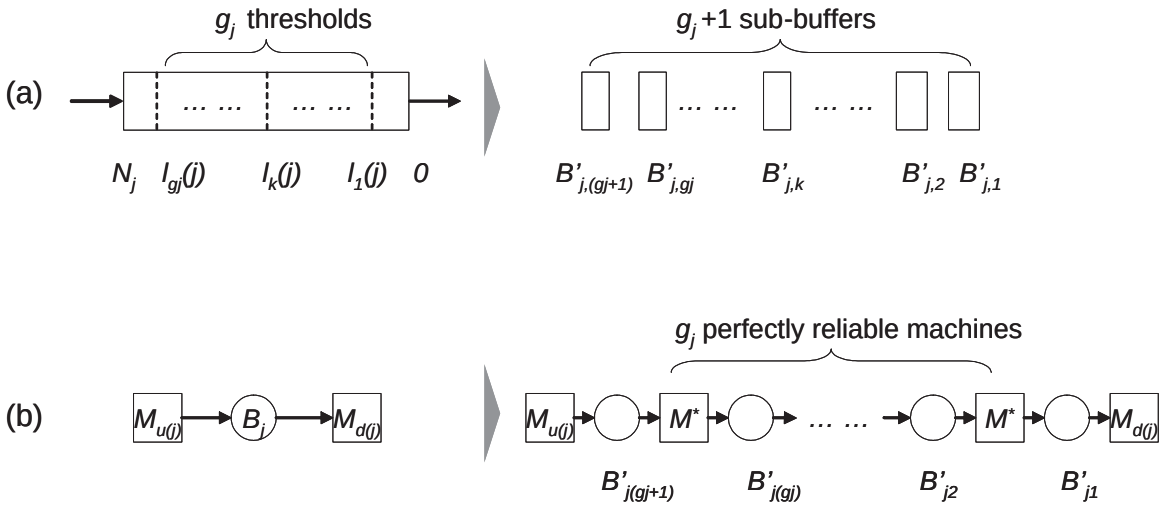


Figure 5-3: System transformation. (a) Divide the buffer into sub-buffers. (b) Two-machine one-buffer component after transformation.

After we divide all the buffers of the network, the number of buffers after the transformation is

$$m' = \sum_{j=1}^{m} (g_j + 1) \tag{5.4}$$

and the number of the set of machines after the transformation is

$$n' = n + \sum_{j=1}^{m} g_j \tag{5.5}$$

Among them, there are $\sum_{j=1}^{m} g_j$ perfectly reliable machines. The number of the loops remains the same, $m - n + 1$:

$$m' - n' + 1 = \sum_{j=1}^{m}(g_j + 1) - (n + \sum_{j=1}^{m} g_j) + 1 = m - n + 1 \tag{5.6}$$

In matrix Θ, we replace column vector $\boldsymbol{\Theta}_{\star j}$ with $g_j + 1$ column vectors $\boldsymbol{\Theta'}_{\star j1}$, $\boldsymbol{\Theta'}_{\star j2}$,...,$\boldsymbol{\Theta}_{\star j(g_j+1)}$ corresponding to buffers $B'_{j1}, B'_{j2}, ..., B'_{j(g_j+1)}$. If $\theta_{ij} = l_k(j)$, then we have

$$\theta'_{i,jv} = \begin{cases} N'_{jv} & v = 1, 2, ..., k \\ 0 & v = k+1, ..., g_j + 1 \end{cases} \tag{5.7}$$

After the column replacement, we get an expanded $n \times m'$ matrix Θ'. Notice that the newly inserted perfectly reliable machines are not listed in the matrix since they will not cause single machine failures. Only unreliable machines are listed in the matrix. The entries of Θ' are either zero or equal to the size of the corresponding buffer.

For example, the two-loop system in Section 4.6 is transformed into the system in Figure 5-4. The 'Machine failure - Buffer level' matrix of the original system is given as follows:

$$\Theta = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 \end{array} \\ \left[\begin{array}{ccccccc} 0 & 0 & 5 & 0 & 10 & 10 & 10 \\ 10 & 0 & 5 & 0 & 10 & 10 & 10 \\ 10 & 10 & 0 & 5 & 5 & 10 & 10 \\ 10 & 10 & 10 & 0 & 0 & 5 & 5 \\ 10 & 10 & 5 & 10 & 0 & 10 & 0 \\ 10 & 5 & 10 & 0 & 10 & 0 & 5 \end{array}\right] \end{array}$$

In the matrix, from column 2 to column 7, each column has a threshold value of 5. Therefore, we insert a perfectly reliable machine to split each buffer into two sub-buffers of size 5. The matrix of the transformed system is:

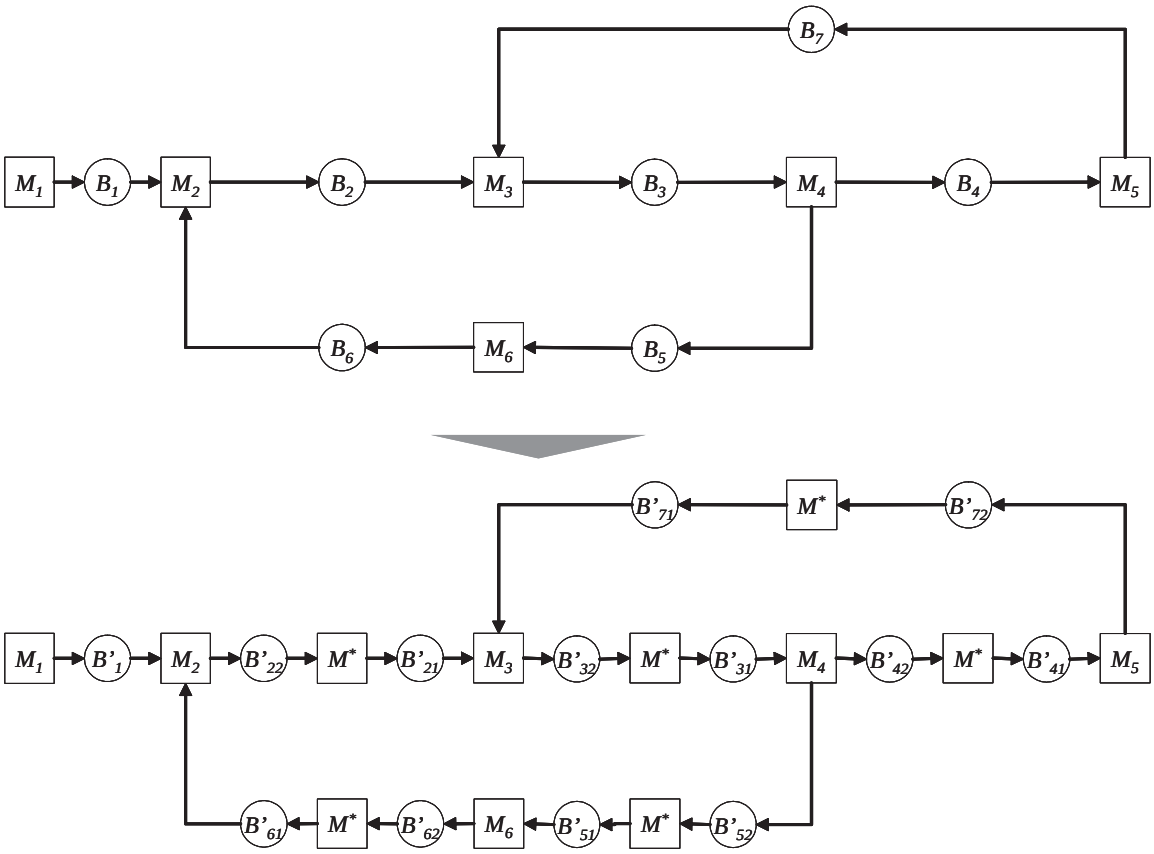


Figure 5-4: Transformation of a two-loop system.

$$\Theta' = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array} \begin{array}{c} \begin{array}{ccccccccccccc} B'_1 & B'_{22} & B'_{21} & B'_{32} & B'_{31} & B'_{42} & B'_{41} & B'_{52} & B'_{51} & B'_{62} & B'_{61} & B'_{72} & B'_{71} \end{array} \\ \left[\begin{array}{ccccccccccccc} 0 & 0 & 0 & 0 & 5 & 0 & 0 & 5 & 5 & 5 & 5 & 5 & 5 \\ 10 & 0 & 0 & 0 & 5 & 0 & 0 & 5 & 5 & 5 & 5 & 5 & 5 \\ 10 & 5 & 5 & 0 & 0 & 0 & 5 & 0 & 5 & 5 & 5 & 5 & 5 \\ 10 & 5 & 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 5 \\ 10 & 5 & 5 & 0 & 5 & 5 & 5 & 0 & 0 & 5 & 5 & 0 & 0 \\ 10 & 0 & 5 & 5 & 5 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 5 \end{array}\right] \end{array}$$

In matrix Θ', the limiting propagation state buffer levels are either full or empty and the buffer thresholds are eliminated. Given a buffer, each machine is either in the range of blocking or in the range of starvation.

The new transformed system has 12 machines and 13 buffers. The performance of the new system is approximately the same as the original system because they have the same blocking and starvation properties, except that each perfectly reliable machine in the new system could add a small amount of time delay because material needs to transverse the perfectly reliable machine. In Gershwin (1994), three models are presented: deterministic processing time model (discrete material, discrete time), exponential processing time model (discrete material, continuous time), and continuous material model (continuous material, continuous time). In continuous material model, there is no delay. In deterministic processing time model and exponential processing time model, the delay is nearly negligible.

5.3 Setup of Building Blocks

After we transform an assembly/disassembly network into a new network without buffer thresholds, the new network can be decomposed into a set of building blocks. Each of them corresponds to a buffer after transformation. This is the same as the methods presented in Werner (2001), Levantesi (2001), Maggio et al. (2006) and Gershwin and Werner (2006).

We assume that the original network Ω has n machines and m buffers, and the new transformed network Ω' has n' machines and m' buffers. The m' buffers are re-numbered from B'_1 through $B'_{m'}$. The $(n'-n)$ perfectly reliable machines that are inserted are denoted by $M_{n+1}, M_{n+2}, ..., M_{n'}$, while the index of the n machines of the original network remains the same. We assume machine $M_i, i = 1, ..., n$ has f_i failure modes. Then the set of failure modes of M_i is denoted by $\lambda_{ih}, h = 1, ..., f_i$. The total number of the failure modes of the network is given by

$$F = \sum_{i=1}^{n} f_i \tag{5.8}$$

The new network can be decomposed into m' building blocks, $BB(j), j = 1, 2, ..., m'$. In each building block $BB(j)$, its buffer has the same size as B'_j. For the pseudo-machines, we should assign the failure modes and specify the parameters.

5.3.1 Assignment of Failure Modes

We use 'Machine failure – Buffer level' matrix Θ' of new network Ω' to assign the failure modes to the pseudo-machines in each building block. In matrix Θ', there are m' columns corresponding to the m' buffers of the network. However, there are only n rows which correspond to the machines in the original network.

We use symbol $RB(j)$ to represent the set of machines in the range of blocking of B_j and symbol $RS(j)$ to represent the set of machines in the range of starvation of B_j. For building block $BB(j)$, we can assign the failure modes of $M^u(j)$ and $M^d(j)$ by observing column j in matrix Θ':

- If $\theta'_{ij} = 0$, then machine M_i is in $RS(j)$ such that the failure modes $\lambda_{ih}, h = 1, ..., f_i$ of M_i are assigned to $M^u(j)$.
- If $\theta'_{ij} = N'_{ij}$, then machine M_i is in $RB(j)$ such that the failure modes $\lambda_{ih}, h = 1, ..., f_i$ of M_i are assigned to $M^d(j)$.

The set of F failure modes $\lambda_{ih}, i = 1, ..., n; h = 1, ..., f_i$ of unreliable machines are separated into two groups in each building block. We use symbol

$$\lambda_{ih}^{u}(j) \qquad i \in RS(j); h = 1, ..., f_i$$

to represent the set of failure modes of upstream pseudo-machine $M^u(j)$. Failure mode $\lambda_{ih}^{u}(j)$ corresponds to failure mode h of machine M_i. M_i is called *source machine* of failure mode $\lambda_{ih}^{u}(j)$.

Similarly, the set of failure modes of downstream pseudo-machine $M^d(j)$ is denoted by

$$\lambda_{ih}^{d}(j) \qquad i \in RB(j); h = 1, ..., f_i$$

In addition, we differentiate the *local failure modes* and *remote failure modes* of upstream or downstream pseudo-machines. Recall that $u(j)$ and $d(j)$ are the upstream and downstream machines of B'_j, respectively. Given a failure mode $\lambda_{ih}^{u}(j)$ of the upstream pseudo-machine:

- If $i = u(j)$, $M^u(j)$ appears to be down because $M_{u(j)}$ is down. This failure mode is a local failure mode of $M^u(j)$.
- If $i \neq u(j)$, $M^u(j)$ appears to be down because machine $M_{u(j)}$ is starved or blocked via some buffer other than B'_j. This failure mode is a remote failure mode of $M^u(j)$.

Likewise, failure mode $\lambda_{ih}^{d}(j)$ of the downstream pseudo-machine is a local failure mode of $M^d(j)$ when $i = d(j)$; otherwise, it is a remote failure mode of $M^d(j)$.

5.3.2 Routing Buffer of Failure Propagation

Furthermore, for each remote failure mode of the upstream or downstream pseudo-machines, we should identify the *routing buffer* via which the failure mode propagates. The routing buffer information is used to set up the decomposition equations in Section 5.4.3.

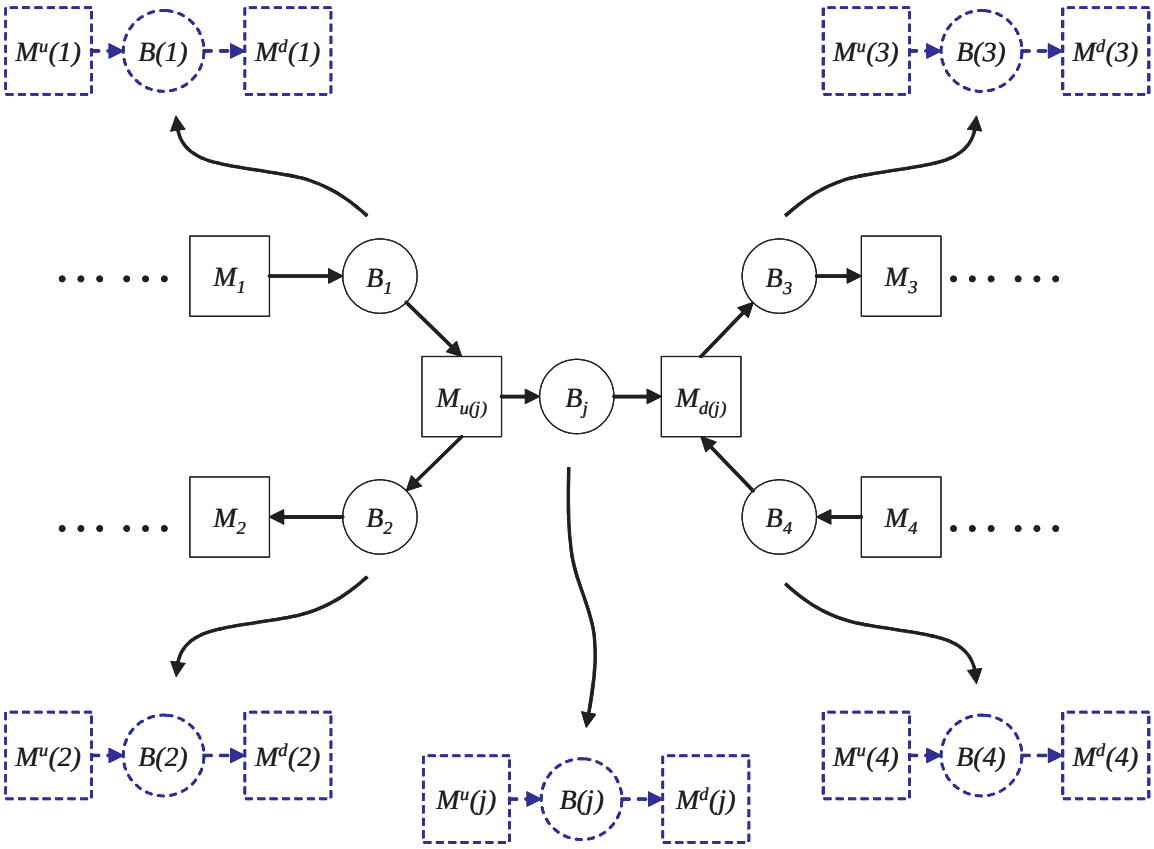


Figure 5-5: Routing buffer of failure propagation

Consider the part of network in Figure 5-5. In the network, machine $M_{u(j)}$ has upstream buffer B_1 and downstream B_2, while machine $M_{d(j)}$ has downstream buffer B_3 and upstream buffer B_4. Building blocks $BB(j)$, $BB(1)$, $BB(2)$, $BB(3)$ and $BB(4)$ correspond to buffers B_j, B_1, B_2, B_3 and B_4. Given a remote failure mode of upstream pseudo-machine $M^u(j)$, it could be propagated through M_1 as B_1 is empty and $M_{u(j)}$ is starved, or through M_2 as B_2 is full and $M_{u(j)}$ is blocked. The local observer in B_j is not able to differentiate these two cases since it can only perceive an upstream failure causing buffer B_j to be empty. However, we must specify the routing buffer of the remote failure modes assigned to the pseudo-machines.

- For remote failure mode $\lambda^u_{ih}(j)$ of upstream pseudo-machine $M^u(j)$:
  - If upstream pseudo-machine $M^u(j)$ appears to be down because $M_{u(j)}$ is starved by B_1, B_1 is the routing buffer of remote failure mode $\lambda^u_{ih}(j)$ of $M^u(j)$. Since B_1 is empty, in building block $BB(1)$, failure mode λ_{ih} is assigned to its upstream pseudo-machine $M^u(1)$.
  - If upstream pseudo-machine $M^u(j)$ appears to be down because $M_{u(j)}$ is blocked by B_2, B_2 is the routing buffer of failure modes $\lambda^u_{ih}(j)$ of $M^u(j)$. Since B_2 is full, in building block $BB(2)$, failure mode λ_{ih} is assigned to its downstream pseudo-machine $M^d(2)$.

- For remote failure mode $\lambda^d_{ih}(j)$ of downstream pseudo-machine $M^d(j)$:
  - If upstream pseudo-machine $M^d(j)$ appears to be down because $M_{d(j)}$ is blocked by B_3, B_3 is the routing buffer of failure modes $\lambda^d_{ih}(j)$ of $M^d(j)$. Since B_3 is full, in building block $BB(3)$, failure mode λ_{ih} is assigned to its downstream pseudo-machine $M^d(3)$.
  - If upstream pseudo-machine $M^d(j)$ appears to be down because $M_{d(j)}$ is starved by B_4, B_4 is the routing buffer of remote failure mode $\lambda^d_{ih}(j)$ of $M^d(j)$. Since B_4 is empty, in building block $BB(4)$, failure mode λ_{ih} is assigned to its upstream pseudo-machine $M^u(4)$.

Notice that, among the remote failure modes of a pseudo-machine, the ones having the same source machine share the same routing buffer.

In production lines or tree-structured networks, the routing buffer of remote failure mode is unique since the path between any machine-buffer pair is unique. However, in networks with closed loops, the failure mode could be propagated via multiple routing buffers. The set of routing buffers of failure modes $\lambda_{ih}^{u}(j), h = 1, ..., f_i$ is denoted by $\eta_i^u(j)$ while the set of routing buffers of failure modes $\lambda_{ih}^{d}(j), h = 1, ..., f_i$ is denoted by $\eta_i^d(j)$.

5.3.3 The Building Block Parameters

To solve the two-machine line with multiple failure modes, Tolio et al. (2002), Levantesi et al. (May, 1999) and Levantesi et al. (September, 1999) have developed three Markov process models: deterministic processing time model (discrete material, discrete time), exponential processing time model (discrete material, continuous time), and continuous material model (continuous material, continuous time). To illustrate the algorithm, we use the continuous material model. The deterministic processing time model and exponential processing time model versions of the algorithm are presented in Appendix C. The complete descriptions and all assumptions of the three models can be referred to Gershwin (1994).

In continuous material model, the parameters of building block $BB(j), j = 1, 2, ..., m'$ are denoted as follows:

- $N'(j)$: the size of the buffer of $BB(j)$. This is the only known parameter.
- $\mu^u(j)$: processing rate of upstream pseudo-machine $M^u(j)$.
- $p_{ih}^u(j)$, $r_{ih}^u(j)$: failure rate and repair rate of failure mode $\lambda_{ih}^u(j)$, respectively.
- $\mu^d(j)$: processing rate of downstream pseudo-machine $M^d(j)$.
- $p_{ih}^d(j)$, $r_{ih}^d(j)$: failure rate and repair rate of failure mode $\lambda_{ih}^d(j)$, respectively.

5.3.4 Summary

The setup of building blocks for a network without buffer thresholds includes five steps as follows:

1. For each building block, separate the machines into two groups and assign the failure modes to upstream and downstream pseudo-machines accordingly.
2. For each failure mode, specify the source machine.
3. For each pseudo-machine, differentiate the local and remote failure modes.
4. For each remote failure mode, identify the routing buffer.
5. For each building block, specify the appropriate parameters for the Markov process model. Section 5.4 shows how the values of these parameters are calculated.

5.4 Derivation of Decomposition Equations

By using the continuous multiple-failure-mode model, we assume that the behavior of the flow in B'_j can be characterized by failure, repair and processing rates with parameters $\mu^u(j)$, $p^u_{ih}(j)$, $r^u_{ih}(j)$, $\mu^d(j)$, $p^d_{ih}(j)$ and $r^d_{ih}(j)$ of building block $BB(j)$. The key to the decomposition method is to find these unknown parameters so that the flow into and out of the buffers of the building blocks closely matches the flow into and out of the corresponding buffers of the network. Recall that the upstream and downstream pseudo-machines together have a set of F failure modes. Each building block has $(2F + 2)$ unknown parameters including the failure, repair and processing rates. Therefore, $(2F + 2)$ equations per buffer, or $(2F + 2) \times m'$ conditions, are required to determine the parameters.

5.4.1 States of Pseudo-machines

There are several sets of states in which the pseudo-machines of the building blocks can be found. The states of the upstream pseudo-machine $M^u(j)$ of building block

$BB(j)$ are:

- $E^u(j)$: the event machine $M^u(j)$ is up.
- $X_h^u(j)$: the event machine $M^u(j)$ is down due to local failure modes $\lambda_{ih}^u(j), i = u(j), h = 1, ..., f_i$.
- $V_{ih}^u(j)$: the event machine $M^u(j)$ is down due to remote failure modes $\lambda_{ih}^u(j), i \in RS(j), i \neq u(j), h = 1, ..., f_i$.
- $W_{ih}^u(j)$: the event machine $M^u(j)$ is blocked due to failure modes $\lambda_{ih}^d(j)$ of downstream pseudo-machine $M^d(j)$, $i \in RB(j), h = 1, ..., f_i$.

In general, the probability of state S is denoted by $\mathbf{p}(S)$. The sum of the probabilities of all states of $M^u(j)$ gives:

$$\mathbf{p}(E^u(j)) + \sum_{h=1}^{f_{u(j)}} \mathbf{p}(X_h^u(j)) + \sum_{\substack{i \in RS(j) \\ i \neq u(j)}} \sum_{h=1}^{f_i} \mathbf{p}(V_{ih}^u(j)) + \sum_{i \in RB(j)} \sum_{h=1}^{f_i} \mathbf{p}(W_{ih}^u(j)) = 1 \tag{5.9}$$

The following are the states in which the downstream pseudo-machine $M^d(j)$ can be found:

- $E^d(j)$: the event machine $M^d(j)$ is up.
- $X_h^d(j)$: the event machine $M^d(j)$ is down due to local failure modes $\lambda_{ih}^d(j), i = d(j), h = 1, ..., f_i$.
- $V_{ih}^d(j)$: the event machine $M^d(j)$ is down due to remote failure modes $\lambda_{ih}^d(j), i \in RB(j), i \neq d(j), h = 1, ..., f_i$.
- $W_{ih}^d(j)$: the event machine $M^d(j)$ is starved due to failure modes $\lambda_{ih}^u(j)$ of upstream pseudo-machine $M^u(j)$, $i \in RS(j), h = 1, ..., f_i$.

Similarly, the sum of the probabilities of all states of $M^d(j)$ must be equal to 1. Thus,

$$\mathbf{p}(E^d(j)) + \sum_{h=1}^{f_{d(j)}} \mathbf{p}(X_h^d(j)) + \sum_{\substack{i \in RB(j) \\ i \neq d(j)}} \sum_{h=1}^{f_i} \mathbf{p}(V_{ih}^d(j)) + \sum_{i \in RS(j)} \sum_{h=1}^{f_i} \mathbf{p}(W_{ih}^d(j)) = 1 \tag{5.10}$$

5.4.2 Resumption of Flow Equations

Here we discuss the resumption of flow equations. This is based on Werner (2001) and Gershwin and Werner (2006).

Every time a machine is failed in any given failure mode, the machine gets repaired. The period in which the machine is under repair is not affected by the states of other machines or buffers. Therefore, the repair rates of the pseudo-machines in the building blocks are exactly the same as the repair rates of the corresponding real machines of the network. We have the following equations for the repair rates:

$$r_{ih}^u(j) = r_{ih} \qquad j = 1, ..., m'; \quad i \in RS(j); \quad h = 1, ..., f_i \tag{5.11}$$

$$r_{ih}^d(j) = r_{ih} \qquad j = 1, ..., m'; \quad i \in RB(j); \quad h = 1, ..., f_i \tag{5.12}$$

5.4.3 Interruption of Flow Equations

The equations presented here are based on Werner (2001) and Gershwin and Werner (2006).

Since every time machine $M^u(j)$ has a failure it has a repair, the failure frequency must equal to the repair frequency for each failure mode. We can therefore write for $M^u(j)$ the following equations:

$$p_{ih}^u(j)\ \mathbf{p}(E^u(j)) = r_{ih}^u(j)\ \mathbf{p}(X_h^u(j)) \qquad h = 1, ..., f_{u(j)} \tag{5.13}$$

$$p_{ih}^u(j)\ \mathbf{p}(E^u(j)) = r_{ih}^u(j)\ \mathbf{p}(V_{ih}^u(j)) \qquad \begin{array}{l} i \in RS(j), i \neq u(j); \\ h = 1, ..., f_i \end{array} \tag{5.14}$$

Similarly, for $M^d(j)$ the following equations hold:

$$p^d_{ih}(j)\ \mathbf{p}(E^d(j)) = r^d_{ih}(j)\ \mathbf{p}(X^d_h(j)) \qquad i = u(j); h = 1, ..., f_{d(j)} \tag{5.15}$$

$$p^d_{ih}(j)\ \mathbf{p}(E^d(j)) = r^d_{ih}(j)\ \mathbf{p}(V^d_{ih}(j)) \qquad \begin{array}{l} i \in RB(j), i \neq d(j); \\ h = 1, ..., f_i \end{array} \tag{5.16}$$

Local Failure Modes

Recall that in Chapter 3, the upstream and downstream machines of B'_j are denoted by $M_{u(j)}$ and $M_{d(j)}$, respectively. $U(M_i)$ represents the set of upstream buffers of machine M_i while $D(M_i)$ represents the set of downstream buffers of machine M_i.

The local failure rates of $M^u(j)$ are exactly the same as the local failure rates of $M^d(v), v \in U(M_{u(j)})$ and $M^u(w), w \in D(M_{u(j)}), w \neq j$ since these pseudo-machines refer to the same real machine $M_{u(j)}$. We can write:

$$\begin{gathered} \mathbf{p}(X^u_h(j)) = \mathbf{p}(X^d_h(v)) = \mathbf{p}(X^u_h(w)) \\ v \in U(M_{u(j)}); w \in D(M_{u(j)}), w \neq j \end{gathered} \tag{5.17}$$

We substitute (5.17) into (5.13) and obtain the following equations for the local failure rates of $M^u(j)$

$$\begin{gathered} p^u_{ih}(j) = \frac{\mathbf{p}(X^d_h(v))}{\mathbf{p}(E^u(j))}\ r^u_{ih}(j) = \frac{\mathbf{p}(X^u_h(w))}{\mathbf{p}(E^u(j))}\ r^u_{ih}(j) \\ i = u(j); v \in U(M_{u(j)}); w \in D(M_{u(j)}), w \neq j \end{gathered} \tag{5.18}$$

Since $r^u_{ih}(j) = r_{ij}$ and every time machine $M^u(j)$ has a failure it has a repair, the local failure rates are actually equal to the corresponding failure rates of the real machine $M_{u(j)}$:

$$p^u_{ih}(j) = p_{ih} \qquad i = u(j); v \in U(M_{u(j)}); w \in D(M_{u(j)}), w \neq j \tag{5.19}$$

Similarly, the local failure rates of $M^d(j)$ are exactly the same as the local failure rates of $M^d(v), v \in U(M_{d(j)}), v \neq j$ and $M^u(w), w \in D(M_{d(j)})$ since these pseudo-machines refer to the same real machine $M_{d(j)}$. The following equations can be derived:

$$\mathbf{p}(X_h^d(j)) = \mathbf{p}(X_h^d(v)) = \mathbf{p}(X_h^u(w)) \tag{5.20}$$
$$v \in U(M_{d(j)}), v \neq j; w \in D(M_{d(j)})$$

Substitute (5.23) into (5.15). Then we derive the local failure rates of $M^u(j)$:

$$p_{ih}^d(j) = \frac{\mathbf{p}(X_h^d(v))}{\mathbf{p}(E^d(j))}\ r_{ih}^u(j) = \frac{\mathbf{p}(X_h^u(w))}{\mathbf{p}(E^d(j))}\ r_{ih}^u(j) \tag{5.21}$$
$$i = d(j); v \in U(M_{d(j)}), v \neq j; w \in D(M_{d(j)})$$

Since $r_{ih}^d(j) = r_{ij}$ and every time machine $M^d(j)$ has a failure it has a repair, the local failure rates are actually equal to the corresponding failure rates of the real machine $M_{d(j)}$:

$$p_{ih}^d(j) = p_{ih} \qquad i = d(j); v \in U(M_{d(j)}), v \neq j; w \in D(M_{d(j)}) \tag{5.22}$$

Remote Failure Modes

The remote failure mode $\lambda_{ih}^u(j)$ of $M^u(j)$ represents the upstream failure of $BB(j)$ when real machine $M_{u(j)}$ is blocked or starved by failure mode λ_{ih} of real machine M_i via routing buffer $\eta_i^u(j)$. This relationship is illustrated in Figure 5-6.

- If $M_{u(j)}$ is the downstream machine of the routing buffer, failure mode λ_{ih} is a failure mode of the upstream pseudo-machine of building block $BB(\eta_i^u(j))$. The probability of $M^u(j)$ being in remote failure state $(V_{ih}^u(j))$ must be equal to the probability of $M^d(\eta_i^u(j))$ being in starvation state $W_{ih}^d(\eta_i^u(j))$ since pseudo-machines $M^u(j)$ and $M^d(\eta_i^u(j))$ refer to the same real machine $M_{u(j)}$.

- On the other hand, if $M_{u(j)}$ is the upstream machine of routing buffer, failure

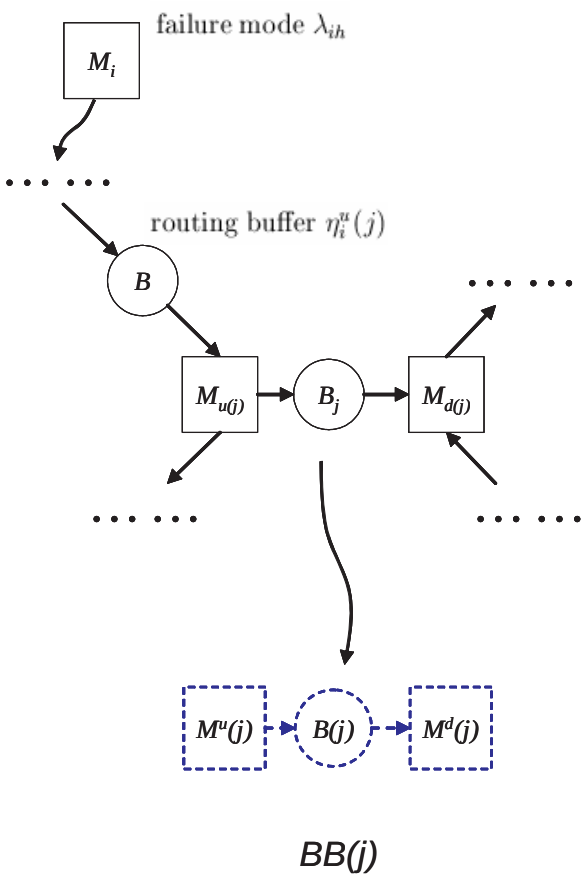


Figure 5-6: Remote failure mode $\lambda_{ih}^u(j)$ of the upstream pseudo-machine of $BB(j)$

mode λ_{ih} is a failure mode of the downstream pseudo-machine of building block $BB(\eta_i^u(j))$. The probability of $M^u(j)$ being in remote failure state $X_{ih}^u(j)$ must be equal to the probability of $M^u(\eta_i^u(j))$ being in blocking state $W_{ih}^u(\eta_i^u(j))$ since pseudo-machines $M^u(j)$ and $M^u(\eta_i^u(j))$ refer to the same real machine $M_{u(j)}$.

Let $v = \eta_i^u(j)$. Then the following equations can be written:

$$\mathbf{p}(V_{ih}^u(j)) = \begin{cases} \mathbf{p}(W_{ih}^d(v)) & u(j) = d(v) \\ \mathbf{p}(W_{ih}^u(v)) & u(j) = u(v) \end{cases} \tag{5.23}$$

in which $i \in RS(j), i \neq u(j); h = 1, ..., f_i$.

Substituting (5.23) into (5.14), we obtain the following equations for the remote failure rates of $M^u(j)$:

$$p_{ih}^u(j) = \begin{cases} \dfrac{\mathbf{p}(W_{ih}^d(v))}{E^u(j)}\, r_{ih}^u(j) & u(j) = d(v) \\ \\ \dfrac{\mathbf{p}(W_{ih}^u(v))}{\mathbf{p}(E^u(j))}\, r_{ih}^u(j) & u(j) = u(v) \end{cases} \tag{5.24}$$

Similarly, consider Figure 5-7. For the remote failure mode of $M^d(j)$, if $M_{d(j)}$ is the downstream machine of routing buffer $\eta_i^d(j)$, the probability of $M^u(j)$ being in remote failure state $X_{ih}^d(j)$ must be equal to the probability of $M^d(\eta_i^d(j))$ being in starvation state $W_{ih}^d(\eta_i^d(j))$ since pseudo-machines $M^d(j)$ and $M^d(\eta_i^d(j))$ refer to the same real machine $M_{d(j)}$.

If $M_{d(j)}$ is the upstream machine of the routing buffer, the probability of $M^d(j)$ being in remote failure state $X_{ih}^d(j)$ must be equal to the probability of $M^u(\eta_i^d(j))$ being in starvation state $PB_{ih}(\eta_i^d(j))$ since pseudo-machines $M^d(j)$ and $M^u(\eta_i^d(j))$ refer to the same real machine $M_{d(j)}$.

Let $w = \eta_i^d(j)$. The following equations hold:

$$\mathbf{p}(V_{ih}^d(j)) = \begin{cases} \mathbf{p}(W_{ih}^d(w)) & d(j) = d(w) \\ \mathbf{p}(W_{ih}^u(w)) & d(j) = u(w) \end{cases} \tag{5.25}$$

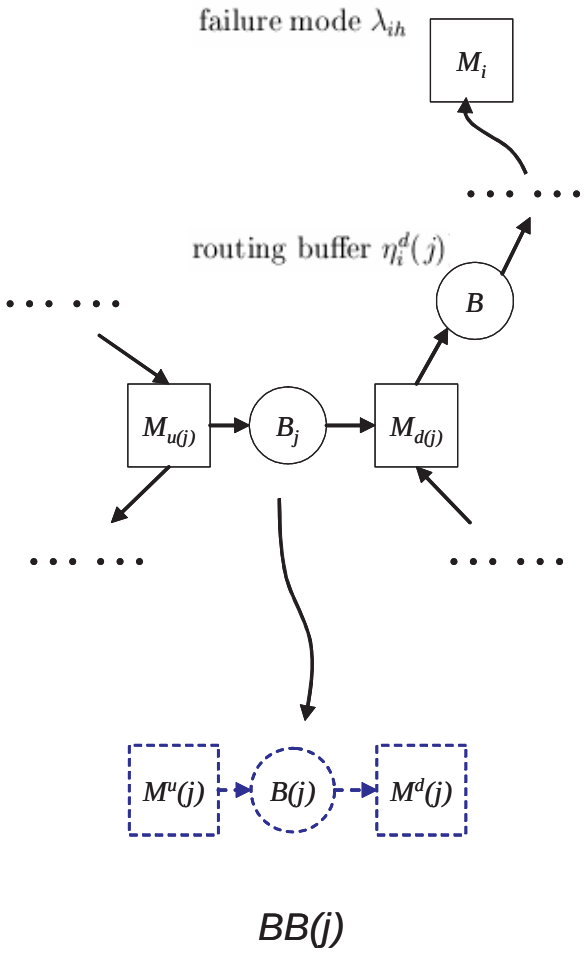


Figure 5-7: Remote failure mode $\lambda_{ih}^d(j)$ of the downstream pseudo-machine of $BB(j)$

in which $i \in RB(j), i \neq d(j); h = 1, ..., f_i$.

Substituting (5.25) into (5.16), we obtain the following equations for the remote failure rates of $M^d(j)$:

$$p^d_{ih}(j) = \begin{cases} \dfrac{\mathbf{p}(W^d_{ih}(w))}{\mathbf{p}(E^d(j))}\ r^d_{ih}(j) & d(j) = d(w) \\ \\ \dfrac{\mathbf{p}(W^u_{ih}(w))}{\mathbf{p}(E^d(j))}\ r^d_{ih}(j) & d(j) = u(w) \end{cases} \tag{5.26}$$

5.4.4 Processing Rate Equations

Finally, we develop the set of equations for the processing rates. We define $P(j)$ as the production rate of building block $BB(j)$. Then we have

$$P(j) = \mu^u(j)\ \mathbf{p}(E^u(j)) = \mu^d(j)\ \mathbf{p}(E^d(j)) \tag{5.27}$$

Consider machine $M_{u(j)}$. The average rate of flow transported from $M_{u(j)}$ to buffer B'_j is equal to the average rate of the flows transported from $M_{u(j)}$'s upstream buffers to $M_{u(j)}$, and the flows transported from $M_{u(j)}$ to its downstream buffers other than B'_j. Therefore, the production rates of building block $BB(j)$ and the building blocks corresponding to the upstream and downstream buffers of $M_{u(j)}$ are the same.

$$P(j) = P(v) \qquad v \in U(M_{u(j)}) \cup D(M_{u(j)}), v \neq j \tag{5.28}$$

Manipulate equations (5.27) and (5.28) to express the processing rate of $M^u(j)$:

$$\mu^u(j) = \frac{P(v)}{\mathbf{p}(E^u(j))} \qquad v \in U(M_{u(j)}) \cup D(M_{u(j)}), v \neq j \tag{5.29}$$

Similarly, conservation of flow at machine $M^d(j)$ implies that the production rates of building block $BB(j)$ and its downstream adjacent building blocks are the same. The processing rate of $M^d(j)$ is:

$$\mu^d(j) = \frac{P(w)}{\mathbf{p}(E^d(j))} \qquad w \in U(M_{d(j)}) \cup D(M_{d(j)}), w \neq j \tag{5.30}$$

Notice that the processing rates of the pseudo-machines should be no greater than the processing rates of the corresponding real machines in the system. In addition, the processing rates of the pseudo-machines should be no less than the lowest processing rate of the original system. We define μ_{min} as below:

$$\mu_{min} = \min_{i=1,\ldots,n'} \mu_i \tag{5.31}$$

Then we have

$$\mu_{min} \leq \mu^u(j) \leq \mu_{u(j)} \tag{5.32}$$

$$\mu_{min} \leq \mu^d(j) \leq \mu_{d(j)} \tag{5.33}$$

5.5 Iterative Evaluation

After specifying the decomposition equations, we are now in a position to describe the algorithm for evaluating assembly/disassembly networks. The algorithm described here is based on the Dallery-David-Xie (DDX) algorithm by Dallery et al. (1988).

