New Results in Flow Line Analysis

by

Mitchell H. Burman

B.A. in Mathematical Economics, Colgate University, USA (1985)
M.S. in Operations Research, Massachusetts Institute of Technology, USA (1993)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctorate of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Author ....
Department of Electrical Engineering and Computer Science
May 11, 1995

Certified by ....
Stanley B. Gershwin
Senior Research Scientist
Thesis Supervisor

Accepted by ....
Tom Magnanti
Co-Director, Operations Research Center
Chairman, Interdepartmental Committee on Operations Research

New Results in Flow Line Analysis

by

Mitchell H. Burman

Submitted to the Department of Electrical Engineering and Computer Science
on May 11, 1995, in partial fulfillment of the
requirements for the degree of
Doctorate of Philosophy in Operations Research

Abstract

We develop an analytic decomposition-type approximation to estimate the throughput of flow lines with processing rates that may differ from stage to stage but where the production times are constant at each stage. We develop the fastest and most reliably convergent algorithm for solving decomposition-type equations known to date. This method has been used in the design of a large automated production system for Hewlett Packard, and it has provided substantial economic benefit. Johnson & Johnson have also adapted the model for their own use. We also develop analysis techniques for dealing with work centers with parallel identical processes and failing buffers.

Thesis Supervisor: Stanley B. Gershwin
Title: Senior Research Scientist

Acknowledgments

Support for this thesis came in many forms, technical, emotional and financial. Without undue exaggeration, the greatest contributor in all three areas was my advisor Stan Gershwin. He kept me going at least 1000 times when I truly believed this work would never end. He is a great friend and advisor.

Expressed thanks are also due to the rest of my committee: Vien Nguyen, Michael Caramanis and Stephen Graves. Each was patient and invaluable in helping me complete this work.

This work was primarily financially supported by the Department of Energy's Integrated Manufacturing Fellowship provided through the National Academy of Science. This research was also partially supported by the National Science Foundation by NSF Grant DDM-9216358, Johnson & Johnson, Hewlett Packard and the Leaders for Manufacturing Program at MIT.

I would also like to thank the people at Hewlett Packard who have so generously provided me with their factory as my laboratory. Special thanks to Jim Burruss, Curtis Suyematsu, Pat Redding, Paul Speer and Brad Freeman.

Special thanks to John Matson for all his help and encouragement.

Thanks again to my father for everything.

Biographical Note

The author received his Bachelor's degree in Mathematical Economics from Colgate University in 1985 and a Master's Degree in Operations Research from MIT in 1993. He has worked on special projects for Intel, General Motors, Johnson & Johnson, Hewlett Packard as well as many other smaller companies. He is the current President of Analytics, Inc., a company established to bring Operations Research techniques for manufacturing and logistic systems from academia into the business community. The author has participated in numerous conferences and has had a number of accepted publications in the areas of manufacturing systems design and real-time control.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A *flow line* is a manufacturing system consisting of work centers separated by storage areas in which material flows in a fixed sequence, visiting each work center once and only once. Figure 1-1 illustrates a simple flow line of k work centers $(M_1, M_2, \ldots, M_k)$ separated by $k-1$ buffers $(B_1, B_2, \ldots, B_{k-1})$. Raw material enters the system from outside through Work Center M_1, where it receives processing (*e.g.*, inspection, forming, assembly, etc.). The material then flows through the system in the direction of the arrows and receives processing at each work center. A final assembly exits from Work Center M_k. In this thesis, we develop a decomposition-type model to estimate the throughput of flow lines with unreliable machines with operation-dependent failures. We deal explicitly with flow lines where processing rates may differ from stage to stage but where the production times are constant at each stage. We also develop the fastest and most reliably convergent algorithm for solving decomposition-type equations known to date.

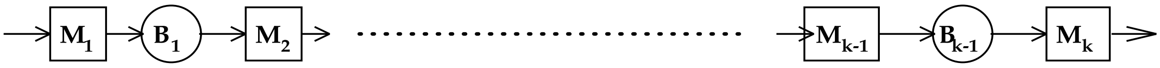

Figure 1-1: Flow Line

Examples of flow lines can be observed in automotive fabrication, electronic components assembly, food packaging, and consumer products manufacturing industries. As a result of increasing competitive requirements in these industries, many of the United States' industrial dinosaurs have embarked on process improvement programs that often include the reengineering of their manufacturing processes (Snodgrass, 1994). An effective implementation of these programs requires performance evaluation techniques for comparing various restructuring options, but many manufacturing facilities lack good methods that can quickly and accurately estimate important performance metrics such as average manufacturing throughput capacity and work-in-process ("WIP").

Flow lines represent large investments for most companies. For example, typical flow lines in the automotive industry cost on the order of hundreds of millions of dollars (Suri and Fu, 1993). Throughput estimates are particularly important for project justification, establishing realistic system objectives and evaluating the financial effects of machine configuration changes (Ransley, 1995).

This thesis is applications-driven and accounts for the following characteristics observed in automated discrete part manufacturing flow lines in industry:

- unreliable machines with operation-dependent failures and deterministic operation times when operating,
- multiple work centers,
- finite material storage areas, called *buffers*,
- asynchronous, non-homogeneous operation times,
- short operation times compared with average failure durations.

In addition, this thesis examines two relatively unexplored characteristics of flow lines:

- buffers that can fail,
- work centers with parallel identical processes.

This research does not account for rework, scrap, and reentrant flows which are also important issues.

The goal of this thesis is to answer the question, "What average throughput rate and WIP are expected from a given flow line design?" A quick and accurate answer to this question will assist in answering further questions such as "What is the best way allocate buffer capacity to achieve the maximum throughput rate for a given process?," and "What is the variability of output?" The literature review in Gershwin and Goldis (1995) discusses the area of line optimization and Gershwin (1993) addresses the variance issue. Financing, quality improvement and customer satisfaction, technology development, environmental impact, employee satisfaction and scheduling, maintenance policies, production scheduling, and other system management issues are also not addressed in this thesis.

Evaluating dynamic manufacturing systems to obtain measures of average throughput and WIP can be very difficult due to the complexity of the system and the uncertainty of the future. Systematic approaches to the problem include real world experimentation, simulation and analytic methods. Real world experimentation is often too expensive, too time-consuming and too much of a disruption to current operations to implement in practice. Simulation, while faster and less disruptive than real world experimentation, can still be very slow. However, it can be very useful for the detailed evaluation of a single final design. Graphics modules included with some simulation packages can also often add insight. Analytic models are by far the fastest method for evaluating system performance, but are very hard to construct. These techniques are usually quicker, cheaper and less disruptive, but are subject to errors associated with the abstracting of reality.

This thesis focuses on furthering the development of analytic models to evaluate real systems by more accurately accounting for key system phenomena. Motivated by the actual planning requirements at Hewlett Packard's Vancouver Division ("HP"), this thesis identifies the relevant and critical phenomena in their production facility. A mathematical abstraction of flow lines is developed that captures these phenomena. It is a Markovian model that uses a continuous material approximation to represent the movement of parts. This approach facilitates the development of a tractable, realistic model.

1.1 Motivation

Several years ago, the process engineering department at the Vancouver Division of Hewlett Packard was planning to build a new large automated manufacturing system.[1] They had already committed to a flow line design with the added complication of multiple subassembly input flow lines (SA) feeding it at various points through buffers (B) as shown in Figure 1-2. The group built financial models to calculate the viability of the investment with an estimate of the system throughput rate as a key input. Third party simulation to estimate system throughput would have delivered results

[1] Details concerning the actual system design and throughput numbers have been omitted for reasons of confidentiality.

too late to influence the design given the aggressive project schedule. Simulation benefits were therefore greatly diminished for this phase of the program. To keep the project moving forward, HP utilized analytic modeling as a tool to help analyze and modify the line architecture and predict the resulting effects on throughput.

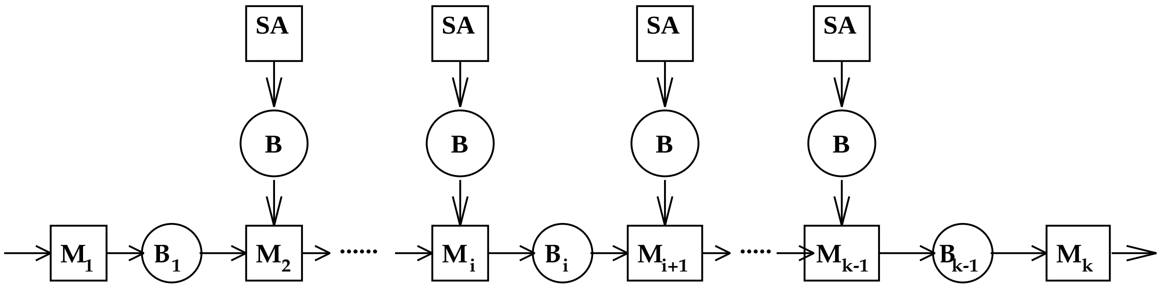

Figure 1-2: Flow Line with Subassemblies

Months later, it was decided to develop a real-time scheduling procedure for the system that would meet the projected system throughput requirements without an excessively amount of *work-in-process* (WIP). This would limit system *flow time* (the time from entering M_1 to leaving M_k, Figure 1-2) and the costs associated with WIP. The small buffers made it easy to estimate system throughput using the zero-buffer model of Buzacott (1968). However, the estimated output of the non-modified line was significantly lower than HP's original estimate. It was at this point that HP's attention shifted towards reengineering their production process.