5.5.1 Initialization

For each building block $BB(j)$, the known parameters are assigned while the unknown parameters are initialized as follows:

$$N'(j) = N'_j \tag{5.34}$$

$$\mu^u(j) = \mu_{u(j)} \tag{5.35}$$

$$p^u_{ih}(j) = p_{ih} \qquad i \in RS(j); h = 1, ..., f_i \tag{5.36}$$

$$r^u_{ih}(j) = r_{ih} \qquad i \in RS(j); h = 1, ..., f_i \tag{5.37}$$

$$\mu^d(j) = \mu_{d(j)} \tag{5.38}$$

$$p^d_{ih}(j) = p_{ih} \qquad i \in RB(j); h = 1, ..., f_i \tag{5.39}$$

$$r^d_{ih}(j) = r_{ih} \qquad i \in RB(j); h = 1, ..., f_i \tag{5.40}$$

5.5.2 Iterations

The algorithm evaluates quantities at all building blocks in a forward sequence and a reverse sequence. The forward sequence is arbitrary[1] and the reverse is the opposite of the forward sequence.

In forward order, we calculate the parameters of upstream pseudo-machines from $M^u(1)$ to $M^u(m')$, whereas in reverse order, we calculate the parameters of downstream pseudo-machines from $M^d(m')$ to $M^d(1)$.

The iterative algorithm is described here:

Step 1: Forward Procedure

1. Evaluate $BB(j)$ to obtain the probabilities of events $E^u(j)$, $E^d(j)$, $W^u_{ih}(j)$, $W^d_{ih}(j)$ and production rate $P(j)$.

2. For $j = 1, ..., m'$

   (a) Update $\mu^u(j)$.

$$\mu^u(j) = \min_{\substack{v \in U(M_{u(j)}) \cup D(M_{u(j)}) \\ v \neq j}} \frac{P(v)}{\mathbf{p}(E^u(j))} \tag{5.41}$$

[1] A method for constructing an evaluation sequences of assembly/disassembly systems is discussed in (Gershwin 1994, Section 5.4). Our experience shows that this is not necessary and an arbitrary evaluation sequence can be used.

Adjust $\mu^u(j)$ to satisfy equation (5.32):

$$\mu^u(j) = \begin{cases} \mu_{min} & \mu^u(j) < \mu_{min} \\ \mu^u(j) & \mu_{min} \leq \mu^u(j) \leq \mu_{u(j)} \\ \mu_{u(j)} & \mu^u(j) > \mu_{u(j)} \end{cases} \tag{5.42}$$

(b) Update remote failure rates $p^u_{ih}(j)$ using equation (5.24). If failure mode $\lambda^u_{ih}(j)$ is propagated through multiple routing buffers, calculate $p^u_{ih}(j)$ for each routing buffer and take the maximal value.

Step 2: Reverse Procedure

1. Evaluate $BB(j)$ to obtain the probabilities of events $E^u(j)$, $E^d(j)$, $W^u_{ih}(j)$, $W^d_{ih}(j)$ and production rate $P(j)$.

2. For $j = m', ..., 1$

   (a) Update $\mu^d(j)$.

$$\mu^d(j) = \min_{\substack{w \in U(M_{d(j)}) \cup D(M_{d(j)}) \\ w \neq j}} \frac{P(w)}{\mathbf{p}(E^d(j))} \tag{5.43}$$

   Adjust $\mu^d(j)$ to satisfy equation (5.33):

$$\mu^d(j) = \begin{cases} \mu_{min} & \mu^d(j) < \mu_{min} \\ \mu^d(j) & \mu_{min} \leq \mu^d(j) \leq \mu_{d(j)} \\ \mu_{d(j)} & \mu^d(j) > \mu_{d(j)} \end{cases} \tag{5.44}$$

   (b) Update remote failure rates $p^d_{ih}(j)$ using equation (5.26). If failure mode $\lambda^d_{ih}(j)$ is propagated through multiple routing buffers, calculate $p^d_{ih}(j)$ for each routing buffer and take the maximal value.

We perform Steps 1 and 2 iteratively until the parameters converge to an acceptable tolerance.

5.5.3 Termination Condition

During iterative evaluation, we calculate the production rate and average buffer level of each building block and update the parameters of the pseudo-machines including remote failure rates and processing rates. When the iterative algorithm converges, the percent changes of the parameters and the performance measures (production rate, buffer level) of each building block between two consecutive iterations should be less than a specified percent tolerance.

Suppose at kth iteration, the value of a specific parameter or performance measure is $Z^{(k)}$. The value in $(k+1)$th iteration is $Z^{(k+1)}$. Given the tolerance ε, the algorithm is terminated when

$$\frac{|Z^{(k+1)} - Z^{(k)}|}{Z^{(k)}} < \varepsilon \tag{5.45}$$

for all parameters and performance measures of all the building blocks.

5.5.4 Record the Performance Measures

Denote P as an estimate of system production rate.

$$P = \sum_{j=1}^{m'} P(j) \Big/ m' \tag{5.46}$$

The steady state average buffer levels $\overline{b}'(j), j = 1, ..., m'$ can be calculated by evaluating building blocks $BB(j), j = 1, ..., m'$ after termination. To obtain the average levels of the real buffers in the original system, form the appropriate sums of the average levels of the corresponding sub-buffers created in transformation in Section 5.2.2.

$$\overline{b}(k) = \sum_{B'_j \subseteq B_k} \overline{b}'(j) \qquad k = 1, ..., m \tag{5.47}$$

5.6 Conclusion

In this chapter, we introduced a transformation method to eliminate the buffer thresholds by inserting perfectly reliable machines. By doing so, we are able to assign the failure modes in each building block so that the building block could be evaluated using Tolio's two-machine line model with multiple failure modes.

To determine the unknown parameters of the building blocks of the transformed system, we derived the decomposition equations for the continuous material model. An iterative algorithm was developed based on the DDX algorithm to derive the parameters and obtain the production rate and average buffer levels.

Chapter 6

Algorithm for Assembly/Disassembly Networks

In this chapter, the blocking and starvation analysis of Chapter 4 and the decomposition evaluation of Chapter 5 are synthesized to present a comprehensive algorithm to predict the performance of assembly/disassembly networks with arbitrary topologies. For convenience, the algorithm is called Assembly/Disassembly Network (ADN) algorithm.

The inputs of the ADN algorithm are described in Section 6.1. Section 6.2 presents two phases of the ADN algorithm. Some important issues regarding the development of the algorithm are discussed in Section 6.3.

6.1 Algorithm Input

In this thesis, we develop three versions of the ADN algorithm: deterministic processing time model version (discrete material, discrete time), exponential processing time model version (discrete material, continuous time), and continuous material model version (continuous material, continuous time). The complete descriptions and all assumptions of the three models can be referred to Gershwin (1994). In this chapter, we present the continuous material model version. The deterministic processing time model and exponential processing time model versions of the algorithm are presented

in Appendix C.

Given an assembly/disassembly network Ω with n machines and m buffers, the algorithm inputs, i.e. the known parameters, are specified as follows:

Machines: $M_i, i = 1, ..., n$

- μ_i: processing rate.
- f_i: number of failure modes of M_i.
- p_{ih}, r_{ih}: failure rate and repair rate of failure mode λ_{ih}, $h = 1, ..., f_i$.

Buffers: $B_j, j = 1, ..., m$

- N_j: size of buffer B_j.

Loops: $L_k, k = 1, ..., m - n + 1$

- I_k: invariant of loop L_k.
- $\Psi = [\psi_{ij}]$: circuit (or loop) matrix with entry ψ_{kj}, $k = 1, ..., m - n + 1; j = 1, ..., m$.

$$\psi_{kj} = \begin{cases} 1 & \text{if } B_j \text{ is in } L_k \text{ and its orientation agrees with the loop orientation;} \\ -1 & \text{if } B_j \text{ is in } L_k \text{ and its orientation does not agree with the loop orientation;} \\ 0 & \text{if } B_j \text{ is not in } L_k. \end{cases} \tag{6.1}$$

Network Topology

- $\Phi = [\phi_{ij}]$: all-vertex incidence matrix of Ω with entry ϕ_{ij}, $i = 1, ..., n; j = 1, ..., m$.

$$\phi_{ij} = \begin{cases} 1 & \text{if } B_j \text{ is incident out of } M_i; \\ -1 & \text{if } B_j \text{ is incident into } M_i; \\ 0 & \text{if } B_j \text{ is not incident on } M_i. \end{cases} \tag{6.2}$$

6.2 Algorithm

6.2.1 Phase I: Blocking and Starvation Analysis

1. Establish the graph model for the input assembly/disassembly system $\Omega = (M, B, L)$ composed of n machines, m buffers and $m-n+1$ loops with specified circuit matrix Ψ and all-incidence matrix Φ. (Refer to Section 3.2.)

2. Identify $m - n + 1$ chords of the graph and create a set of subsystems $\Omega_k, k = 0, ..., m - n + 1$, in which Ω_0 is a spanning tree of Ω. (Refer to Section 4.5.2.)

3. For subsystem Ω_0, construct the 'Machine failure – Buffer level' matrix $\Theta(\Omega_0)$ using 'Connectivity rule' in analyzing tree-structured networks. (Refer to Section 4.3.1.)

4. Construct matrices $\Theta(\Omega_k), k = 1, ..., m - n + 1$ using the induction method. In the kth induction step, assume the single machine failure at M_i, solve buffer level vector $\mathbf{\Theta_{i\star}}(\Omega_k)$ using induction operator $\Gamma(\mathbf{\Theta_{i\star}}(\Omega_{k-1}), I_k)$. (Refer to Section 4.5.3.)

In the $(m-n+1)$th induction step, after we obtain the 'Machine failure – Buffer level' matrix of Ω_{m-n+1}, the blocking and starvation properties of the whole system are determined.

6.2.2 Phase II: Decomposition Evaluation

1. Eliminate buffer thresholds. (Refer to Section 5.2.)

   (a) In 'Machine failure – Buffer level' matrix $\Theta(\Omega_{m-n+1})$, identify the threshold values of each buffer.

(b) For each buffer, insert perfectly reliable machines in threshold positions and split the buffer into a set of sub-buffers.

After eliminating the buffer thresholds, we renumber the new buffers from B'_1 through $B'_{m'}$. The newly inserted perfectly reliable machines are denoted by $M_{n+1}, ..., M_{n'}$. Therefore, the transformed system consists of n' machines and m' buffers. The 'Machine failure – Buffer level' matrix corresponding to the transformed system is an $n \times m'$ matrix $\Theta'_{n \times m'}$.

2. Decompose the transformed system and set up building blocks. (Refer to Section 5.3.)

   (a) Decompose the new transformed system into m' building blocks, $BB(j), j = 1, ..., m'$.

   (b) Assign the failure modes to the upstream and downstream pseudo-machines of each building block according to 'Machine failure – Buffer level' matrix $\Theta'_{n \times m'}$.

   (c) For each pseudo-machine:

   - Specify the source machines of all failure modes;
   - Differentiate the local failure modes and remote failure modes;
   - Identify the routing buffer for the remote failure modes.

3. Evaluate the unknown parameters. The unknown parameters are the processing rates and the remote failure rates of upstream and downstream pseudo-machines in each building block. (Refer to Section 5.5.)

   (a) Initialize the buffer size, failure, repair and processing rates of building blocks $BB(j), j = 1, ..., m'$.

   (b) Evaluate the building blocks using the algorithm presented in Levantesi et al. (September, 1999) and update the unknown parameters in forward and reverse orders iteratively until the termination condition is satisfied.

(c) Record the performance measures.

- Record production rate P.
- Calculate average buffer levels $\overline{b}'(j), j = 1, ..., m'$.
- For the original buffers $B_k, k = 1, ..., m$ before transformation, obtain their average levels by summing up the average levels of the corresponding sub-buffers in the transformed system using (5.47).

Notice that the three versions of the algorithm only differ from each other in Phase II: Decomposition Evaluation. They have the same Phase I algorithm since the underlying graph models of these three version are the same.

6.3 Important Issues

In this section, we describe some important issues which will be encountered while developing computer programs for the algorithm.

6.3.1 Aggregation of Failure Modes

While applying the multiple-failure-mode model to evaluate two-machine building blocks after decomposition, analytical methods are used to solve Markov transition equations and obtain the probabilities of each pseudo-machine's states: up, down, starved, and blocked. For a pseudo-machine with repeated repair rates, those failure modes are considered to be the same failure mode as in Syrowicz (1998).

When an upstream pseudo-machine has multiple failure modes with the same repair rate (and different failure rates), these modes are aggregated as one failure mode. For example, in a model with continuous processing time, if failure modes λ_{a1}, λ_{a2},...,λ_{ak} of upstream pseudo-machine have the same repair rate, we aggregate these modes into one failure mode λ_m using following equations:

$$r_m = r_{ai} \qquad i = 1, ..., k \tag{6.3}$$

$$p_m = \sum_{i=1}^{k} p_{ai} \tag{6.4}$$

After solving a two-machine building block, the probability of the downstream pseudo-machine being in starvation due to failure mode λ_m should be disaggregated such that the probabilities of being in starvation state due to each specific failure mode before aggregation can be determined. Suppose the probability of being in starvation state due to failure mode λ_m is $\mathbf{p}(W_m^d)$. Then the probabilities of being in starvation state due to failure modes λ_{a1}, λ_{a2},..., λ_{ak} are:

$$\mathbf{p}(W_{ai}^d) = \mathbf{p}(W_m^d)\frac{p_{ai}}{p_m} \qquad i = 1, ..., k \tag{6.5}$$

When a downstream pseudo-machine has multiple failure modes λ_{a1}, λ_{a2},..., λ_{ak} with the same repair rate (different failure rate), the aggregation of failure modes follows (6.3) and (6.4). After evaluation, the probability of the upstream pseudo-machine being in blocking state W_m^u is disaggregated as:

$$\mathbf{p}(W_{ai}^u) = \mathbf{p}(W_m^u)\frac{p_{ai}}{p_m} \qquad i = 1, ..., k \tag{6.6}$$

6.3.2 Accuracy of Convergence

According to conservation of flow, the production rates of all the building blocks should be the same. However, we have observed that the analytical algorithm does not exactly satisfy conservation of flow and there exist differences between the production rates of the building blocks. (This is also observed by Werner (2001) and Gershwin and Werner (2006).)

To measure the accuracy of convergence, we define an indicator as *convergence error of production rate*. First, identify the maximal production rate P_{max} and minimal

production rate P_{min} among all the building blocks:

$$P_{max} = \max_{j=1,\ldots,m'} P(j) \tag{6.7}$$

$$P_{min} = \min_{j=1,\ldots,m'} P(j) \tag{6.8}$$

Then the convergence error of production rate is given by

$$\xi = \frac{P_{max} - P_{min}}{\sum_{j=1}^{m'} P(j) \Big/ m'} \times 100\% \tag{6.9}$$

For an acyclic system, the convergence error is usually at the same magnitude as ε, the specified percent tolerance of convergence. When a system has multiple closed loops, convergence error becomes significant compared to the specified percent tolerance.

6.3.3 Equivalence

The *Equivalence Theorem* of asssembly/disassembly networks is presented in Gershwin (1994) Section 5.6. That theorem is a general version which requires specifying the initial states of all buffers. Here, we use invariance condition presented in Section 3.3.2 to extend a specific version of equivalence theorem for assembly/disassembly systems with multiple closed loops.

In an assembly/disassembly network Ω with n machines, m buffers and $m-n+1$ closed loops, suppose we transform the network into Ω' by reversing the flow direction of buffer B_s. If B_s is only in loop L_k and the corresponding entry in loop matrix is ψ_{ks}, the invariant of loop L'_k after transformation is:

$$I'_k = I_k - \psi_{ks} N_s \tag{6.10}$$

The intuition of (6.10) is presented as follows. Suppose the performance measures of the system after transformation satisfy:

$$P' = P \tag{6.11}$$

$$\overline{b}'(s) = N_s - \overline{b}(s) \tag{6.12}$$

$$\overline{b}'(j) = \overline{b}(j) \qquad j = 1, ..., m; j \neq s \tag{6.13}$$

where P, $\overline{b}(s)$, $\overline{b}(j)$ are respectively the production rate, average level of B_s, and average levels of other buffers in network Ω; P', $\overline{b}'(s)$, $\overline{b}'(j)$ are respectively the production rate, the average level of B_s after the transformation, and the average levels of other buffers in network Ω';

In loop L'_k, since the flow direction of B_s has been changed, the entries of loop vector $\mathbf{\Psi}'_{k\star}$ are given by:

$$\psi'_{ks} = -\psi_{ks} \tag{6.14}$$

$$\psi'_{kj} = \psi_{kj} \qquad j = 1, ..., m; j \neq s \tag{6.15}$$

The loop invariant of L'_k is:

$$I'_k = [\psi'_{k1}, \ldots, \psi'_{ks}, \ldots, \psi'_{km}] \begin{bmatrix} \overline{b}'(1) \\ \vdots \\ \overline{b}'(s) \\ \vdots \\ \overline{b}'(m) \end{bmatrix} = [\psi_{k1}, \ldots, -\psi_{ks}, \ldots, \psi_{km}] \begin{bmatrix} \overline{b}(1) \\ \vdots \\ N_s - \overline{b}(s) \\ \vdots \\ \overline{b}(m) \end{bmatrix}$$

$$= [\psi_{k1}, \ldots, -\psi_{ks}, \ldots, \psi_{km}] \left\{ \begin{bmatrix} \bar{b}(1) \\ \vdots \\ -\bar{b}(s) \\ \vdots \\ \bar{b}(m) \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ N_s \\ \vdots \\ 0 \end{bmatrix} \right\}$$

$$= [\psi_{k1}, \ldots, \psi_{ks}, \ldots, \psi_{km}] \begin{bmatrix} \bar{b}(1) \\ \vdots \\ \bar{b}(s) \\ \vdots \\ \bar{b}(m) \end{bmatrix} - \psi_{ks} N_s = I_k - \psi_{ks} N_s$$

Consequently, condition (6.10) is generalized to extend the equivalence theorem.

Equivalence Theorem Let Ω and Ω' be two assembly/disassembly networks with the same number of machines and buffers. Assume there is a subset of buffers B^* such that the all-vertex incidence matrices and circuit matrices of Ω and Ω' satisfy:

$$\phi_{ij} = \begin{cases} \phi'_{ij} & B_j \notin B^* \\ -\phi'_{ij} & B_j \in B^* \end{cases} \qquad i = 1, ..., n \tag{6.16}$$

$$\psi_{kj} = \begin{cases} \psi'_{kj} & B_j \notin B^* \\ -\psi'_{kj} & B_j \in B^* \end{cases} \qquad k = 1, ..., m - n + 1 \tag{6.17}$$

That is, the flow directions of the buffers in B^* are reversed in two networks.

In addition, the loop invariant vectors $\mathbf{I}$ and $\mathbf{I}'$ of two networks satisfy:

$$I'_k = I_k - \sum_{B_j \in \Omega} \psi_{kj} N_j \tag{6.18}$$

As a consequence,

$$P' = P \tag{6.19}$$

$$\overline{b}'(j) = \overline{b}(j) \qquad j \notin B^* \tag{6.20}$$

$$\overline{b}'(j) = N_j - \overline{b}(j) \qquad j \in B^* \tag{6.21}$$

That is, the production rates of the two networks are the same, and the average levels of all the buffers in the networks whose direction of flow has not been changed are the same, The average levels of all the buffers in the networks whose direction of flow has been changed are complementary; the average number of parts in one is equal to the average amount of space in the other.

For example, consider the two-loop system Ω in Figure 6-1(a). Suppose all the machines have $p_i = 0.01$ and $r_i = 0.1$; and all the buffer sizes are 10. The invariants of loops L_1 and L_2 are 25 and 15 respectively.

The flow direction of buffer B_3 has been changed in the equivalent network Ω' in Figure 6-1(b).The all-vertex incidence matrices and circuit matrices of two networks are:

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

$$\Phi' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

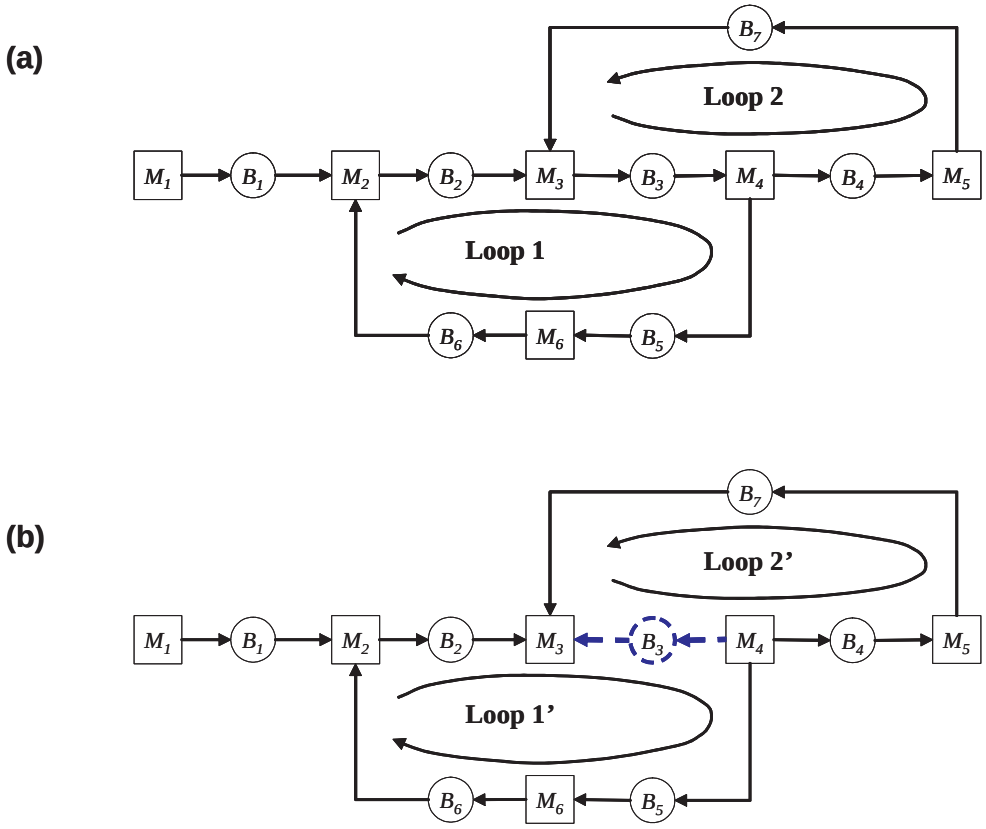


Figure 6-1: An example of network equivalence

$$\Psi = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\Psi' = \begin{bmatrix} 0 & 1 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The loop invariants of Ω' are:

$$I'_1 = I_1 - \psi_{13} N_3 = 25 - 10 = 15 \tag{6.22}$$

$$I'_2 = I_2 - \psi_{23} N_3 = 15 - 10 = 5 \tag{6.23}$$

Using the ADN algorithm to evaluate these two networks, the performance measures are compared in Table 6.1 (deterministic processing time model) and Table 6.2 (continuous material model).

	Network Ω	Network Ω'
Production rate	0.722905	0.722902
$\overline{b}(1)$	7.50150	7.50363
$\overline{b}(2)$	5.78215	5.78258
$\overline{b}(3)$	5.62328	4.37579
$\overline{b}(4)$	3.58613	3.58582
$\overline{b}(5)$	5.53872	5.53831
$\overline{b}(6)$	6.49916	6.49886
$\overline{b}(7)$	5.80020	5.80016

Table 6.1: Comparison of performance of equivalent networks (deterministic processing time model)

In the steady states, production rates are identical. The levels of buffer B_3 are complementary, while the levels of other buffers are the same.

	Network Ω	Network Ω'
Production rate	0.748502	0.748502
$\overline{b}(1)$	7.65710	7.65830
$\overline{b}(2)$	5.99654	5.99677
$\overline{b}(3)$	5.86282	4.13538
$\overline{b}(4)$	3.14046	3.13919
$\overline{b}(5)$	5.83434	5.83453
$\overline{b}(6)$	7.02815	7.02782
$\overline{b}(7)$	6.01997	6.01969

Table 6.2: Comparison of performance of equivalent networks (continuous material model)

Chapter 7

Performance of Algorithm

In this chapter, we describe numerical experiments that were performed in an effort to demonstrate the performance of the ADN (Assembly/Disassembly Network) algorithm developed in previous chapters.

The goal of the experiments is twofold. First, we want to demonstrate that the ADN algorithm is accurate, reliable and efficient while evaluating large scale assembly/disassembly networks with complex topologies. Second, we are interested in understanding the algorithm performance as a function of network scale and topological complexity.

In Chapter 8, we perform additional numerical experiments to give an understanding of the behavior of assembly/disassembly networks with multiple-loop structures.

7.1 Algorithm Performance Measures

With the purpose of comparing the performance measures of the ADN algorithm with those of discrete event simulation, all cases in this chapter are evaluated by using the deterministic processing time model version of the ADN algorithm. (Refer to Appendix C.)

7.1.1 Accuracy

Production Rate Error

The accuracy of the production rate is measured by comparing the results of the ADN algorithm with those of discrete event simulation. Two indicators are defined:

- Percent error of production rate:

$$Error_P = \frac{P_{ADN} - P_{simulation}}{P_{simulation}} \times 100\% \tag{7.1}$$

  in which P_{ADN} is the production rate evaluated by the ADN algorithm; $P_{simulation}$ is the simulation result of production rate.

- Absolute percent error of production rate:

$$Error_{|P|} = \frac{|P_{ADN} - P_{simulation}|}{P_{simulation}} \times 100\% \tag{7.2}$$

Buffer Level Error

For buffer levels, we calculate the average percent error and average absolute percent error level as follows:

- Average percent error of buffer levels:

$$Error_{BL} = \frac{\sum_{j=1}^{m} \frac{\overline{b}(j)_{ADN} - \overline{b}(j)_{simulation}}{N_j/2}}{m} \times 100\% \tag{7.3}$$

  where $\overline{b}(j)_{ADN}, j = 1, ..., m$ are the average buffer levels evaluated by the ADN algorithm; $\overline{b}(j)_{simulation}, j = 1, ..., m$ are the simulation results of average buffer levels.

- Average absolute percent error of buffer levels:

$$Error_{|BL|} = \frac{\sum_{j=1}^{m} \frac{|\bar{b}(j)_{ADN} - \bar{b}(j)_{simulation}|}{N_j/2}}{m} \times 100\% \tag{7.4}$$

The reason to use $N_j/2$ in (7-6) and (7-7) is that we want to avoid biased results — not to magnify the errors in small buffer levels while, at the same time, reducing the apparent errors in large buffer levels. More detailed discussion can be found in **?**.

Loop Invariant Error

Additionally, the average buffer levels within each loop should satisfy the loop invariant. The loop invariants of the ADN algorithm are given by

$$I_{k,ADN} = [\psi_{k1}, \psi_{k2}, \ldots, \psi_{km}] \begin{bmatrix} \bar{b}(1)_{ADN} \\ \bar{b}(2)_{ADN} \\ \vdots \\ \bar{b}(m)_{ADN} \end{bmatrix} \tag{7.5}$$

$$k = 1, 2, ..., n - m + 1$$

We calculate the average percent error and average absolute percent error as follows:

- Average percent error of loop invariants:

$$Error_I = \frac{\sum_{k=1}^{m-n+1} \frac{I_{k,ADN} - I_k}{I_k}}{m - n + 1} \times 100\% \tag{7.6}$$

- Average absolute percent error of loop invariants:

$$Error_{|I|} = \frac{\sum_{k=1}^{m-n+1} \frac{|I_{k,ADN} - I_k|}{I_k}}{m - n + 1} \times 100\% \tag{7.7}$$

Notice that the invariants are compared between the ADN results and the specified values. The simulation results are equal to the specified values exactly.

Convergence Error

The convergence error ξ of the production rate is examined:

$$\xi = \frac{P_{max} - P_{min}}{\sum_{j=1}^{m'} P(j) \Big/ m'} \times 100\% \tag{7.8}$$

where $P(j), j = 1, ..., m'$ are the production rates of all building blocks; P_{max}, P_{min} are respectively the maximal and minimal values among $P(j), j = 1, ..., m'$.

7.1.2 Convergence Reliability

As we use an iterative method to update the parameters (remote failure rates, processing rates) of the building blocks in the ADN algorithm, the parameters and performance measures (production rate, average buffer level) of the building blocks should reach convergence such that the iterations can be terminated. The convergence reliability of the ADN algorithm is indicated by the ratio of the number of cases which converge successfully to the total number of cases.

7.1.3 Speed

The most significant advantage of the ADN algorithm compared to simulation is its computational efficiency. As the ADN algorithm includes two phases, the computation time of each phase is recorded separately:

- t_I: computation time of Phase I: Blocking and Starvation Analysis;
- t_{II}: computation time of Phase II: Decomposition Evaluation.

7.2 Experiment Design

In this section, we first define the terminology used in experiments and then present the experiment architecture.

7.2.1 Definition of Terminology

Here, we define three levels of a hierarchy of experiments: scale, structure, and case.

Scale This level describes a set of networks with the same number of machines and number of loops. (The number of buffers is determined by the number of machines and loops — an n-machine k-loop system has $n - 1 + k$ buffers.) The networks have different topologies. The parameters of machines, buffers, and loops also differ from each other.

Structure This level describes a set of networks with the same topological structure and the specified scale. The parameters of machines, buffers, and loops vary from each other.

Case This level describes a specific network (the parameters of machines and buffers) with the specified scale and structure.

7.2.2 Experiment Architecture

In Figure 7-1, we illustrate the experiment architecture. The components in the graph are explained as follows:

Experiment Description This component is the input of the experiment architecture. We need to specify the scales for random case generation. The description of each scale includes the number of machines, the number of loops, the number of structures per scale, and the number of cases per structure. Each random case is identified by a unique combination of scale code, structure code, and case code.

SysG (System Generator) This is an algorithm for random case generation. It can generate a set of connected networks with random structures and random machine and buffer parameters. Before generating a random case, the scale code, structure code, and case code should be assigned. The details of the SysG algorithm are presented in Appendix D.

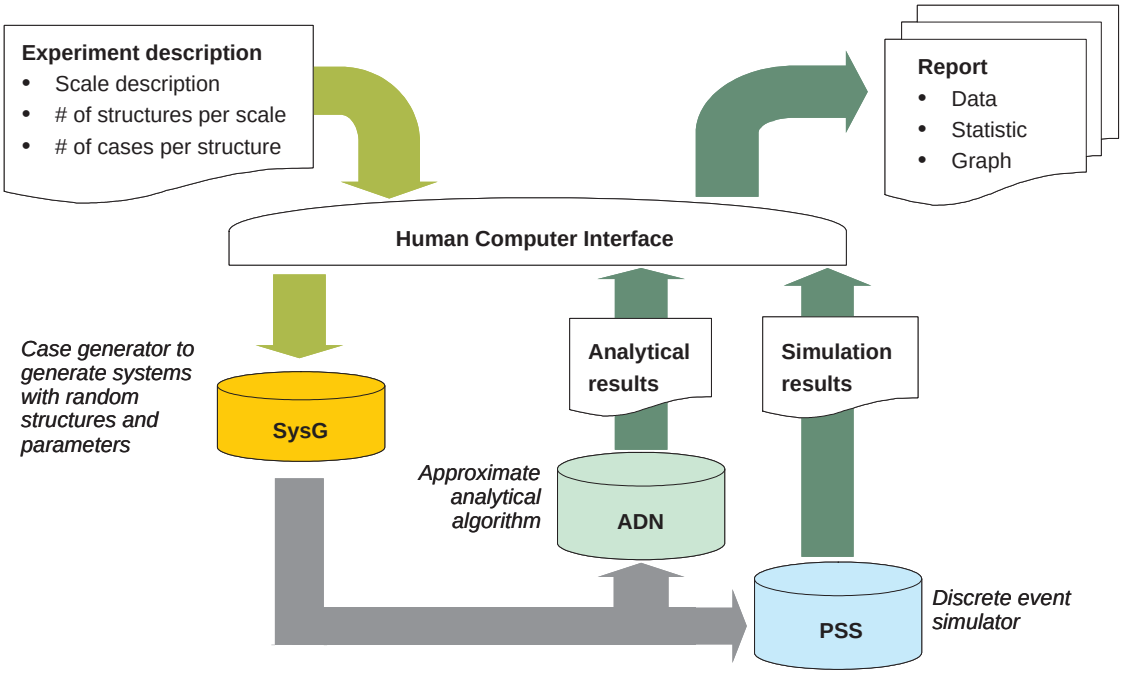


Figure 7-1: Experiment architecture

ADN (Assembly/Disassembly Network) algorithm This is the evaluation algorithm developed in previous chapters. This algorithm can predict the production rate and average buffer levels of assembly/disassembly networks with arbitrary topologies.

PSS (Production System Simulator) This is a discrete event simulation program. The transient period is set to 10,000 time units and data collection period is set to 1,000,000 time units. For the experiments in this thesis, the half width of the 98% confidence interval on the production rate falls below 1% of its mean value and the half widths of the 98% confidence intervals on the average buffer levels are less than 3.5% of the mean value of average buffer levels.

Report This component is the output of the experiment architecture. In addition to exporting the results of the ADN algorithm and simulation separately, we also compare the results and export the errors defined in Section 7.1.

Both the ADN algorithm and PSS are developed on the Visual C++ 6.0 platform. All experiments in this thesis were run on a 2.4 GHz Pentium IV desktop computer with 512 MB of RAM.

7.3 Basic Experiments

7.3.1 Experiment Description

The objective of this set of basic experiments is to obtain the key performance measures in terms of accuracy, convergence reliability, and speed. Given a scale of 15 machines and 4 loops, we generate 50 random network structures. For each network structure, we generate 50 cases. Therefore, we have 1,500 random cases in total.

7.3.2 Performance Measures

Accuracy

The average percent errors of production rate $Error_P$ of 1500 cases are plotted in Figure 7-2. In Figure 7-3, the distribution of $Error_P$ shows a mean value of 0.14% and a standard deviation of 1.17%. The distribution is not biased and therefore the measurement in (5.46) is a good estimation of production rate.

The absolute percent errors of production rate $Error_{|P|}$ of 1500 cases are shown in Figure 7-4. The distribution of $Error_{|P|}$ is illustrated in Figure 7-5. The mean and standard deviation of $Error_{|P|}$ is 0.89% and 0.77%, respectively.

To examine the accuracy of buffer levels, the average percent errors $Error_{BL}$ and the average absolute percent errors $Error_{|BL|}$ of buffer levels are plotted in Figure 7-6 and Figure 7-7. The mean and standard deviation of $Error_{BL}$ are respectively -0.04% and 0.89%. The mean and standard deviation of $Error_{|BL|}$ are respectively 7.93% and 2.93%. The mean value of $Error_{|BL|}$ is much larger than that of $Error_{|P|}$ but it is typical for analytical results according to Dallery and Gershwin (1992) and Burman (1995).