Since the system was already in fabrication, individual machines in the production system could not be changed without a significant disruption to the product development cycle. Therefore, there was no easy way to change the isolated efficiencies of the components of the production system. The best way to improve the throughput would be to selectively install limited buffers at strategic points. This would dampen the effects of machine failures by limiting the propagation effects through the rest of the production line. These buffers were themselves machines such as conveyers, accumulators and other such material handling equipment. This type of machinery was also subject to failures. No modeling technique in the literature at the time accounted for failing buffers. A second and much more costly option for improving expected throughput was to increase the capacity of the slowest stages in the system by installing additional parallel machines. With this approach, when a machine fails, its processing responsibility is immediately covered by another machine, preventing an overall disruption to the line. However, the problem remained to estimate the impact on throughput of installing new in-line pallet accumulators or other extra machinery.

Due to the magnitude and sensitivity of the business investments, HP required a more rigorous approach to system design. Simulation lead-time, complexity, sensitivity to inputs, and iteration difficulties greatly diminished its usefulness. Therefore, we decided to use some of the quicker analytic models found in the flow line literature (Dallery and Gershwin, 1992). We chose the decomposition method introduced in Gershwin (1987a), the decomposition equations developed in Gershwin (1989) and the DDX decomposition algorithm (Dallery, David and Xie, 1988). The assembly aspect of the system was handled less formally by speeding up the subassembly cells to a level where they virtually never starved the main line of material. The failing buffers were accounted for by multiplying the results of the decomposition estimates by a scaling factor. This overall approach provided a means of comparing the relative improvements associated with different buffer configurations on the production line. Multiple experiments were conducted and within one week, recommendations were in place that promised to substantially increase the expected system throughput (approximately double) with a minimum impact on flow time and WIP, and a relatively low price tag. The benefit of this work is on the order of millions of dollars per month.

The HP story highlights typical problems faced by process engineers. Our goal is to make this entire process easier and better in the future. Product introductions and changes are occurring so rapidly that process engineers are faced with increasing pressure to reduce development cycles. Judgments made on simplified capacity models are not adequate to manage investment risks of this magnitude on aggressive schedules. Typical simulation exercises take far too long to model even very few design options. Although there were limitations in the analytical models used at HP (specifically, the inability to account for parallel machines, deterministic asynchronous processing times, failures in the material handling system and subassembly cells) there was the opportunity to employ some creative adjustments in using the models. It was these creative approaches that motivated this thesis.

1.2 Classification Scheme

A *manufacturing flow line* (also called *transfer line*, *production line* and within the queueing literature, *finite buffered, tandem queueing system with blocking*) consists of material, *work areas* or *stages*, and finite storage areas or *buffers* (Figure 1-1). Finite material storage areas distinguish flow lines from other queueing systems. Material flows from work area to storage area to work area, visiting each work and storage area exactly once in a fixed sequence. Work areas may consist of a single machine or multiple machines in parallel, in what we will call *series-parallel systems*. In this section, we establish a classification scheme for identifying the different assumptions made in the literature regarding flow lines. We begin with the more general assumptions, common to many types of production models, and continue with the characteristics which tend to distinguish flow lines from other queuing models. Many of the definitions are obtained directly from Dallery and Gershwin (1992).

1.2.1 General Modeling Assumptions

There are a number of assumptions and classifications that are used to differentiate general production queueing models. We define these concepts in this section.

Independence of Events *Independence of events* implies that events in the future are only contingent on the present state of the system and are otherwise independent of each other and the past events (Ross, 1983 or Gershwin, 1994). In production models, this means that failure times and repair times on different machines are independent of one another. It also implies that the times between failures of a given machine are independent of previous failure times, repair times and the amount of material in the buffer, assuming the machine is not blocked or starved. This independence assumption allows the system to be modelled as a *Markov process.* Markovian models are frequently employed in analyzing production systems because of their tractability (Buzacott and Shanthikumar, 1993).

Saturated System A production line that always has raw material available and unlimited demand is called *saturated.* Unsaturated systems may be approximated using a saturated model by emulating the arrival process with an additional machine upstream of the true first machine.

Work-Conserving Operating Policies A simple operating policy that requires that a machine never be stopped from operating unless it is blocked or starved is discussed in Buzacott (1967a). Using dynamic programming methods, Buzacott (1982) shows the optimality of this policy with respect to maximizing throughput in a transfer line of the type described in his 1967 paper. This control policy is very commonly used in simulation packages as well as queueing models.

Open System An *open queueing system* is based on the assumption of independence between arriving and departing parts. While in most of the automated flow lines in which we are concerned there is independence between arriving and departing parts, there may be a dependence between the arriving and departing pallets on which the parts are transported. The degree of this dependence may influence the validity of our model; closed-loop models that do not assume this independence may be more appropriate (Frein *et al.*, 1992)in some cases. Others have tried to suggest the use of closed-loop models to approximate the behavior of open networks with blocking (Suri and Diehl, 1986). However, these models have very limited applicability and are more computationally complex than the decomposition methods explored in this thesis.

In addition to overall closed-loop systems, a second type of non-open system exists called a *reentrant-flow network* where parts return to the same machine more than once for processing. Examples of reentrant flows are common in semiconductor manufacturing and products that require multiple coatings (*e.g.*, cars that require 15 coats of paint). These systems are important and very hard to analyze, and are not addressed in this thesis.

1.2.2 Machine Behavior

In this section, we examine the distinguishing characteristics associated with machine reliability and material processing on flowlines.

Failures A *failure* occurs when a flow line stage ceases to run due to a malfunction. Failures can be *single stage failures* involving only one stage that may be due among other reasons to tool wear or jamming. *Total line failures* simultaneously shut down all stages in a line and may be caused by events such as power failures. Total line failures can be easily modeled and are not discussed further in this thesis (Buzacott and Hanifin, 1978b). *Failing buffers* are buffers that are subject to stoppages that simultaneously prevent them from receiving, transporting and delivering material.

There are two important categories of single stage failures. *Operation-dependent failures*, or ODFs, occur as a function of the amount of material processed since the last time a machine was repaired. *Time-dependent failures*, or TDFs, occur as a function of the time a machine runs since its last failure. There is a significant impact on flow line model estimate based on whether one assumes TDFs or ODFs. However, casual observations have shown that TDFs are not very prevalent. Models that incorporate failures are also distinguished by the type of distribution used to define a machine's *time to failure* and the *time to repair*. Due to the independence of events assumption, geometric, exponential and Coxian distributions (Buzacott and

Shanthikumar, 1993) are the most common distributions used to model these phenomena.

Processing Time Variations The *cycle time* is the time required for a single operation on an isolated machine. Cycle times are considered *deterministic* when they do not vary from one part to the next on a specific process. *Stochastic* cycle times vary randomly from part to part. Common distributions associated with processing times in Markovian models are again geometric, exponential, and Coxian. However, flow lines are usually designed to produce similar or identical mature products in large quantities, and while work centers may jam or fail completely, they usually perform their task with a low level of variability when operational (*i.e.*, deterministic operation times).

Lines can be further delineated based on the coordination of operations at each work center. A *synchronous* system is one in which events only occur at certain times. Every work center starts and stops at the same time and process at the same rate. *Asynchronous* machines are not constrained to stop and start at the same time. Even when machines have fixed, equal cycle times, they may stop or start independently due to failures in the system. Systems are often assumed to be synchronous for the purpose of model tractability. *Homogeneous* systems exist when all the production stages produce at the same rate but not necessarily at the same time instant as with synchronous systems. In *non-homogeneous* systems, different stages can produce at different rates.

FLUMs **vs.** *FLRMs* Most flow line analyses account for machine failures and processing time variations in one of two ways. One method absorbs the failure time into the processing time distribution in what Gaver (1954) calls *completion time*. Sometimes these models are called *flow lines with reliable machines* or *FLRMs* (Dallery and Gershwin, 1992). A description of the literature related to long flow lines with FLRMs can be found in Hillier and Boling (1967). In automated systems, such as the ones in which we are most interested in this thesis, machines have deterministic operation times. We believe that the general distributions required for operation times to correctly reflect this behavior would be intractable for long lines.

The second method of dealing with unreliable machines is to explicitly have machines that fail and get repaired. These models are called *flow lines with unreliable machines* or *FLUMs* (Dallery and Gershwin, 1992). We feel this is more appropriate in situations of automated production and have based our work on this model.

Yield *Yield* is the number of non-defective parts produced divided by the total number of parts produced. When no material is scrapped, yield is considered *perfect*, and *imperfect* yield indicates scrapping. In real systems, inspection sometimes indicates a need to re-process a part until it meets specifications. Another basic phenomenon occurs when something is perishable. In such cases, parts are scrapped. We are aware of two papers that deal with scrapping (Shanthikumar and Tien, 1983; Jafari and Shanthikumar, 1987). The first paper only deals with simple two-stage lines and the second uses an *aggregation method* (Section 2.1). In general, aggregation methods are less accurate than decomposition. Although we recognize yield as an important issue, it is assumed to be perfect in this thesis because of the computational complexity associated with accounting for it.

1.2.3 Buffers

Buffer is another word for storage for in-process inventory. The assumption of *finite capacity buffers* most clearly distinguish the flow line models from standard queuing models. With the advent of Just-in-Time ("JIT"), small buffers have become the norm in many manufacturing operations (Schonberger, 1986). When buffers are small, system disruptions rapidly propagate through the entire manufacturing system. This creates starvation, blockage, and reduces overall throughput. *Starvation* occurs when a process is unable to produce at its full capacity because it lacks sufficient input material from upstream[2]. *Blockage* occurs when a process is unable to produce at its full capacity because the buffer immediately downstream is filled.

Blockage and starvation have a critical impact on system performance. Manufacturers can keep WIP low if they are willing to accept a low throughput rate (Buzacott, 1967a). For example, imagine a ten-stage line in which each machine has a 90% isolated efficiency[3] and each processes at a rate of one part per time unit. Under the assumption of TDFs, if there were no buffers, the average line throughput would be approximately 0.9^{10} or 0.35. With infinite buffers, average line throughput would be 0.9.

Buffer transit time is the time from when a part enters an empty buffer that is not blocked by a downstream machine until the time that part is able to leave the buffer. All the flow line models, with the exception of Commault and Semery (1990) and Liu (1990), assume a zero transit time in the buffer. Observations of industry indicate

[2] *Upstream* refers to the part of the process that occurs before a particular stage, and *downstream* refers to the part that occurs after.

[3] *Isolated efficiency* is the percentage of time an isolated machine would produce if it is never intentionally shutoff.

that many systems use transport systems as buffers. These systems violate the zero transit time assumption. Buffers can be designed to significantly reduce transport times and this can substantially improve throughput as well as the usefulness of these models. However, users of the decomposition techniques in this thesis must be wary of the potential positive bias in the throughput estimate caused by the buffer transit time.

1.3 Literature Review

This review focuses on analytical flow line models. No references are made to simulation or the general queueing literature. In this thesis, we further the development of the research in the area of analytical models by developing new decomposition methods for estimating throughput and average buffer levels. We also account for new flow line characteristics such as failing buffers and flow lines with multiple sets of parallel machines. These specific issues are reviewed separately in Chapters 2, 4 and 5.

Previous surveys of the flow line literature include Buxey *et al.* (1973) which dealt with a wide variety of flow line phenomena. Buzacott and Hanifin (1978b) provided a state-of-the-art literature review for the time it was published. Smunt and Perkins (1985) focused on asynchronous flow lines, including a great deal of the simulation literature; Perros (1986) reviewed general results of queueing networks with blocking; and Awate and Sastry (1987) summarized the solution methods of the important papers. Dallery and Gershwin (1992) is the most recent and complete survey that relates to this thesis. The interested reader is directed to their review for a more detailed description of the current literature and a further list of other reviews. The terminology and notation of this thesis is consistent with that of the Dallery and Gershwin review.

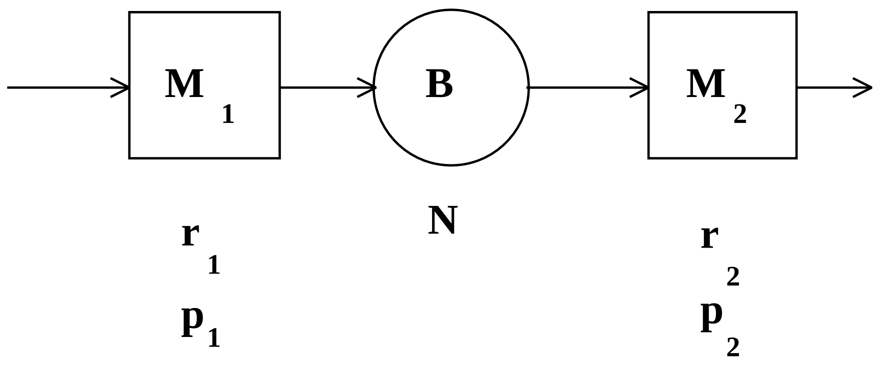

Figure 1-3: Buzacott-Type System

Buzacott (1967b) modeled a synchronous, saturated, finite buffer, single machine per stage, two-stage system (Figure 1-3) as a Markov process. Zero buffer transit times were assumed as well as reliable buffers. The model was a FLUM with ODFs and operated on the work conserving policy described in the previous section. Failure and repair times were geometrically distributed; and processing times were identical and deterministic with a value of 1. Although earlier works had dealt with similar analyses, (see review by Koenigsberg, 1959), Buzacott is generally credited with its popularization.

The model assumed *unlimited repair personnel* so that the repairs at different machines would be independent of one another. Informal studies by this author[4] and documented studies (Vladzievski, 1952; Buzacott, 1967b) have shown their assumption of geometric times between failures to be reasonable. As a justification for using

[4]Confidentiality restricts the publication of these results.

the ODFs assumption, Buzacott and Hanifin (1978a) provide data from Chrysler Corporation that showed that the majority of failures are ODFs. They also claimed other informal studies conclude the insignificance of TDFs impact on production lines.

Schick and Gershwin (1978) developed a similar model to Buzacott's that served as the foundation for much of the work in this thesis. (See Gershwin (1994) for details.) The failing buffer model in Chapter 4 is an extension of this model. They defined for i = 1,2

$$p_i = \begin{cases} \text{the probability that machine } i \text{ will fail in the next time unit} \\ \text{given that it is currently up and not blocked or starved by} \\ \text{any other machine,} \end{cases}$$

$$r_i = \begin{cases} \text{probability that machine } i \text{ will be repaired in the next time} \\ \text{unit given that it is currently down,} \end{cases}$$

$$N = \text{the capacity of the buffer.}$$

The state of the Markov process was defined as:

$$s(t) = (n(t), \alpha_1(t), \alpha_2(t))$$

where $s(t)$ is the state of the system at time t. We drop the t from now on unless it is required. The parameter n was the buffer level ($0 \leq n \leq N$) and α_i was the repair state of Machine i ($i = 1, 2$). $\alpha_i = 0$ if machine i was down and $\alpha_i = 1$ if it was up. Although ODFs were assumed, no description of the amount of material processed since the last failure was required in the system state because of the Markovian assumption concerning the occurrence of failures. The steady-state probability of state s was denoted as

$$\mathbf{p}(s) \quad = \quad \mathbf{p}(n, \alpha_1, \alpha_2). \tag{1.1}$$

Schick and Gershwin (1978) accounted for the occurrence of events only slightly more accurately than Buzacott (1967a). However, we focus on the Schick and Gershwin model because it eventually led to the first analytical decomposition for analyzing long flow lines of the type developed in Chapter 2.

Both models analyzed the trade-offs between buffer capacity and throughput and provided analytical solutions for determining the average throughput and average buffer level. The production rate or throughput of Machine i in parts per time unit, denoted by E_i, was determined as follows:

$$\begin{aligned} E_1 &= \text{prob } (\alpha_1 = 1, n < N) \\ E_2 &= \text{prob } (\alpha_2 = 1, n > 0). \end{aligned} \tag{1.2}$$

Since material is neither created nor destroyed in the line, E_1 equals E_2 when the system is in steady state. This concept is referred to as *conservation of flow* and is discussed in more detail in Section 2.1.

The *average buffer level* $\overline{n}$ is by definition:

$$\overline{n} \quad = \quad \sum_{n=0}^{N} n \sum_{\substack{\alpha_1=0,1 \\ \alpha_2=0,1}} \mathbf{p}(n, \alpha_1, \alpha_2). \tag{1.3}$$

These discrete time Markov process models only allowed the analysis of synchronized, homogeneous systems. This is often an unreasonable assumption on the manufacturing floor. Buzacott (1972), by assuming exponential service times, was able to introduce a model with a third parameter (in addition to p_i and r_i) for the processing rate of each stage. This allowed for the analysis of non-synchronized, non-homogeneous systems. Gershwin and Berman (1981) also developed a model that accounted for non-homogeneous processing rates. They defined

$$\mu_i = \begin{cases} \text{Average rate of production by Stage } i \text{ when it is opera-} \\ \text{tional, not starved and not blocked.} \end{cases} \tag{1.4}$$

They assumed operation times at Stage i were exponentially distributed with parameter μ_i. They also modelled the time between failures and the time to repair Machine i as random variables with exponential distributions with parameters p_i and r_i $(i = 1, 2)$. In this way, the system could be modeled as a discrete state/continuous time Markov process. This three-parameter system is illustrated in Figure 1-4.

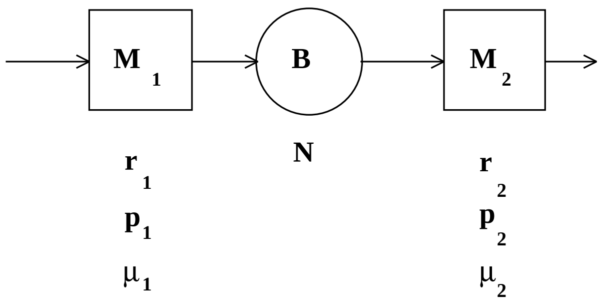

Figure 1-4: Three-Parameter Model

In this new model the efficiency E_i is not necessarily the same as the throughput and so they defined

$$\begin{aligned} P_i &= \text{the steady state non-isolated production rate of Stage } i \\ &= E_i\mu_i \qquad\qquad\qquad i = 1, 2. \end{aligned} \tag{1.5}$$

The variability of exponentially distributed processing times is high and many actual flow lines have deterministic processing times. This variability has the effect of blocking and starving the system more often than would actually take place, sometimes significantly. The variability in processing time would bias the average throughput estimate downward. One approach for dealing with non-homogeneous lines with deterministic processing times was to construct a homogeneous line that was approximately equivalent to the original system in that both would provide similar performance characteristic estimates (Okamura and Yamashina, 1977; Yeralan and Muth, 1987; Dallery *et al.*, 1989). In this way, existing homogeneous models could be used to analyze non-homogeneous production systems. However, these methods only had a limited success in accurately predicting performance measures.

Continuous fluid approximations treat discrete material traveling through the production system as if it were a continuous fluid. As early as Zimmern (1956) for TDFs and Sevast'yanov (1962) for ODFs, authors have used continuous material approximations to model discrete part flow lines to simultaneously deal with deterministic, non-homogeneous processing times. Other recent examples include Dubois and Forestier (1982), David *et al.* (1990), and Mitra (1988). Mitra (1988) identifies the use of continuous models as "...relatively uncommon and the subject of active research." Suri and Fu (1993) have investigated the reliability of continuous models.

They concluded the models are useful as long as production times are relatively small in relation to failure and repair times and that buffers are of a reasonable size.