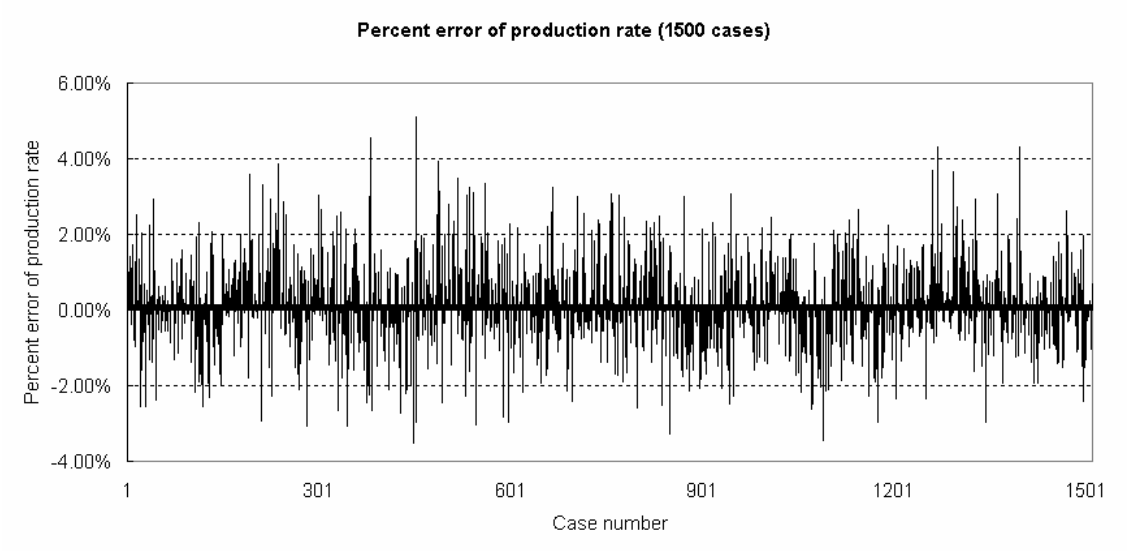


Figure 7-2: The percent errors of production rate of 1500 cases

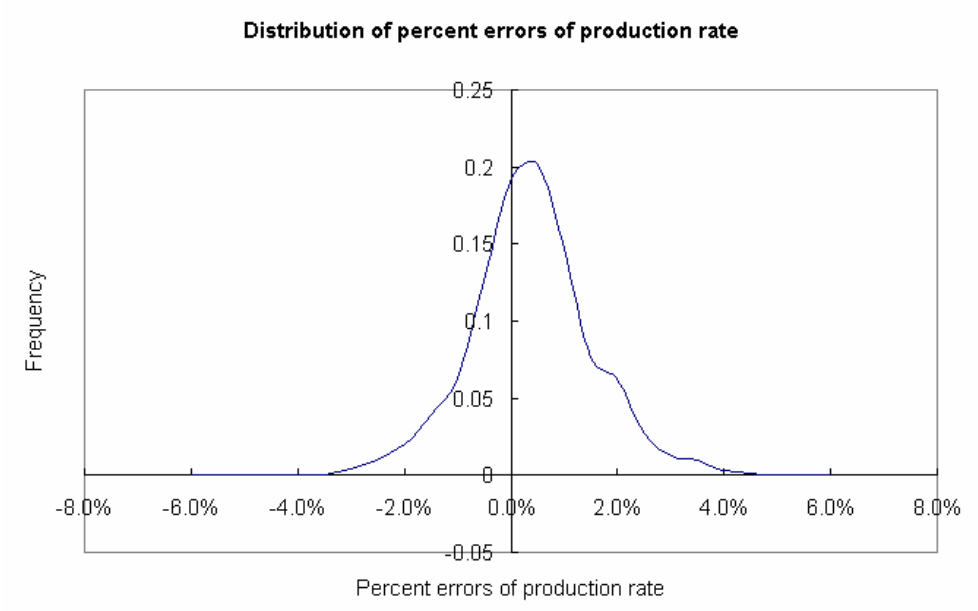


Figure 7-3: The distribution of the percent errors of production rate

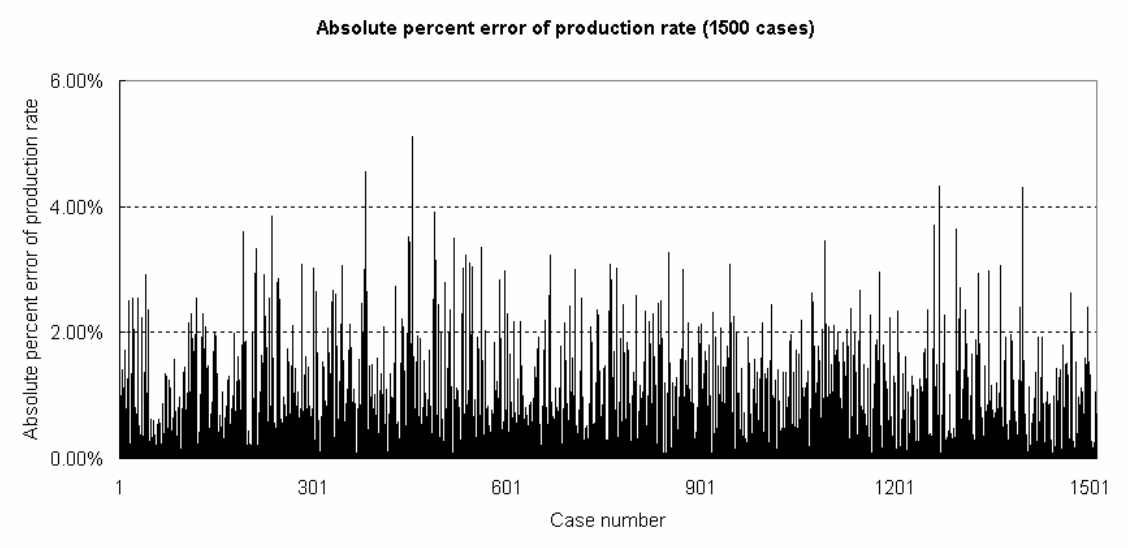


Figure 7-4: The absolute percent errors of production rate of 1500 cases

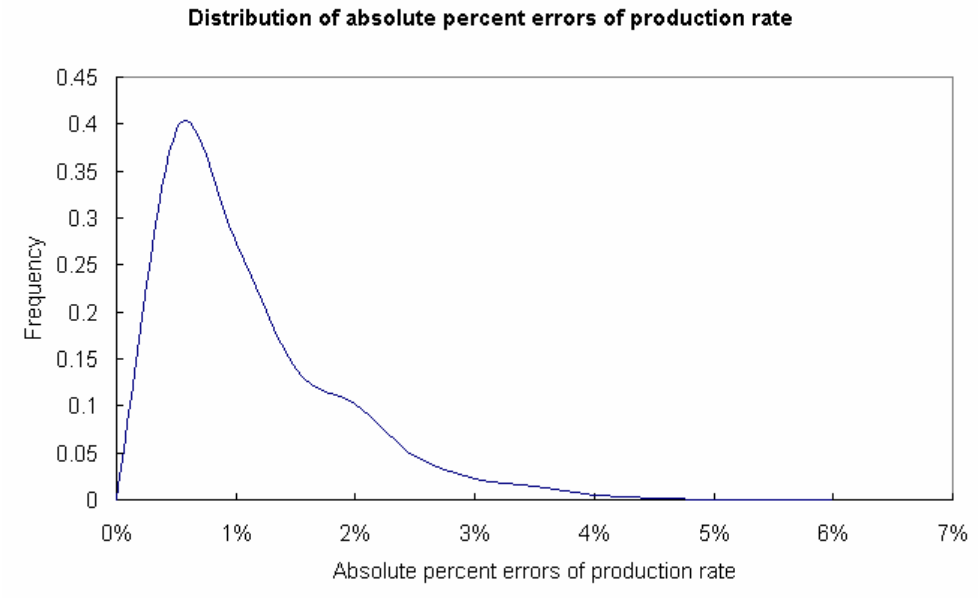


Figure 7-5: The distribution of the absolute percent errors of production rate

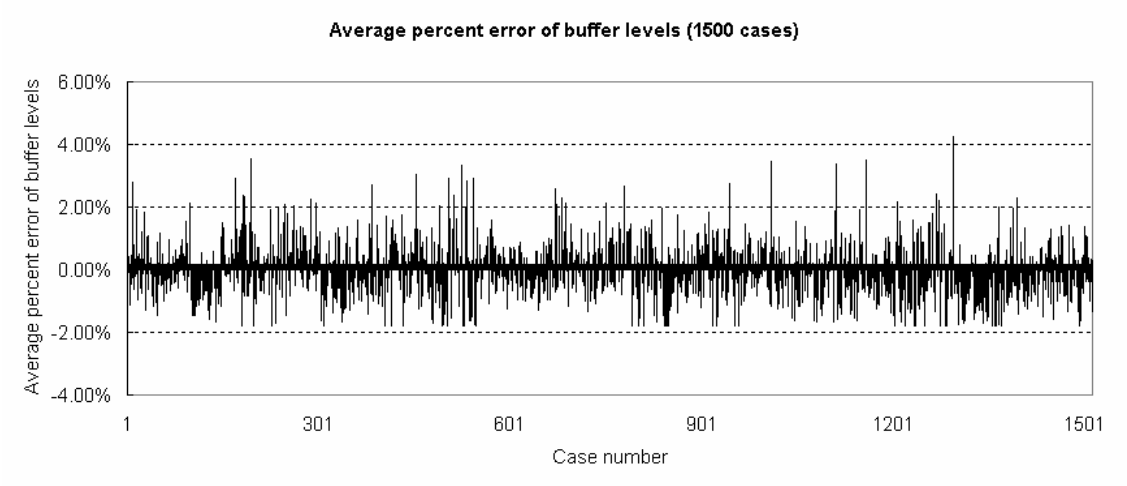


Figure 7-6: The average percent errors of buffer levels of 1500 cases

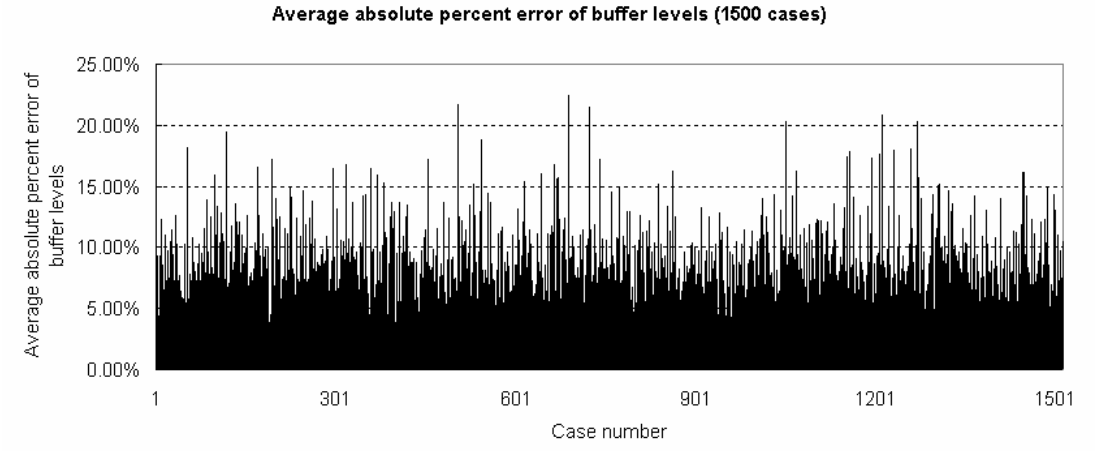


Figure 7-7: The average absolute percent errors of buffer levels of 1500 cases

In Figure 7-8, the percent errors of loop invariants $Error_I$ have a mean of -0.02% and a standard deviation of 0.68%. The average absolute errors of loop invariants $Error_{|I|}$ are plotted in Figure 7-9, with a mean of 1.81% and a standard deviation of 1.23%.

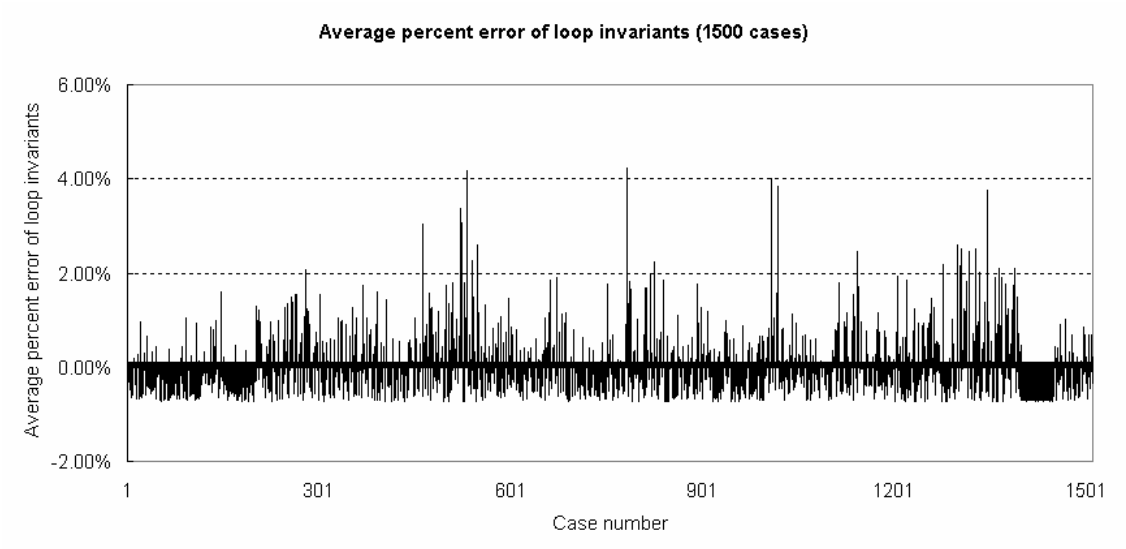


Figure 7-8: The average percent errors of loop invariants of 1500 cases

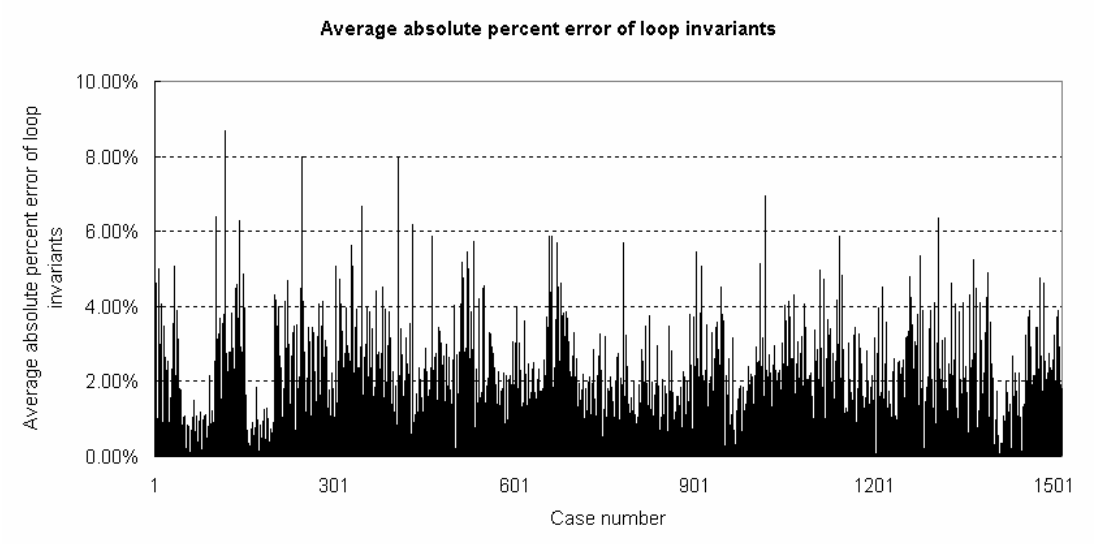


Figure 7-9: The average absolute percent errors of loop invariants of 1500 cases

Convergence Reliability

We define the indicator of convergence reliability as the ratio of the number of cases which converge successfully to the total number of cases. Among 1,500 cases, only 2 cases fail to reach convergence. The convergence reliability is therefore 99.87%.

Recall the definition of convergence error ξ in Section 6.3.3. The mean value of the convergence errors of 1500 cases is 2.24%, which implies there exists a gap between P_{max} and P_{min}. However, the estimate of the production rate is good as the mean of $Error_{|P|}$ is only 0.89%.

Speed

To evaluate 1500 cases using ADN algorithm, the average value of t_I is 0.06 second. The average value of t_{II} is 8.75 seconds per case, while the simulation takes approximately 12 minutes to finish one case. t_{II} is proportional to the number of building blocks and the number of iterations. The number of iterations depends on the specified percent tolerance of termination condition in Section 5.5.3. The percent tolerance ε of the experiments in this thesis is set to 0.1%.

To summarize, the ADN algorithm provides satisfactory accuracy of production rate and average buffer levels. The algorithm is very reliable and has significant advantage in computational efficiency compared to simulation.

7.4 Advanced Experiments

7.4.1 Experiment Description

With the purpose of understanding the production rate error and computation time as functions of network scale and topological complexity, we experiment with a set of networks with different scales. The number of machines varies from 2 to 40. The number of loops varies from 0 to 4. For each specified network scale, we generate 5 random network structures and five random cases per network structure. Therefore,

we have approximately 5,000 random cases in total.

7.4.2 Production Rate Error

We plot the absolute percent errors of production rate $Error_{|P|}$ in Figure 7-10. Each data point represents the average value of 25 cases with the same scale. The data points are linked according to the number of loops such that five curves are formed in the figure. We observe that the curves first increase slowly and then fluctuate around a mean value. The production rate error $Error_{|P|}$ is not very sensitive to the number of machines.

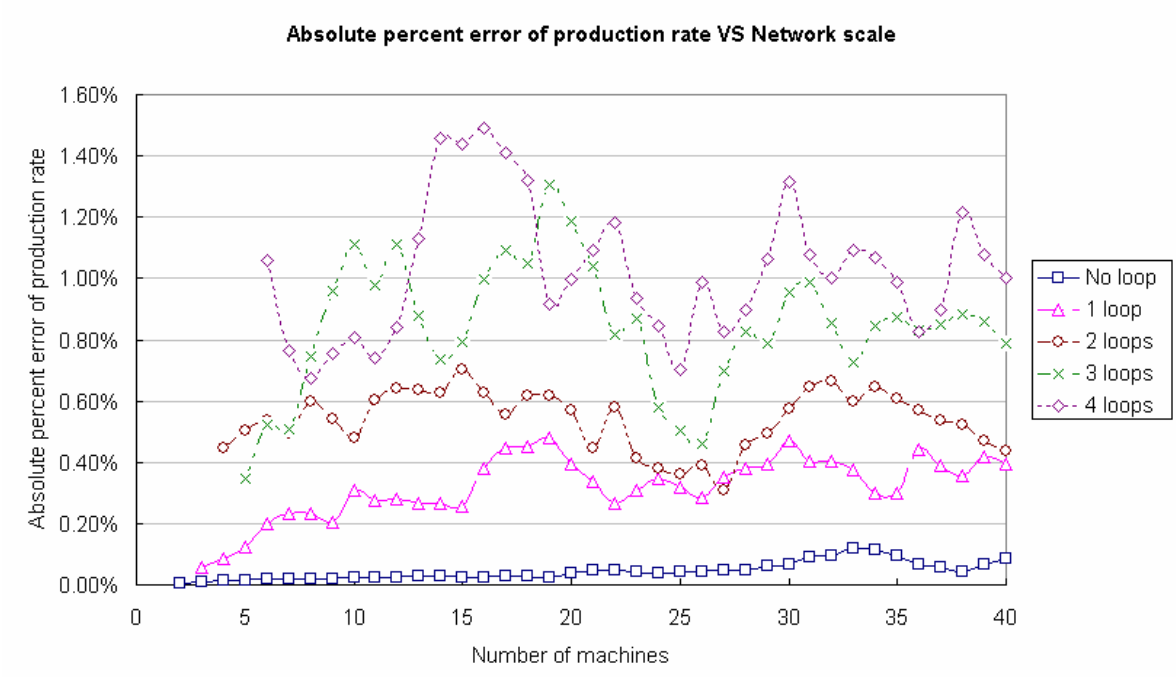


Figure 7-10: The absolute percent errors of production rate VS Network scale

In Figure 7-11, we plot the statistics (mean, standard deviation, max, and min) of the cases of each curve in Figure 7-10. We observe that, when the number of loops increases, the average value of production rate errors increases gradually.

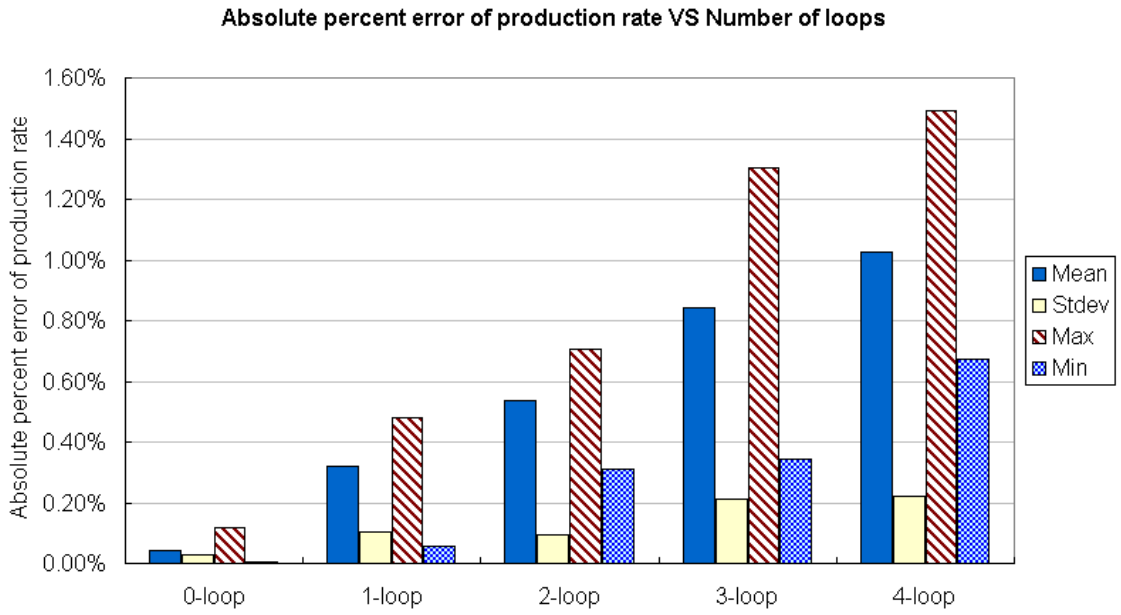


Figure 7-11: The absolute percent errors of production rate VS Number of loops

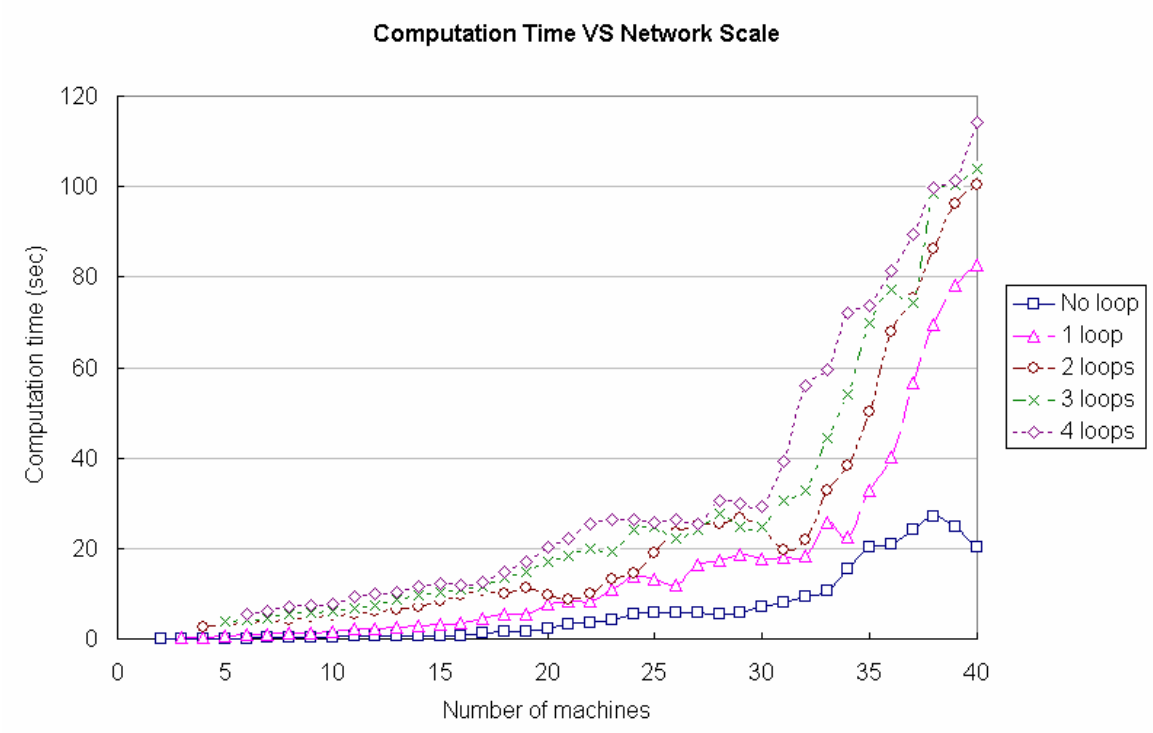


Figure 7-12: The computation time VS Network scale

7.4.3 Computation Time

In Figure 7-12, we plot the computation time t_{ADN} which is the sum of t_I and t_{II}. When there is no loop in networks, the computation time increases slowly with the number of machines in the system. When networks contain closed loops, the computation time increases rapidly with the number of machines. One possible reason is that closed loops could cause thresholds in buffers such that the number of building blocks after transformation increases rapidly.

Table 7.1 shows the computational efficiency of ADN algorithm compared with simulation. For 25 cases with the same scale, the average values of computation time of ADN algorithm and simulation are recorded. The efficiency ratio is given by

$$\text{Efficiency ratio} = t_{simulation}/t_{ADN} \tag{7.9}$$

The efficiency ratio decreases slowly with the scale of networks.

Scale (# of machines, # of loops)	(4, 2)	(20, 3)	(40, 4)
ADN computation time (sec)	2.58	16.93	114.17
Simulation time (sec)	216	858	4,784
Efficiency ratio	83.56	50.69	41.70

Table 7.1: Comparison of computation time of ADN algorithm and simulation

Chapter 8

Behavior Analysis and Design Insights on Closed Loop Control

In the previous chapters, we developed an efficient algorithm for evaluating assembly/disassembly systems with arbitrary topologies. With this algorithm, we are able to efficiently evaluate a large number of variations of systems with different structures and parameters in order to investigate the behavior of multiple-loop structures which will help design multiple-kanban-controlled systems.

In this chapter, we focus on the behavior of production control using multiple-loop structures. Specifically, we study the control structures and invariants of multiple closed loops. Preliminary design insights are presented as well.

8.1 Introduction

Consider the 5-machine CONWIP-controlled production line in Figure 8-1. Machines are identical with processing rate $\mu = 1.0$, failure rate $p = 0.01$, and repair rate $r = 0.1$. All material buffer sizes are 50. The size of the kanban buffer is 80. Kanbans are costless in this example. All the systems in this chapter are treated with the continuous processing time model.

Initially, the target production rate is set at 0.8 units per time unit. When the number of kanbans is 29, the production rate is 0.8011 and the total inventory is 23.2,

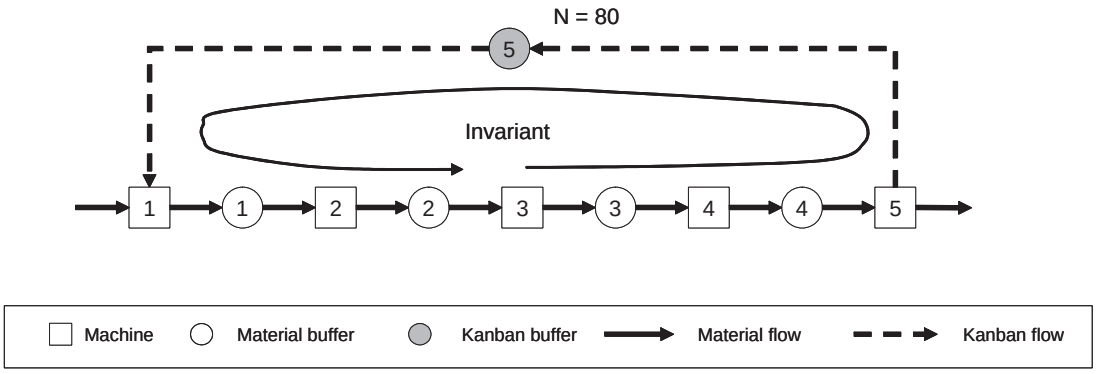


Figure 8-1: A five-machine CONWIP-controlled production line

the summation of the levels of buffers B_1, B_2, B_3, and B_4 (NOT B_5).

Assume the target production rate decreases from 0.8 to 0.7 due to a demand drop. The objective of design is to minimize the total inventory level while satisfying the target production rate at 0.7.

In order to reduce the production rate, we examine three options:

- Option 1: reduce processing rate μ_1 of the first machine M_1 while keeping the invariant of the CONWIP loop, i.e. $I = 29$.
- Option 2: reduce processing rate μ_5 of the last machine M_5 while keeping the invariant of the CONWIP loop, i.e. $I = 29$.
- Option 3: vary loop invariant I of the CONWIP loop without changing the processing rate of any machine.

When three options all meet the target production rate at 7.0, the parameters, production rate, and total inventory level are summarized in Table 8.1.

Three curves are plotted in Figure 8-2. We observe that:

- Option 1 and Option 2 have similar curves. Reducing the processing rate of M_1 or M_5 effectively reduces the production rate. However, the total inventory level stays almost constant as it is controlled by the loop invariant.

Option	μ_1	μ_5	I	Production rate	Total inventory level
1	0.875	1.0	29	0.700781	22.948
2	1.0	0.875	29	0.700781	22.681
3	1.0	1.0	4	0.704403	3.2

Table 8.1: Comparison of three control options (target production rate = 0.7)

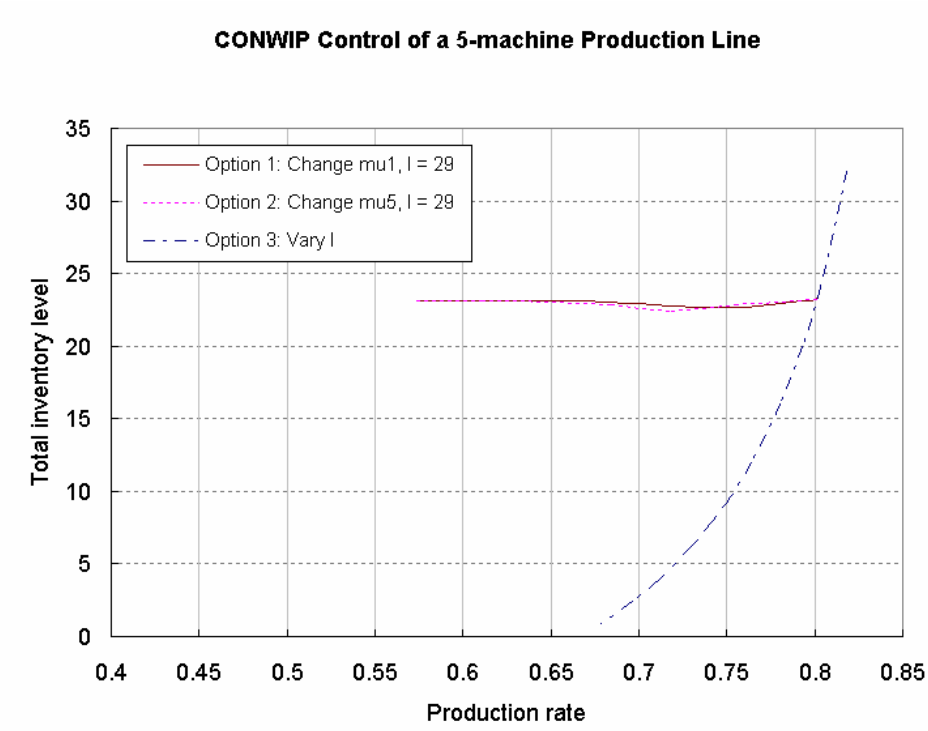


Figure 8-2: Comparison of three control options

- In Option 3, reducing the loop invariant not only effectively reduces the production rate, but also reduces the total inventory level. However, the production rate reaches the lower limit at 0.6775 when the loop invariant is 1. Therefore, if the target production rate is smaller than the lower limit, e.g. 0.65, we must also adjust other parameters, e.g. reducing the processing rate of M_1.

In summary, to meet a lower target production rate, the best option is to decrease the loop invariant without changing machine parameters. Therefore, in this chapter, we restrict the analyses to the following assumptions:

- Machine parameters (processing, failure, and repair rates) and buffer sizes are not control variables. In other words, they are fixed.
- The control variables are loop structures and invariants.

In Section 8.2, we present a behavior analysis of production line control. The control of assembly systems is discussed in Section 8.3. A summary of design insights and practical guidelines are presented in Section 8.3.

8.2 Control of Production Lines

8.2.1 Fundamental Features

With the purpose of understanding the fundamental features of single-loop control, we examine the behavior of production rate and total inventory level by varying the loop invariant.

We still study the CONWIP-controlled production line in Figure 8-1. Notice that the summation of all the buffer space (including the kanban buffer) is

$$N_1 + N_2 + N_3 + N_4 + N_5 = 50 + 50 + 50 + 50 + 80 = 280$$

Therefore, the number of kanbans should be less than 280 and greater than 0. We vary the loop invariant from 1 to 279 and plot the production rate and total inventory level in Figure 8-3.

The graph of production rate is exactly symmetric and the total inventory level graph is rotationally symmetric. In the graph of production rate, there are several bumps on the curve. Some of them (the big ones) are caused by the behavior of the loop, while some of them (the small ones) are caused by the error of the evaluation algorithm. There is a flat area in the middle of the curve. In the total inventory level graph, bumps and a flat area can also be observed. To fully understand the reasons for the shape of the curve, further analyses are required. However, in this thesis, we describe the behavior of loops.

In the previous case, the machines are identical. Suppose we change the parameters of M_2 and M_5:

$$\mu_2 = 1.0, p_2 = 0.015, r_2 = 0.1;$$
$$\mu_5 = 1.0, p_5 = 0.012, r_5 = 0.1.$$

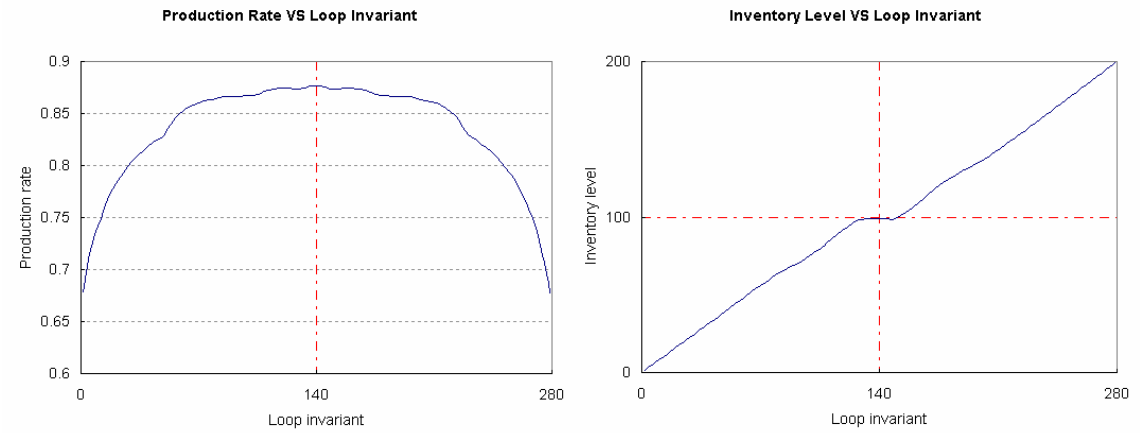


Figure 8-3: Production rate and total inventory level vs. Loop invariant (identical machines)

We plot the production rate and total inventory level again in Figure 8-4. The graphs of production rate and total inventory now have only approximate symmetry.

In the graphs of production rate and total inventory level (Figure 8-3 and Figure 8-

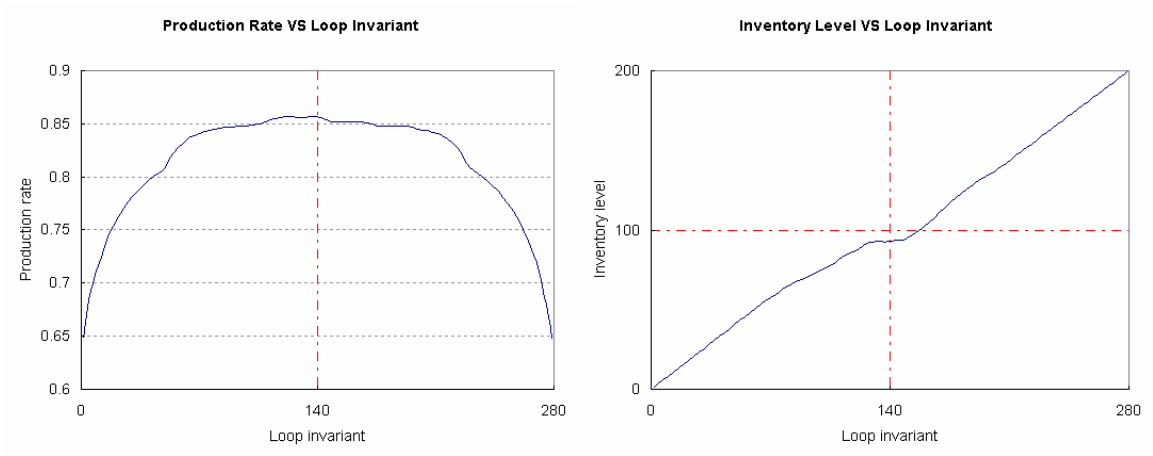


Figure 8-4: Production rate and total inventory level vs. Loop invariant (nonidentical machines)

4), inflection points and flat areas in the middle can be identified. These features lead to an interesting shape of the 'Production rate – Total inventory level' curve in Figure 8-5.

The curve in Figure 8-5 is called the *control curve*. When we adjust the loop invariant, we can identify a unique point on the curve with corresponding production rate and total inventory level. For the previous case with identical machines, a control curve can also be plotted.

The control curve in Figure 8-5 has rough symmetry. For a given production rate, two points could be identified on the curve. For example, suppose the target production rate is 0.8,

- Point 1: $I = 43$, production rate = 0.80071, total inventory level = 35.71.
- Point 2: $I = 237$, production rate = 0.80072, total inventory level = 165.43.

To achieve the same production rate, the point on the upper part of the curve has higher total inventory level than the point on the lower part of the curve. Therefore, the points on the upper part of the curve are inefficient solutions for a given production rate. We should always choose points on the lower part of the curve. The lower part of the curve is called the *efficient frontier*.

Figure 8-5: Control curve: production rate vs. total inventory level (nonidentical machines)

On the control curve, several inflection points can be identified. In Figure 8-5, there are two big bumps when the production rate is near 0.81. When the production rate is near the maximum at 0.85, a strange shape can be observed. This is because the control curve is the combination of two curves in Figure 8-4, which both have bumps and flat areas. These inflection points and flat areas are important features for choosing optimal control structures. Details are discussed in Section 8.2.2.

In summary, a control curve of a single-loop control is a roughly symmetric curve with infection points. The lower part of the curve is the efficient frontier of the control curve. In the remainder of this thesis, we ignore the upper part, so only the efficient frontier of a control curve is referred to as a control curve.

8.2.2 Single-Loop Control

Control Structures

Consider the 5-machine production line in Figure 8-6(a). Machines are identical with processing rate $\mu = 1.0$, failure rate $p = 0.01$, and repair rate $r = 0.1$. All material buffer sizes are 50. Five variations of single-loop control structures are illustrated in Figure 8-6(b). The size of kanban buffer B_5 is 80. Each control structure is labelled by the name of an added chord. For example, Structure #3 (loop: $M_3 \rightarrow M_1$) means a chord from M_3 to M_1 is added to the production line to form a closed loop. In the remainder of this chapter, we use the same labelling method.

The performance measures include production rate P and total inventory level T_{inv} . Total inventory level is defined as

$$T_{inv} = \overline{b}(1) + \overline{b}(2) + \overline{b}(3) + \overline{b}(4) \tag{8.1}$$

in which $\overline{b}(j)$ is the average buffer level of B_j.

Inventory Comparison

We vary the loop invariant of each variation and plot the control curves in Figure 8-7. Given the target production at 0.825, the total inventory levels and loop invariants

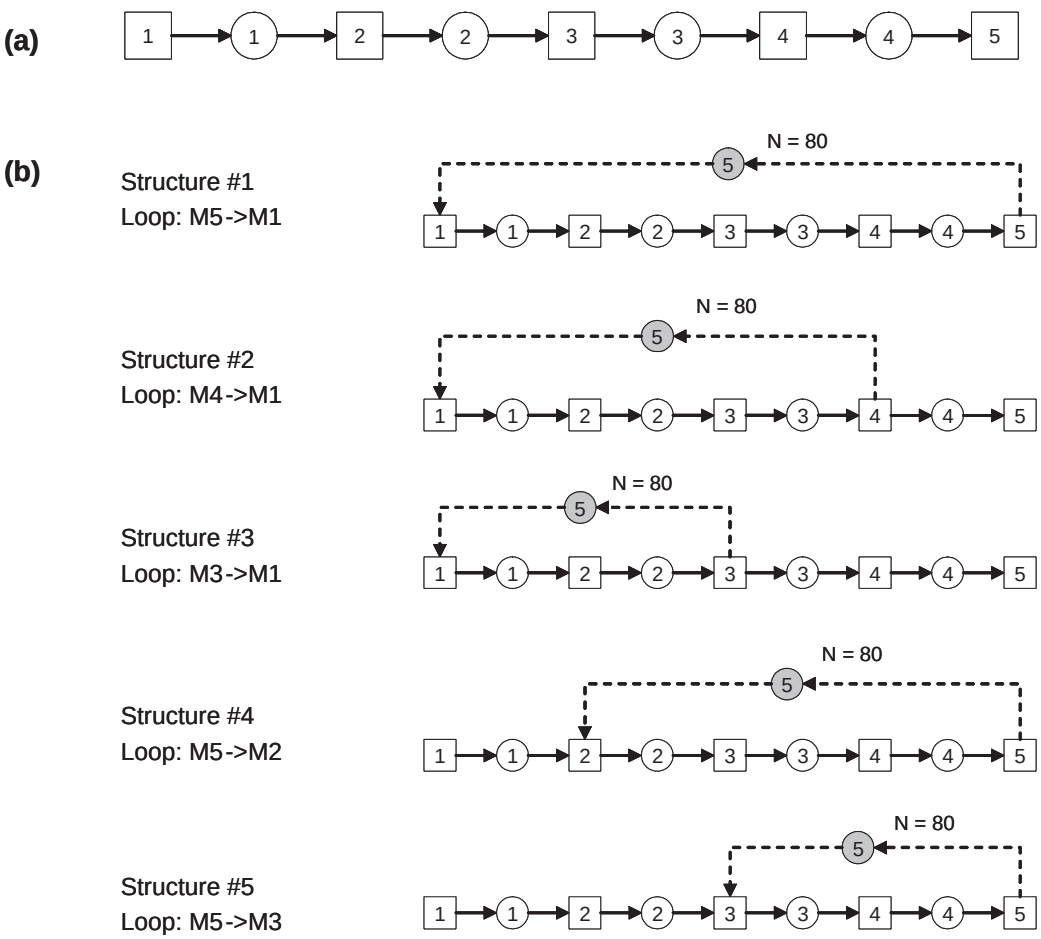


Figure 8-6: A 5-machine production line and five variations of control structures

of five variations are summarized in Table 8.2.