Gershwin and Schick (1980) developed a two-stage, ODF formulation based on a continuous material representation that simultaneously handled deterministic and non-homogeneous processing times by also incorporating μ as a parameter representing a deterministic processing rate in addition to p and r. A detailed description of this model can be found in Gershwin (1994). The system was represented using a mixed state/continuous time Markov process. It is this model that we use as the basis of our long line decomposition. We, like them, define the system state as

$$s(t) \quad = \quad (n(t), \alpha_1(t), \alpha_2(t)).$$

However, since n is a continuous random variable, the probability mass function (1.1) is not adequate for defining the steady state probability of state s. Therefore, in addition to a mass function for the probabilities for boundary states $(N, 0, 0)$, $(N, 1, 0)$, $(N, 0, 1)$, $(N, 1, 1)$, $(0, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, and $(0, 1, 1)$, Gershwin and Schick defined the following density function

$$\int_x^{x+\delta x} f(y, \alpha_1, \alpha_2) dy \quad = \quad \begin{cases} \text{probability of finding Stage 1 in state} \\ \alpha_1\text{, Stage 2 in state } \alpha_2 \text{ with between} \\ x \text{ and } x+\delta x \text{ units of material in the} \\ \text{buffer.} \end{cases} \tag{1.6}$$

Small adjustments are also required to convert (1.2) and (1.3) to account for continuous material in determining expected throughputs and buffer levels.

Wijngaard (1979) developed a two-stage model based on a continuous material representation with TDFs that was virtually the same as earlier work by Zimmern

(1956). The Wijngaard and Gershwin and Schick models are significantly more important than the other two-stage continuous models mentioned above because they were done in a fashion which allowed the exploitation of the decomposition techniques which are addressed in Chapter 2. In fact, the first continuous material decomposition for non-homogeneous lines by Glassey and Hong (1993) was based on an adaptation of Wijngaard's model presented in Glassey and Hong (1986). The work in Chapter 4 on series-parallel machines is a direct extension of continuous fluid approximation models. Previous work in this area is discussed in the literature review in that chapter.

1.4 Thesis Outline

We develop a new set of decomposition equations for continuous material flow lines in Chapter 2. Chapter 3 focuses on improvements in speed, convergence and accuracy in the Dallery *et al.* (1988) algorithm, which is the standard method for solving sets of equations of the type developed in Chapter 2. Chapter 4 investigates the failing buffer phenomenon. Chapter 5 introduces methods of analyzing work centers with parallel machines. Finally, a summary of results, a brief description of future applications and research directions are presented in a conclusion in Chapter 6.

Chapter 2

A New Set of Flow Line Decomposition Equations

New decomposition equations are proposed for analyzing the behavior of saturated, asynchronous, unreliable, k-stage, finite buffer, continuous material flow lines (Figure 1-1). These new equations more accurately than earlier models account for many of the phenomena that are observed on the factory floor of mass production facilities. A continuous fluid model of the flow line is represented as a mixed state, continuous time Markov process. The steady state behavior of this model is determined using an already developed decomposition technique that builds on the results of exact two-stage analytic models. Quick steady state behavior estimates are important because they facilitate the rapid analysis of new manufacturing systems.

Transfer line decomposition methods typically work as follows. An original line is broken into $k-1$ two-stage lines as illustrated in Figure 2-1. The two stages of Line i represent the aggregate behavior of the flow line up- and down-stream of

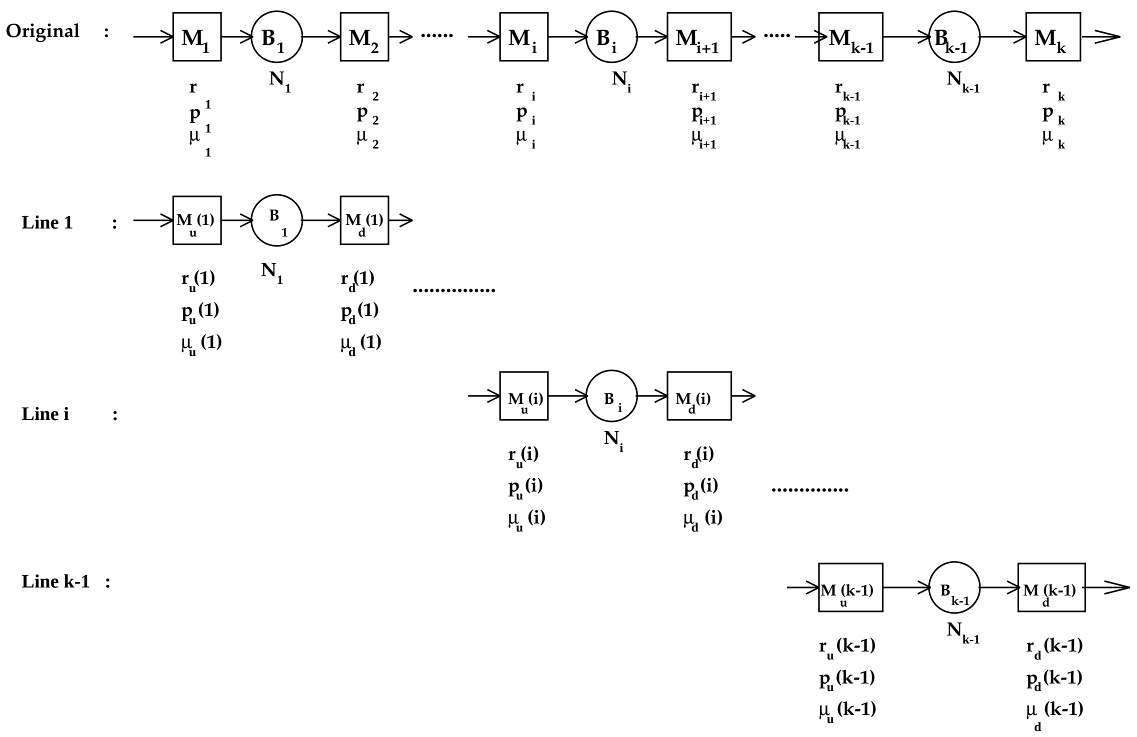

Figure 2-1: Decomposition Method

buffer B_i. The method requires the derivation of a set of equations that link the decomposed two-stage systems together. It is these equations that are developed in this chapter. Algorithms for solving these equations appear in Chapter 3. Such methods are efficient because two-stage systems can be rapidly analyzed.

The literature review in Section 2.1 describes existing decomposition techniques. We describe the continuous material model in Section 2.2. Section 2.3 presents the derivation of the new equations. A conclusion of the chapter is presented in Section 2.4.

2.1 Literature Review

A literature review of flow line decomposition is presented in this section. A more general literature review of flow lines is presented in Section 1.3.

As illustrated in the *Original* line in Figure 2-1, a flow line consists of a series of k stages ($M_i, i = 1, ..., k$) separated by storage buffers or inventory stores ($B_i, i = 1, ..., k-1$), each with finite capacity. Each stage i has parameters p_i, r_i, and μ_i, $i = 1, ..., k$ that are the failure, repair, and processing rates of that stage. Each buffer i is characterized by N_i, $i = 1, ..., k-1$, which is its capacity. In order to determine some of the flow line's performance characteristics (average throughput and average buffer levels), the system is modeled as a Markov process for which the steady state behavior is determined.

The state of the system is defined by the state of each stage (up or down) and the amount of material in each buffer. In a similar manner as with the two-stage system, the system state of the entire k-stage flow line is denoted as:

$$s = (n_1, \cdots, n_{k-1}, \alpha_1, \cdots, \alpha_k).$$

where n_i is the level of buffer B_i ($0 \leq n_i \leq N_i$) and α_i is the repair state of Machine i . The steady state probability of state s is denoted as

$$\mathbf{p}(s) = \mathbf{p}(n_1, \cdots, n_{k-1}, \alpha_1, \cdots, \alpha_k).$$

The state space of even moderately complex lines grows inordinately large for

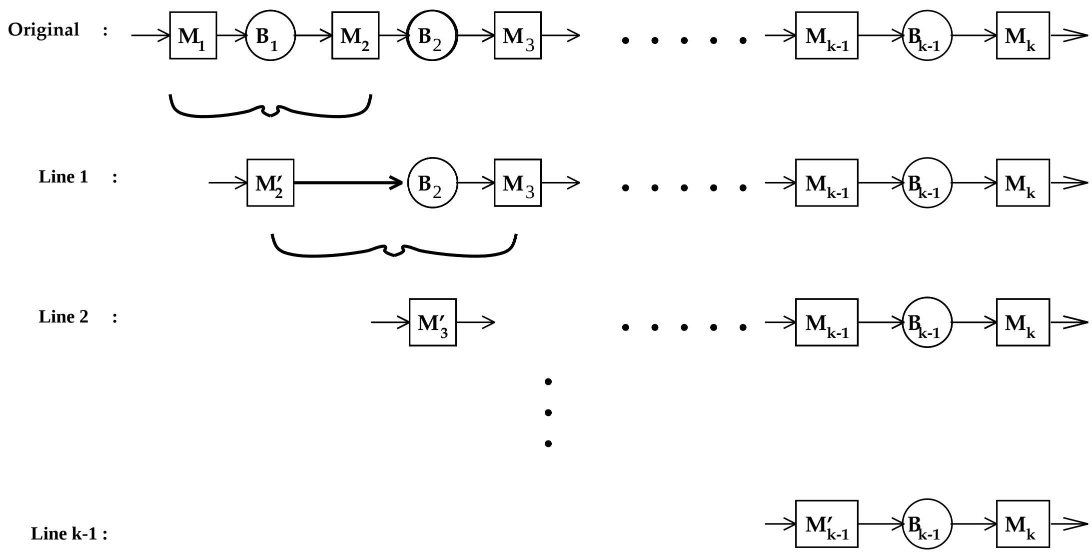

Figure 2-2: Aggregation Method

Markovian analysis. *Aggregation* is sometimes used to handle this complexity by replacing two-machine-one buffer sections of the line by a single equivalent machine (Ancelin and Semery, 1987; Terracol and David, 1987; De Koster 1987). Sections of the line are repeatedly aggregated until only one two-stage system remains. Figure 2-2 illustrates the modified lines obtained by continually aggregating the first two machines of the sequentially derived flow lines. The major deficiency in the aggregation models is that they only account for a unidirectional propagation of events. For example in Figure 2-2, when the first two stages are aggregated, there is no accounting for the effects that downstream blocking might have on the parameters of a previously aggregated stage.

In using the aggregation procedure, one can chose to aggregate in different orders. For example, one could proceed by continually grouping the first two machines in the line as in the example above. A second possibility is to aggregate by continually aggregating the last two machines. Different aggregations are required to determine different parameters such as the average inventory in the first versus last buffer. The results of various aggregation procedures can be quite different. This can be seen in the results of the numerical experiments of aggregation by Terracol and David (1987).

Sevast'yanov (1962) introduced a second approach that generalizes aggregation which we refer to as *decomposition*. The method breaks k-stage lines into $k-1$ two-stage lines as illustrated in Figure 2-1. Line i ($L(i)$) consists of two stages or *pseudo-machines*, $M_u(i)$ and $M_d(i)$, that represent the aggregate behavior of the flow line up- and down-stream respectively of buffer B_i. The line also contains a copy of buffer B_i, which is assumed to behave identically to the corresponding buffer in the original line. ıDecomposition equations establish the parameters for each of the pseudo-machines in the new two-stage systems. This can be viewed as an aggregation that accounts for both the up- and down-stream propagation of events. The state of the Markov process of $L(i)$ is defined as:

$$s_i = (n_i, \alpha_u(i), \alpha_d(i)) \text{ for } i = 1 \text{ to } k. \tag{2.1}$$

where n_i is a continuous random variable representing the buffer level of buffer B_i ($0 \leq n_i \leq N_i$), $\alpha_u(i)$ is the repair state of pseudo-machine $M_u(i)$, and $\alpha_d(i)$ is the repair state of pseudo-machine $M_d(i)$. We let $\alpha_u(i)$ and $\alpha_d(i)$ be 1 when the pseudo-machine is *up* and 0 when it is not operating or *down*.

The following are definitions from Gershwin (1994) of $M_u(i)$ and $M_d(i)$ being *up* or *down*:

$$\begin{aligned} M_u(i)\ \textit{down} &= \text{material is not flowing into } B_i \text{ because of a failure upstream} \\ &= \begin{cases} M_i\ \textit{down} & i=1 \\ M_i\ \textit{down} \text{ or } (M_u(i-1)\ \textit{down} \text{ and } n_{i-1}=0) & i=2,\cdots,k-1 \end{cases} \\ M_u(i)\ \textit{up} &= M_u(i) \text{ not } \textit{down} \qquad i=1,\cdots,k-1 \end{aligned} \tag{2.2}$$

$$\begin{aligned} M_d(i)\ \textit{down} &= \text{material is not flowing out of } B_i \text{ because of a failure downstream} \\ &= \begin{cases} M_{i+1}\ \textit{down} & i=k-1 \\ M_{i+1}\ \textit{down} \text{ or } (M_d(i+1)\ \textit{down} \text{ and } n_{i+1}=N_{i+1}) & i=1,\cdots,k-2 \end{cases} \\ M_d(i)\ \textit{up} &= M_d(i) \text{ not } \textit{down} \qquad i=1,\cdots,k-1. \end{aligned} \tag{2.3}$$

For discrete material models, the steady state probability of state s_i is denoted as

$$\mathbf{p_i}(s_i) = \mathbf{p_i}(n_i, \alpha_u(i), \alpha_d(i)) \qquad i=1,\cdots,k. \tag{2.4}$$

For the continuous material model, for values of n_i other than 0 and N_i, we define:

$$\int_x^{x+\delta x} f_i(y, \alpha_u(i), \alpha_d(i))dy = \begin{cases} \text{probability of finding } M_u(i) \text{ in state} \\ \alpha_u(i),\ M_d(i) \text{ in state } \alpha_d(i) \text{ and find-} \\ \text{ing between } x \text{ and } x+\delta x \text{ units of} \\ \text{material in } B_i. \end{cases} \tag{2.5}$$

The efficiency of $M_u(i)$ and $M_d(i)$ are $E_u(i)$ and $E_d(i)$, and are by definition:

$$\begin{aligned} E_u(i) &= \text{prob } (\alpha_u(i) = 1, n_i < N_i \text{ at } t) + \mathbf{p_i}(N_i, 1, 1)\frac{\mu_d(i)}{\mu_u(i)} \\ E_d(i) &= \text{prob } (\alpha_d(i) = 1, n_i > 0 \text{ at } t) + \mathbf{p_i}(0, 1, 1)\frac{\mu_u(i)}{\mu_d(i)} \quad i = 1, \cdots, k, \end{aligned} \tag{2.6}$$

where $\mu_u(i)$ and $\mu_d(i)$ are the effective average production rates of $M_u(i)$ and $M_d(i)$ defined in (2.16). The second terms in (2.6) are required for the continuous material model because a machine can be partially starved or blocked and producing at a reduced rate. Gershwin (1994) did not include this term in his definitions of E for the continuous material model.

The *non-isolated production rate* of machine i is the effective production rate of the machine when it is part of a larger system. We define

$$\begin{aligned} P_u(i) &= \text{the steady state non-isolated production rate of } M_u(i) \\ &= E_u(i)\mu_u(i) \qquad\qquad i = 1, \cdots, k-1 \\ P_d(i) &= \text{the steady state non-isolated production rate of } M_d(i) \\ &= E_d(i)\mu_d(i) \qquad\qquad i = 1, \cdots, k-1 \\ P(i) &= \text{the steady state non-isolated production rate of } L(i) \\ &= P_u(i) = P_d(i) \qquad\qquad i = 1, \cdots, k-1 \\ P &= \text{the steady state production rate of a } k\text{-stage line .} \end{aligned} \tag{2.7}$$

Since no mechanism creates nor destroys material in the entire line, flow is conserved. $P_u(i)$ and $P_d(j)$ for all i and j from 1 to k should be equal. Therefore, the calculation of $P_u(i)$ or $P_d(i)$ for any $L(i)$ should provide the throughput of the entire flowline (P). In principle, we have to show that the decomposition equations derived in this chapter satisfy this condition.

Since B_i is assumed to have identical behavior in both the long line and in $L(i)$, the average buffer level of B_i, $\overline{n_i}$ is by definition

$$\overline{n_i} = \sum_{\alpha_u(i)=0}^{1} \sum_{\alpha_d(i)=0}^{1} \left[\int_0^{N_i} n_i f_i(n_i, \alpha_u(i), \alpha_d(i)) dn_i + N_i \mathbf{p}(N_i, \alpha_u(i), \alpha_d(i)) \right] \quad i = 1, \cdots, k-1. \tag{2.8}$$

Following the classification scheme of Dallery and Gershwin (1992), there are a number of types of decompositions that differ according to the number of parameters describing the behavior of $M_u(i)$ and $M_d(i)$. To solve for the unknown parameters of the pseudo-machines, an equal number of equations as unknowns is required.

One-parameter machines Sevast'yanov (1962) extended a two-stage model to longer lines using decomposition. All the machines in the line had the same processing and repair rates. Therefore, he needed to estimate the $2(k-1)$ unknown failure rates (one value of $p_u(i)$ for each $M_u(i)$ and one value of $p_d(i)$ for each $M_d(i)$). To do this, he derived equations that described the failure behavior from the perspective of buffer. Unfortunately, Sevast'yanov had no numerical results because computers were not available at the time.

Choong and Gershwin (1987) later referred to these as the *interruption of flow equations* ($IOFs$). We use IOF equations in our decomposition and define for $i = 1, \cdots, k-1$

$$p_u(i)\delta t + o(\delta t) = \begin{cases} \text{the probability that } M_u(i) \text{ goes from } up \text{ to } down \text{ when} \\ n_i < N_i \text{ in } (t, t+\delta t) \text{ for small } \delta t \end{cases}$$

$$= \text{prob}\left[\alpha_u(i) = 0 \text{ at } t+\delta t | \alpha_u(i) = 1 \text{ and } n_i < N_i \text{ at } t\right]$$

$$p_d(i-1)\delta t + o(\delta t) = \begin{cases}\text{the probability that } M_d(i-1) \text{ goes from } up \text{ to } down \\ \text{when } n_i > 0 \text{ in } (t, t+\delta t) \text{ for small } \delta t\end{cases}$$

$$= \text{prob}\left[\alpha_d(i-1) = 0 \text{ at } t+\delta t | \alpha_d(i-1) = 1 \text{ and } n_{i-1} > 0 \text{ at } t\right]. \tag{2.9}$$

For one-parameter k-stage lines, (2.9) provides $2(k-2)$ equations. The $o(\delta t)$ is needed to represent multiple failures and repairs that could happen during $(t, t+\delta t)$ or other extremely unlikely events. Because it is small, this term plays no role in our analysis and is disregarded in all future expressions.