Structure	I	Production rate P	Total inventory level T_{inv}
1	46	0.8253	43.57
2	32	0.8255	38.98
3	16	0.8250	35.35
4	32	0.8254	65.53
5	16	0.8250	87.32

Table 8.2: Comparison of five control structures (target production rate = 0.825)

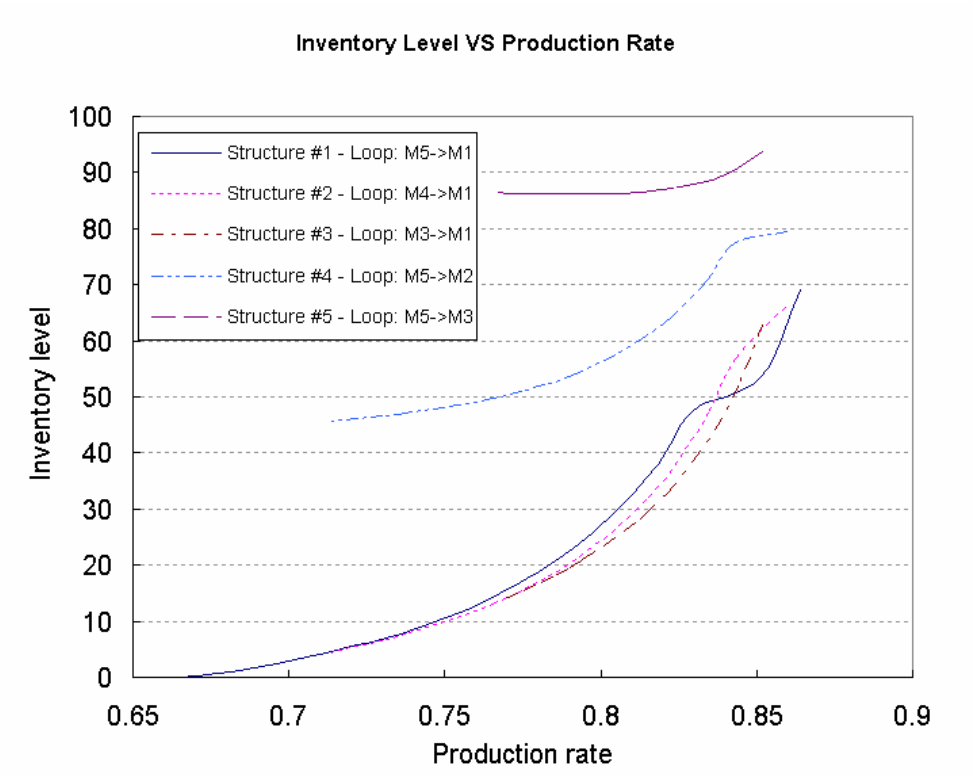


Figure 8-7: Comparison of the total inventory levels of five control structures

Comparing the total inventory levels of five structures in Figure 8-7 and Table 8.2, we observe that:

- When the target production rate is 0.825, the best control structure is #3 (loop: $M_3 \rightarrow M_1$), while the worst is #5 (loop: $M_5 \rightarrow M_3$). The difference of total inventory levels between the worst and the best structures is $(87.32 - 35.35) =$

51.97, approximately 1.5 times of the best total inventory level at 35.35. The difference between the best and second best structures (#2) is $(38.98 - 35.35) = 3.63$, around 10.3% of the best total inventory level.

- Control structure #3 reaches the lower limit at 0.7733 when the invariant is 1. Therefore, in order to achieve a target production rate lower than 0.7733, we should use other structures. For example, when the target production rate is at 0.7, Structure #1 (loop: $M_5 \rightarrow M_1$) is the optimal choice.

- When the target production rate is higher than 0.844, Structure #3 is no longer the optimal one as Structure #1 (loop: $M_5 \rightarrow M_1$) has the lowest total inventory level. This is because the infection points on the curve of Structure #1 leads the curve across the control curves of Structure #2 and Structure #3 (Figure 8-8). Different control curves have inflection points at different positions. The optimal structure changes when there is a crossing point on the current optimal control curve.

Although the dominant optimal control structure is Structure #3, due to lower limits and inflection points, there is no absolute optimal single control curve when target production rate varies. We define the *optimal control frontier* which envelops multiple curves of difference structures. The challenge of closed loop control is to determine both control structure and loop invariants, which are coupled design factors.

Profit Optimization

In the previous example, the comparison is based on total inventory levels while satisfying a given production rate. Now we optimize the profit by setting up margin per unit production rate C_P and inventory holding cost C_T for all material buffers. Kanban buffer B_5 has no inventory holding cost. There is no constraint on the target production rate. We define profit Y:

$$Y = C_P P - C_T T_{inv} \tag{8.2}$$

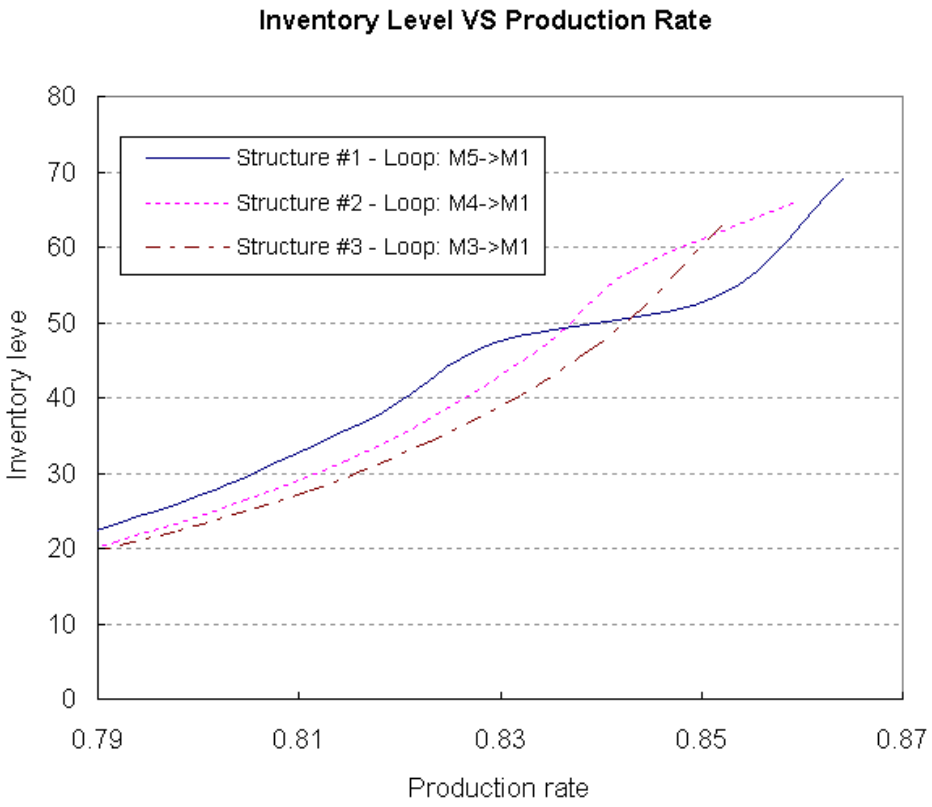


Figure 8-8: Crossing between control curves due to infection points

where P is the production rate; T_{inv} is the total inventory level (8.1).

Consider three scenarios:

- Scenario 1: $C_P = 1000$, $C_T = 1$;
- Scenario 2: $C_P = 1000$, $C_T = 2$;
- Scenario 3: $C_P = 1000$, $C_T = 10$.

We plot the profits of five variations as functions of loop invariant I:

- Figure 8-9 shows the profit curves in Scenario 1. The optimal solution is Structure #1 (loop: $M_5 \rightarrow M_1$), when $I = 62$, $P = 0.8537$, and $T_{inv} = 54.62$. The optimal profit $Y = 799.08$.
- Figure 8-10 shows the profit curves in Scenario 2. The optimal solution is Structure #3 (loop: $M_3 \rightarrow M_1$), when $I = 12$, $P = 0.8162$, and $T_{inv} = 30.23$. The optimal profit $Y = 755.74$.
- Figure 8-11 shows the profit curves in Scenario 3. The optimal solution is Structure #1 (loop: $M_5 \rightarrow M_1$), when $I = 3$, $P = 0.6960$, and $T_{inv} = 2.53$. The optimal profit $Y = 670.70$.

We observe that margin and cost coefficients have significant effects on the optimal profit solution. When the inventory holding cost is low, Structure #1 (loop: $M_5 \rightarrow M_1$) gives the optimum. When the inventory holding cost increases, Structure #3 (loop: $M_3 \rightarrow M_1$) becomes the optimal structure. When the inventory holding cost further increases, Structure #1 (loop: $M_5 \rightarrow M_1$) becomes the optimal solution again.

To investigate the switches between different control structures, we plot objective coefficient vectors in Figure 8-12. For a given objective coefficient vector, an optimal normal line could be identified which is normal to the objective coefficient vector, tangent to one or multiple control curves simultaneously, and has no intersection points with other control curves. The control curve which is tangent to the optimal normal line corresponds to the optimal control structure. When we vary the margin

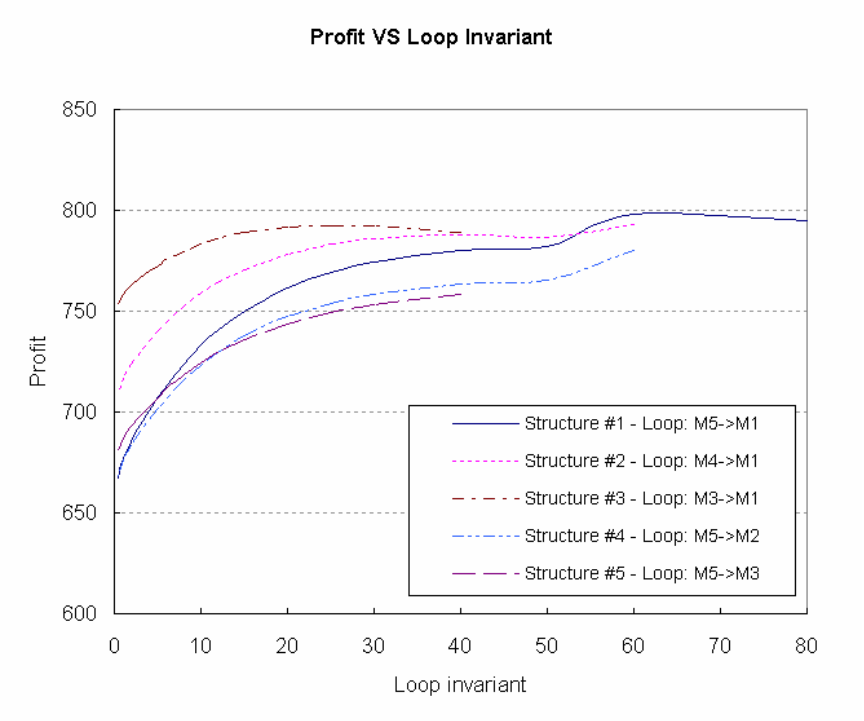


Figure 8-9: Comparison of the profits of five control structures ($C_P = 1000$, $C_T = 1$)

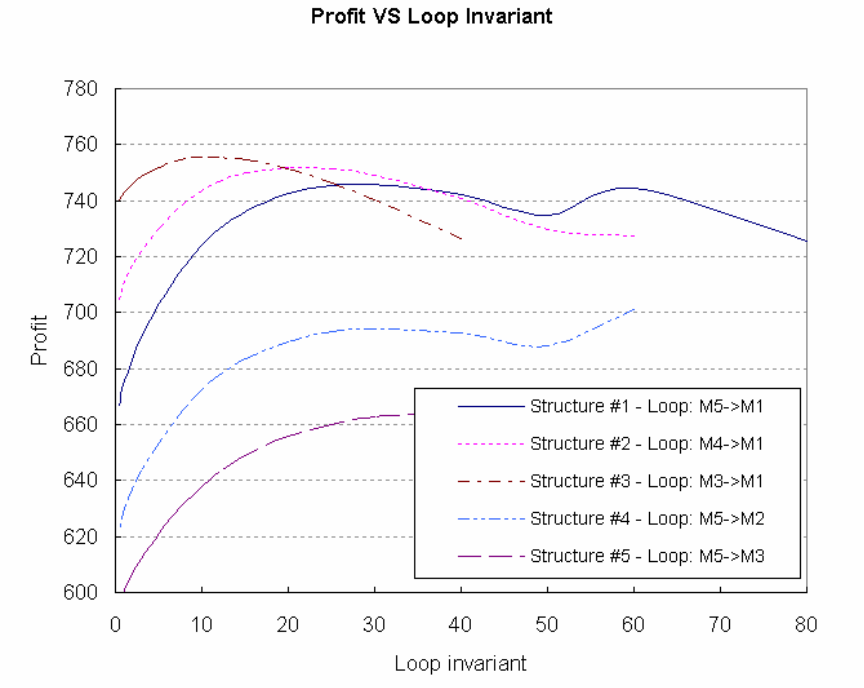


Figure 8-10: Comparison of the profits of five control structures ($C_P = 1000$, $C_T = 2$)

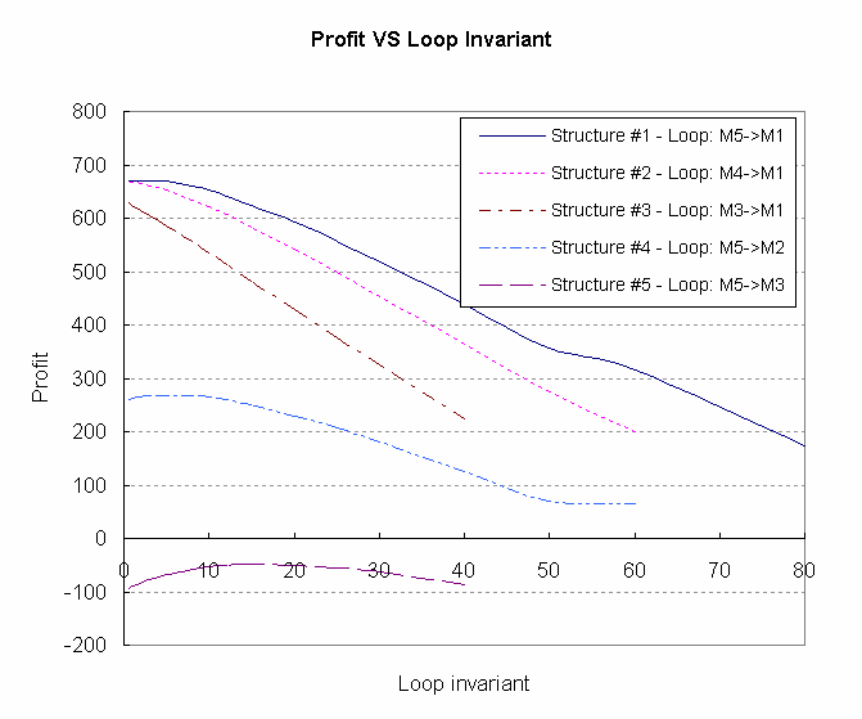


Figure 8-11: Comparison of the profits of five control structures ($C_P = 1000$, $C_T = 10$)

and cost coefficients, i.e. the C_P/C_T ratio, the objective coefficient vector changes direction and its optimal normal line is tangent to different control curves.

- When the C_P/C_T ratio is high, the optimal normal line of objective coefficient vector is tangent to the control curve of Structure #1 (loop: $M_5 \rightarrow M_1$).
- When the C_P/C_T ratio is intermediate, the optimal normal line of objective coefficient vector is tangent to the control curve of Structure #3 (loop: $M_3 \rightarrow M_1$). This is due to the crossing between the control curves of Structure #1 and Structure #3.
- When the C_P/C_T ratio is low, the optimal normal line of objective coefficient vector is tangent to the control curves of Structure #2 or Structure #1. This is because of the lower limit of the control curve of Structure #3.

To summarize, switches between different structures are caused by inflection points and lower limits of different control curves.

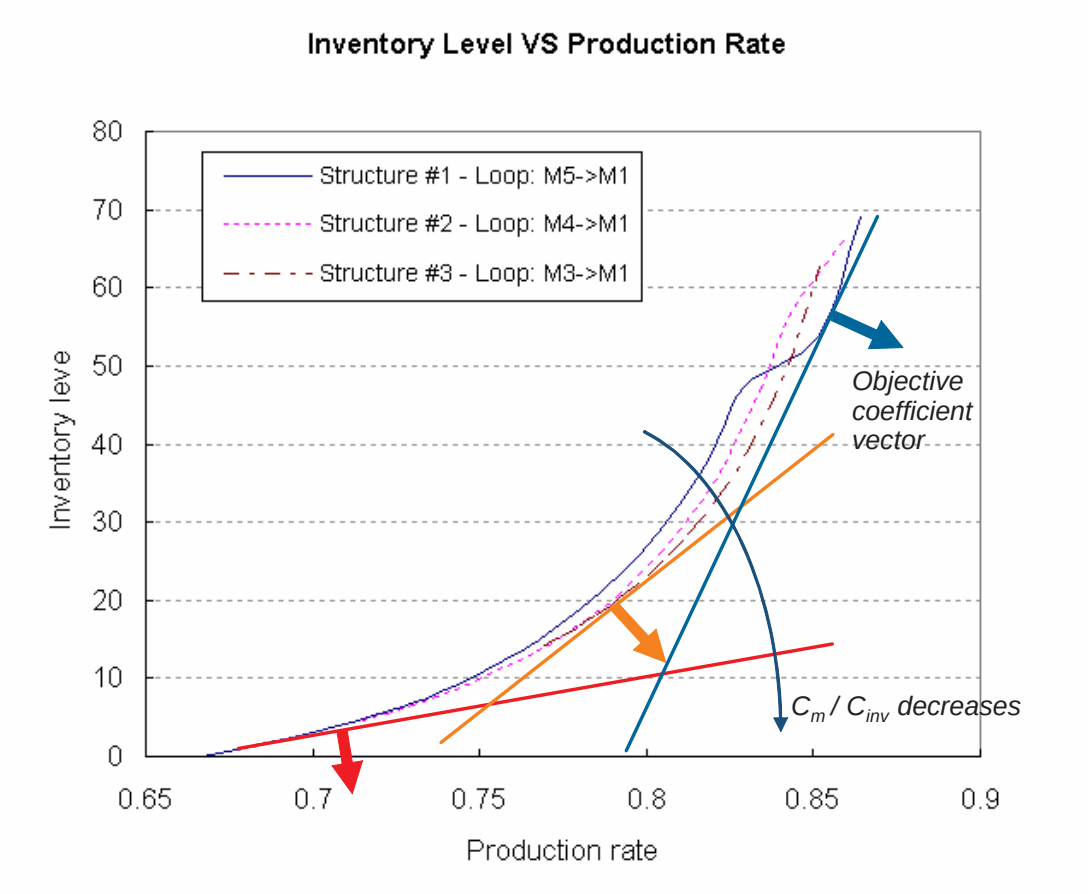


Figure 8-12: Control curves and objective coefficient vectors

Inventory Holding Cost Comparison

In the previous discussion, all material buffers have the same cost coefficients. In this section, we introduce a *cost scheme* which specifies the inventory holding costs of individual buffers. The inventory holding cost of buffer B_j is denoted by C_j.

Consider the CONWIP-controlled production line in Figure 8-1. We study six cost schemes in Table 8.3.

Cost Scheme #	Description	C_1	C_2	C_3	C_4
1	Constant	100	100	100	100
2	Increasing	10	40	70	100
3	Decreasing	100	70	40	10
4	Cost jump at B_4	1	1	1	100
5	Cost jump at B_3	1	1	100	100
6	Cost jump at B_2	1	100	100	100

Table 8.3: Six cost schemes of a production line

Notice that kanban buffer B_5 has no inventory holding cost in any of the schemes. The total inventory holding cost H_{inv} is:

$$H_{inv} = C_1\overline{b}(1) + C_2\overline{b}(2) + C_3\overline{b}(3) + C_4\overline{b}(4) \tag{8.3}$$

We present some examples of these six cost schemes in the real world:

- Cost scheme #1 has constant cost distribution. This could happen when raw material is very expensive and processes are low value-added.

- Cost scheme #2 has increasing cost distribution. This is common when the added value at each process is significant compared to the value of raw material.

- Cost scheme #3 has decreasing cost distribution. This is unusual when we consider the inventory holding cost from value-added perspective. However, when we take account into the cost of keeping parts in specific buffers, we could identify a number of examples in industry. One example is canned food production. Before food is enclosed in a can, it must be refrigerated and kept clean

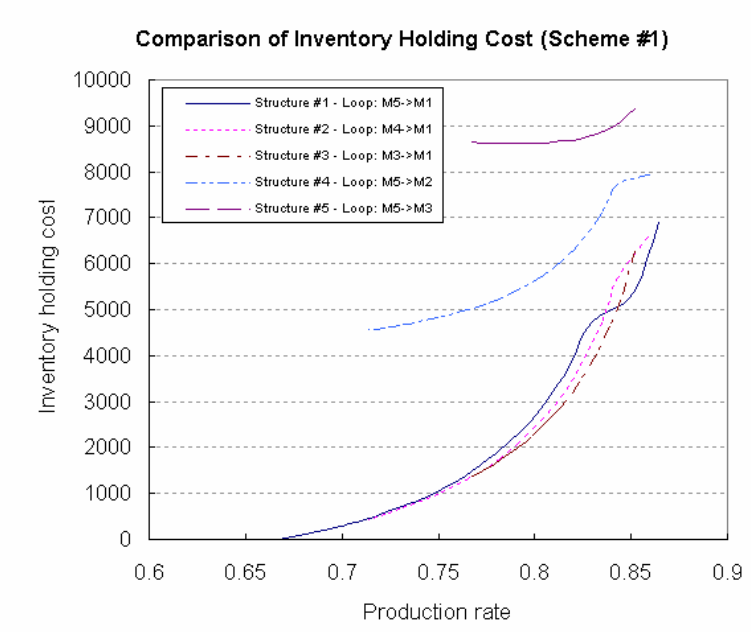


Figure 8-13: Comparison of the inventory holding costs of five control structures (Cost scheme #1)

which incur higher maintenance and operating cost. After it is canned, it can be stored in an ordinary low-cost location. Another example is semiconductor manufacturing. Chips are kept in a high-cost clean room before packaging. After packaging processes, they can be held in low-cost storage space. Similar examples can be found in pharmaceutical, steel, and other consumer goods industries.

- Cost schemes #4, #5, and #6 have cost jumps. This could happen when a very expensive process is performed, or a very expensive component is assembled at a certain process. The downstream buffer of this process presents a cost jump.

The inventory holding costs of five control structures in six cost schemes are plotted in Figure 8-13 to Figure 8-18. In these six graphs, the positions and shapes of the curves of five control structures vary from each other, although they have the same 'Production rate – Total inventory level' graph in Figure 8-7.

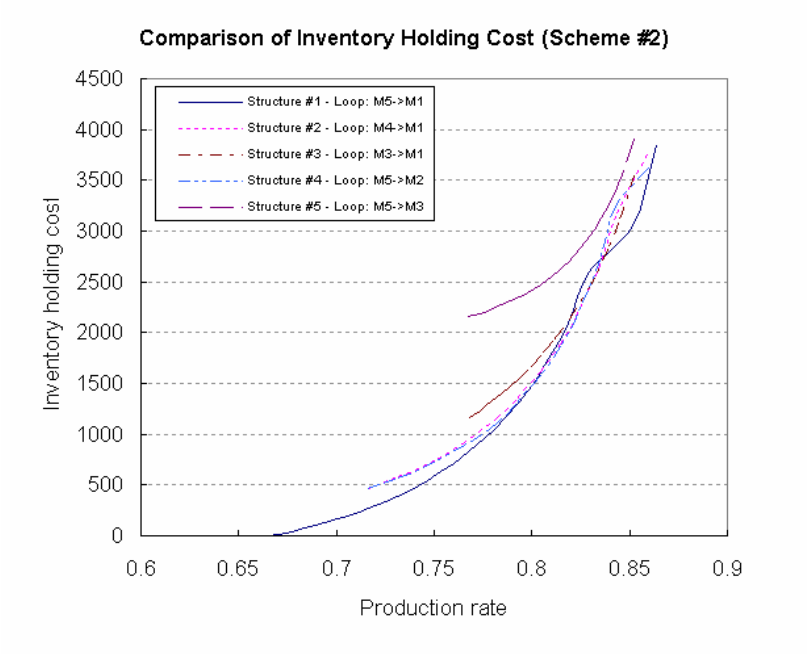


Figure 8-14: Comparison of the inventory holding costs of five control structures (Cost scheme #2)

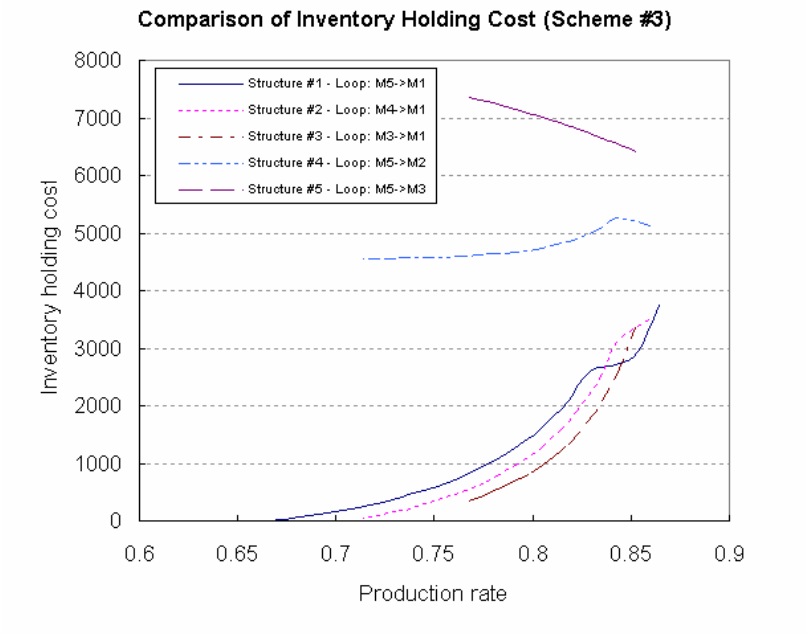


Figure 8-15: Comparison of the inventory holding costs of five control structures (Cost scheme #3)

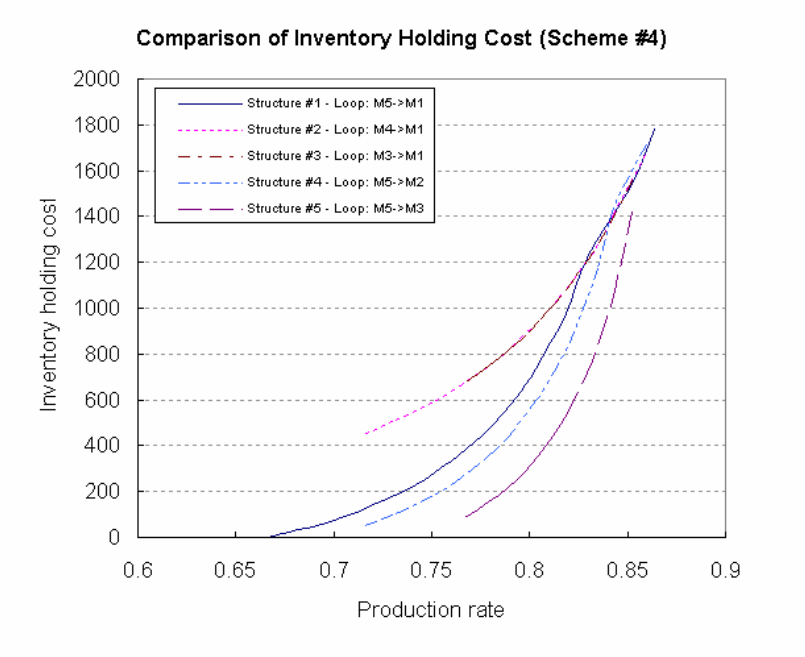


Figure 8-16: Comparison of the inventory holding costs of five control structures (Cost scheme #4)

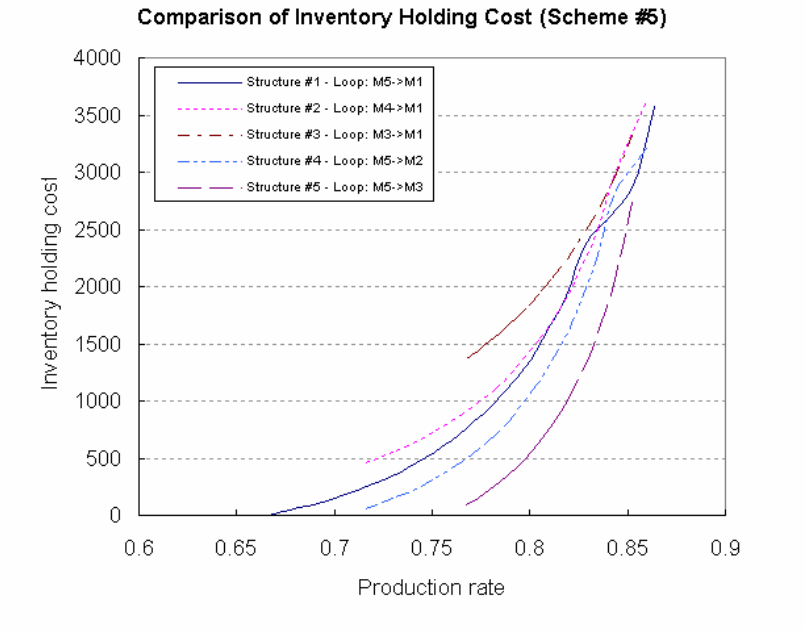


Figure 8-17: Comparison of the inventory holding costs of five control structures (Cost scheme #5)

Consider the comparison of inventory holding costs in cost scheme #5 (Figure 8-17). The best control structure is #5 as it has the least inventory holding cost. However, it is the worst case in terms of total inventory level in the previous example (Figure 8-7). To explain this phenomenon, we plot the distribution of buffer levels of Structures #1, #3, and #5 in Figure 8-19 when the production rate is 0.793 for all three structures. From the graph, we observe that:

- Although Structure #3 (loop: $M_3 \rightarrow M_1$) has tight control over B_1 and B_2, these buffers are costless in cost scheme #4.
- Structure #5 (loop: $M_6 \rightarrow M_3$) focuses on the control of the most expensive buffers B_3 and B_4. This control structure also allows high inventory levels in costless buffers B_1 and B_2 in order to reduce the starvation from upstream low cost buffers.
- Structure #1 (loop: $M_5 \rightarrow M_1$) implements the control over all the material buffers. However, this structure is not able to decouple two critical tasks: re-

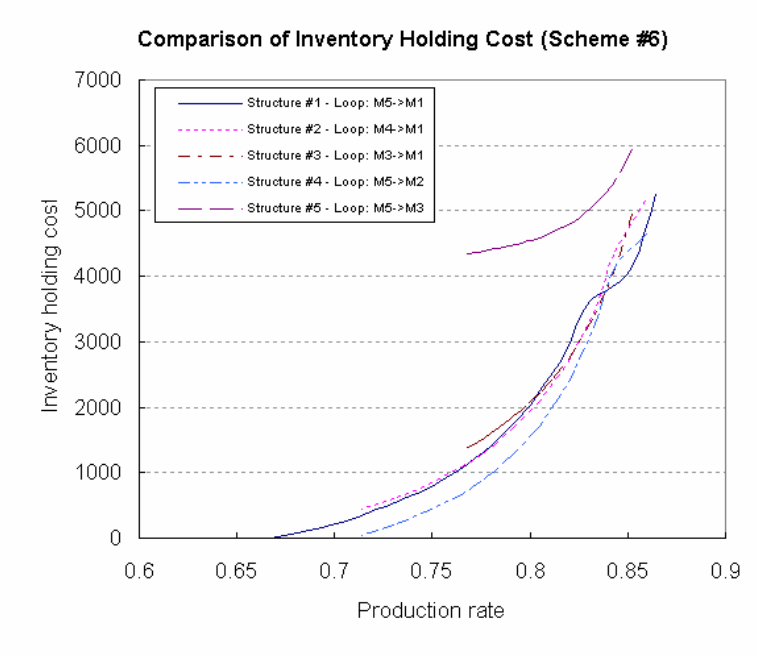


Figure 8-18: Comparison of the inventory holding costs of five control structures (Cost scheme #6)

ducing inventory levels in expensive buffers, and minimizing the starvation from upstream low cost buffers.

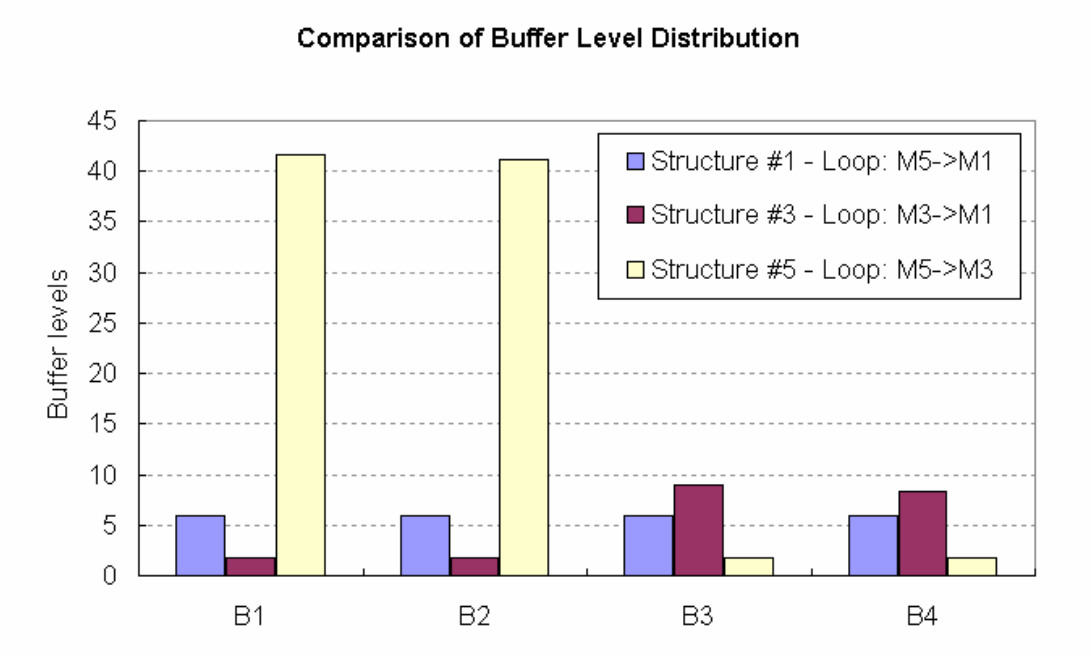


Figure 8-19: Comparison of inventory distributions (production rate = 0.793)

Therefore, when we determine the optimal control structure based on inventory holding cost, a good control structure should have two features:

- implementing tight inventory control over the most expensive buffers in order to cut inventory holding cost.
- relaxing inventory control over low cost buffers in order to reduce the interruption of flows due to blocking or starvation.

Design Insights

The cost scheme is emphasized in behavior analysis because it is a critical variable in factory design. When we analyze manufacturing systems in different industries, their inventory holding cost schemes vary from each other. Even for the same manufacturing system, the changing of part types might change the cost scheme. Significant

changes in equipment or the supply cost structure require reconsideration of the cost scheme.

Based on the analyses of single-loop control of production lines, we develop the following preliminary insights:

- **It is usually better to control the upstream portion of a production line.** When we consider only the total inventory level, Structure #3 is the dominant optimal structure. It has lower inventory levels in upstream buffers but higher inventory levels in downstream buffers than other control structures that provide the same production rate. The total inventory level of Structure #3 is the least one. This is because, to maintain a certain production rate, the negative effect on the production rate due to decreases of inventory levels in upstream buffers could be offset by a smaller total increase in downstream buffer levels. In other words, inventory levels in downstream buffers are usually more useful for maintaining a certain production rate than those in upstream buffers.

- **However, the cost scheme is a factor which might be more important.** To maintain a certain production rate, when we decrease the levels of upstream buffers B_1 and B_2, we must increase the levels of downstream buffers B_3 and B_4. When cost scheme #5 is applied, the additional inventory holding cost incurred by the increase in downstream buffer levels is much higher than the saving of inventory holding cost from the decrease in upstream buffer levels. Therefore, it is not wise to decrease levels in upstream buffers which are low cost buffers. Structure #5 is better than Structure #3. In Structure #5, keeping high inventory levels in low cost buffers could improve the interruption of flow due to blocking or starvation.

Therefore, to design closed loop controls, we should first specify the cost scheme. Cost jumps and the segment of expensive buffers must be identified. A closed loop

should be implemented to control over the flow into and through the expensive buffers if the optimal structure is determined based on total inventory holding cost.

8.2.3 Double-Loop Control

Control Structure

We control a 5-machine production line using two closed loops (Figure 8-20). The production line machine parameters and buffer sizes are the same as those of the production line in Figure 8-6. The sizes of kanban buffers of two closed loop are both 80.

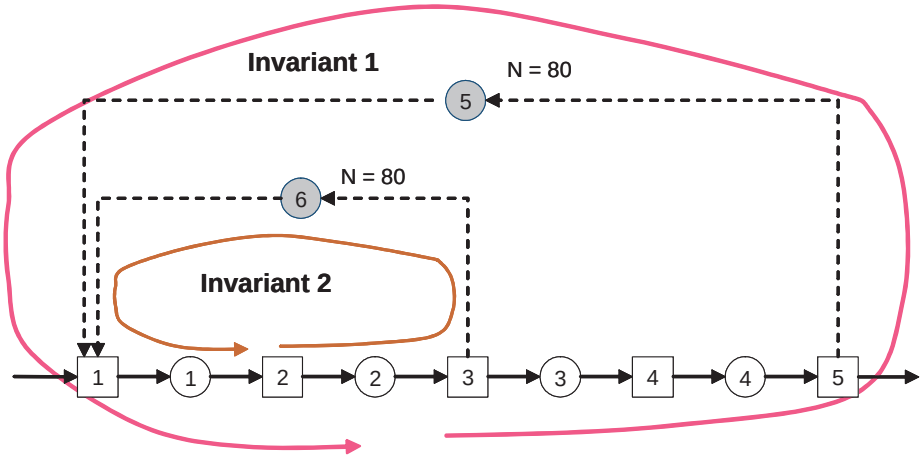


Figure 8-20: A double-loop control of a 5-machine production line

To design a double-loop control structure, we should determine both the structure of two closed loops and loop invariants I_1 and I_2.

Behavior Analysis

Currently, we assume the control structure is given in Figure 8-20. Then the control curve of double-loop control depends on the selection of loop invariants. In Figure 8-21 and Figure 8-22, we plot the production rate and total inventory level on the 'I_1 – I_2' plane.

In the production rate plot, a set of iso-curves can be drawn. Any point on an production rate iso-curve corresponds to a combination of (I_1, I_2) that determines a

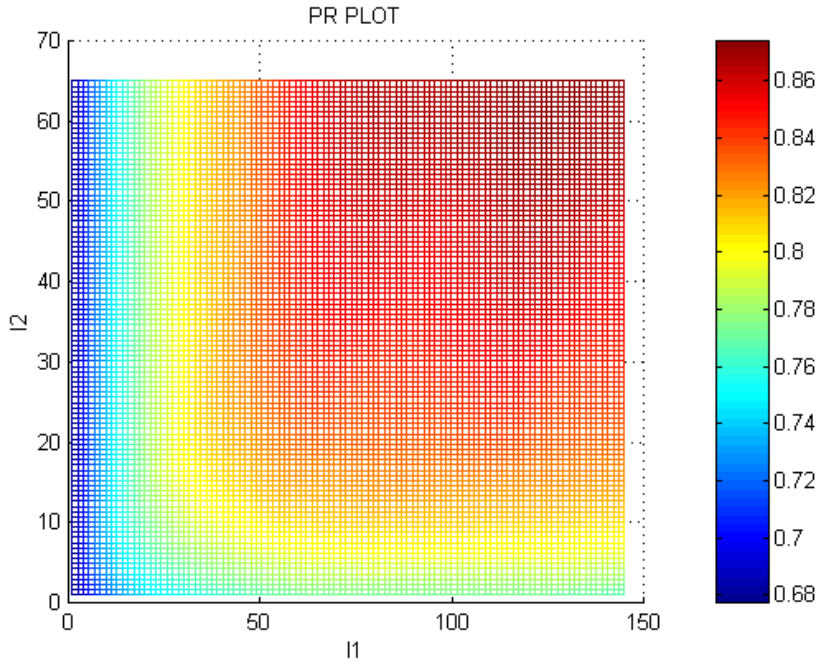


Figure 8-21: Production rate plot of a double-loop control structure

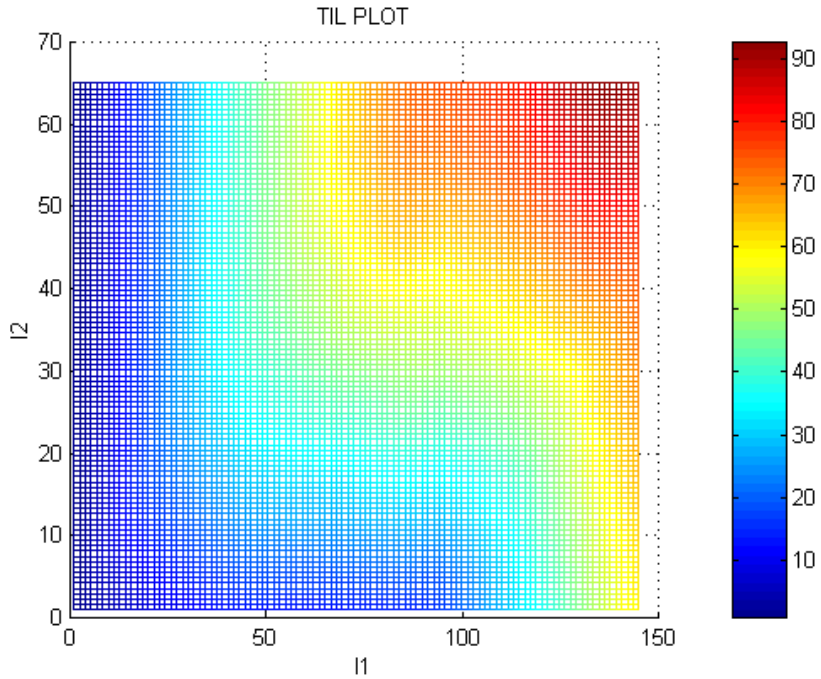


Figure 8-22: Total inventory level plot of a double-loop control structure

specific production rate. It is worthwhile to mention that the production rate plot looks similar to the production plot for the buffer space allocation of a three-machine line (Schor May 1995, Gershwin and Schor 2000). The graph in this thesis is plotted on the 'I_1 – I_2' plane, while that graph is plotted on the 'N_1 – N_2' plane.

In the total inventory level plot, these combinations have different values. The one has the lowest value is the solution of optimizing total inventory level. In Figure 8-23, we illustrate an iso-curve mapping method to determine the loop invariants which optimize total inventory level.

- For a given target production rate, identify the corresponding iso-curve in the production rate plot.
- Map this iso-curve to the total inventory level plot.
- On the mapped curve in the total inventory level plot, identify a point which has the lowest value. If there exist multiple points, select the one which has the least gradient.

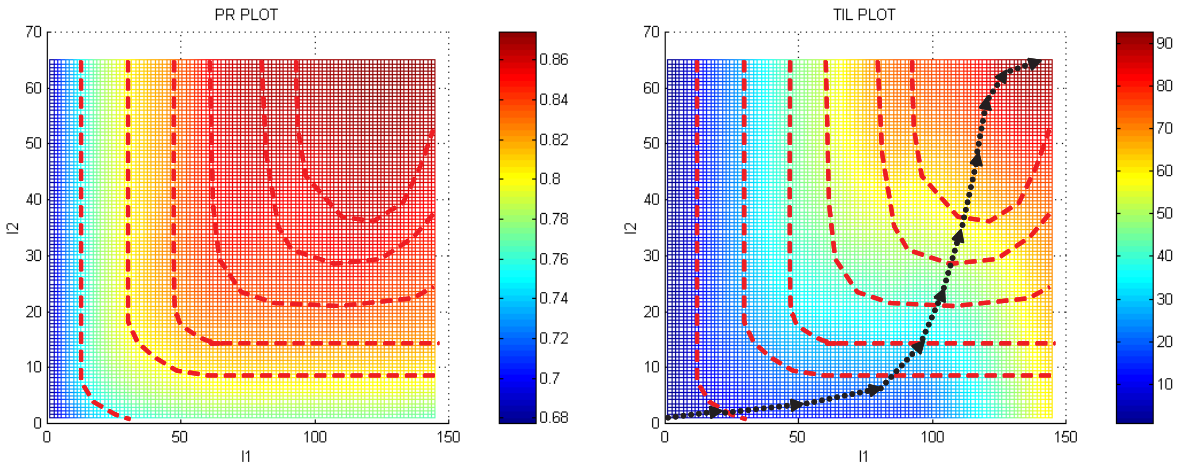


Figure 8-23: Determine the optimal loop invariants of a double-loop control

In Figure 8-23, a set of production rate iso-curves are drawn on the production rate plot. These iso-curves are mapped to the total inventory level plot. On each mapped curve, a point of the optimal combination of (I_1, I_2) is illustrated. These

points are linked into a dot-arrowed-curve in the total inventory level plot. This curve is defined as *optimal invariant curve*.

In order to compare the effects of double-loop control and single-loop control, the control curve of the double-loop control structure in Figure 8-20 is plotted in Figure 8-24 together with those of Structures #1 and #3 in Figure 8-7. The double-loop control structure is a combination of Structures #1 and #3.

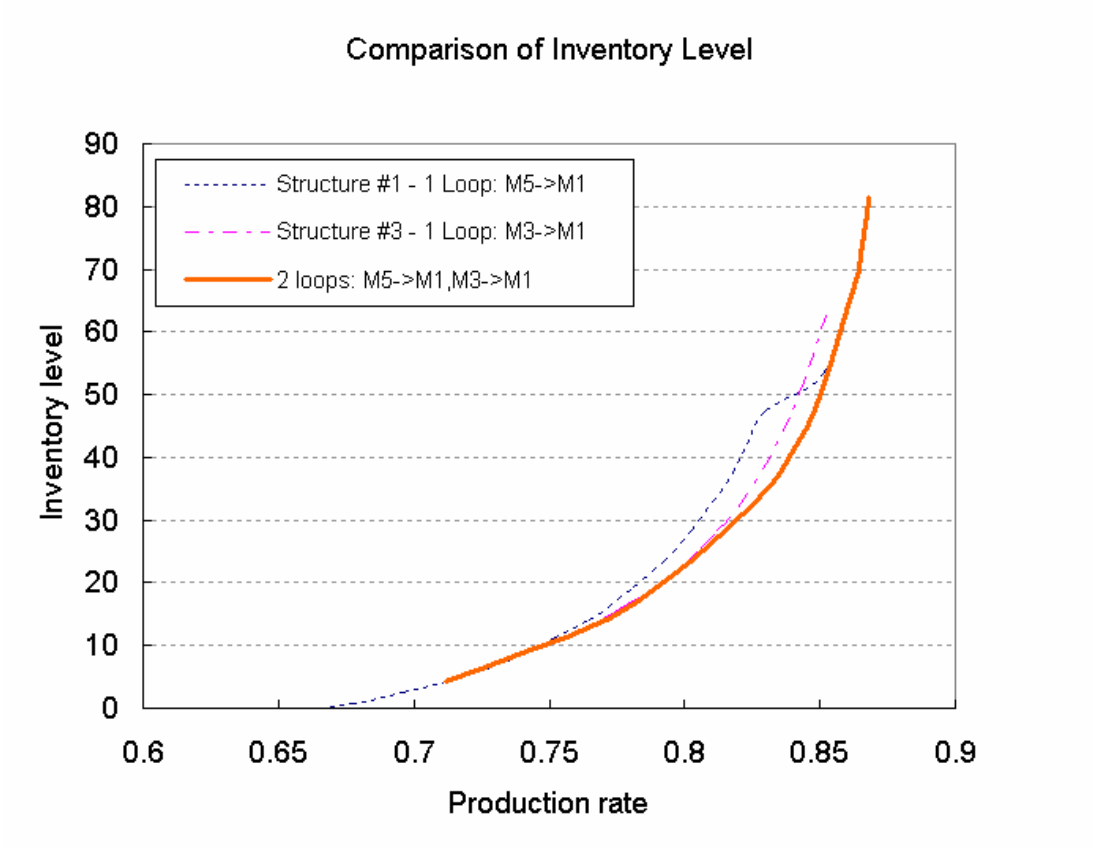


Figure 8-24: Comparison of the total inventory levels of double-loop control and single-loop control

From Section 8.2.2, we know that the envelop of the control curves of Structures #1 and #3 is the optimal control frontier of single-loop control. In the graph, the control curve of the double-loop control structure is below or covers this optimal control frontier. Therefore, the double-loop control structure with optimal loop invariants has better performance than single-loop control structures. To explain this phenomenon, a close observation on Figure 8-24 discloses that:

- When the production rate is low ($P < 0.75$), the control curve of the double-loop control structure coincides with that of Structure #1. In Figure 8-23, the overlapped portion corresponds to the lower-left part of the optimal invariant curve. According to the direction of the optimal invariant curve, increasing I_1 is more effective at increasing the production rate while minimizing the increase of total inventory level than increasing I_2 by the same amount. This indicates that, when the production rate is low, loop ($M_5 \rightarrow M_1$) has dominant effect on production rate control while loop ($M_3 \rightarrow M_1$) almost has no effect. Therefore, the double-loop control structure almost has the same behavior as single-loop control structure #1 (loop: $M_5 \rightarrow M_1$) such that their control curves are superposed.

- When production rate is intermediate ($0.77 < P < 0.82$), the control curve of the double-loop control structure overlaps that of Structure #3. Similarly, in Figure 8-23, the corresponding part of optimal invariant curve indicates that loop ($M_3 \rightarrow M_1$) has dominant effect on production rate control while loop ($M_5 \rightarrow M_1$) almost has no effect. This explains the overlap between the control curves of double-loop control and Structure #3.

- When production rate is high ($P > 0.85$), the curve of double-loop control overlaps that of Structure #1 again. This is because loop ($M_5 \rightarrow M_1$) retakes the dominant position in controlling production rate and total inventory level when production rate is high.

From the above analysis, we develop an insight which says: a double-loop control structure which is a combination of two single-loop control structures can have a better control curve than the envelop curve of those single-loop control structures.

After having investigated the selection of loop invariants, we now discuss how to determine the structure of double-loop control. This is a more important step which should be performed before selecting the loop invariants. Based on the insight in the previous paragraph, a combination of the best single-loop structures can give

better control performance than the optimal control frontier of single-loop control. Therefore, we develop a two-step method:

- Plot the control curves of all single-loop structures and identify the two major control curves which compose the optimal control frontier of single-loop control.
- Design a double-loop structure which is a combination of two single-loop control structures which correspond to the above two major control curves.

Besides the above 5-machine production line example, we perform the behavior analysis on a set of additional experiments (from 7-machine production lines to 25-machine production lines) that are not included in this thesis. Based on the observation, we believe the above insight can be generalized. An optimal control frontier of single-loop control might compose more than two control curves. We can design a multiple-loop control structure which combines the single-loop control structures corresponding to the curves composing the optimal control frontier. We believe that this multiple-loop control should generally have better control performance than any single-loop control structure. It might be better than double-loop control. However, to determine the invariants of more than three loops requires very complicated behavior analysis. In addition, it will be easier to implement fewer loops. Therefore, two-loop control or three-loop control is recommended since it seems to have sufficiently good control performance and it is easy to implement.

Design Insights

To control a production line, we believe that double-loop control could achieve better control performance than any single-loop control structure by choosing appropriate loop invariants. It is probably not wise to implement more than three control loops as this may cause significant challenges in loop invariant design and implementation.

The potential applications of double-loop control are twofold:

- For a given production rate, double-loop control could provide a better total inventory level than single-loop control. For example, in Figure 8-24, when the

target production rate is 0.84, the control curve of double-loop control is below the control curves of Structures #1 and #3.

- When the target production rate varies in a certain range, using single-loop control might require switching between different control structures in order to achieve optimal total inventory level. In other words, on the graph of control curves, we can identify crossing points between single-loop control curves. Double-loop control can provide a better optimal control frontier without changing control structures. Therefore, when target production rate varies, double-loop control could provide robust control. For example, when the target production rate increases from 0.84 to 0.85, in Figure 8-24, the optimal control frontier of single-loop control switches from Structure #3 to Structure #1. A control structure change is required. If the double-loop control structure is used, we only need to adjust loop invariants to make smooth control.

To use double-loop control, a two-step design should be conducted:

- **Determine the control structure.** If the number of variations of single-loop control structures are small, plot the control curves of all variations and identify the major single-loop control structures. If there are too many variations, predict two major single-loop structures using insights developed in Section 8.2.2. The double-loop control structure is designed as a combination of major single-loop control structures.

- **Select loop invariants.** Plot the production rate and total inventory level graphs on the 'I_1 – I_2' plane. Use the iso-curve mapping method to determine the optimal combinations of (I_1, I_2).

When the optimization objective is to minimize total inventory holding cost, we must first specify the cost scheme. Double-loop control structure and optimal loop invariants might vary depending on the specified cost scheme. To optimize total inventory holding cost of a double-loop control, a similar two-step design method can be developed to determine the double-loop structure and optimal loop invariants.

In summary, double-loop control has a better control performance than single-loop control. It is simple enough to implement and could provide flexibility and robustness of control in some cases.

Notice that the above conclusions are all preliminary and lots of experimentations and analyses are required.

8.2.4 Production Lines with Bottlenecks

Control Structure

The previous sections discuss the control of production lines without bottleneck machines. In this section, we examine the cases when there exists a bottleneck machine in a production line.

Consider the 10-machine production line in Figure 8-25(a). Machine M_5 is a bottleneck machine with $\mu_5 = 1.0, p_5 = 0.03, r_5 = 0.07$. Its isolated efficiency e_5 is

$$e_5 = \frac{r_5}{r_5 + p_5} = 0.7 \tag{8.4}$$

All the other machines have following parameters:

$$\mu_i = 1.0, p_i = 0.01, r_i = 0.1, e_i = 0.909 \qquad i = 1, ..., 10; i \neq 5$$

All the buffer sizes are 50.

Five variations of closed loop control are illustrated in Figure 8-25(b). The first four structures are single-loop control. Structure #5 is a double-loop control structure which is the combination of Structures #1 and #3. The sizes of the kanban buffers are 90 in all variations.

Behavior Analysis

We plot the control curves of these five variations in Figure 8-26 and Figure 8-27. The observations made by comparing control curves are summarized as follows:

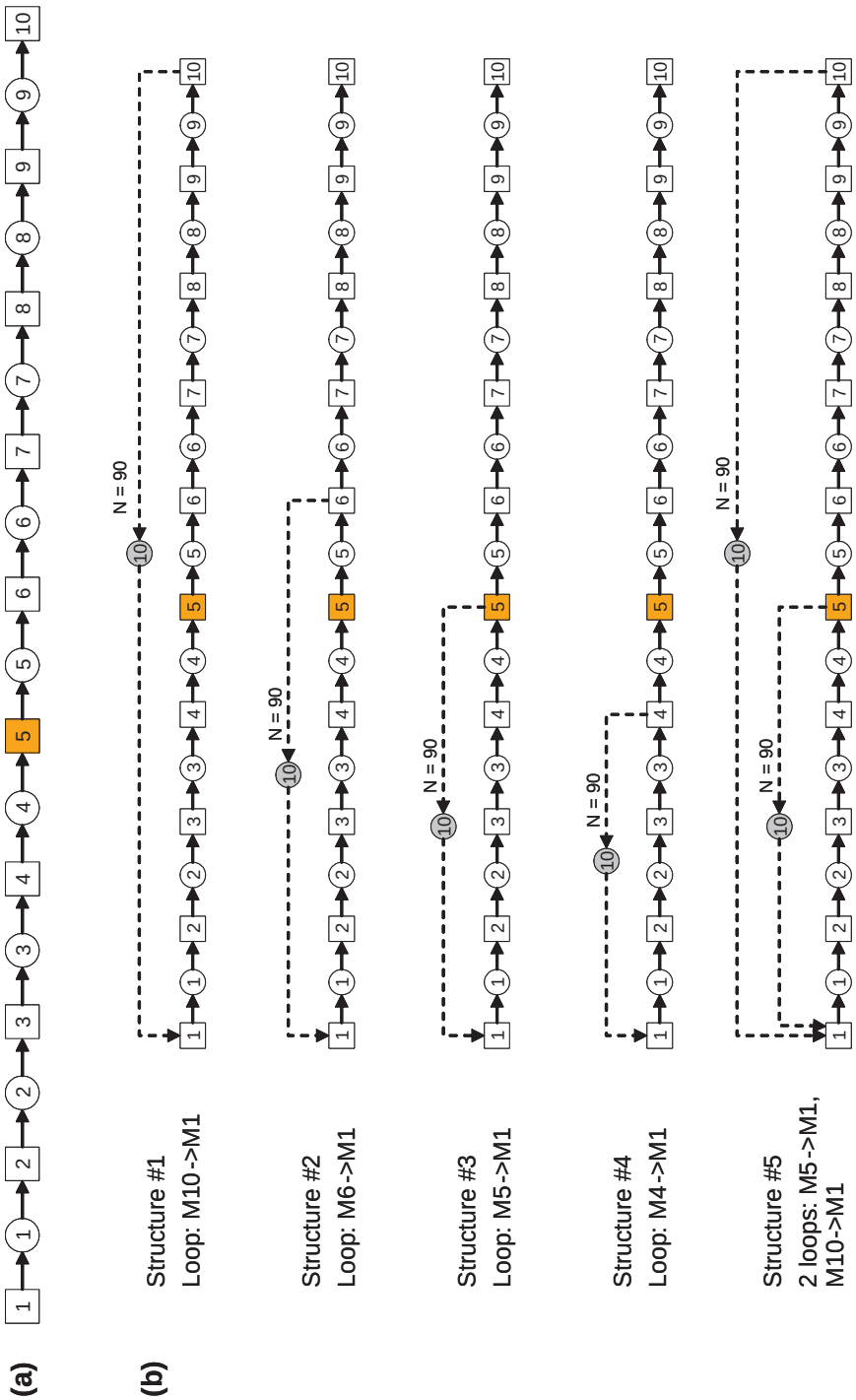


Figure 8-25: A 10-machine production line with bottleneck and five variations of closed loop control

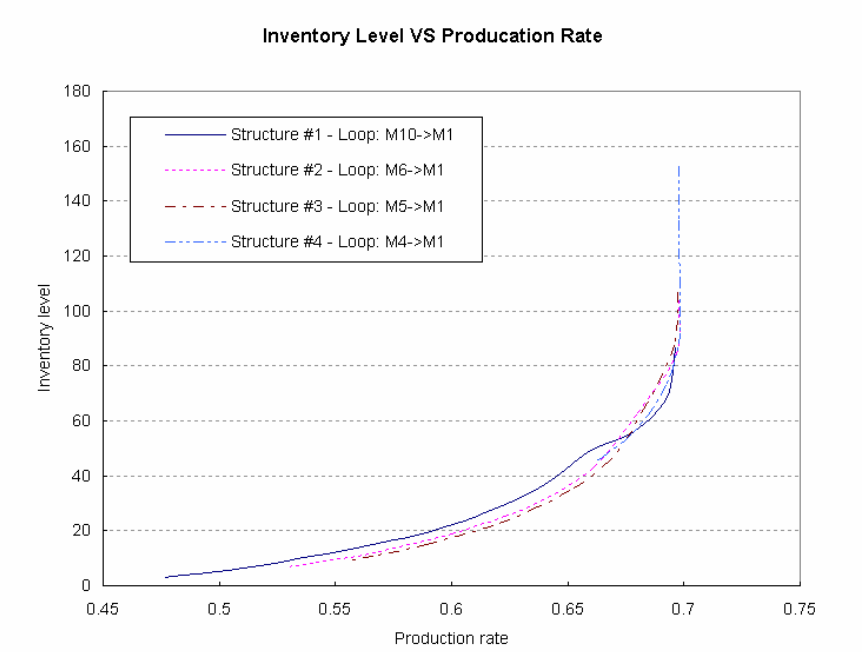


Figure 8-26: Comparison of the total inventory levels of single-loop control structures

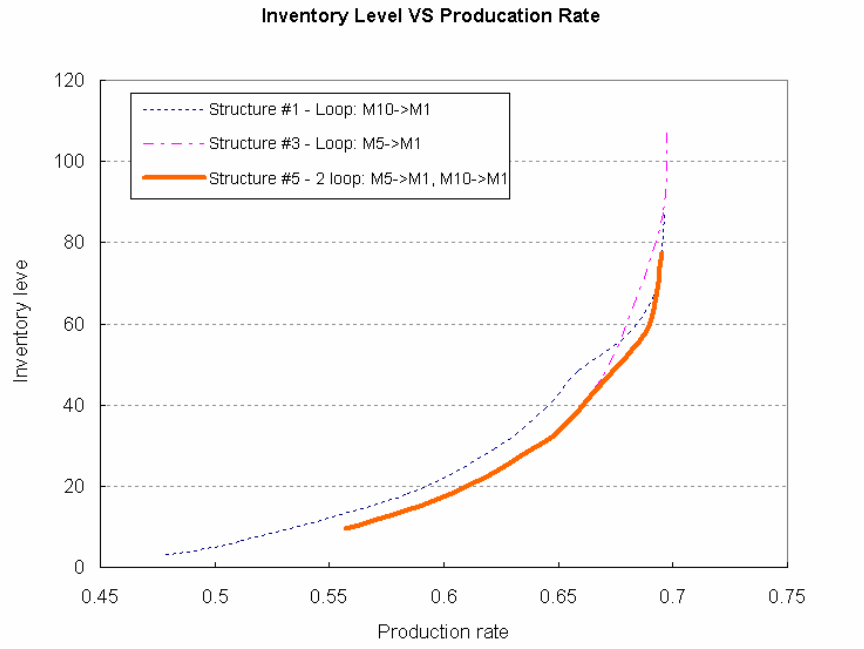


Figure 8-27: Comparison of the total inventory levels of two single-loop control structures and a double-loop control structure

- Structure #3 (loop: $M_5 \rightarrow M_1$) is the optimal control structure when the target production rate is less than 0.66. This structure implements a closed loop control from the bottleneck machine M_5 to M_1.

- Structure #2 (loop: $M_6 \rightarrow M_1$) has a control loop starting from M_6, the downstream machine of M_5. The control curve of Structure #2 is not as good as that of Structure #3. This phenomenon agrees with the design insights of single-loop control in Section 8.2.2: It is better to control the upstream of a production line.

- Structure #4 (loop: $M_4 \rightarrow M_1$) implements a closed loop from M_4, which is the upstream machine of M_5. According to the 'Control the upstream' rule, it should have better control curve than that of Structure #3 (loop: $M_5 \rightarrow M_1$). However, when the target production rate is between 0.660 and 0676, the control curve of Structure #4 is a little above that of Structure #3.

  To explain this phenomenon, we plot the inventory distributions of three control structures in Figure 8-28 when the target production rate is 0.662. The inventory distribution shows that Structure #4 has tight control over buffer B_1, B_2, and B_3. However, B_4, which is not in the control loop, has a much higher buffer level that than in the other two control structures. The reason is that, although the upstream part from M_1 to M_4 is controlled by the closed loop with a very small invariant, the production rate of that part is still much higher than the isolated production rate of the bottleneck machine M_5. This results a high inventory level in buffer B_4, which is between the upstream part and M_5. The high inventory level of B_5 might cause the total inventory level to be higher than that of Structure #3.

  In the graph, when the target production rate increases, the control curve of Structure #4 crosses that of Structure #3 eventually. In addition, the lower limit of Structure #4's control curve is much higher than that of other control curves. Therefore, Structure #4 has limited control capability over production rate — decreasing the loop invariant can not effectively reduce the production

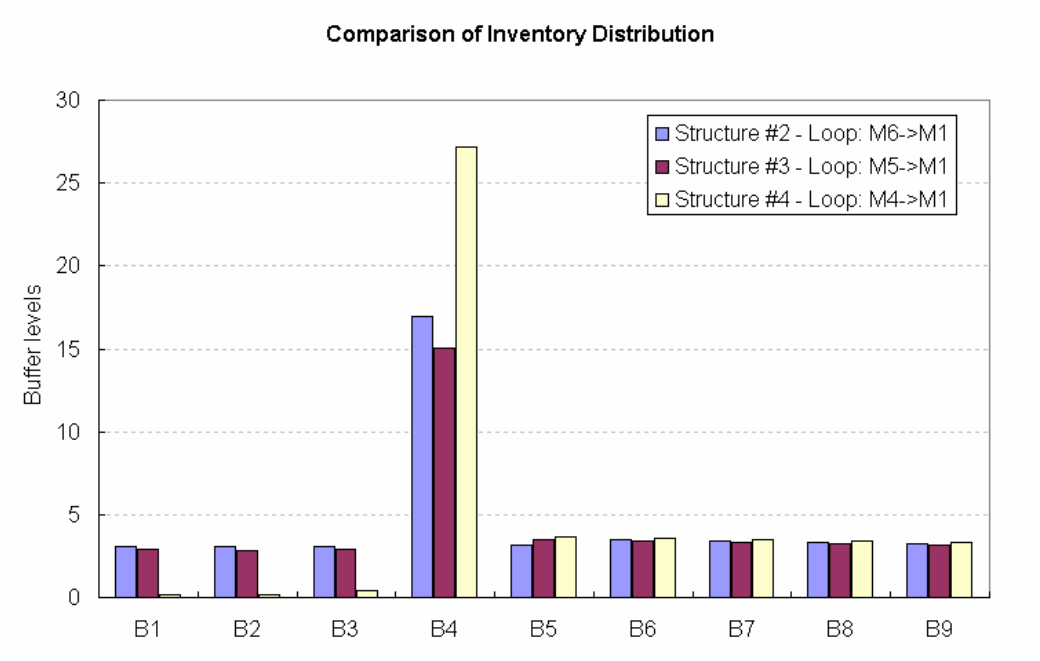


Figure 8-28: Comparison of inventory distributions (production rate = 0.662)

rate.

- Structure #1 (loop: $M_{10} \rightarrow M_1$) becomes the optimal one when the target production rate is greater than 0.678. This is because the inflection points lead the curve across other control curves.

- Structure #5 (double-loop: $M_5 \rightarrow M_1$, $M_{10} \rightarrow M_1$) combines Structures #1 and #3. In Figure 8-27, its control curve is below other two curves when loop invariants are appropriately selected. This feature agrees with the design insights of double-loop control in Section 8.2.3.

Overall, for single-loop control, a control loop from the bottleneck machine to the first machine shows good control over the total inventory level.

Design Insights

When there exists a bottleneck machine in a production line, a single-loop control from the bottleneck machine to the first machine is usually the dominant optimal

control structure. This agrees with the concept 'Drum-Buffer-Rope' (DBR) presented in Goldratt and Cox (1986).

Controlling the upstream part of the bottleneck machine but without including that machine is usually a bad design. This is because a lot of inventory is accumulated in the buffers between the controlled segment and the bottleneck machine.

Using double-loop control could provide better performance than single-loop control when optimal combinations of loop invariants are selected. A set of additional cases have been investigated from 7-machine production lines with bottlenecks to 25-machine production lines with bottlenecks. In most cases, one of the double loops is a loop starting from the bottleneck machine to the first machine.

Last but not least, the cost scheme is still a more important factor to consider than the bottleneck position when the optimization objective is total inventory holding cost. For example, if there is a cost jump at buffer B_4, a single-loop control structure from M_{10} to M_4 has less inventory holding cost than a single-loop control structure from M_5 to M_1.

Notice that the design insights presented here are preliminary and lots of experimentations and analyses are required.

8.3 Control of Assembly Systems

8.3.1 Control Structures

In this section, we discuss the control of assembly systems using multiple-loop structures. Consider the assembly system in Figure 8-29(a). There are two sub-lines and each of them includes five machines. These two sub-lines meet at the assembly machine M_{11}.

Three control structures are illustrated in Figure 8-29(b).

- Structure #1 has a symmetric control structure. We get downstream information from machine M_{11} and the upstream control points are at M_1 and M_6. The sizes of two kanban buffers are both 90. Loop invariants are defined as follows:

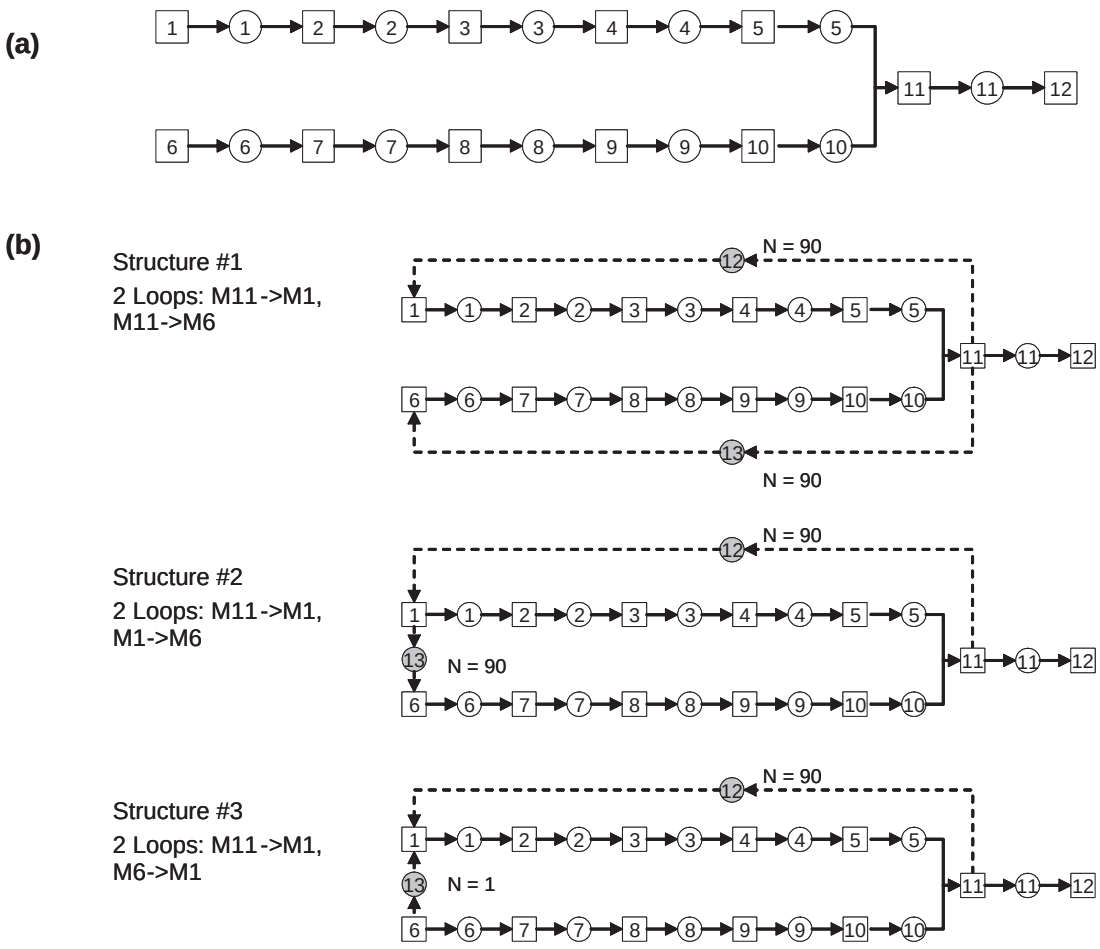


Figure 8-29: An assembly system with two sub-lines and three variations of closed loop control

$$
\begin{aligned}
I_1 &= b(1) + b(2) + b(3) + b(4) + b(5) + b(12) \\
I_2 &= b(6) + b(7) + b(8) + b(9) + b(10) + b(13)
\end{aligned}
$$

Release of material into the upper and lower sub-lines are triggered by the same downstream point, machine M_{11}.

- Structure #2 has a closed loop from M_{11} to M_1. The second closed loop is implemented from M_1 to M_6. The sizes of B_{12} and B_{13} are both 90. Loop invariants are defined as follows:

$$
\begin{aligned}
I_1 &= b(1) + b(2) + b(3) + b(4) + b(5) + b(12) \\
I_2 &= b(1) + b(2) + b(3) + b(4) + b(5) - b(10) - b(9) - b(8) - b(7) - b(6) - b(13)
\end{aligned}
$$

When a workpiece is finished at machine M_1 and buffers B_1 and B_{13} are not full, this workpiece is sent to B_1. At the same time, M_1 takes a kanban from B_{12}, and sends a token to machine M_6. The production of the lower sub-line is controlled by the upper sub-line. Notice that the size of B_{13} is 90, which is relatively large. So the probability that the upper sub-line is blocked by the lower sub-line is near zero.

The structure from machine M_1 to machine M_{11} shows a disassembly/assembly structure. The invariant of the second loop is the difference between the inventory in the upper sub-line and that in the lower sub-line.

- Structure #3 has a closed loop from M_{11} to M_1. The second closed loop is implemented from M_6 to M_1. The size of buffer B_{12} is 90. The size of B_{13} is 1, which is near zero. Loop invariants are defined as follows:

$$
\begin{aligned}
I_1 &= b(1) + b(2) + b(3) + b(4) + b(5) + b(12) \\
I_2 &= b(1) + b(2) + b(3) + b(4) + b(5) - b(10) - b(9) - b(8) - b(7) - b(6) + b(13)
\end{aligned}
$$

When a workpiece is finished at machine M_6 and buffers B_6 and B_{13} are not full, this workpiece is sent to B_6. At the same time, a token is sent to B_{13} by M_1. Notice that the size of buffer B_{13} is 1. On one hand, the production of the upper sub-line is tightly controlled by the lower sub-line because the upper sub-line can produce only when M_6 finishes a workpiece and one token is present in B_{13}. On the other hand, buffer B_{13} ensures the synchronization between the upper and lower sub-lines. The lower sub-line is blocked until one token in B_{13} is consumed by M_1.

Similarly, a disassembly/assembly structure is formed from M_1 to M_{11}. The difference between the inventory in the upper sub-line and that in the lower sub-line is loop invariant I_2.

8.3.2 Identical Sub-lines

In this section, we assume two sub-lines have identical machines and buffers. All the buffer sizes are 20. All the machines in the system are identical with $\mu = 1.0$, $p = 0.01$, and $r = 0.1$. The efficiencies of all machines are

$$e_i = \frac{r_i}{r_i + p_i} = \frac{0.1}{0.1 + 0.01} = 0.909 \qquad i = 1, 2, ..., 12 \tag{8.5}$$

Inventory Comparison

We analyze the behavior of three control structures. Loop invariants are selected to optimize the control performance, i.e. to minimize total inventory level for a given target production rate. The total average inventory level T_{inv} is:

$$T_{inv} = \sum_{j=1}^{11} \overline{b}(j) \tag{8.6}$$

To obtain the optimal control curve, we should design the loop invariants to keep the upper and lower sub-lines balanced in terms of production rate. When the upper and lower sub-lines are identical, the assembly system is symmetric.

- In Structure #1, as the control loops of upper and lower sub-lines are symmetric, it is optimal to choose the same value for invariants I_1 and I_2. The inventory distributions in the upper and lower sub-lines are exactly the same.

- In Structure #2, the lower sub-line is controlled by tokens sent from M_1. I_2 is the difference between the inventory in the upper sub-line and that in the lower sub-line. In order to keep the sub-lines balanced, I_2 is set to 0 such that similar inventory distributions appear in the upper and lower sub-lines.

- In Structure #3, the production of the upper sub-line is triggered by the lower sub-line. Loop invariant I_2 is set to 0 in order to achieve similar inventory distributions in the upper and lower sub-lines.

In Figure 8-30, we plot the control curves of three control structures. The control curve of Structure #1 is obtained by varying I_1 and I_2 simultaneously. For the control curves of Structures #1 and #2, we vary I_1 and keep I_2 as 0. In the graph, three control structures have very similar control curves as they all get control feedback from the same downstream position — machine M_{11}.

Inventory Holding Cost Comparison

To compare the control performance based on total inventory holding cost, we need to specify a cost scheme. Three cost schemes for the assembly system in Figure 8-29(a) are specified in Table 8.4.

In cost scheme #1, the upper and lower sub-lines has the same inventory holding costs. In cost scheme #2, the inventory holding cost of the upper sub-line is higher than that of the lower sub-line. Cost scheme #3 is the opposite of cost scheme #2.