Two-parameter machines Gershwin (1987a) extended the decomposition technique by building on the Schick and Gershwin (1978) two-stage model as its basic building block. Like Sevast'yanov, the model assumed homogeneous processing rates. However, the Gershwin model allowed repair probabilities to vary from machine to machine. This required additional equations to estimate the $2(k-1)$ unknown repair parameters (one value of $r_u(i)$ for each $M_u(i)$ and one value of $r_d(i)$ for each $M_d(i)$). He defined from first principles what he called the *resumption of flow* equations ($ROFs$) which represents the repair behavior of the system from the perspective of each buffer. We use ROF equations in our decomposition and define for $i = 2, \cdots, k-1$,

$$r_u(i)\delta t = \begin{cases}\text{the probability that } M_u(i) \text{ goes from } down \text{ to } up \text{ in} \\ (t, t+\delta t) \text{ for small } \delta t\end{cases}$$

$$= \text{prob}\left[\alpha_u(i) = 1 \text{ at } t+\delta t | \alpha_u(i) = 1 \text{ at } t\right]$$

$$
\begin{aligned}
r_d(i-1)\delta t &= \begin{cases}\text{the probability that } M_d(i-1) \text{ goes from } \textit{down} \text{ to } \textit{up} \\ \text{in } (t, t+\delta t) \text{ for small } \delta t\end{cases} \\
&= \text{prob}\left[\alpha_d(i-1) = 1 \text{ at } t+\delta t | \alpha_d(i-1) = 0 \text{ at } t\right]. \qquad (2.10)
\end{aligned}
$$

For k-stage lines, (2.10) provides an additional $2(k-2)$ equations.

Gershwin (1987a) did not use IOFs (2.9). Instead he introduced two other sets of equations. One he called the *conservation of flow equations* ($COFs$). These equations assumed that the throughput rates at all machines must be equal because material is neither destroyed nor created within the system. The $(k-2)$ COF equations are:

$$
P(i) = P(1) \qquad i = 2, \cdots, k-1 \tag{2.11}
$$

Gershwin also derived what he called the *flow rate/idle time* (FR/IT) equations. The probability of a machine being idle can be expressed as the sum of the probabilities of it being blocked and starved. He defined

$$
\begin{aligned}
e_i &= \text{The steady state isolated efficiency of Stage } i \\
&= \frac{r_i}{r_i + p_i}, \qquad i = 1, \cdots, k, \qquad (2.12) \\
p_s(i) &= \text{The steady state probability of Stage } i \text{ being starved} \\
&= \mathbf{p}_{i-1}(0, 0, 1) \qquad i = 2, \cdots, k, \qquad (2.13) \\
p_b(i) &= \text{The steady state probability of Stage } i+1 \text{ being blocked} \\
&= \mathbf{p}_i(N_i, 1, 0) \qquad i = 1, \cdots, k-1. \qquad (2.14)
\end{aligned}
$$

The FR/IT relationship is

$$
P(i) = e_i \mu_i (1 - p_s(i-1) - p_b(i)) \qquad i = 2, \cdots, k-1. \tag{2.15}
$$

In Gershwin (1987a), this equation is only an approximation. However, in the continuous material case, the starvation and blockage is defined slightly differently and this equation is exact because a machine processing continuous material being simultaneously blocked and starved has probability 0 (Dallery *et al.*, 1989).

The Gershwin model results 1) were close to simulation and 2) demonstrated the tradeoffs between buffer capacity and throughput. Gershwin's algorithm required several levels of iteration to solve the equations. The method worked in some cases, but was generally slow and did not converge for a significant number of important cases. Soon after Gershwin (1987a), Dallery, David and Xie (1988) proposed a new algorithm (DDX) that solved the Gershwin equations in substantially less computing time than Gershwin's algorithm and was much more robust.

Three-parameter machines The system of interest to us is non-homogeneous for continuous material and requires a third parameter for each stage, in addition to r_i and p_i, to account for the different isolated production times. Some early attempts were made to treat three-parameter lines as modified two-parameter lines by altering parameters so that all the processing rates are equal (Dallery *et al.*, 1989) in a process called *homogenization*. Gershwin (1987b) tried to combine two equal processing rate machines together. One machine accounted for the failure rate and the other modified the machine speed. However, simulation experiments demonstrated that this method was not a good approximation. There were also significant problems with convergence caused by the required differences in parameters on adjacent machines.

The first three-parameter decomposition for discrete material systems was presented in Choong and Gershwin (1987), which used the Gershwin and Berman (1981) two-stage, discrete material, exponential processing time model as its basic building block. In addition to estimating $r_u(i)$, $p_u(i)$, $r_d(i)$, and $p_d(i)$, there were an additional $2(k-1)$ unknowns (one vale of $\mu_u(i)$ for each $M_u(i)$ and one value of $\mu_d(i)$ for each $M_d(i)$). The ROF $(2(k-2))$, IOF $(2(k-2))$, FR/IT $(k-2)$ and COF $(k-2)$ equations together provided $6(k-2)$ equations. The boundary equations (2.63) provided the remaining 6 required equations. Choong and Gershwin (1987) developed an algorithm similar to Gershwin (1987a). It had similar, but more severe difficulties in converging.

Gershwin (1989) solved the Choong and Gershwin equations with a DDX-type procedure. Using the IOF and the FR/IT equations, he derived equations for the expected production rate of each machine in the system from the perspective of each buffer. We use almost identical equations in our decomposition and define for $i = 2,\cdots,k-1$

$$\mu_u(i)\delta t = \begin{cases} \text{the amount of material produced by } M_u(i) \text{ in } (t, t+\delta t) \\ \text{for small } \delta t \text{ when } \alpha_u(i) = 1,\ n_{i-1} > 0, \text{ and } N_i < n_i \end{cases}$$

$$\mu_d(i-1)\delta t = \begin{cases} \text{the amount of material produced by } M_d(i-1) \text{ in } (t, t+ \\ \delta t) \text{ for small } \delta t \text{ when } \alpha_d(i-1) = 1,\ n_{i-1} > 0, \text{ and} \\ N_i < n_i. \end{cases} \tag{2.16}$$

Recognizing the need to simultaneously handle asynchronous lines and deterministic processing times, a number of papers focused on continuous material decompositions that dealt with both. Semery (1987) and Hong and Seong (1989) provided a continuous fluid decomposition in which they let $\mu_u(i) = \mu_i$ and $\mu_d(i) = \mu_{i+1}$ as one set of their equations. This procedure seemed to work well only when all $\mu's$ were identical. Glassey and Hong (1993) developed decomposition equations for the continuous material cases based on their earlier two-stage continuous material model

(Glassey and Hong, 1986) that was based on Wijngaard (1979). The development of their equations was also similar to Gershwin (1989). However, they substituted an equation derived from first principles to determine the $\mu_u(i)$ and $\mu_d(i)$ instead of using the COFs and the FR/ITs. We base our decomposition on the Gershwin and Schick (1980) two-stage ODF continuous material model. Like Choong and Gershwin (1987), we use the ROF $(2(k-2))$, IOF $(2(k-2))$, COF $(k-2)$, FR/IT $(k-2)$ and boundary equations (6) to provide the $6(k-1)$ required equations. We address the differences between our work and that of Glassey and Hong in the conclusion of this chapter.

2.2 Continuous Material Flow Model

We model the flow line as a three-parameter, continuous time, mixed state Markov process. We assume that machine processing rates are non-homogeneous, machines are unreliable with ODFs, yield is 100%, and failure and repair times are independent. Conceptually, we can describe the dynamics of a k-stage flow line with a complicated set of partial differential equations that represent the Markov process. From the solution to this set of equations, one can derive the steady state probability of any system state. However, solving such a set of equations is impossible at this time. Therefore, we employ the decomposition technique to approximate the steady state expected throughput and average buffer levels of the system. Before proceeding, we define the continuous material flow line model.

The continuous material model is unique in the way it treats the material being processed. The material is treated as though it is a continuous fluid. Each work center (M_i) is capable of processing material at a maximum rate (μ_i). It produces material at this rate unless impeded by a failure, blockage or a starvation of raw material. Unlike discrete material flow line models (Gershwin, 1987; Gershwin and Choong, 1987, etc.), the work center holds no material. The work center functions like a random water faucet. It allows the liquid to pass, but all the liquid is stored either before or after the faucet itself.

Buffer B_i can hold a maximum amount of material equal to N_i. The fact that buffers are finite clearly distinguishes the flow line models from other queuing models. The reason for this is that if a buffer becomes full, the machine immediately upstream cannot process material faster than the machine immediately downstream of the buffer. Similarly, if a buffer becomes empty, the machine immediately downstream cannot process material faster than the machine immediately upstream of the buffer. To capture the changing of processing rates, we first define

$$n_i(t) = \text{the amount of material in buffer } i \text{ at } t \qquad i = 1, \cdots, k-1.$$

By convention, we let

$$\begin{aligned} n_0(t) &= \infty \\ n_k(t) &= 0 \end{aligned} \tag{2.17}$$

for all t. Then we define

$$j_u^*(i,t) = max[j : j \leq i \text{ and } n_{j-1}(t) > 0] \qquad i = 1, \cdots, k-1 \tag{2.18}$$

$$j_d^*(i,t) = min[j : j \geq i \text{ and } n_j(t) < N_j] \qquad i = 1, \cdots, k-1. \tag{2.19}$$

Equation (2.18) states that all buffers between $M_{j_u^*(i,t)}$ and M_i are empty ($M_{j_u^*(i,t)}$

can be M_i.) This means that M_i cannot be producing at a rate faster than $M_{j_u^*(i,t)}$. Similarly, (2.19) indicates that M_i cannot be producing at a rate faster than $M_{j_d^*(i,t)}$. By finding the minimum of μ_i, $\mu_{j_u^*(i,t)}$, and $\mu_{j_d^*(i,t)}$, we can determine the actual rate of production M_i at t. Therefore,

$$\begin{aligned}
\mu_i(t) &= \text{the } \textit{instantaneous rate of production} \text{ at } M_i \text{ at time } t \\
&= min[\mu_i, \mu_{j_u^*(i,t)}, \mu_{j_d^*(i,t)}] \qquad (2.20) \\
\mu_i(t)\delta t &= \begin{cases} \text{the amount of material processed by } M_i \text{ in } (t, t+\delta t) \\ \text{for small } \delta t. \qquad i = 1, \cdots, k-1. \end{cases} \qquad (2.21)
\end{aligned}$$

Each machine in the system is considered to be unreliable and goes up and down intermittently. Failures are operation dependent (Section 1.2.2). Formally, we define

$$\begin{aligned}
p_i\delta t &= \begin{cases} \text{the probability that } M_i \text{ goes from } \textit{up} \text{ to } \textit{down} \text{ dur-} \\ \text{ing } (t, t+\delta t) \text{ when } n_i < N_i \text{ for small } \delta t \end{cases} \\
r_i\delta t &= \begin{cases} \text{the probability that } M_i \text{ goes from } \textit{down} \text{ to } \textit{up} \text{ dur-} \\ \text{ing } (t, t+\delta t) \text{ for small } \delta t. \end{cases} \qquad (2.22)
\end{aligned}$$

Notice that the failure rate is dependent on whether the machine is blocked or starved, while the repair rate is independent of such external activity. We assume that p_i is the failure rate of M_i when it is operating at rate of μ_i. However, when M_i is either blocked or starved its failure rate is reduced proportionally to the reduction in processing speed. Therefore, we have for the failure rate of machine i

$$\begin{aligned}
p_i(t)\delta t &= \begin{cases} \text{the probability that } M_i \text{ goes from } \textit{up} \text{ to } \textit{down} \text{ in } (t, t+ \\ \delta t) \text{ for small } \delta t \text{ at } t \end{cases} \\
&= \frac{\mu_i(t)}{\mu_i} p_i \delta t. \qquad (2.23)
\end{aligned}$$

Once the machine dynamics have been defined, the dynamics of the buffer level are not difficult to establish. Since no material is destroyed or created in the operation of the line, the following describes the dynamics of the buffer level

$$\dot{n}_i(t) = \mu_i(t) - \mu_{i+1}(t). \tag{2.24}$$

Now that the system has been described, we proceed in the next section to demonstrate how to approximate this system using the decomposition technique.

2.3 Derivation of Decomposition Equations

As discussed in the introduction to this chapter, the decomposition method breaks a k-stage line into k-1 two-stage systems. To employ the flow line decomposition method, parameters for each of the new two-stage lines must be established. Since each $L(i)$ has 6 unknown parameters ($r_u(i)$, $r_d(i)$, $p_u(i)$, $p_d(i)$, $\mu_u(i)$, and $\mu_d(i)$), an equal number of equations are required to solve for these parameters. In this section, the $6(k$-1) equations required for a k-stage line are derived. The equations are similar to those developed for the exponential decomposition of Gershwin (1989) which can be found more easily in Gershwin (1994). In Section 2.3.1, the IOF (2.9) equations are derived. In Section 2.3.2, the ROF (2.10) equations are derived. In Section 2.3.3, the production rate equations that are based on the COF (2.11) and FR/IT (2.15) equations are given. Finally, in Section 2.3.4, the 6 boundary conditions are provided.

2.3.1 Interruption of Flow Equations

The first of the decomposition equations are the IOF equations. They provide the failure rates of $M_u(i)$ and $M_d(i)$. It is important to note that an upstream failure, from the perspective of B_i, could be caused by either M_i actually failing or a starvation caused by B_{i-1} being empty simultaneously with $M_u(i-1)$ being down. The IOF equations incorporate both of these phenomena.

The IOF equations most clearly distinguish among the different sets of decomposition equations (Gershwin, 1987; Gershwin, 1989; Glassey and Hong, 1993, etc). These equations are especially complicated to derive in the continuous material case because of varying failure rates. For example, in the earlier decomposition methods, a stage is either starved or not starved because material and movements of material are discrete. In the case of continuous material, it is possible that an immediate upstream buffer of Stage i is empty and the upstream process of Stage i is slower than Stage i itself. During this period, Stage i is producing more slowly than μ_i. This reduces the failure rate in an ODF system.

The derivation of the IOF equations begins with the definition of the upstream failure rate of $L(i)$.

$$\begin{aligned} p_u(i)\delta t &= \text{prob}[M_u(i)\ \textit{down} \text{ at } t+\delta t | M_u(i)\ \textit{up} \text{ and } n_i < N_i \text{ at } t] \\ & \qquad\qquad\qquad\qquad i = 2, \cdots, k-1. \end{aligned} \tag{2.25}$$

For the next and all following equations related to $L(i)$, $i = 2,\cdots,k-1$. We substitute the recursive definition of $M_u(i)$ *down* (2.2) to obtain

$$\begin{aligned} p_u(i)\delta t &= \text{prob}[M_i \textit{ down} \text{ or } (n_{i-1}=0 \text{ and } M_u(i-1) \textit{ down}) \text{ at } t+\delta t \\ &\qquad\qquad |M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t]. \end{aligned} \tag{2.26}$$

We can refine this further by making a few observations concerning the relationship between $\{M_i \textit{ down}\}$ and $\{M_u(i-1) \textit{ down} \text{ and } n_{i-1}=0\}$. If M_i is *down*, $M_d(i-1)$ is *down* by definition (2.3). Gershwin (1994) demonstrates that (0,0,0) is a transient state of a two-stage line. Therefore, $\{M_i \textit{ down}\}$ and $\{M_u(i-1) \textit{ down} \text{ and } n_{i-1}=0\}$ are mutually exclusive and this results in

$$\begin{aligned} p_u(i)\delta t &= \text{prob}[M_i \textit{ down} \text{ at } t+\delta t|M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t] + \\ &\quad \text{prob}[n_{i-1}=0 \text{ and } M_u(i-1) \text{ is } \textit{down} \text{ at } t+\delta t \\ &\qquad |M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t] \end{aligned} \tag{2.27}$$

We define the mutually exclusive events

$$\begin{aligned} a &= M_i \textit{ up} \text{ at } t+\delta t \text{ and } M_u(i) \textit{ up}, n_i < N_i \text{ and } n_{i-1}=0 \text{ at } t \\ b &= M_i \textit{ down} \text{ at } t+\delta t \text{ and } M_u(i) \textit{ up}, n_i < N_i \text{ and } n_{i-1}=0 \text{ at } t \\ c &= M_i \textit{ down} \text{ at } t+\delta t \text{ and } M_u(i) \textit{ up}, n_i < N_i \text{ and } n_{i-1}\neq 0 \text{ at } t \\ d &= M_i \textit{ up} \text{ at } t+\delta t \text{ and } M_u(i) \textit{ up}, n_i < N_i \text{ and } n_{i-1}\neq 0 \text{ at } t. \end{aligned}$$

The following identities can be derived from first principles

$$\text{prob}(A \text{ and } B|B) = \frac{\text{prob}(A \text{ and } B)}{\text{prob}(B)} = \text{prob}(A|B) \tag{2.28}$$

$$\text{prob}[b \text{ or } c|a \text{ or } b \text{ or } c \text{ or } d] = \frac{\text{prob}[b|a+b]\text{prob}[a \text{ or } b] \text{ or } \text{prob}[c|c \text{ or } d]\text{prob}[c \text{ or } d]}{\text{prob}[a \text{ or } b \text{ or } c \text{ or } d]}. \tag{2.29}$$

Using (2.28), the first term of (2.27) is equal to

$$\text{prob}[\overbrace{M_i \textit{ down} \text{ at } t+\delta t,\ M_u(i) \textit{ up}, \text{ and } n_i < N_i \text{ at } t}^{b \text{ or } c} \mid \underbrace{M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t}_{a \text{ or } b \text{ or } c \text{ or } d}]. \tag{2.30}$$

We use (2.29) to analyze (2.30). The first term of (2.27) then becomes equal to

$$\left[\left(\text{prob}[\overbrace{M_i \textit{ down} \text{ at } t+\delta t,\ M_u(i) \textit{ up},\ n_i < N_i, \text{ and } n_{i-1} = 0 \text{ at } t}^{b} \mid \underbrace{M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ and } n_{i-1} = 0 \text{ at } t}_{a \text{ or } b}] \right) \times \underbrace{\left(\frac{\text{prob}(M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ and } n_{i-1} = 0 \text{ at } t)}{\text{prob}(M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t)} \right)}_{\frac{\text{prob }(a \text{ or } b)}{\text{prob }(a \text{ or } b \text{ or } c \text{ or } d)}} \right] + .$$

$$\left[\left(\text{prob}[\overbrace{M_i \textit{ down} \text{ at } t+\delta t,\ M_u(i) \textit{ up},\ n_i < N_i \text{ and } n_{i-1} \neq 0 \text{ at } t}^{c} \mid \underbrace{M_u(i) \textit{ up},\ n_i < N_i, \text{ and } n_{i-1} \neq 0 \text{ at } t}_{c \text{ or } d}] \right) \times \underbrace{\left(\frac{\text{prob}(M_u(i) \textit{ up},\ n_i < N_i, \text{ and } n_{i-1} \neq 0 \text{ at } t)}{\text{prob}(M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t)} \right)}_{\frac{\text{prob }(c \text{ or } d)}{\text{prob }(a \text{ or } b \text{ or } c \text{ or } d)}} \right]$$

We then use (2.28) to obtain

$$\left[\begin{array}{c}\text{prob}[M_i\ \mathit{down}\text{ at }t+\delta t|M_u(i)\ \mathit{up},\ n_i<N_i,\text{ and }n_{i-1}=0\text{ at }t]\times\\ \left(\frac{\text{prob}(M_u(i)\ \mathit{up},\ n_i<N_i,\text{ and }n_{i-1}=0\text{ at }t)}{\text{prob}(M_u(i)\ \mathit{up}\text{ and }n_i<N_i\text{ at }t)}\right)\end{array}\right]+$$
$$\left[\begin{array}{c}\text{prob}[M_i\ \mathit{down}\text{ at }t+\delta t|M_u(i)\ \mathit{up},\ n_i<N_i\text{ and }n_{i-1}\neq 0\text{ at }t]\times\\ \left(\frac{\text{prob}(M_u(i)\ \mathit{up},\ n_i<N_i\text{ and }n_{i-1}\neq 0\text{ at }t)}{\text{prob}(M_u(i)\ \mathit{up}\text{ and }n_i<N_i\text{ at }t)}\right)\end{array}\right]. \tag{2.31}$$