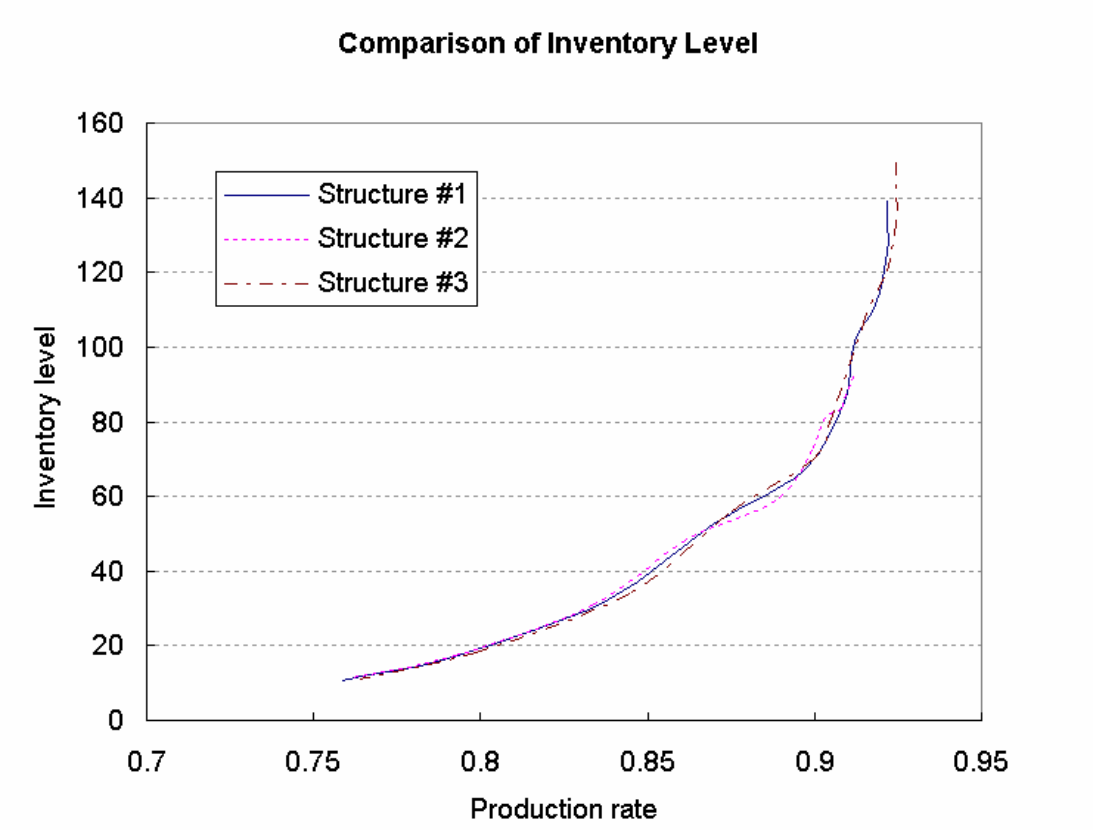


Figure 8-30: Comparison of the total inventory levels of three control structures

Cost Scheme #	Description	Upper sub-line	Lower sub-line	Main line
		C_1, C_2, C_3, C_4, C_5	C_6, C_7, C_8, C_9, C_{10}	C_{11}
1	Upper = Lower	1000	1000	2000
2	Upper > Lower	1000	100	1100
3	Upper < Lower	100	1000	1100

Table 8.4: Three cost schemes of an assembly system

We study control structure #3 in Figure 8-29(b) by varying loop invariant I_2. In the previous section, we assume $I_2 = 0$ such that the upper and lower sub-lines are balanced. However, when the optimization objective is inventory holding cost instead of inventory level, keeping I_2 at 0 might give worse performance.

While varying I_1,we set three values for I_2: 0, 15, and -15. The inventory holding costs of Structure #3 in three cost schemes are plotted in Figure 8-31 to Figure 8-33. The analyses of three cost schemes are presented as follows:

- In cost scheme #1, the costs of the upper and lower sub-lines are the same. In Figure 8-31, the curves corresponding to $I_2 = 15$ and $I_2 = -15$ are above the curve corresponding to $I = 0$. This is because, either increasing or decreasing I_2 causes two sub-lines to be unbalanced — the difference between the inventory in the upper sub-line and that in the lower sub-lines increases.

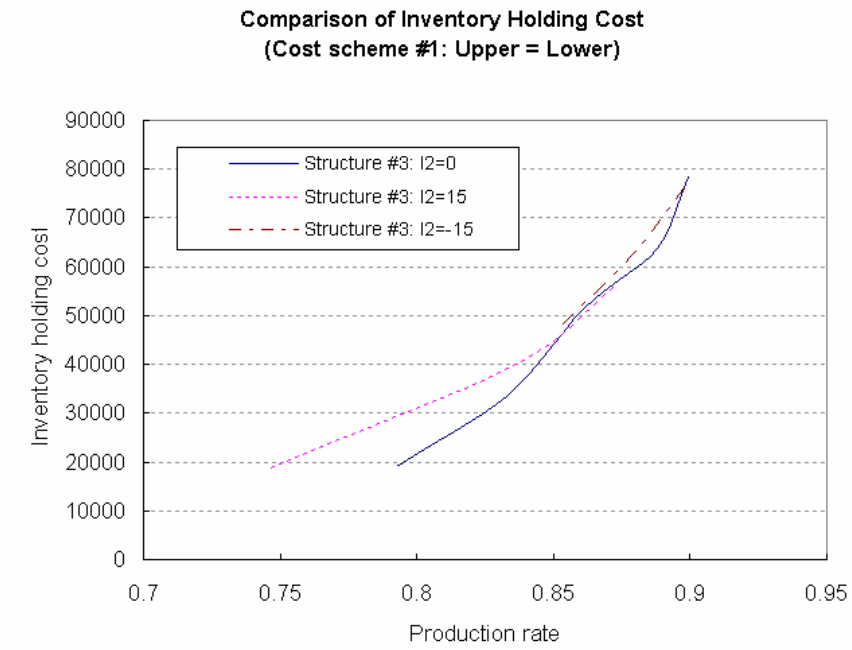


Figure 8-31: Comparison of the total inventory holding costs of Structure # 3 while varying I_2 (cost scheme #1)

- In cost scheme #2, the cost of upper sub-line is much higher than that of the lower sub-line. Intuitively, to increase the target production rate, holding more

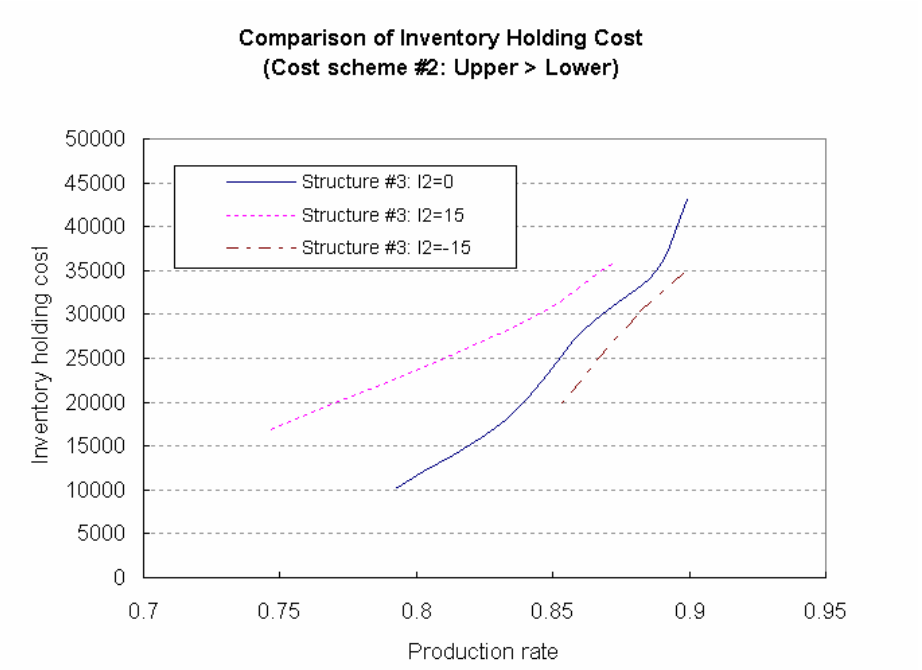


Figure 8-32: Comparison of the total inventory holding costs of Structure # 3 while varying I_2 (cost scheme #2)

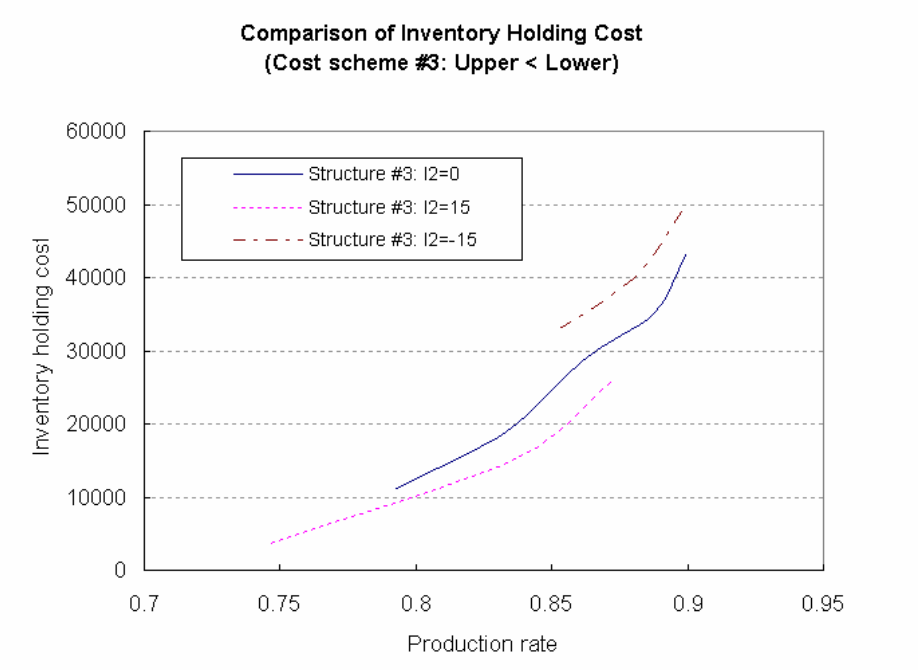


Figure 8-33: Comparison of the total inventory holding costs of Structure # 3 while varying I_2 (cost scheme #3)

inventory in less expensive buffers could reduce the increase of total inventory holding cost. Therefore, we keep less inventory in the upper sub-line than that in the lower sub-line. Invariant I_2 is set to be less than 0. In Figure 8-32, the curve corresponding to $I_2 = -15$ is below the curves corresponding to $I = 0$ and $I = 15$. A negative I_2 is better than $I_2 = 0$.

- In cost scheme #3, the cost of the lower sub-line is much higher than that of the upper sub-line. This is the opposite case of cost scheme #2. Therefore, we keep less inventory in the lower sub-line than that in the upper sub-line. Invariant I_2 is set to be greater than 0. In Figure 8-33, the curve corresponding to $I_2 = -15$, which is the best case in cost scheme #2, becomes the worst case in cost scheme #3. The curve corresponding to $I_2 = 15$ is best among the three curves. A positive I_2 is better than $I_2 = 0$.

If we use control structure #1 in Figure 8-29(b), the invariant of the closed loop which controls the high cost sub-line should be less than that of the closed loop which controls the low cost sub-line. If Structure #2 is used, invariant I_2 should be negative when the upper sub-line is high cost. When the lower sub-line is high cost, it should be positive.

To summarize, when cost scheme of an assembly system varies, the loop invariant design of a multiple-loop control structure should be as follows:

- keeping less inventory in high cost sub-lines to reduce total inventory holding cost.

- keeping more inventory in low cost sub-lines to reduce the interruption of flows due to starvation or blocking.

In all three cost schemes in Table 8.4, the cost scheme within each sub-line is constant. Suppose the cost scheme within a sub-line varies, such as decreasing, increasing, cost jump, etc. Then the design insights of production line control presented in Section 8.2 could be applied to improve production control within a single sub-line. Notice that lots of cases are needed to verify this.

8.3.3 Nonidentical Sub-lines

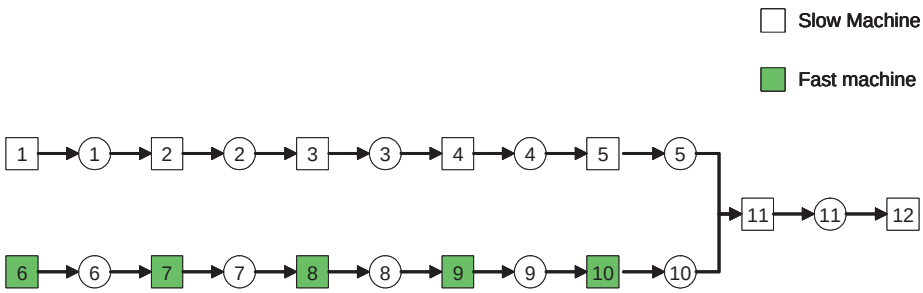


Figure 8-34: An assembly system with nonidentical sub-lines

In the previous section, the machines and buffers in two sub-lines are identical such that the efficiencies of the upper and lower sub-lines are equal. In this section, we increase the repair rates of the machines in the lower sub-line from $r = 0.1$ to $r = 0.2$ (Figure 8-34). Therefore, the production rate of the lower sub-line is greater than that of the upper one:

$$e_i = \frac{r_i}{r_i + p_i} = \frac{0.2}{0.2 + 0.01} = 0.952 \qquad i = 6, 7, ..., 10 \tag{8.7}$$

The parameters of all other machines and buffer sizes remain unchanged.

Inventory Comparison

While using symmetric control structure (Structure #1) to control two identical sub-lines, it is optimal to choose the same value for invariants I_1 and I_2. To verify this statement, in Figure 8-35, we plot the production rate and total inventory level on the 'I_1 – I_2' plane. As the production rate and total inventory level plots are diagonally symmetric. The optimal invariant curve illustrated in the graph is the diagonal which indicates $I_1 = I_2$.

When two sub-lines have different production rates, we should carefully select loop invariants. We plot the production rate and total inventory level in Figure 8-36. We observe that, the production rate and total inventory level plots are no longer diagonally symmetric. By using Structure #1, simultaneously varying the

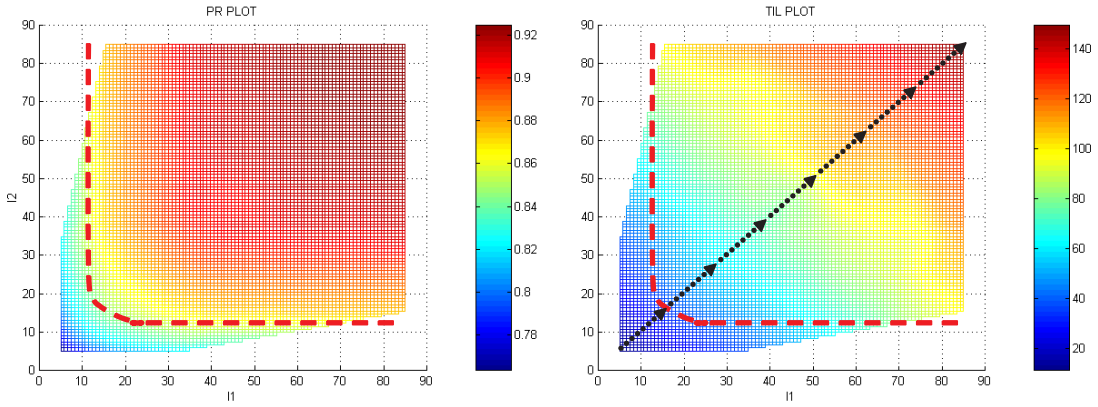


Figure 8-35: Determine the optimal loop invariants of an assembly system with identical sub-lines controlled by Structure #1

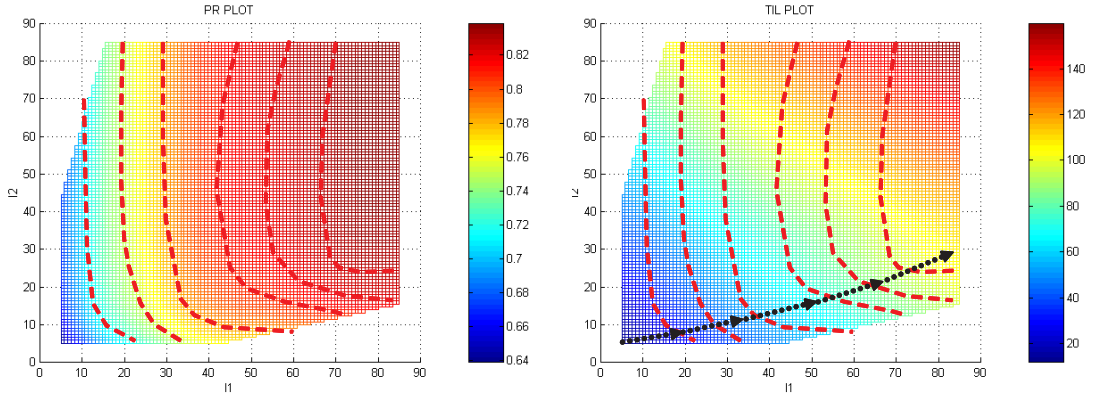


Figure 8-36: Determine the optimal loop invariants of an assembly system with non-identical sub-lines controlled by Structure #1

loop invariants does not give the optimal control performance. According to the optimal invariant curve, we observe that I_2 is always smaller than I_1 and the increase speed of I_2 is much slower than I_1. This is because when the production rates of the upper and lower sub-lines are equal, the lower sub-line require much less buffer inventory since its machines are more efficient than those of the upper sub-line.

If we use control structures #2 or #3 in Figure 8-29(b) to control the assembly system in Figure 8-34, loop invariant I_2 should be positive because we need less inventory in the lower sub-line than in the upper sub-line to balance the production rate of these two sub-lines.

Inventory Holding Cost Comparison

When a cost scheme is included to optimize total inventory holding cost, the design of loop invariants becomes more complicated because the difference of production rate and the difference of inventory holding cost are combined.

Using the cost scheme in Table 8.4, three combinations are illustrated in Table 8.5 and Figure 8-37.

Combination #	Production rates	Cost Scheme
1	Upper < Lower	Upper = Lower
2	Upper < Lower	Upper > Lower
3	Upper < Lower	Upper < Lower

Table 8.5: Three combinations of different production rates and cost schemes of an assembly system

We discuss the optimal loop invariant selection of these three combinations.

- In Combination #1, because the costs of two sub-lines are the same, the selection of loop invariants to optimize total inventory holding cost is the same as that to optimize total inventory level. Suppose we use control structure #1 in Figure 8-29(b). The optimal invariant curve is illustrated in Figure 8-38.

- Combination #2 has conflicting features. When we consider the balance of flow, the upper sub-line should hold more inventory than the lower sub-line since the

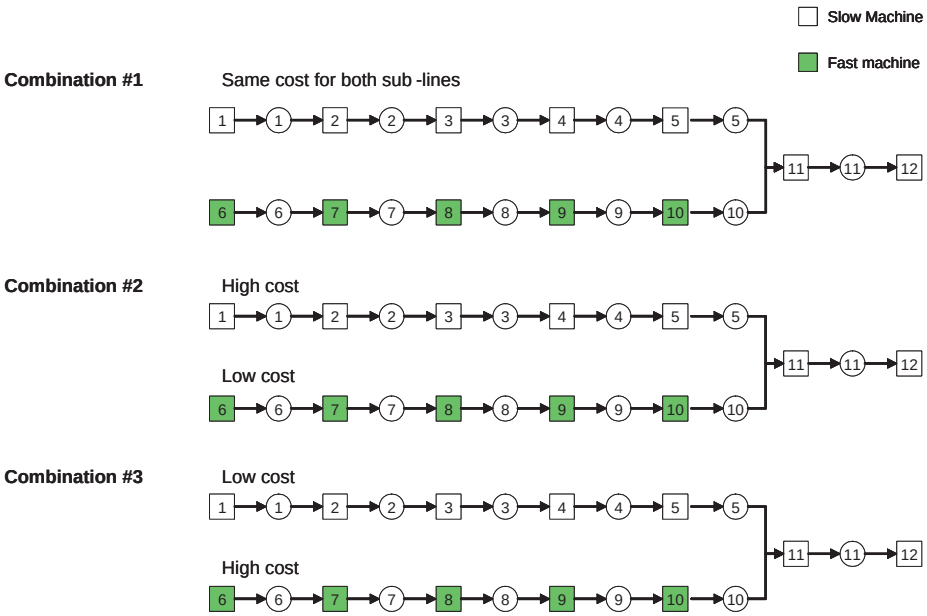


Figure 8-37: Three combination of different production rates and cost schemes of an assembly system

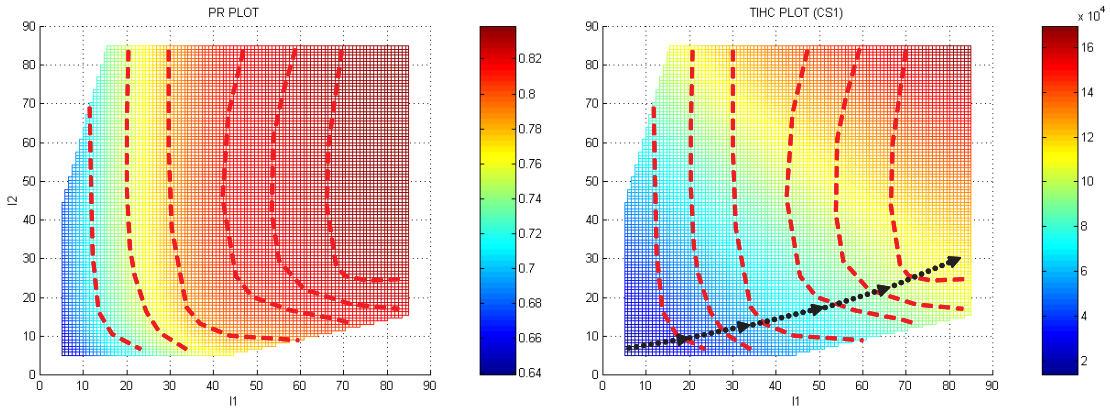


Figure 8-38: Determine the optimal loop invariants of an assembly system with non-identical sub-lines controlled by Structure #1 (cost scheme #1)

upper sub-line is slower than the lower one. However, from the cost perspective, less inventory should be kept in the upper sub-line than the lower one. Suppose control structure #1 in Figure 8-29(b) is used. The optimal invariant curve of Combination #2 in Figure 8-39 is above that of Combination #1 in Figure 8-38. This indicates that, for the same I_1, optimal I_2 in Combination #2 is larger than that in Combination #1.

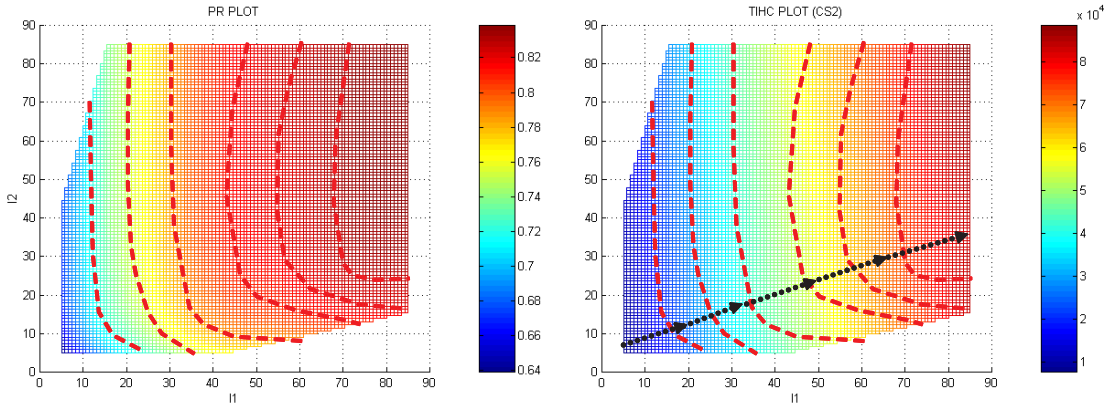


Figure 8-39: Determine the optimal loop invariants of an assembly system with non-identical sub-lines controlled by Structure #1 (cost scheme #2)

- In Combination #3, the differences of production rate and cost both indicate that less inventory should be kept in the lower sub-line than in the upper because the lower sub-line has higher speed and higher inventory holding cost. When the system is controlled by Structure #1 in Figure 8-29(c), we plot the optimal invariant curve in Figure 8-40. The optimal invariant curve is below that of Combination #1 in Figure 8-38. This indicates that, for the same I_1, optimal I_2 in Combination #3 is smaller than that in Combination #1.

To explain the above phenomena, an analysis of the production rate and total inventory holding cost plots are described here.

- By comparing Figure 8-35 and Figure 8-36, the difference of production rate causes asymmetry of the production rate plot, while the total inventory level plot has slight change. When the upper sub-line is slower than the lower one, the production rate plot deviates to the lower-right corner. Otherwise, the plot

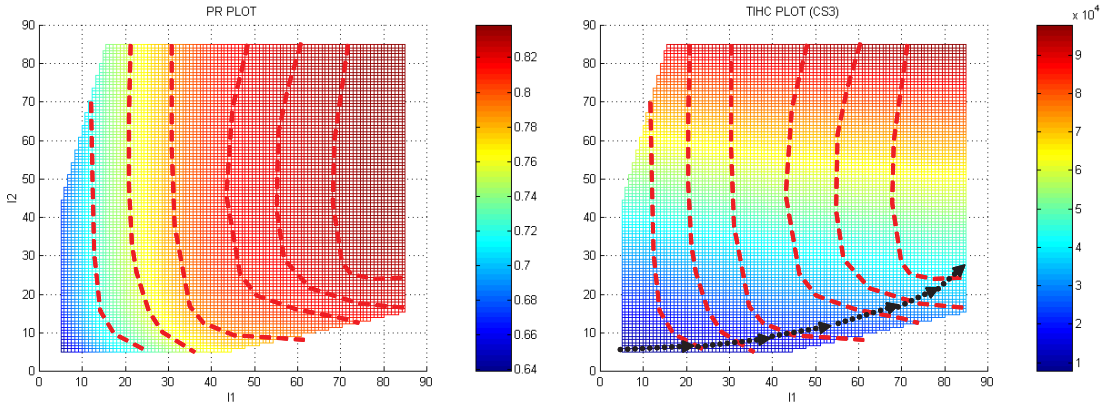


Figure 8-40: Determine the optimal loop invariants of an assembly system with non-identical sub-lines controlled by Structure #1 (cost scheme #3)

deviates to the upper-left corner when the upper sub-line is faster than the lower one.

- Compare Figure 8-38, Figure 8-39, and Figure 8-40. The cost scheme does not affect the production rate and total inventory level plots. However, it rotates the plot of total inventory holding cost either clockwise (cost scheme #2) or counterclockwise (cost scheme #3). Compared to the optimal invariant curve in cost scheme #1, the clockwise rotation in cost scheme #2 raises the optimal invariant curve, while counterclockwise rotation in cost scheme #3 lowers it.

When the changes of the production rate and total inventory holding cost plots are combined, we observe that the deviation of the optimal invariant curve with reference to the optimal invariant curve when sub-lines are identical and have same costs.

- In Combination #1, the production rate plot deviates to the lower-right corner and total inventory holding cost plot does not rotate. The optimal invariant curve deviates downward.

- In Combination #2, the production rate plot deviates to the lower-right corner and the total inventory holding cost plot rotates clockwise. The optimal invariant curve deviates downward and then bends upward. The final deviation of a point on the optimal invariant curve depends on the magnitudes of these two opposite movements.

- In Combination #3, the production rate plot deviates to the lower-right corner and total inventory holding cost plot rotates anticlockwise. The optimal invariant curve deviates downward and then bend downward. Downward deviation and downward bend both cause the whole optimal invariant curve deviate downward.

In summary, when the optimization objective is the total inventory holding cost, the deviation of the optimal invariant curve is caused by the differences of production rate and inventory holding cost.

8.3.4 Design Insights

To control an assembly system using multiple-loop structures, beside selecting multiple-loop control structures, the key design issue is to determine optimal loop invariants, i.e. determining the optimal invariant curve, to optimize total inventory level, total inventory holding cost, etc.

When the optimization objective is total inventory holding cost, it is important to consider the production rate difference and cost scheme as a combination to affect the deviation of optimal invariant curve. The difference of production rate is usually a major drive of the deviation of optimal invariant curve. However, sometimes the deviation depends on cost scheme rather than the difference of production rate. Be cautious that a certain combination of sub-lines' production rates and cost scheme might have conflicting effects on the deviation of optimal invariant curve.

Therefore, we must carefully compare the directions and magnitudes of the deviations due to difference of production rate and cost scheme. Analyzing the production rate and total inventory holding cost plots provides an effective method to identify the behavior of optimal invariant curve.

8.4 Conclusion

In this chapter, we performed the behavior analysis of using closed loop structures to control production lines and assembly systems. Design insights are presented at the end of each section. In principle, the overall objective of the production control using multiple-loop structures is to balance the flows and trade-off the cost distribution between the upstream and the downstream of a line segment, and across a set of line segments.

Key messages from behavior analysis are summarized as follows:

1. It is better to control from the upstream of a production line. However, the cost scheme is a factor that might be more important.

2. To control a production line, double-loop control has better performance than single-loop control in some cases.

3. To control a production line with bottleneck machines, the bottleneck machines are usually the starting point of closed loop control.

4. To control a complex assembly/disassembly system, we must consider the combination of production rate differences and the cost scheme of the whole system.

Design guidelines are developed for production control using multiple-loop structures. Notice that these guidelines are tentative and much more experiments and analyses are required.

1. Fully understand the production requirements (e.g. average demand and demand fluctuation) and the objective of production control (e.g. produce at a target production rate, maximize profit, minimize inventory holding cost).

2. Clearly specify the cost scheme of the whole system. Identify the positions of cost jumps of drops. Implement closed loop control wherever necessary.

3. Identify bottlenecks and prioritize them from high cost segments to low cost segments. Apply closed loop control from the bottleneck if the control structure does no conflict with the control structure design based on cost scheme.

4. Use double-loop or three-loop control for line segments wherever necessary.

5. After determining control structures, design loop invariants to balance the flows and trade-off the cost distribution.

These guidelines provide an efficient method to select a set of good variations of control structures and loop parameters for evaluation during design. Among the selected set of variations, we could achieve good enough sub-optimal solutions after performing the analysis and design of loop invariants. After that, simulation should be conducted to verify and refine the design.

To reach a true optimal design requires very sophisticated modeling and exhaustive analysis. Unfortunately, the optimal solution might be fairly difficult to implement on a shop floor due to its complexity of control structures. Therefore, we recommend using sub-optimal control solutions which are good enough, easy to implement, and robust.

To conclude, multiple-loop structures could provide effective methods to control production systems in order to meet target performance measures.

Chapter 9

Conclusion

This thesis develops a tool to analyze multiple-loop structures and it develops an understanding of manufacturing systems controlled by such structures. First, we develop a graph model and an induction method to efficiently analyze the blocking and starvation properties of complex assembly/disassembly networks with arbitrary topologies. Based on decomposition, three versions of an efficient algorithm are developed for evaluating assembly/disassembly networks with arbitrary topologies. We describe some important behaviors of the production lines and assembly systems controlled by multiple-loop structures. Finally, design insights developed based on this behavior analysis are summarized as guidelines for production system design and operational control using multiple-loop structures.

The key contributions of this thesis and their importance are highlighted here:

- **A unified assembly/disassembly network model.** This model represents the essence of the kanban control mechanism by using assembly/disasssembly operations. Information flows and material flows are integrated into a unified model. It provides a framework for the development of the graph model and evaluation algorithms in this thesis.

- **A graph model and an induction method for blocking and starvation analysis.** The graph model represents a set of important features of assem-

bly/disassembly networks, including flows, assembly/disassembly operations, loop invariants, machine failures, and blocking and starvation. A systematic approach was developed for analyzing blocking and starvation properties. Three major steps are addressed:

  - A 'single machine failure assumption' is made to understand the maximal propagation effect of a machine failure.
  - A 'Machine failure – Buffer level' matrix is used to represent the blocking and starvation relationships between machine failures and the limiting state buffer levels under the single machine failure assumption.
  - An induction method was invented to efficiently find the limiting state buffer levels and derive the 'Machine failure - Buffer level' matrix of the whole system.

  The above approach enables us to analyze the blocking and starvation property of large-scale complex assembly/disassembly systems with arbitrary topologies. In additional, the graph model can be further explored from the perspective of flow network analysis.

- **An efficient algorithm for evaluating large-scale assembly/disassembly systems with arbitrary topologies.** This algorithm has two phases. Phase I: blocking and starvation analysis provides essential information for Phase II: decomposition evaluation. We performed a set of numerical experiments which show that the algorithm is accurate, reliable, and fast. We emphasize the efficiency of the algorithm because production situations (e.g. demand, process, quality, cost) are always changing and factories or production systems require frequent redesign or adjustments. Therefore, it is extremely important to have a method which could efficiently evaluate a large number of variations with different structures and parameters.

- **Behavior analysis and design insights on closed loop control.** The behavior analysis shows the dynamics of the selection of optimal control struc-

tures and parameters when the optimization objective varies. We developed design insights on closed loop control of production lines and assembly systems. Preliminary practical guidelines for the design and operational control of multiple-kanban production systems are summarized at the end. The preliminary guidelines further improve the efficiency of design by selecting good variations for evaluation.

The contributions of this thesis, especially the ADN algorithm and the behavior analysis of closed loop control, lay a foundation for the optimal design and control of production systems using multiple-loop structures. The following items are recommended for future research:

- Develop optimization methods to minimize inventory level, minimize inventory holding cost, and trade-offs across multiple performance measures.
- Expand the models and evaluation algorithms to include flow merge and split, quality issues (e.g. yield rate, rework flow), multiple-part type, reentrant processes, etc.
- Based on the expanded new evaluation methods, understand the behavior and develop design insights on some new types of systems featured by quality feedback and rework flow, multiple-part type, reentrant flow, etc.
- For industrial practitioners, develop guidelines which are new, valuable, intuitive, and easy to implement. The development of practical guidelines requires much more experimentations and analyses. The first item, the development of optimization methods, will help.

Appendix A

The Uniqueness of Blocking and Starvation

The goal of the blocking and starvation analysis is to obtain the 'Machine failure – Buffer level' matrix (Section 4.2.4) which we can use for decomposition to set up the failure modes for the pseudo-machines in each building block (Section 5.3). In order to ensure that the assignment of failure modes during decomposition is both possible and unique, we must prove that the 'Machine failure – Buffer level' matrix is unique. Equivalently, we must prove that, under the single machine failure assumption, the limiting propagation state of buffer levels due to a specific single machine failure exists, is independent of the initial state, and is unique.

In this proof, we first state the objective and notation and we formulate the problem. The proof steps are summarized and followed by the details of each proof step.

A.1 Objective

This proof demonstrates the *uniqueness property* of connected flow networks: in the limiting propagation state due to a single machine failure at M_s, the cumulative flows of all the machines are simultaneously maximized. The level of each buffer is unique and independent of the initial buffer levels.

A.2 Notation and Definitions

Given a connected flow network Ω with n machines and m buffers, we specify constants, time-varying quantities, and flow networks.

Constants (including structural matrices)

- *Buffer size* N_j is defined as the size of buffer B_j. The sizes of all the buffers are denoted by buffer size vector: $\mathbf{N} = [N_1, ..., N_m]^T$.

- *All-vertex incidence matrix* $\Phi = [\phi_{ij}]$ has n rows, one for each machine, and m columns, one for each buffer. The element ϕ_{ij} of Φ is defined as follows:

$$\phi_{ij} = \begin{cases} 1 & \text{if the } j\text{th buffer is incident out of the } i\text{th machine;} \\ -1 & \text{if the } j\text{th buffer is incident into the } i\text{th machine;} \\ 0 & \text{if the } j\text{th buffer is not incident on the } i\text{th machine.} \end{cases}$$

  Note: the rank of Φ is $n-1$.

- *Incidence matrix* Φ_f is an $(n-1)$-rowed submatrix of Φ. The machine which corresponds to the row of Φ which is not in Φ_f is called the *reference machine* of Φ_f. The incidence matrix with M_s as reference machine is denoted by $\Phi_f(s)$. Notice that matrix $\Phi_f(s)$ is a matrix with rank $n-1$.

- A *spanning tree* T is a tree made from elements of network Ω. It contains all n machines and $n-1$ of the buffers. B_T is the set of buffers of spanning tree T. $B_{\overline{T}}$ is the set of buffers not in the spanning tree.

- A *path* $p(s,i)$ is a finite alternating sequence of machines and buffers beginning with M_s and ending with M_i. All the machines of $p(s,i)$ are distinct. In a spanning tree, there exists a unique path between any given pair of machines.

- *Path matrix* $\Gamma(T,s) = [\gamma_{ij}(T,s)]$ is an $(n-1)\times m$ matrix defined on the selected spanning tree T with machine M_s as reference machine:

$$\gamma_{ij}(T,s) = \begin{cases} 1 & \text{if the } j\text{th buffer is in the unique path } p(s,i) \text{ and connected} \\ & \text{to the } s\text{th machine via its upstream machine } u(j); \\ -1 & \text{if the } j\text{th buffer is in the unique path } p(s,i) \text{ and connected} \\ & \text{to the } s\text{th machine via its downstream machine } d(j); \\ 0 & \text{if the } j\text{th buffer is not in the unique path } p(s,i) \end{cases}$$

The rank of path matrix $\Gamma(T,s)$ is $n-1$. All its row vectors $\mathbf{\Gamma}_{i\star}(T,s), i = 1,...,n; i \neq s$ are linearly independent. Row vector $\mathbf{\Gamma}_{i\star}(T,s)$ corresponds to path $p(s,i)$. The row vector corresponds to path $p(i,k), i,k = 1,...,n; i,j \neq s$ is given by

$$\mathbf{\Gamma}_{k\star}(T,s) - \mathbf{\Gamma}_{i\star}(T,s) \tag{A.1}$$

For a spanning tree T of Ω, a non-singular submatrix $\Phi_f(T,s)$ of incidence matrix $\Phi_f(s)$ can be identified. The $n-1$ columns of $\Phi_f(T,s)$ correspond to $n-1$ buffers in T. A relationship between path matrix $\Gamma(T,s)$ and incidence matrix $\Phi_f(T,s)$ is described in Hale (1962) and Resh (1988):

$$\Gamma(T,s) = -\left(\Phi_f(T,s)^T\right)^{-1} \tag{A.2}$$

- *Fundamental circuit matrix* $\Psi = [\psi_{ij}]$ has m columns, one for each buffer, and $m-n+1$ rows, one for each fundamental circuit or loop with respect to tree T. The entry ψ_{ij} is defined as follows:

$$\psi_{ij} = \begin{cases} 1 & \text{if the } j\text{th buffer is in the } i\text{th loop and its orientation agrees with the loop orientation;} \\ -1 & \text{if the } j\text{th buffer is in the } i\text{th loop and its orientation does not agree with the loop orientation;} \\ 0 & \text{if the } j\text{th buffer is not in the } i\text{th loop.} \end{cases}$$

Note: the rank of Ψ is $m - n + 1$.

- *Loop invariant* I_k is defined as the buffer invariant of closed loop L_k. The invariant of all the fundamental loops can be denoted by loop invariant vector $\mathbf{I} = [I_1, ..., I_{m-n+1}]^T$.

Time-Varying Quantities

- *Cumulative flow* $q(i,t)$ is defined as the amount of material that is transported through machine M_i during time $(0,t)$. $q(i,t)$ is non-decreasing. The cumulative flows of all the machines are denoted by the cumulative flow vector: $\mathbf{q}(t) = [q(1,t), ..., q(n,t)]^T$. The cumulative flow vector without the component corresponding to reference machine M_s is denoted by $\mathbf{q}_s(t)$.

  Note: when we neglect time scale, we can simply write the cumulative flow of M_i as $q(i)$.

- *Buffer level* $b(j,t)$ is defined as the amount of material in buffer B_j at time t. The levels of all the buffers are denoted by buffer level vector: $\mathbf{b}(t) = [b(1,t), ..., b(m,t)]^T$. Given a spanning tree T, the vector of buffer levels of B_T is called $\mathbf{b}_T(t)$. The vector of buffer levels of $B_{\overline{T}}$ is $\mathbf{b}_{\overline{T}}(t)$.

  Note: when we neglect time, we simply write the buffer level of B_j as $b(j)$.

- *Blocking and starvation* of machine M_i can be defined as follows:

  - M_i is blocked at time t, if

$$\exists w \in D(M_i) \qquad b(w,t) = N_w$$

where $D(M_i)$ is the set of downstream buffers of M_i.

– M_i is starved at time t, if

$$\exists v \in U(M_i) \qquad b(v,t) = 0$$

where $U(M_i)$ is the set of upstream buffers of M_i.

Connected Flow Networks with a Single Machine Failure

In Section 3.4, we describe two types of connected flow networks. Here, we describe connected flow networks with a single machine failure.

- A *static connected flow network with a single machine failure* can be described using the following system of equations and inequalities:

$$\mathbf{b} = \mathbf{b}^0 + \Phi^T \mathbf{q} \tag{A.3}$$

$$\Psi \mathbf{b} = \mathbf{I} \tag{A.4}$$

$$0 \leq \mathbf{b} \leq \mathbf{N} \tag{A.5}$$

$$\mathbf{q} \geq 0 \tag{A.6}$$

$$q(s) = 0 \tag{A.7}$$

where $\mathbf{b}^0$ is called the *initial buffer level vector*. Machine M_s is the machine which is down. Vectors $\mathbf{I}$ and $\mathbf{N}$ are assumed to be feasible for the system. That is, there is at least one $(\mathbf{b}, \mathbf{q})$ that satisfies (A.3)–(A.7).

Substituting (A.7) into (A.3), we can use incidence matrix $\Phi_f(s)$ and cumulative flow vector $\mathbf{q}_s$ to represent (A.3). $\mathbf{q}_s$ is the cumulative flow vector without the component corresponding to reference machine M_s.

The whole system can be rewritten as follows:

$$\mathbf{b} = \mathbf{b}^0 + \Phi_f(s)^T \mathbf{q}_s \tag{A.8}$$

$$\Psi \mathbf{b} = \mathbf{I} \tag{A.9}$$

$$0 \leq \mathbf{b} \leq \mathbf{N} \tag{A.10}$$

$$\mathbf{q} \geq 0 \tag{A.11}$$

$$q(s) = 0 \tag{A.12}$$

- A *dynamic connected flow network with a single machine failure* can be described using the following system of equations and inequalities:

For all $t \geq 0$

$$\mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) \tag{A.13}$$

$$\Psi \mathbf{b}(t) = \mathbf{I} \tag{A.14}$$

$$0 \leq \mathbf{b}(t) \leq \mathbf{N} \tag{A.15}$$

$$\mathbf{q}(t) \quad \text{non-decreasing} \tag{A.16}$$

$$\mathbf{q}(0) = 0 \tag{A.17}$$

$$q(s,t) = 0 \tag{A.18}$$

where $\mathbf{b}(0)$ is called the *initial buffer level vector*. Machine M_s is the machine which is down for all $t \geq 0$. $\mathbf{I}$ and $\mathbf{N}$ are assumed to be feasible for the system.

Substituting (A.18) into (A.13), we use incidence matrix $\Phi_f(s)$ and cumulative flow vector $\mathbf{q}_s$ to represent (A.13). The whole system can be rewritten as follows:

For all $t \geq 0$

$$\mathbf{b}(t) = \mathbf{b}(0) + \Phi_f(s)^T \mathbf{q}_s(t) \tag{A.19}$$

$$\Psi \mathbf{b}(t) = \mathbf{I} \tag{A.20}$$

$$0 \leq \mathbf{b}(t) \leq \mathbf{N} \tag{A.21}$$

$$\mathbf{q}(t) \quad \text{non-decreasing} \tag{A.22}$$

$$\mathbf{q}(0) = 0 \tag{A.23}$$

$$q(s,t) = 0 \tag{A.24}$$

A.3 Problem Formulation

Given any $\mathbf{q}(t)$ that satisfies:

For all $t \geq 0$

$$\mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) \tag{A.25}$$

$$\Psi \mathbf{b}(t) = \mathbf{I} \tag{A.26}$$

$$0 \leq \mathbf{b}(t) \leq \mathbf{N} \tag{A.27}$$

$$\mathbf{q}(t) \quad \text{non-decreasing} \tag{A.28}$$

$$\mathbf{q}(0) = 0 \tag{A.29}$$

$$q(s,t) = 0 \tag{A.30}$$

$$\forall M_i, i \neq s \tag{A.31}$$

$$s.t. \quad b(w,t) < N_w, \forall w \in D(M_i) \quad \text{and}$$
$$b(v,t) > 0, \forall v \in U(M_i),$$
$$q(i,t) \text{ is strictly increasing.}$$

Note that $q(s,t) = 0$ for all $t \geq 0$ because machine M_s is down for all $t \geq 0$. Condition (A.31) says that, $q(i,t)$, the cumulative flow through M_i is strictly

increasing whenever M_i is neither starved nor blocked.

We will demonstrate the uniqueness property which is:

$\mathbf{b}(\infty|M_s) = \lim_{t\to\infty} \mathbf{b}(t)$ exists, is independent of $\mathbf{b}(0)$, and is unique.

A.4 Proof

We will prove the following lemmas and the theorem. Each lemma is required for a later lemma or the theorem. These are informal statements of them:

- **Lemma A.4.1**: In a static connected flow network with a single machine failure, any given machine's cumulative flow can be maximized.
- **Lemma A.4.2**: In a static connected flow network with a single machine failure, the cumulative flows of all the machines can be simultaneously maximized.
- **Lemma A.4.3**: In a static connected flow network with a single machine failure, when the cumulative flows of all the machines are simultaneously maximized, the corresponding buffer level vector is independent of the initial buffer level vector and unique.
- **Lemma A.4.4**: In a static connected flow network with a single machine failure, when all operational machines are blocked, starved, or both, then the cumulative flows of all the machines are maximized.
- **Theorem A.4.5**: In a dynamic connected flow network with a single machine failure, when time goes to infinity, the system must reach a limiting propagation state in which all the cumulative flows are simultaneously maximized and the corresponding buffer level vector reaches a limit which is independent of the initial buffer level vector and is unique.

Lemma A.4.1 *In a static connected flow network with single machine failure at M_s, we have $q(s) = 0$. For a given machine $M_i, i \neq s$, we define problem P_i as follows:*

$$
\begin{aligned}
P_i: \quad \max \quad & q(i) && \text{(A.32)}\\
s.t. \quad & \mathbf{b} = \mathbf{b}^0 + \Phi_f(s)^T \mathbf{q}_s && \text{(A.33)}\\
& \Psi \mathbf{b} = \mathbf{I} && \text{(A.34)}\\
& 0 \leq \mathbf{b} \leq \mathbf{N} && \text{(A.35)}\\
& \mathbf{q} \geq 0 && \text{(A.36)}\\
& q(s) = 0 && \text{(A.37)}
\end{aligned}
$$

Then $\max q(i)$ *exists and is bounded and unique.*

Proof For a given network Ω with n machines and m buffers, in problem P_i, we have $n + m - 1$ variables:

$$
\begin{aligned}
& q(i) \quad i = 1, ..., n; i \neq s\\
& b(j) \quad j = 1, ..., m.
\end{aligned}
$$

Notice that from equation (A.37), we have $q(s) = 0$.

The rank of the incidence matrix $\Phi_f(s)$ is $n-1$. Then the linear system (A.33) has $n-1$ independent equations. Since fundamental circuit matrix Ψ has rank $m-n+1$, the linear system (A.34) has $m-n+1$ independent equations. Rearrange (A.33) and write it together with (A.34):

$$
\begin{bmatrix} E & -\Phi_f(s)^T \\ \Psi & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{q}_s \end{bmatrix} = \begin{bmatrix} \mathbf{b}^0 \\ \mathbf{I} \end{bmatrix} \qquad \text{(A.38)}
$$

in which E is an identity matrix of order m.

There are m rows in matrix

$$
\begin{bmatrix} E & -\Phi_f(s)^T \\ \Psi & 0 \end{bmatrix}
$$

All the rows are linearly independent. Therefore, the number of independent equations in (A.38) is m. Since we have m independent equations in $n+m-1$ variables, we can choose any $n-1$ variables independently.

In network Ω, choose a spanning tree T. We will show that buffer level vector $\mathbf{b}_{\overline{T}}$ of $m-n+1$ buffers $B_{\overline{T}}$ is dependent on $\mathbf{b}_T$ and can be eliminated from (A.33)–(A.37).

- We divide the matrix Ψ into two matrices $\Psi(T)$ and $\Psi(\overline{T})$ corresponding to buffer sets B_T and $B_{\overline{T}}$, respectively. The invariant constraint can be written as follows:

$$\begin{bmatrix}\Psi(T) & \Psi(\overline{T})\end{bmatrix}\begin{bmatrix}\mathbf{b}_T \\ \mathbf{b}_{\overline{T}}\end{bmatrix} = \Psi(T)\mathbf{b}_T + \Psi(\overline{T})\mathbf{b}_{\overline{T}} = \mathbf{I} \tag{A.39}$$

  Consider the row vector $\mathbf{\Psi}_{k\star}(T)$ corresponds chord buffer B_k in $B_{\overline{T}}$. It is equal to a path vector which starts with B_k's downstream machine of $M_{d(k)}$ and ends with its upstream machine $M_{u(k)}$. Denote the path by $p(d(k), u(k))$. Therefore, from (A.1), we can write

$$\mathbf{\Psi}_{k\star}(T) = \mathbf{\Gamma}_{u(k)\star}(T,s) - \mathbf{\Gamma}_{d(k)\star}(T,s) \tag{A.40}$$

- According to the definition of *the set of fundamental circuits* with respect to T (Swamy 1981), matrix $\Psi_{\overline{T}}$ is a non-singular matrix of order $(m-n+1)$. Therefore, the inverse $\Psi_{\overline{T}}^{-1}$ exists. $\mathbf{b}_{\overline{T}}$ can be computed from $\mathbf{b}_T$ by using following equation:

$$\mathbf{b}_{\overline{T}} = \Psi(\overline{T})^{-1}[\mathbf{I} - \Psi(T)\mathbf{b}_T] \tag{A.41}$$

- Since $\mathbf{b}_{\overline{T}}$ satisfies the buffer size constraint $\mathbf{0} \le \mathbf{b}_{\overline{T}} \le \mathbf{N}_{\overline{T}}$, we have

$$\Psi(\overline{T})^{-1}[\mathbf{I} - \Psi(T)\mathbf{b}_T] \le \mathbf{N}_{\overline{T}} \tag{A.42}$$

$$\Psi(\overline{T})^{-1}[\mathbf{I} - \Psi(T)\mathbf{b}_T] \ge \mathbf{0} \tag{A.43}$$

We manipulate (A.42) and (A.43) and simplify them as follows:

$$\xi_k^L \leq \mathbf{\Psi}_{k\star}(T)\mathbf{b}_T \leq \xi_k^U \qquad k = 1, ..., m-n+1 \tag{A.44}$$

where ξ_k^L and ξ_k^U are respectively the lower and upper bounds of $\mathbf{\Psi}_{k\star}(T)\mathbf{b}_T$ derived from (A.42) and (A.43).

Substitute (A.40) into (A.53),

$$\xi_k^L \leq \mathbf{\Gamma}_{u(k)\star}(T,s)\mathbf{b}_T - \mathbf{\Gamma}_{d(k)\star}(T,s)\mathbf{b}_T \leq \xi_k^U \qquad k = 1, ..., m-n+1 \tag{A.45}$$

Now, we will show that buffer level vector $\mathbf{b}_T$ of $n-1$ buffers B_T can be represented by cumulative flow vector $\mathbf{q}_s$.

- Recall the relationship between $\Gamma(T,s)$ and $\Phi_f(T,s)$ described in (A.2). Rearrange (A.2) and we have

$$\Phi_f(T,s)^T = -\Gamma(T,s)^{-1} \tag{A.46}$$

Substituting (A.46) into (A.33), we have

$$\mathbf{b}_T = \mathbf{b}_T^0 - \Gamma(T,s)^{-1}\mathbf{q}_s \tag{A.47}$$

Manipulate (A.47) and we get

$$\mathbf{q}_s = \Gamma(T,s)(-\mathbf{b}_T + \mathbf{b}_T^0) \tag{A.48}$$

Therefore, the cumulative flow of machine $M_k, k \neq s$ is:

$$\begin{aligned} q(i) &= \mathbf{\Gamma}_{i\star}(T,s)(-\mathbf{b}_T + \mathbf{b}_T^0) \\ &= \textstyle\sum_{j\in T} \gamma_{ij}(T,s)(-b(j) + b^0(j)) \qquad i = 1, ..., n; i \neq s \end{aligned} \tag{A.49}$$

Notice that $\gamma_{ij}(T,s)$ is non-zero when buffer B_j is within path $p(s,k)$. Therefore, we can write

$$q(i) = \sum_{j \in p(s,i)} \gamma_{ij}(T,s)(-b(j) + b^0(j)) \qquad i = 1, ..., n; i \neq s \tag{A.50}$$

- Since $\mathbf{b}_T$ satisfies the buffer size constraint $\mathbf{0} \leq \mathbf{b}_T \leq \mathbf{N}_T$, we have

$$\mathbf{b}_T^0 - \Gamma(T,s)^{-1}\mathbf{q}_s \leq \mathbf{N}_T \tag{A.51}$$

$$\mathbf{b}_T^0 - \Gamma(T,s)^{-1}\mathbf{q}_s \geq \mathbf{0} \tag{A.52}$$

We manipulate (A.51) and (A.52) and simplify them as follows:

$$\eta_i^L \leq q(i) \leq \eta_i^U \qquad i = 1, ..., n; i \neq s \tag{A.53}$$

where η_i^L and η_i^U are respectively the lower and upper bounds of $q(i)$ derived from (A.51) and (A.52).

- Manipulate (A.49) and we can write

$$\mathbf{\Gamma}_{i\star}(T,s)\mathbf{b}_T = -q(i) + \mathbf{\Gamma}_{i\star}(T,s)\mathbf{b}_T^0 \qquad i = 1, ..., n; i \neq s \tag{A.54}$$

When $i = u(k)$ and $d(k)$, substitute (A.54) into (A.45):

$$\xi_k^L \leq \big(-q(u(k)) + \mathbf{\Gamma}_{u(k)\star}(T,s)\mathbf{b}_T^0 \big) - \big(-q(d(k)) + \mathbf{\Gamma}_{u(k)\star}(T,s)\mathbf{b}_T^0 \big) \leq \xi_k^U \qquad k = 1, ..., m-n+1 \tag{A.55}$$

To simplify, we can write

$$\zeta_k^L \leq q(u(k)) - q(d(k)) \leq \zeta_k^U \qquad k = 1, ..., m-n+1 \tag{A.56}$$

where ζ_k^L and ζ_k^U are the lower and upper bounds of $q(u(k)) - q(d(k))$ respectively.

Therefore, we can choose cumulative flow vector $\mathbf{q}_s = [\ q(1), ..., q(s-1), q(s+1), ..., q(n)\ ]$ to be the set of independent variables of P_i. After eliminating dependent variables $\mathbf{b}_T$ and $\mathbf{b}_{\overline{T}}$, problem P_i can be rewritten as

$$
\begin{aligned}
P_i' : \quad \max \quad & q(i) && && \text{(A.57)}\\
\text{s.t.} \quad & \eta_j^L \le q(j) \le \eta_j^U \qquad j = 1, ..., n; j \ne s && && \text{(A.58)}\\
& \zeta_k^L \le q(u(k)) - q(d(k)) \le \zeta_k^U \qquad k = 1, ..., m-n+1 && && \text{(A.59)}
\end{aligned}
$$

Since cumulative flows $q(j), j = 1, ..., n; j \ne s$ are bounded in a convex set C given by constraints (A.58) and (A.59) in problem P_i', cumulative flow $q(i)$, as a linear function on C, can be maximized at a unique optimum:

$$q^*(i) = \max q(i) \tag{A.60}$$

where $q^*(i)$ is the optimum of P_i.

Therefore, $q^*(i)$ exists and is bounded and unique. Let $\mathbf{q}^*$ be an optimal solution of P_i'. Components $q^*(j), j = 1, ..., n; j \ne s, n$ satisfy constraints (A.58) and (A.59). Corresponding optimal buffer level vectors $\mathbf{b}_T^*$ and $\mathbf{b}_{\overline{T}}^*$ can be determined using (A.47) and (A.41). Notice that the solution $(\mathbf{b}^*, \mathbf{q}^*)$ of P_i is not unique.

q.e.d.

In Lemma A.4.2, we show that $q^*(l)$, the unique optimum of $P_l, l \neq i$, is part of a vector that is a solution of P_i.

Lemma A.4.2 *Consider a static connected flow network with a single machine failure at M_s. For given machines M_i and $M_l, i \neq s, l \neq s$, let $q_i^*(i)$, $q_l^*(l)$ be the optimums of problems P_i and P_l. Then there exists $\mathbf{q}'$, $\mathbf{b}'$ that satisfy P_i such that:*

$$\begin{aligned} q'(i) &= q_i^*(i) \\ q'(l) &= q_l^*(l) \end{aligned}$$

That is, $(\mathbf{q}', \mathbf{b}')$ is an optimal solution for both P_i and P_l.

Proof We first write problem P_l':

$$\begin{aligned} P_l': \quad \max \quad & q(l) && && \text{(A.61)} \\ \text{s.t.} \quad & \eta_j^L \leq q(j) \leq \eta_j^U && j = 1, ..., n; j \neq s && \text{(A.62)} \\ & \zeta_k^L \leq q(u(k)) - q(d(k)) \leq \zeta_k^U && k = 1, ..., m - n + 1 && \text{(A.63)} \end{aligned}$$

Consider constraint (A.63). This is the only relationship between any pair of cumulative flows. We will show that there is no trade-off between any pair of cumulative flows. That is, increasing one cumulative flow does not require decreasing any other cumulative flows.

Move $q(d(k))$ in (A.63) and we have

$$\zeta_k^L + q(d(k)) \leq q(u(k)) \leq \zeta_k^U + q(d(k)) \qquad k = 1, ..., m - n + 1 \qquad \text{(A.64)}$$

Therefore, the upper bound of $q(u(k))$ increases with $q(d(k))$.

Then, we move $q(u(k))$ in (A.63):

$$q(u(k)) - \zeta_k^U \leq q(d(k)) \leq q(u(k)) - \zeta_k^L \qquad k = 1, ..., m - n + 1 \tag{A.65}$$

Therefore, the upper bound of $q(d(k))$ increases with $q(u(k))$.

In summary, there is no trade-off between any pair of cumulative flows in constraint (A.63). Increasing one cumulative flow does not require decreasing any other cumulative flows.

Now we consider two cases: when $q(l)$ is not present in constraint (A.63), and when $q(l)$ is present in constraint (A.63).

Case 1: $q(l)$ is not present in constraint (A.63)

In this case, M_l is not the upstream or downstream machine of any chord buffer. $q(l)$ is only constrained by (A.62) and is independent of all other cumulative flows. Therefore, $q(l)$ can be maximized when $q(i) = q_i^*(i)$. That is, there exists an optimal solution $\mathbf{q}'$ of P_l' which satisfies

$$\begin{aligned} q'(i) &= q_i^*(i) \\ q'(l) &= q_l^*(l) \end{aligned}$$

Corresponding buffer level vectors $\mathbf{b}_T'$ and $\mathbf{b}_{\overline{T}}'$ can be determined using (A.47) and (A.41).

Case 2: $q(l)$ is present in constraint (A.63)

In this case, M_l is the upstream or downstream machine of a chord buffer. If $q(i)$ does not present in (A.63), $q(l)$ can be maximized independently. If $q(i)$ presents in (A.63), $q(l)$ can be maximized when $q(i)$ is maximized, i.e. $q(i) = q_i^*(i)$. Therefore, there exists an optimal solution $\mathbf{q}'$ of P_l' which satisfies

$$q'(i) = q_i^*(i)$$
$$q'(l) = q_l^*(l)$$

Corresponding buffer level vectors $\mathbf{b}'_T$ and $\mathbf{b}'_{\overline{T}}$ can be determined using (A.47) and (A.41).

In summary, there exists $\mathbf{q}'_s$, $\mathbf{b}'$ which is an optimal solution for both P_i and P_l.

q.e.d.

Corollary Problems $P_i, i = 1, ..., n; i \neq s$ can be simultaneously maximized.

Proof We use an induction method to construct an optimal cumulative flow vector $\mathbf{q}'_s$ such that $P_i, i = 1, ..., n; i \neq s$ are all optimized.

- First solve problem P'_1 as in Lemma A.4.1 to obtain an optimal solution $\mathbf{q}_s^{(1)}$ which satisfies:

$$q^{(1)}(1) = q_1^*(1) \tag{A.66}$$

- Suppose in the kth step, we construct a cumulative flow vector $\mathbf{q}_s^{(k)}$ such that $P'_1, P'_2, ..., P'_k$ are satisfied:

$$q^{(k)}(i) = q_i^*(i) \qquad i = 1, 2, ..., k \tag{A.67}$$

- In the $(k+1)$th step, to construct a cumulative flow vector $\mathbf{q}_s^{(k+1)}$ to optimize $P'_i, i = 1, 2, ..., k+1$, we solve problem P'_{k+1} with the conditions:

$$q(i) = q^{(k)}(i) \qquad i = 1, 2, ..., k \tag{A.68}$$

From Lemma A.4.2, we can optimize $q(k+1)$ while satisfying (A.68). Therefore, we can construct $\mathbf{q}_s^{(k+1)}$ which satisfies

$$\begin{aligned} q^{(k+1)}(i) &= q_i^*(i) \qquad i = 1, ..., k \\ q^{(k+1)}(k+1) &= q_{k+1}^*(k+1) \end{aligned} \tag{A.69}$$

That is, (A.67) implies (A.69).

- When $k = n - 1$: from Lemma A.4.2, there exists a cumulative flow vector $\mathbf{q}_s^{(n)}$ which satisfies

$$q^{(n)}(i) = q_i^*(i) \qquad i = 1, ..., n; j \neq s \tag{A.70}$$

Let $\mathbf{q}'_s = \mathbf{q}_s^{(n)}$ and calculate corresponding buffer level vector $\mathbf{b}'$ using (A.47) and (A.41). In conclusion, $(\mathbf{q}'_s, \mathbf{b}')$ is an optimal solution such that $P_i, i = 1, ..., n; i \neq s$ are all optimized.

q.e.d.

Lemma A.4.3 *Given problem VOP (Vector optimization problem):*

$$
\begin{aligned}
VOP: \quad \mathbf{q}^* = \max \quad & q(1), q(2), ..., q(n) \\
s.t. \quad & \mathbf{b} = \mathbf{b}^0 + \Phi_f(s)^T \mathbf{q}_s \\
& \Psi \mathbf{b} = \mathbf{I} \\
& 0 \leq \mathbf{b} \leq \mathbf{N} \\
& q(s) = 0 \\
& q(j) \geq 0, j = 1, ..., n \quad j \neq s
\end{aligned}
$$

The optimal solution $\mathbf{q}^*$ *exists and the corresponding buffer level vector* $\mathbf{b}^*$ *is independent of* $\mathbf{b}_0$ *and is unique.*

Proof Recall the relationship between cumulative flows and buffer levels in Lemma A.4.1 (A.50):

$$
q(i) = \sum_{j \in p(s,i)} \gamma_{ij}(T, s)(-b(j) + b^0(j)) \qquad i = 1, ..., n; i \neq s
$$

Notice that maximizing $q(i)$ is equivalent to minimizing a specific linear combination of the buffer levels:

$$
\max q(i) \iff \min \sum_{j \in p(s,i)} \gamma_{ij}(T, s) b(j) \tag{A.71}
$$

Therefore, the objectives of problem VOP can be represented in buffer levels and the constraint of flow conservation can be eliminated. After eliminating the dependent variables $\mathbf{b}_{\overline{T}}$ as in Lemma A.4.1, we can rewrite problem VOP as follows:

$$\text{VOP2:} \quad \min \quad \sum_{j \in p(s,i)} \gamma_{ij}(T,s)b(j) \qquad i = 1, ..., n; i \neq s \tag{A.72}$$

$$\text{s.t.} \quad \Psi_{\overline{T}}^{-1}[\mathbf{I} - \Psi_T \mathbf{b}_T] \leq \mathbf{N}_{\overline{T}} \tag{A.73}$$

$$\Psi_{\overline{T}}^{-1}[\mathbf{I} - \Psi_T \mathbf{b}_T] \geq \mathbf{0} \tag{A.74}$$

$$0 \leq \mathbf{b}_T \leq \mathbf{N}_T \tag{A.75}$$

From Lemmas A.4.1 and A.4.2, we know that all the objectives of VOP can be simultaneously maximized at unique global maximums. Therefore, all the objectives of VOP2 can be simultaneously minimized at unique global minimums.

Let the solution of VOP2 be the buffer level vector $\mathbf{b}_T^*$. It has two properties:

- *Independence*: As there is no initial buffer level vector $\mathbf{b}^0$ in problem VOP2, so $\mathbf{b}_T^*$ is independent of $\mathbf{b}^0$.
- *Uniqueness*: Denote the unique minimal objective values by

$$\mathbf{p}^* = [p_1^*, ..., p_{s-1}^*, p_{s+1}^*, ..., p_n^*]^T \tag{A.76}$$

We can write the following equation:

$$\mathbf{p}^* = \Gamma(T,s)\mathbf{b}_T^* \tag{A.77}$$

Recall that all row vectors $\mathbf{\Gamma}_{i\star}(T,s), i = 1, ..., n; i \neq s$ are linearly independent. Therefore, matrix $\Gamma(T,s)$ is a non-singular matrix of order $n-1$. A unique solution $\mathbf{b}_T^*$ can be solved from (A.77).

Finally, buffer level vector $\mathbf{b}_{\overline{T}}^*$ of buffer set $B_{\overline{T}}$ can be uniquely determined from $\mathbf{b}_T^*$ using (A.41) in Lemma A.4.1.

Incidentally, optimal cumulative flow vector $\mathbf{q}_s^*$ can be determined using (A.4). However, it depends on the initial buffer level vector $\mathbf{b}^0$.

q.e.d.

Lemma A.4.4 *In a static flow network with single machine failure at M_s, for any operational machine M_i such that*

$$\exists w \in D(M_i) \qquad b(w) = N_w$$

and/or

$$\exists v \in U(M_i) \qquad b(v) = 0$$

Then the cumulative flows of all the machines are maximized:

$$q(i) = q^*(i) \qquad \forall i \neq s$$

Proof Suppose all the machines are in one of the three states: blocked, starved, or simultaneously blocked and starved. According to the induction steps of the corollary of Lemma A.4.2, it is feasible to construct $\hat{\mathbf{q}}, \hat{\mathbf{b}}$ such that there exists only one machine M_g for which

$$\hat{q}(g) < q^*(g) \qquad g \neq s \tag{A.78}$$

$$\hat{q}(i) = q^*(i) \qquad \forall i \neq g, s \tag{A.79}$$

Since $\hat{q}(g)$ is not maximized, define $\Delta q(g)$:

$$\Delta q(g) = q^*(g) - \hat{q}(g) > 0$$

Notice that the cumulative flows of all other machines are maximized. Therefore

$$\Delta q(i) = 0, i \neq g \tag{A.80}$$

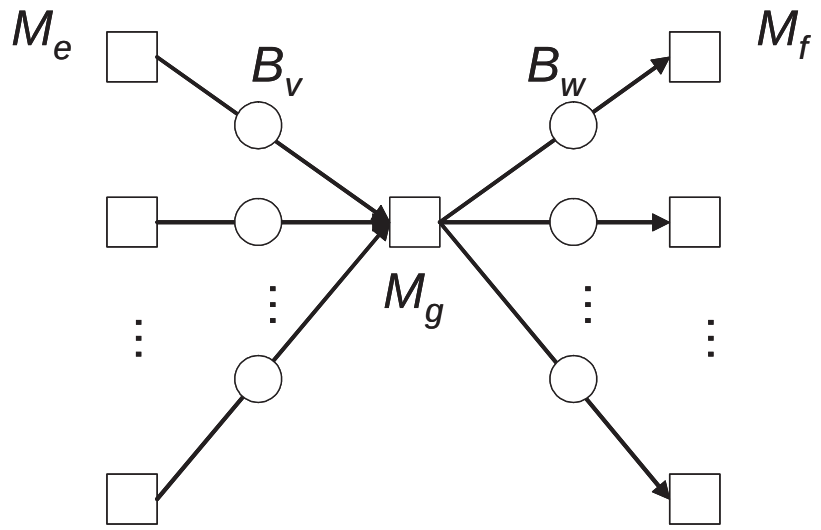


Figure A-1: Machine M_g and its upstream and downstream buffers and machines

- Consider one of the downstream buffers of M_g (Figure A-1). For a given downstream buffer B_w, denote its downstream machine by M_f. According to flow conservation, on path $p(g, f)$:

$$\Delta q(f) - \Delta q(g) = -b^*(w) + \hat{b}(w) \qquad \forall w \in D(M_g) \tag{A.81}$$

where $b^*(w)$ is the corresponding buffer level of B_w when $q(g) = q^*(g)$.

According to (A.80), $\Delta q(f) = 0$, then we have

$$\Delta q(g) = b^*(w) - \hat{b}(w) > 0 \quad \forall w \in D(M_g) \tag{A.82}$$

which implies

$$b^*(w) > \hat{b}(w) \quad \forall w \in D(M_g) \tag{A.83}$$

To satisfy the buffer constraint

$$\hat{b}(w) < b^*(w) \leq N_w \quad \forall w \in D(M_g) \tag{A.84}$$

Therefore, machine M_g is not blocked since $\hat{b}(w) < N_w, \forall w \in D(M_g)$.

- Consider one of the upstream buffers of M_g (Figure A-1). For a given upstream buffer B_v, denote its upstream machine by M_e. According to flow conservation, on path $p(g, e)$:

$$\Delta q(e) - \Delta q(g) = b^*(v) - \hat{b}(v) \qquad \forall v \in U(M_g) \tag{A.85}$$

According to (A.80), $\Delta q(h) = 0$, then we have

$$\Delta q(g) = -b^*(v) + \hat{b}(v) > 0 \quad \forall v \in U(M_g) \tag{A.86}$$

which implies

$$b^*(v) < \hat{b}(v) \quad \forall v \in U(M_g) \tag{A.87}$$

To satisfy the buffer constraint

$$\hat{b}(v) < b^*(v) \leq N_v \quad \forall v \in U(M_g) \tag{A.88}$$

Therefore, machine M_g is not starved since $\hat{b}(v) > 0, \forall v \in U(M_g)$.

In summary

$$\begin{aligned} &\hat{b}(w) < b^*(w) \leq N_w \quad \forall w \in D(M_g) \\ &\hat{b}(v) > b^*(v) \geq 0 \qquad \forall v \in U(M_g) \end{aligned}$$

Therefore, machine M_g is not starved and not blocked. This contradicts the assumption: all the machines are in one of the three states: blocked, starved, or simultaneously blocked and starved.

q.e.d.

Corollary In a static connected flow network with single machine failure, given machine M_i

$$q(i) < q^*(i) \qquad i \neq s$$

Then there must exist $M_g, g \neq s$ such that

$$\begin{aligned} b(w) &< N_w \quad \forall w \in D(M_g) \\ b(v) &> 0 \qquad \forall v \in U(M_g) \end{aligned}$$

Theorem A.4.5 *Given a single machine failure at M_s and an initial buffer level vector $\mathbf{b}(0)$, consider problem LT:*

$$
\begin{aligned}
LT: \lim_{t\to\infty} \quad & \mathbf{q}(t), \mathbf{b}(t) \\
s.t. \quad & \mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) \\
& \Psi \mathbf{b}(t) = \mathbf{I} \\
& 0 \le \mathbf{b}(t) \le \mathbf{N} \\
& q(s,t) = 0 \qquad t \ge 0 \\
& \mathbf{q}(t) \quad \textit{non-decreasing} \\
& \textit{for} \quad t \ge 0 \\
& \forall M_i, i \ne s \\
& \quad s.t. \quad b(w,t) < N_w, \forall w \in D(M_i) \\
& \qquad\quad b(v,t) > 0, \forall v \in U(M_i) \\
& \textit{Then } q(i,t) \textit{ is strictly increasing.}
\end{aligned}
$$

Then we have

- $\mathbf{q}(\infty|M_s) = \lim_{t\to\infty} \mathbf{q}(t)$ *exists and is unique.* $\mathbf{q}(\infty|M_s) = \mathbf{q}^*$.
- $\mathbf{b}(\infty|M_s) = \lim_{t\to\infty} \mathbf{b}(t)$ *exists, is independent of* $\mathbf{b}(0)$*, and is unique.* $\mathbf{b}(\infty|M_s) = \mathbf{b}^*$.

Proof Choose a spanning tree T which contains $n-1$ buffers B_T with buffer level vector $\mathbf{b}_T(0)$. Given a machine $M_i, i \ne s$, identify path $p(s,i)$ from M_s to M_i in T. Using (A.50) in Lemma A.4.1 and considering time scale, the cumulative flow $q(i,t)$ is:

$$
\begin{aligned}
q(i,t) &= \textstyle\sum_{j\in p(s,i)} \gamma_{ij}(T,s)(-b(j,t) + b(j,0)) \\
&= \textstyle\sum_{j\in p(s,i)} \gamma_{ij}(T,s)(-b(j,t)) + \sum_{j\in p(s,i)} \gamma_{ij}(T,s) b(j,0) \qquad t \ge 0
\end{aligned}
\tag{A.89}
$$

Because $q(i,t)$ is non-decreasing and buffer levels $b(j,t)$ are bounded, $\lim_{t\to\infty} q(i,t)$ exists.

Then we will prove

$$\lim_{t\to\infty} \mathbf{q}(t) = \mathbf{q}^* \tag{A.90}$$

Suppose at time t', the cumulative flows of all the machines have reached their limit values but there exists machine M_i whose limit value of $q(i,t')$ is smaller than $q^*(i)$. From the corollary of Lemma A.4.4, there must exist a machine M_g which is not blocked and not starved. Given the assumption that $q(g,t)$ is strictly increasing when M_g is not starved and not blocked, the cumulative flow of M_g will increase with time. Then $q(g,t')$ is not its limit value. Contradiction.

Therefore, the limit of $q(i,t)$ is its maximal value:

$$\lim_{t\to\infty} q(i,t) = \max q(i,t) = q^*(i) \tag{A.91}$$

When $\mathbf{b}(0) = \mathbf{b}^0$, problem LT is equivalent to VOP in Lemma A.4.3. According to Lemmas A.4.2 and A.4.3, $q(i,t), i = 1,...,n; i \neq s$ can be maximized simultaneously. The limit values $\mathbf{q}(\infty|M_s)$ and corresponding buffer level vector $\mathbf{b}(\infty|M_s)$ are unique and satisfy:

$$\mathbf{q}(\infty|M_s) = \lim_{t\to\infty} \mathbf{q}(t) = \mathbf{q}^* \tag{A.92}$$

$$\mathbf{b}(\infty|M_s) = \lim_{t\to\infty} \mathbf{b}(t) = \mathbf{b}^* \tag{A.93}$$

where $\mathbf{b}(\infty|M_s)$ is independent of $\mathbf{b}(0)$.

q.e.d.

Appendix B

Examples of Blocking and Starvation Properties

B.1 Example I: a 15-machine 4-loop assembly / disassembly network (Figure B-1)

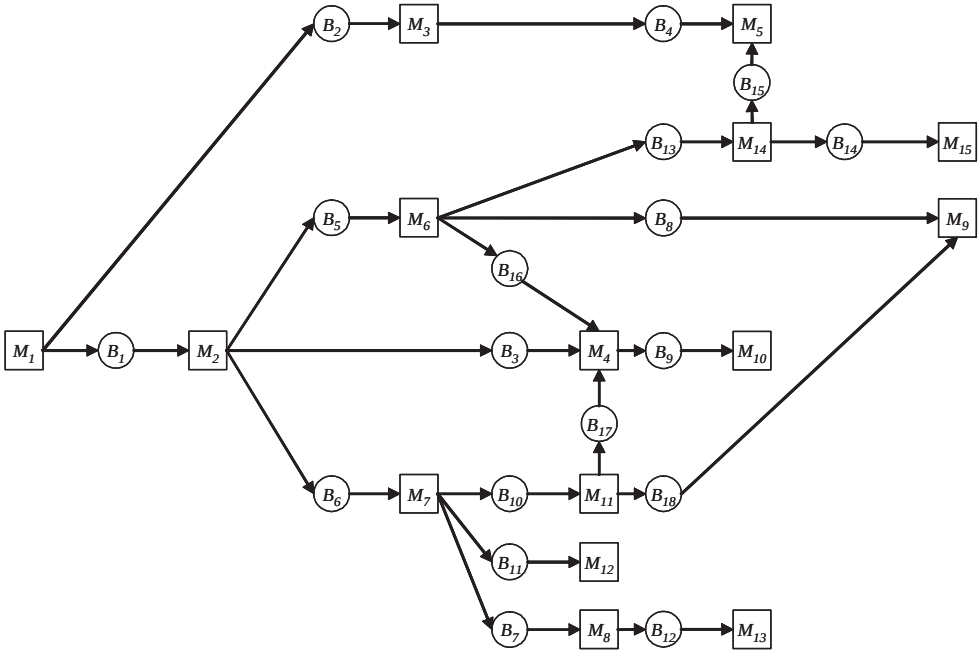


Figure B-1: A 15-machine 4-loop assembly/disassembly network

ID of Buffer	ID of upstream machine	Buffer size	ID of downstream machine
1	1	30	2
2	1	7	3
3	2	35	4
4	3	26	5
5	2	28	6
6	2	11	7
7	7	24	8
8	6	31	9
9	4	33	10
10	7	36	11
11	7	10	12
12	8	25	13
13	6	28	14
14	14	10	15
15	14	30	5
16	6	8	4
17	11	26	4
18	11	31	9

ID of Loop	Invariance equation
1	$b(15,t) - b(4,t) - b(2,t) + b(1,t) + b(5,t) + b(13,t) = 62$
2	$b(16,t) - b(3,t) + b(5,t) = -3$
3	$b(17,t) - b(3,t) + b(6,t) + b(10,t) = 6$
4	$b(18,t) - b(8,t) - b(5,t) + b(6,t) + b(10,t) = -5$

$$\Theta = \begin{bmatrix}
0 & 0 & 7 & 0 & 4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 28 & 0 & 30 & 0 & 12 & 0 \\
30 & 0 & 3 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 & 0 & 30 & 0 & 8 & 0 \\
0 & 7 & 14 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 28 & 0 & 30 & 0 & 19 & 6 \\
30 & 0 & 35 & 0 & 24 & 0 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 0 & 8 & 8 & 26 & 5 \\
9 & 7 & 31 & 26 & 28 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 28 & 0 & 30 & 0 & 26 & 13 \\
30 & 0 & 31 & 0 & 28 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 4 & 0 & 26 & 13 \\
30 & 0 & 6 & 0 & 0 & 11 & 0 & 16 & 0 & 0 & 0 & 0 & 2 & 0 & 30 & 3 & 0 & 0 \\
30 & 0 & 6 & 0 & 0 & 11 & 24 & 16 & 0 & 0 & 0 & 0 & 2 & 0 & 30 & 3 & 0 & 0 \\
30 & 0 & 31 & 0 & 28 & 0 & 0 & 31 & 0 & 23 & 0 & 0 & 0 & 0 & 4 & 0 & 13 & 31 \\
30 & 0 & 35 & 0 & 24 & 0 & 0 & 0 & 33 & 14 & 0 & 0 & 0 & 0 & 8 & 8 & 26 & 5 \\
30 & 0 & 35 & 0 & 24 & 4 & 0 & 21 & 0 & 36 & 0 & 0 & 0 & 0 & 8 & 8 & 0 & 0 \\
30 & 0 & 6 & 0 & 0 & 11 & 0 & 16 & 0 & 0 & 10 & 0 & 2 & 0 & 30 & 3 & 0 & 0 \\
30 & 0 & 6 & 0 & 0 & 11 & 24 & 16 & 0 & 0 & 0 & 25 & 2 & 0 & 30 & 3 & 0 & 0 \\
30 & 0 & 31 & 24 & 28 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 28 & 0 & 0 & 0 & 26 & 13 \\
30 & 0 & 31 & 24 & 28 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 28 & 10 & 0 & 0 & 26 & 13
\end{bmatrix}$$

B.2 Example II: an 18-machine 8-loop assembly / disassembly network (Figure B-2)

ID of Buffer	ID of upstream machine	Buffer size	ID of downstream machine
1	1	25	2
2	1	26	3
3	3	2	4
4	2	22	5
5	5	21	6
6	6	14	7
7	4	9	8
8	2	44	9
9	8	14	10
10	9	45	11
11	9	46	12
12	11	38	13
13	6	40	14
14	11	3	15
15	12	7	16
16	4	20	17
17	7	39	18
18	4	28	2
19	6	22	1
20	16	22	15
21	15	36	13
22	1	15	8
23	9	45	7
24	7	35	12
25	8	45	5

ID of Loop	Invariance equation
1	$b(18,t) - b(1,t) + b(2,t) + b(3,t) = 1$
2	$b(19,t) + b(1,t) + b(4,t) + b(5,t) = 44$
3	$b(20,t) - b(14,t) - b(10,t) + b(11,t) + b(15,t) = 8$
4	$b(21,t) - b(12,t) + b(14,t) = -15$
5	$b(22,t) - b(7,t) - b(3,t) - b(2,t) = -15$
6	$b(23,t) - b(6,t) - b(5,t) - b(4,t) + b(8,t) = 3$
7	$b(24,t) - b(11,t) - b(8,t) + b(4,t) + b(5,t) + b(6,t) = -27$
8	$b(25,t) - b(4,t) - b(1,t) + b(2,t) + b(3,t) + b(7,t) = 3$

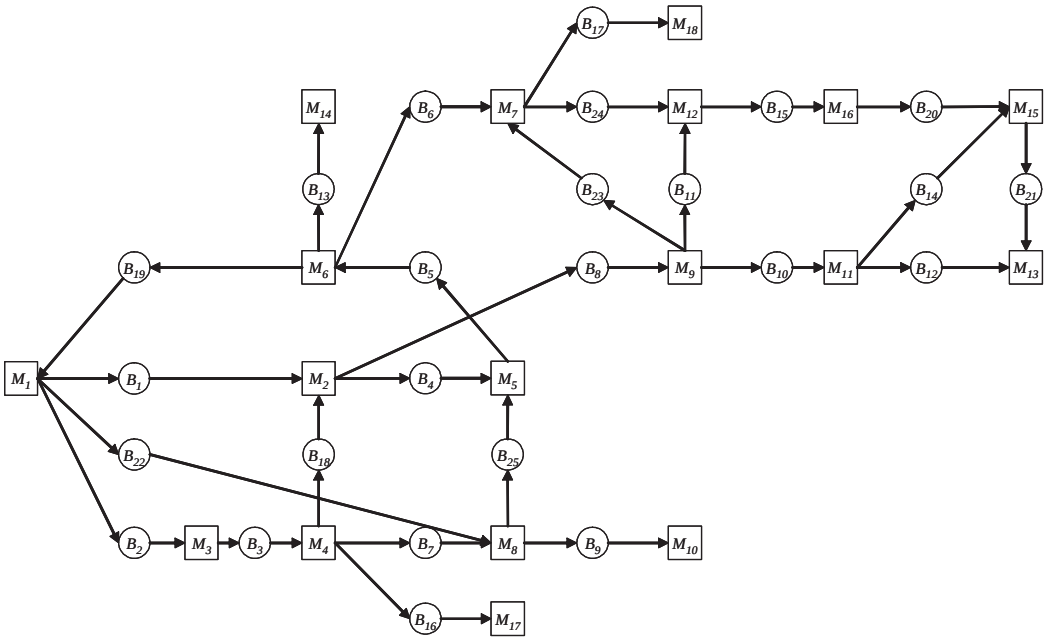


Figure B-2: An 18-machine 8-loop assembly/disassembly network

$$\Theta = \begin{bmatrix}
5 & 4 & 2 & 7 & 10 & 0 & 9 & 0 & 0 & 33 & 44 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 22 & 0 & 0 & 0 & 20 & 0 & 0 \\
25 & 4 & 2 & 0 & 0 & 0 & 9 & 0 & 0 & 16 & 27 & 18 & 0 & 3 & 0 & 0 & 0 & 20 & 19 & 0 & 0 & 0 & 3 & 0 & 13 \\
25 & 26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 27 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 11 & 3 & 0 & 2 \\
25 & 24 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 27 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 11 & 3 & 0 & 2 \\
22 & 4 & 2 & 22 & 0 & 0 & 9 & 3 & 0 & 35 & 46 & 18 & 0 & 3 & 0 & 0 & 0 & 17 & 0 & 0 & 0 & 0 & 22 & 0 & 32 \\
5 & 4 & 2 & 18 & 21 & 0 & 9 & 20 & 0 & 35 & 46 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 0 & 11 \\
5 & 4 & 2 & 18 & 21 & 14 & 9 & 34 & 0 & 35 & 46 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 0 & 11 \\
20 & 19 & 2 & 7 & 0 & 0 & 9 & 0 & 0 & 23 & 34 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 17 & 0 & 0 & 15 & 10 & 0 & 0 \\
17 & 4 & 2 & 6 & 21 & 14 & 9 & 44 & 0 & 13 & 24 & 18 & 0 & 3 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 11 \\
20 & 19 & 2 & 7 & 0 & 0 & 9 & 0 & 14 & 23 & 34 & 18 & 0 & 3 & 0 & 0 & 0 & 0 & 17 & 0 & 0 & 15 & 10 & 0 & 0 \\
17 & 4 & 2 & 6 & 21 & 14 & 9 & 44 & 0 & 45 & 24 & 15 & 0 & 0 & 7 & 0 & 0 & 12 & 0 & 22 & 0 & 0 & 0 & 0 & 11 \\
17 & 4 & 2 & 6 & 21 & 14 & 9 & 44 & 0 & 35 & 46 & 18 & 0 & 3 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 22 & 11 \\
17 & 4 & 2 & 6 & 21 & 14 & 9 & 44 & 0 & 45 & 24 & 38 & 0 & 0 & 7 & 0 & 0 & 12 & 0 & 22 & 23 & 0 & 0 & 0 & 11 \\
5 & 4 & 2 & 18 & 21 & 0 & 9 & 20 & 0 & 35 & 46 & 18 & 40 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 0 & 11 \\
17 & 4 & 2 & 6 & 21 & 14 & 9 & 44 & 0 & 45 & 27 & 18 & 0 & 3 & 7 & 0 & 0 & 12 & 0 & 22 & 0 & 0 & 0 & 3 & 11 \\
17 & 4 & 2 & 6 & 21 & 14 & 9 & 44 & 0 & 42 & 46 & 18 & 0 & 3 & 7 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 22 & 11 \\
25 & 24 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 27 & 18 & 0 & 3 & 0 & 20 & 0 & 0 & 19 & 0 & 0 & 11 & 3 & 0 & 2 \\
5 & 4 & 2 & 18 & 21 & 14 & 9 & 34 & 0 & 35 & 46 & 18 & 0 & 3 & 0 & 0 & 39 & 0 & 0 & 0 & 0 & 0 & 22 & 0 & 11
\end{bmatrix}$$

B.3 Example III: a 20-machine 10-loop assembly / disassembly network (Figure B-3)

ID of Buffer	ID of upstream machine	Buffer size	ID of downstream machine
1	1	19	2
2	2	34	3
3	2	38	4
4	3	7	5
5	2	37	6
6	1	5	7
7	7	37	8
8	4	16	9
9	7	29	10
10	7	21	11
11	11	7	12
12	11	30	13
13	5	26	14
14	4	13	15
15	3	3	16
16	8	5	17
17	7	36	18
18	16	24	19
19	1	29	20
20	11	2	18
21	3	8	7
22	4	8	16
23	4	2	6
24	20	2	2
25	12	34	17
26	11	6	14
27	17	30	16
28	16	34	9
29	2	28	5

ID of Loop	Invariance equation
1	$b(20,t) - b(17,t) + b(10,t) = -8$
2	$b(21,t) - b(6,t) + b(1,t) + b(2,t) = 29$
3	$b(22,t) - b(15,t) - b(2,t) + b(3,t) = 24$
4	$b(23,t) - b(5,t) + b(3,t) = 6$
5	$b(24,t) - b(1,t) + b(19,t) = 17$
6	$b(25,t) - b(16,t) - b(7,t) + b(10,t) + b(11,t) = 20$
7	$b(26,t) - b(13,t) - b(4,t) - b(2,t) - b(1,t) + b(6,t) + b(10,t) = -36$
8	$b(27,t) - b(15,t) - b(2,t) - b(1,t) + b(6,t) + b(7,t) + b(16,t) = -19$
9	$b(28,t) - b(8,t) - b(3,t) + b(2,t) + b(15,t) = -9$
10	$b(29,t) - b(4,t) - b(2,t) = -2$

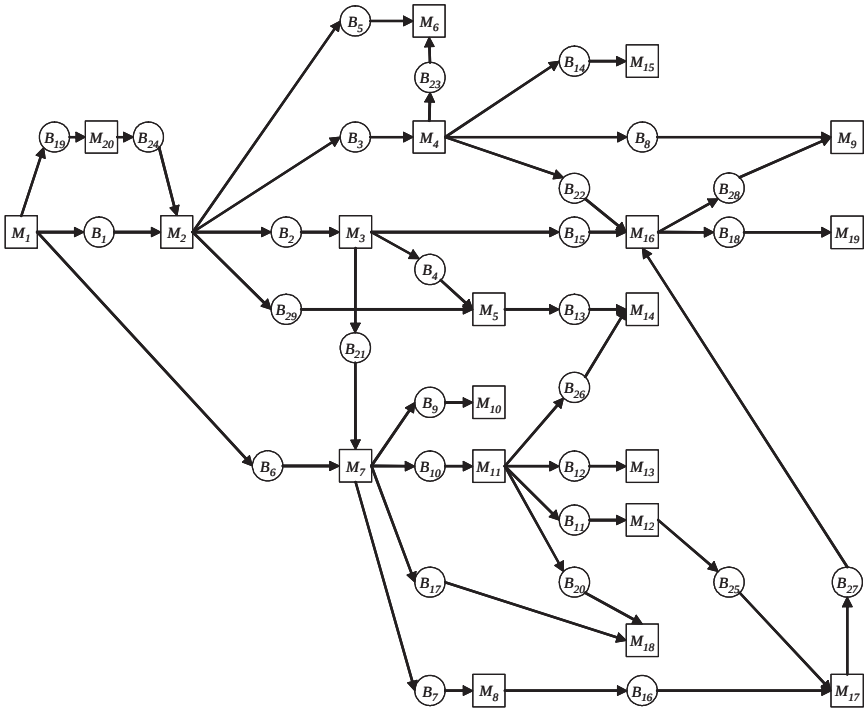


Figure B-3: A 20-machine 10-loop assembly/disassembly network

$$\Theta = \begin{bmatrix}
0 & 21 & 37 & 0 & 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 8 & 0 & 15 & 0 & 8 & 8 & 0 & 2 & 20 & 0 & 2 & 7 & 19 \\
14 & 7 & 23 & 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 8 & 0 & 29 & 0 & 8 & 8 & 0 & 2 & 20 & 0 & 2 & 7 & 5 \\
12 & 22 & 38 & 0 & 32 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 8 & 0 & 27 & 0 & 0 & 8 & 0 & 2 & 20 & 0 & 10 & 7 & 20 \\
14 & 11 & 38 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 3 & 0 & 8 & 0 & 29 & 0 & 4 & 0 & 0 & 2 & 20 & 0 & 9 & 15 & 9 \\
12 & 22 & 38 & 7 & 32 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 27 & 0 & 0 & 8 & 0 & 2 & 20 & 0 & 10 & 7 & 27 \\
14 & 11 & 38 & 0 & 34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 3 & 0 & 8 & 0 & 29 & 0 & 4 & 0 & 2 & 2 & 20 & 0 & 9 & 15 & 9 \\
4 & 22 & 38 & 0 & 32 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 8 & 0 & 19 & 0 & 8 & 8 & 0 & 2 & 20 & 0 & 2 & 7 & 20 \\
14 & 19 & 38 & 0 & 32 & 4 & 13 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 3 & 0 & 8 & 0 & 29 & 0 & 0 & 8 & 0 & 2 & 33 & 0 & 0 & 7 & 17 \\
14 & 11 & 38 & 0 & 32 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 11 & 0 & 3 & 0 & 8 & 0 & 29 & 0 & 4 & 0 & 0 & 2 & 20 & 0 & 9 & 31 & 9 \\
4 & 22 & 38 & 0 & 32 & 5 & 0 & 0 & 29 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 8 & 0 & 19 & 0 & 8 & 8 & 0 & 2 & 20 & 0 & 2 & 7 & 20 \\
7 & 22 & 38 & 7 & 32 & 5 & 0 & 0 & 0 & 21 & 0 & 0 & 26 & 0 & 0 & 1 & 29 & 0 & 22 & 0 & 5 & 8 & 0 & 2 & 0 & 0 & 4 & 7 & 27 \\
10 & 19 & 38 & 7 & 32 & 5 & 3 & 0 & 0 & 21 & 7 & 0 & 26 & 0 & 3 & 5 & 29 & 0 & 25 & 0 & 5 & 8 & 0 & 2 & 0 & 0 & 0 & 7 & 24 \\
7 & 22 & 38 & 7 & 32 & 5 & 0 & 0 & 0 & 21 & 0 & 30 & 26 & 0 & 0 & 1 & 29 & 0 & 22 & 0 & 5 & 8 & 0 & 2 & 0 & 0 & 4 & 7 & 27 \\
12 & 22 & 38 & 7 & 32 & 5 & 0 & 0 & 0 & 20 & 0 & 0 & 26 & 0 & 0 & 0 & 28 & 0 & 27 & 0 & 0 & 8 & 0 & 2 & 0 & 6 & 10 & 7 & 27 \\
14 & 11 & 38 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 13 & 3 & 0 & 8 & 0 & 29 & 0 & 4 & 0 & 0 & 2 & 20 & 0 & 9 & 15 & 9 \\
14 & 19 & 38 & 0 & 32 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 3 & 0 & 8 & 0 & 29 & 0 & 0 & 8 & 0 & 2 & 20 & 0 & 13 & 7 & 17 \\
14 & 19 & 38 & 0 & 32 & 4 & 8 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 3 & 5 & 8 & 0 & 29 & 0 & 0 & 8 & 0 & 2 & 33 & 0 & 0 & 7 & 17 \\
7 & 22 & 38 & 7 & 32 & 5 & 0 & 0 & 0 & 21 & 0 & 0 & 26 & 0 & 0 & 1 & 31 & 0 & 22 & 2 & 5 & 8 & 0 & 2 & 0 & 0 & 4 & 7 & 27 \\
14 & 19 & 38 & 0 & 32 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 3 & 0 & 8 & 24 & 29 & 0 & 0 & 8 & 0 & 2 & 20 & 0 & 13 & 7 & 17 \\
12 & 9 & 25 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 8 & 0 & 29 & 0 & 8 & 8 & 0 & 0 & 20 & 0 & 2 & 7 & 7
\end{bmatrix}$$

Appendix C

Additional Versions of the ADN Algorithm

This appendix presents two additional versions of the ADN (assembly/disassembly networks) algorithm. In Chapter 6, the ADN algorithm for the continuous material model are presented. Here, we discuss versions for deterministic and exponential processing time model. The complete descriptions and all assumptions of the three models can be referred to Gershwin (1994).

Notice that three versions of the algorithm only differ from each other in Algorithm inputs and Phase II: decomposition evaluation. The three versions have the same Phase I algorithm since the underlying graph models of these three versions are the same. Therefore, in this appendix, we only present the Algorithm input and Phase II algorithm. For the Phase I algorithm, refer to Section 6.2.1.

C.1 Deterministic Processing Time Model Version

C.1.1 Algorithm Input

Given an assembly/disassembly network Ω with n machines and m buffers, the algorithm inputs include:

Machines: $M_i, i = 1, ..., n$

- f_i: number of failure modes of M_i.
- p_{ih}, r_{ih}: failure probability and repair probability of failure mode λ_{ih}, $h = 1, ..., f_i$.

Buffers: $B_j, j = 1, ..., m$

- N_j: size of buffer B_j.

Loops: $L_k, k = 1, ..., m - n + 1$

- I_k: invariant of loop L_k.
- $\Psi = [\psi_{ij}]$: circuit (or loop) matrix with entry ψ_{kj}, $k = 1, ..., m - n + 1; j = 1, ..., m$.

Network Topology

- $\Phi = [\phi_{ij}]$: all-vertex incidence matrix of Ω with entry ϕ_{ij}, $i = 1, ..., n; j = 1, ..., m$.

C.1.2 Algorithm Phase II: Decomposition Evaluation

I. Transformation (Refer to Section 5.2.)

1. In 'Machine failure – Buffer level' matrix $\Theta(\Omega_{m-n+1})$, identify the threshold values of each buffer.
2. For each buffer, insert perfectly reliable machines in threshold positions and split the buffer into a set of sub-buffers.

The transformed system consists of n' machines and m' buffers. The 'Machine failure – Buffer level' matrix corresponding to the transformed system is an $n \times m'$ matrix $\Theta'_{n \times m'}$.

II. Decomposition (Refer to Section 5.3.)

1. Decompose the new transformed system into m' building blocks, $BB(j), j = 1, ..., m'$.

2. Assign the failure modes to the upstream and downstream pseudo-machines of each building block according to 'Machine failure – Buffer level' matrix $\Theta'_{n \times m'}$.

3. For each pseudo-machine:

   - Specify the source machines of all failure modes;
   - Differentiate the local failure modes and remote failure modes;
   - Identify the routing buffer for the remote failure modes.

III. Aggregation of Failure Modes

When a downstream pseudo-machine has multiple failure modes λ_{a1}, λ_{a2},..., λ_{ak} with the same repair probability (different failure probability), we aggregate these modes into one failure mode λ_m using following equations:

$$r_m = r_{ai} \qquad i = 1, ..., k \tag{C.1}$$

$$p_m = 1 - \prod_{i=1}^{k} (1 - p_{ai}) \tag{C.2}$$

Notice that (C.2) ensures that the failure probability of the aggregated failure mode in deterministic processing time model is less than 1.

After solving a two-machine building block, the probability of the downstream pseudo-machine being in starvation state W_m^d is disaggregated as:

$$\mathbf{p}(W_{ai}^d) = \mathbf{p}(W_m^d) \frac{p_{ai}}{\sum_{i=1}^{k} p_{ai}} \qquad i = 1, ..., k \tag{C.3}$$

When a downstream pseudo-machine has multiple failure modes λ_{a1}, λ_{a2},..., λ_{ak} with the same repair probability (different failure probability), the aggregation of failure modes follows (C.1) and (C.2). After evaluation, the probability of the upstream pseudo-machine being in blocking state W_m^u is disaggregated as:

$$\mathbf{p}(W_{ai}^u) = \mathbf{p}(W_m^u)\frac{p_{ai}}{\sum_{i=1}^{k} p_{ai}} \qquad i = 1, ..., k \tag{C.4}$$

IV. Iterations

1. Initialize the buffer size, failure, and repair probabilities of building blocks $BB(j), j = 1, ..., m'$.

$$N'(j) = N'_j \tag{C.5}$$
$$p_{ih}^u(j) = p_{ih} \qquad i \in RS(j); h = 1, ..., f_i \tag{C.6}$$
$$r_{ih}^u(j) = r_{ih} \qquad i \in RS(j); h = 1, ..., f_i \tag{C.7}$$
$$p_{ih}^d(j) = p_{ih} \qquad i \in RB(j); h = 1, ..., f_i \tag{C.8}$$
$$r_{ih}^d(j) = r_{ih} \qquad i \in RB(j); h = 1, ..., f_i \tag{C.9}$$

In the deterministic processing time model of two-machine line in Tolio et al. (2002), the minimal feasible buffer size is 3. Therefore, for building block $BB(j)$:

$$\text{If} \quad N'(j) < 3 \text{ , set } N'(j) = 3 \tag{C.10}$$

2. Evaluate building blocks and update unknown parameters in forward and reverse orders iteratively until the termination condition is satisfied.

Step 1: Forward Procedure

(a) Evaluate $BB(j)$ to obtain the probabilities of events $E^u(j)$, $E^d(j)$, $W_{ih}^u(j)$,

$W^d_{ih}(j)$ and production rate $P(j)$ using the algorithm presented in Tolio et al. (2002).

(b) For $j = 1, ..., m'$

Update remote failure rates $p^u_{ih}(j)$ of $M^u(j)$:

$$p^u_{ih}(j) = \begin{cases} \dfrac{\mathbf{p}(W^d_{ih}(v))}{E^u(j)}\ r^u_{ih}(j) & u(j) = d(v) \\ \\ \dfrac{\mathbf{p}(W^u_{ih}(v))}{\mathbf{p}(E^u(j))}\ r^u_{ih}(j) & u(j) = u(v) \end{cases} \tag{C.11}$$

in which

- $v = \eta^u_i(j)$, the routing buffer.
- $E^u(j)$ is the event machine $M^u(j)$ is up.
- $W^u_{ih}(v)$ is the event machine $M^u(v)$ is blocked due to failure modes $\lambda^d_{ih}(v)$ of downstream pseudo-machine $M^d(v)$.
- $W^d_{ih}(v)$ is the event machine $M^d(v)$ is starved due to failure modes $\lambda^u_{ih}(v)$ of upstream pseudo-machine $M^u(v)$.

If failure mode $\lambda^u_{ih}(j)$ is propagated through multiple routing buffers $\eta^u_i(j)$, calculate $p^u_{ih}(j)$ for each routing buffer and take the maximal value.

Step 2: Reverse Procedure

(a) Evaluate $BB(j)$ to obtain the probabilities of events $E^u(j)$, $E^d(j)$, $W^u_{ih}(j)$, $W^d_{ih}(j)$ and production rate $P(j)$.

(b) For $j = m', ..., 1$

Update remote failure rates $p^d_{ih}(j)$ of $M^d(j)$:

$$p^d_{ih}(j) = \begin{cases} \dfrac{\mathbf{p}(W^d_{ih}(w))}{\mathbf{p}(E^d(j))}\ r^d_{ih}(j) & d(j) = d(w) \\ \\ \dfrac{\mathbf{p}(W^u_{ih}(w))}{\mathbf{p}(E^d(j))}\ r^d_{ih}(j) & d(j) = u(w) \end{cases} \tag{C.12}$$

in which

- $w = \eta_i^d(j)$, the routing buffer.
- $E^d(j)$ is the event machine $M^d(j)$ is up.
- $W_{ih}^u(w)$ is the event machine $M^u(w)$ is blocked due to failure modes $\lambda_{ih}^d(w)$ of downstream pseudo-machine $M^d(w)$.
- $W_{ih}^d(w)$ is the event machine $M^d(w)$ is starved due to failure modes $\lambda_{ih}^u(w)$ of upstream pseudo-machine $M^u(w)$.

If failure mode $\lambda_{ih}^d(j)$ is propagated through multiple routing buffers $\eta_i^d(j)$, calculate $p_{ih}^d(j)$ for each routing buffer and take the maximal value.

We perform Steps 1 and 2 iteratively until the parameters converge to an acceptable tolerance.

V. Record the Performance Measures

- Record production rate P.

$$P = \sum_{j=1}^{m'} P(j) \Big/ m' \tag{C.13}$$

- Calculate average buffer levels $\overline{b}'(j), j = 1, ..., m'$.
- For the original buffers $B_k, k = 1, ..., m$ before transformation, obtain their average levels by summing up the average levels of the corresponding sub-buffers in the transformed system.

$$\overline{b}(k) = \sum_{B_j' \subseteq B_k} \overline{b}'(j) \qquad k = 1, ..., m \tag{C.14}$$

C.2 Exponential Processing Time Model Version

The only difference between the exponential processing time model version and the continuous material model is the algorithm for evaluating two-machine one-buffer

building blocks. For building block $BB(j)$, it is evaluated using the algorithm presented in Levantesi et al. (May, 1999) to obtain the probabilities of events $E^u(j)$, $E^d(j)$, $W^u_{ih}(j)$, $W^d_{ih}(j)$ and production rate $P(j)$. For the steps of the algorithm, refer to Section 6.2.

Appendix D

Algorithm for Random Case Generation

This appendix presents an algorithm, called 'System Generator (SysG) algorithm', for generating random cases of assembly/disassembly networks.

In Chapter 6, we have developed the evaluation algorithm (ADN algorithm) for evaluating assembly/disassembly networks with arbitrary topologies. In order to determine whether the ADN algorithm is accurate, reliable, and efficient while evaluating large-scale networks with complex topologies, we should develop an algorithm (that we call SysG) to generate random cases of assembly/disassembly networks. The SysG algorithm is summarized in Section 7.2.2 as a component of experiment architecture.

SysG algorithm includes four steps:

1. Import network scale description;

2. Generate random network structure;

3. Generate random parameters of machines and buffers;

4. Export the network.

Step 1: Import Network Description

The experiment description includes the network scale description, scale code, structure code, and case code. The network scale description is specified by the number of machines, buffers, and loops in the network.

In addition to the above parameters, we need to specified some initial values, upper and lower bounds for similarity of machine and buffer parameters such that manufacturing systems with realistic parameters can be generated. Details are presented in Step 4.

Step 2: Generate Random Network Structure

To randomly generate a network structure, we must ensure that the underlying digraph of the network is connected. A method is presented for generating a random network structure with n machines and m buffers:

1. Generate a random spanning tree with n machines and $n-1$ buffers.

   - $k = 1$: We create the first machine M_1. It is the first vertex in the underlying digraph.
   - $k = 2, ..., n$: To add machine M_k, we randomly select a machine from $\{M_1, ..., M_{k-1}\}$ and create a buffer B_{k-1} to connect the selected machine with M_k. The flow direction of buffer B_{k-1} is randomly selected by flipping a coin.

2. Add chords B_{n-1+k} and form a set of fundamental loops $L_k, k = 1, ..., m-n+1$.

   - To add chord B_{n-1+k}, we randomly select two different machines from $\{M_1, ..., M_n\}$ and connect them using B_{n-1+k}. The flow direction of buffer B_{n-1+k} is randomly selected by flipping a coin.
   - To determine loop L_k corresponding to chord B_{n-1+k}, in the spanning tree composed of $\{M_1, ..., M_n\}$ and $\{B_1, ..., B_{n-1}\}$, we identify a unique path that connects the end vertices of B_{n-1+k}. Loop L_k is composed of B_{n-1+k} and the unique path.

3. Construct all-vertex incidence matrix Φ.

   For each buffer, identify its upstream and downstream machines in the network and specify the corresponding column vector of Φ.

4. Construct loop matrix Ψ.

   We choose the loop orientation of L_k to agree with the flow direction of chord B_{n-1+k} and specify the corresponding loop vector $\mathbf{\Psi}_{k\star}, k = 1, ..., m - n + 1$.

Step 3: Generate Random Parameters of Machines and Buffers

After generating the network structure, we need to generate random machine parameters, buffer sizes, and loop invariants.

Machine and Buffer Parameters We generate parameters including the processing, failure and repair rates of the machines and the sizes of the buffers in the network. We list the key quantities, equations, and procedures here. Detailed discussion and derivation please refer to Tanizar (2001) Appendix A.

A set of constant quantities are defined here:

- Lower and upper bounds of repair rate similarity: L_r, U_r
- Lower and upper bounds of failure rate similarity: L_p, U_p
- Lower and upper bounds of efficiency similarity: L_E, U_E
- Lower and upper bounds of production rate rate similarity: L_P, U_P
- Lower and upper bounds of processing rate similarity: L_μ, U_μ
- Lower bound of the ratio of processing rate to repair rate ratio: η
- Lower and upper bounds of buffer sizes: L_N, U_N

The above 13 constants should be specified in network description in Step 1.

Similarity constraints are:

$$L_r \leq \frac{r_i}{r_j} \leq U_r \tag{D.1}$$

$$L_p \leq \frac{p_i}{p_j} \leq U_p \tag{D.2}$$

$$L_E \leq \frac{r_i}{r_j + p_j} \leq U_E \tag{D.3}$$

$$L_P \leq \frac{P_i}{P_j} \leq U_P \tag{D.4}$$

$$L_\mu \leq \frac{\mu_i}{\mu_j} \leq U_\mu \tag{D.5}$$

$$\forall i, j;\ i, j = 1, ..., k$$

The ratio of processing rate to repair rate should satisfies:

$$\mu_i \neq \eta r_i \qquad \forall i \in 1, 2, ..., k \tag{D.6}$$

Buffer size constraint is:

$$L_N \leq \frac{N_j}{N^*} \leq U_N \qquad j = 1, ..., m \tag{D.7}$$

where $N^* = \sum_{i=1}^{k} \frac{\mu_i}{r_i}$.

After specifying the quantities and constraints, we present an algorithm to generate n random machines which satisfy the above constraints (D.1)–(D.6):

- For the first machine M_1:
  - Generate initial repair rate of the first machine: r_1.
  - Rearrange (D.3) to generate p_1:

$$\frac{1 - U_E}{U_E} r_1 \leq p_1 \leq \frac{1 - L_E}{L_E} r_1 \tag{D.8}$$

  - For the continuous or exponential models, generate μ_1 using (D.6).

- For machines $M_k, k = 2, 3, ..., n$:

  - Generate repair rate of machine M_k:

$$\max_{i=1,2,...,k-1} L_r r_i \leq r_k \leq \min_{i=1,2,...,k-1} U_r r_i \tag{D.9}$$

  - Combine (D.3) and (D.2) to generate p_k

$$\max_{i=1,2,...,k-1} \{\frac{1-U_E}{U_E} r_i, L_P P_i\} \leq p_k \leq \min_{i=1,2,...,k-1} \{\frac{1-L_E}{L_E} r_i, U_P P_i\} \tag{D.10}$$

  - For the continuous or exponential models, generate μ_k by combining (D.5), (D.4), and (D.6):

$$\max_{i=1,2,...,k-1} \{L_\mu \mu_i, L_P P_i \frac{r_k + p_k}{r_k}, \eta r_k\} \leq p_k \leq \min_{i=1,2,...,k-1} \{U_\mu \mu_i, U_P P_i \frac{r_k + p_k}{r_k}\} \tag{D.11}$$

The sizes of m random buffers are generated by

$$L_N N^* \leq N_j \leq U_N N^* \qquad j = 1, ..., m \tag{D.12}$$

Loop Invariants In addition, we need to generate the feasible loop invariants of $m - n + 1$ loops.

In Section 3.4.3, we discuss the feasibility of loop invariants **I** and buffer sizes **N**. When multiple loops are coupled together, it is difficult to generate feasible loop invariants directly. Notice that each loop invariant is a linear combination of the buffer levels:

$$I_k = [\psi_{k1}, \psi_{k2}, \ldots, \psi_{km}] \begin{bmatrix} b(1) \\ b(2) \\ \vdots \\ b(m) \end{bmatrix} \tag{D.13}$$

$$k = 1, 2, ..., n - m + 1$$

Since the buffer levels are a set of independent variables, we generate the loop invariants using two steps:

1. Randomly generate buffer levels $b(1), b(2), ..., b(m)$;
2. Derive loop invariants $I_k, k = 1, ..., n - m + 1$ using (D.13).

Therefore, feasible loop invariants are randomly generated.

Step 4: Export The Network

The final step is to export the network with the generated random structure and parameters. The exported file contains the scale code, structure code and case code specified in Step 1. This file serves as the input file of the evaluation using the ADN algorithm.

Bibliography

T. Altiok. *Performance Analysis of Manufacturing Systems*. Springer Verlag, New York, 1997.

T. B. Boffey. *Graph Theory in Operations Research*. THE MACMILLAN PRESS LTD, 1982.

A. M. Bonvik. Performance analysis of manufacturing systems under hybrid control policies. Technical Report Ph.D. dissertation, MIT Operations Research Center, 1996.

A. M. Bonvik, C. E. Couch, and S. B. Gershwin. A comparison of production-line control mechanism. *International Journal of Production Research*, 35(3):789–804, 1997.

A. M. Bonvik, Y. Dallery, and S. B. Gershwin. Approximate analysis of production systems operated by a conwip/finite buffer hybrid control policy. *International Journal of Production Research*, 38(13):2845–2869, 2000.

M. H. Burman. New results in flow line analysis. Technical Report Ph.D. dissertation, MIT Operations Research Center, 1995.

J. A. Buzacott and J. G. Shanthikumar. *Stochastic Models of Manufacturing Systems*. Prentice Hall, 1993.

B. Carre. *Graphs and Networks*. Oxford University Press, 1979.

Y. Dallery and S. B. Gershwin. Manufacturing flow line systems: A review of models and analytical results. *Queueing Systems Theory and Applications, Special Issue on Queuing Models of Manufacturing Systems*, 12:3–94, 1992.

Y. Dallery and G. Liberopoulos. Extended kanban control system: combining kanban and base stock. *IIE Transactions*, 32(4):369–386, 2000.

Y. Dallery, R. David, and X.-L. Xie. An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffer. *IIE Transactions*, 20(3):280–283, 1988.

Y. Frein, C. Commault, and Y. Dallery. Modeling and analysis of closed-loop production lines with unreliable machines and finite buffers. *IIE Transactions*, 28(7): 369–386, 1996.

E. G. A. Gaury, H. Pierreval, and J. P. C. Kleijnen. An evolutionary approach to select a pull system among kanban, conwip and hybrid. *Journal of Intelligent Manufacturing*, 11(2):157–167, 2000.

E. G. A. Gaury, J. P. C. Kleijnen, and H. Pierreval. A methodology to customize pull control systems. *Journal of the Operational Research Society*, 52(7):789–799, 2001.

S. B. Gershwin. Design and operation of manufacturing systems: the control-point policy. *IIE Transactions*, 32(10):891–906, 2000.

S. B. Gershwin. Factory models for manufacturing systems engineering. SMA IMST Symposium, 2003.

S. 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.

S. B. Gershwin. Assembly/disassembly systems: An efficient decomposition algorithm for tree-structured networks. *IIE Transactions*, 23(4):302–314, 1991.

S. B. Gershwin. *Manufacturing Systems Engineering.* Prentice-Hall, 1994.

S. B. Gershwin and J. E. Schor. Efficient algorithms for buffer space allocation. *Annals of Operations Research*, 93:117–144, 2000.

S. B. Gershwin and L. M. Werner. An approximate analytical method for evaluating the performance of closed loop flow systems with unreliable machines and finite buffers, 2006. To appear.

E. M. Goldratt and J. Cox. *The goal : a process of ongoing improvement*. North River Press, New York, 1986.

H. Hale. The inverse of a nonsingular submatrix of an incidence matrix. *Circuit Theory, IRE Transactions on*, 9(3):299–300, 1962.

S. Helber. *Performance Analysis of Flow Lines with Non-Linear Flow of Material*, volume 473. Springer, 1999.

W. J. Hopp and M. L. Roof. Setting wip levels with statistical throughput control (stc) in conwip production lines. *International Journal of Production Research*, 36 (4):867–882, 1998.

W. J. Hopp and M. L. Spearman. *Factory Physics: Foundations of Manufacturing Management*. Irwin, 1996.

R. Levantesi. Analysis of multiple loop assembly/disassembly networks. Technical Report Ph.D. dissertation, Politecnico di Milano, 2001.

R. Levantesi, A. Matta, and T. Tolio. Exponential two-machine lines with multiple failure modes and finite buffer capacity. Second Aegean International Conference on the, May, 1999.

R. Levantesi, A. Matta, and T. Tolio. Continuous two-machine line with multiple failure modes and finite buffer capacity, September, 1999.

N. Maggio, A. Matta, S. B. Gershwin, and T. Tolio. An approximate analytical method for evaluating the performance of closed loop flow systems with unreliable machines and finite buffers. *IIE Transactions*, 2006. To appear.

M. D. Mascolo, R. David, and Y. Dallery. Modeling and analysis of assembly systems with unreliable machines and finite buffers. *IIE Transactions*, 23(4):315–331, 1991.

Y. Monden. *Toyota Production System: an integrated approach to just-in-time*. Engineering & Management Press, 1983.

J. Resh. The inverse of a nonsingular submatrix of an incidence matrix. *Circuits and Systems, IEEE Transactions on*, 10(1):131–132, 1988.

S. M. Ryan and F. F. Choobineh. Total wip and wip mix for a conwip controlled job shop. *IIE Transactions*, 35(5):405–418, 2003.

S. M. Ryan, B. Baynat, and F. F. Choobineh. Determining inventory levels in a conwip controlled job shop. *IIE Transactions*, 32(2):105–114, 2000.

J. E. Schor. Efficient algorithms for buffer allocation. Technical Report M.S. thesis, MIT EECS, May 1995.

M. N. S. Swamy. *Graphs, Networks, and Algorithms*. Jon Wiley & Sons, 1981.

D. A. Syrowicz. Decomposition analysis of a deterministic, multiple-part-type, multiple-failure-mode production line. Technical Report M.E. thesis, MIT EECS, 1998.

K. Tanizar. Alternatives to gradients in transfer line buffer space allocation. Technical Report M.S. thesis, MIT Operations Research Center, 2001.

T. Tolio and A. Matta. A method for performance evaluation of automated flow lines. *Annals of the CIRP*, 47(1):373–376, 1998.

T. Tolio, S. B. Gershwin, and A. Matta. Analysis of two-machine lines with multiple failure modes. *IIE Transactions*, 34(1):51–62, 2002.

L. Werner. Analysis and design of closed loop manufacturing systems. Technical Report M.S. thesis, MIT Operations Research Center, 2001.