If $M_u(i)$ is up and $n_{i-1}=0$ at t, M_i and M_{i-1} must be up at t by definition of $M_u(i)$ (2.3). The only way for $n_{i-1}=0$, M_i *up* and M_{i-1} *up* at t to be true when the system is in steady state is if the upstream machine is not faster than the downstream machine (*i.e.*, $\mu_u(i-1)\leq\mu_d(i-1)$). If the upstream machine were faster, there would always be material in B_{i-1} when $M_u(i)$ is up. So, when $\mu_u(i-1)\leq\mu_d(i-1)$, the probability of M_i failing at time $t+\delta t$ is reduced proportionately to the reduction in processing rate under the assumption of ODFs. If $\mu_u(i-1)>\mu_d(i-1)$, the second factor in the first term of (2.31) has a value of zero because it implies that $n_{i-1}=0$, M_i *up* and M_{i-1} *up* all be true at t, which is impossible by the same argument as above. Therefore, the first factor of the first term of (2.31) is

$$\text{prob}[M_i\ \mathit{down}\text{ at }t+\delta t|M_u(i)\ \mathit{up}\text{ and }n_i<N_i\text{ and }n_{i-1}=0\text{ at }t]=$$
$$\begin{cases}\frac{\mu_u(i-1)}{\mu_i}p_i\delta t & \text{when }\mu_u(i-1)\leq\mu_d(i-1)\\ 0 & \text{otherwise.}\end{cases} \tag{2.32}$$

Dallery *et al.* (1989) proves the following for continuous material flow lines:

$$\text{prob}(n_{i-1} = 0, \text{ and } N_i = n_i \text{ at time } t) = 0 \quad i = 2, \cdots, k-1. \tag{2.33}$$

By the definition of $M_u(i)$ *up*, (2.4) and (2.33) we have

$$\begin{aligned}
\mathbf{p}_{i-1}(0,1,1) &= \text{prob}(M_u(i-1)\ up\ , M_d(i-1)\ up \text{ and } n_{i-1} = 0 \text{ at } t) \\
&= \text{prob}(M_u(i-1)\ up\ , M_i\ up, (n_i < N_i \text{ or } \{n_i = N_i \text{ and } M_d(i)\ up\}) \\
&\qquad \text{and } n_{i-1} = 0 \text{ at } t) \\
&= \text{prob}(M_u(i-1)\ up\ , M_i\ up, (n_i < N_i \text{and } n_{i-1} = 0 \text{ at } t) \\
&= \text{prob}(M_u(i)\ up, n_i < N_i \text{ and } n_{i-1} = 0 \text{ at } t)
\end{aligned} \tag{2.34}$$

By (2.6), we have

$$E_u(i) = \text{prob}(M_u(i)\ up \text{ and } n_i < N_i) \text{ at t}) + \mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)} \tag{2.35}$$

By (2.34) and (2.35) the second factor of the first term of (2.31) is then

$$\left(\frac{\text{prob}(M_u(i)\ up,\ n_i < N_i \text{ and } n_{i-1} = 0 \text{ at } t)}{\text{prob}(M_u(i) \text{ up and } n_i < N_i \text{ at } t)}\right) = \frac{\mathbf{p}_{i-1}(0,1,1)}{E_u(i) - \mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}. \tag{2.36}$$

The first factor of the second term of (2.31), which accounts for when stage i is neither blocked nor starved, is, by definition (2.9),

$$\text{prob}[M_i\ down \text{ at } t + \delta t | M_u(i)\ up \text{ and } n_i < N_i \text{ and } n_{i-1} \neq 0 \text{ at } t] = p_i \delta t. \tag{2.37}$$

Since $n_{i-1} = 0$ and $n_{i-1} \neq 0$ are mutually exclusive, collectively exhaustive events, we have

$$\begin{aligned}
&\text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{at}\ t) = \\
&\quad \text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{and}\ n_{i-1} = 0\ \text{at}\ t) + \\
&\quad \text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{and}\ n_{i-1} \neq 0\ \text{at}\ t).
\end{aligned}$$

After a little algebraic manipulation, the following results:

$$\begin{aligned}
&\frac{\text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{and}\ n_{i-1} \neq 0\ \text{at}\ t)}{\text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{at}\ t)} \\
&= 1 - \left(\frac{\text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{and}\ n_{i-1} = 0\ \text{at}\ t)}{\text{prob}(M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{at}\ t)} \right) \\
&= 1 - \frac{\mathbf{p}_{i-1}(0,1,1)}{E_u(i) - \mathbf{p}_\mathbf{i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}. \qquad (2.38)
\end{aligned}$$

We then substitute (2.32), (2.36), (2.37), and (2.38) into (2.31) to obtain

$$\text{prob}[M_i\ \textit{down}\ \text{at}\ t+\delta t | M_u(i)\ \textit{up}\ \text{and}\ n_i < N_i\ \text{at}\ t]$$

$$\begin{aligned}
&= \frac{\mu_u(i-1)}{\mu_i} p_i \delta t \left(\frac{\mathbf{p}_{i-1}(0,1,1)}{E_u(i) - \mathbf{p}_\mathbf{i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}} \right) \\
&\quad + p_i \delta t \left(1 - \frac{\mathbf{p}_{i-1}(0,1,1)}{E_u(i) - \mathbf{p}_\mathbf{i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}} \right). \qquad (2.39)
\end{aligned}$$

When $\mu_d(i-1) > \mu_u(i-1)$, $\mathbf{p}_{i-1}(0,1,1)$ must equal 0 because it is a transient state and the value of the first factor of the first term becomes unimportant. Therefore, a special equation is not required when $\mu_d(i-1) > \mu_u(i-1)$, as would first appear from (2.32).

We now examine the second term of (2.27). Since $M_d(i)$ *up* and $M_d(i)$ *down* are mutually exclusive and collectively exhaustive, we have

$$\text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ at } t+\delta t | M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t]$$

$$\begin{aligned} = \quad & \text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ and } M_d(i-1) \textit{ up} \text{ at } t+\delta t \\ & \qquad\qquad |M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t] + \\ & \text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ and } M_d(i-1) \textit{ down} \text{ at } t+\delta t \\ & \qquad\qquad |M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t]. \end{aligned} \tag{2.40}$$

$M_u(i-1)$	$M_d(i-1)$	$n_{i-1} = 0$	$0 < n_{i-1} < \mu_d(i-1)\delta t$
down	up	1	2
up	down	$--$	$--$
down	down	$--$	$--$
up	up	3	$--$

Table 2.1: Previous State Possibilities

$\mathbf{p}_{i-1}(0,0,0)$ is transient and so the second term of (2.40) is eliminated. Table 2.1 enumerates all possible states of $M_u(i-1)$, $M_d(i-1)$ and n_{i-1} at t. The symbol $--$ indicates zero probability of $n_{i-1} = 0$, $M_u(i-1)$ *down*, and $M_d(i-1)$ *up* at $t+\delta t$ given that the system is in the state listed in the table at t. Most cases have probability of zero. Consequently, the only possible mutually exclusive states that the system could be in at t that could lead to the state of $n_{i-1} = 0$, $M_u(i-1)$ *down*, and $M_d(i-1)$ *up* at $t + \delta t$ are

1. $n_{i-1} = 0$, $M_u(i-1)$ *down* and $M_d(i-1)$ *up* at t, and nothing happens in (t,$t + \delta t$), with probability $1 - r_u(i-1)\delta t$ *or,*
2. $0 < n_{i-1} < \mu_d(i-1)\delta t$, $M_u(i-1)$ *down* and $M_d(i-1)$ *up* at t, the buffer empties in (t,$t + \delta t$), with probability $1 - r_u(i-1)\delta t - p_d(i-1)\delta t$ *or,*
3. $n_{i-1} = 0$, $M_u(i-1)$ *up* and $M_d(i-1)$ *up* at t, and $M_u(i-1)$ fails in (t,$t + \delta t$), with probability $p_u(i-1)\delta t$.

Equation (2.40) can therefore be written

$$\text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ at } t + \delta t | M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t]$$

$$\begin{aligned}
&= \text{prob}\left[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ and } M_d(i-1) \textit{ up} \text{ at } t\right. \\
&\qquad\qquad \left.| M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t\right] (1 - r_u(i-1)\delta t) + \\
&\text{prob}\left[0 < n_{i-1} < \mu_d(i-1)\delta t \text{ and } M_u(i-1) \textit{ down} \text{ and } M_d(i-1) \textit{ up} \text{ at } t\right. \\
&\qquad\qquad \left.| M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t\right] (1 - r_u(i-1)\delta t - p_d(i-1)\delta t) + \\
&\text{prob}\left[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ up} \text{ and } M_d(i-1) \textit{ up} \text{ at } t\right. \\
&\qquad\qquad \left.| M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t\right] p_u(i-1)\delta t.
\end{aligned} \tag{2.41}$$

If $M_u(i)$ is up at t, then M_i is up at t by the definition of $M_u(i)$ *up*. If M_i is up and $n_i < N_i$ at t, then $M_d(i-1)$ must also be up by definition. Therefore, the first term of (2.41) is zero. The second term is, by the definition of conditional probability,

$$\text{prob}[0 < n_{i-1} \leq \mu_d(i-1)\delta t \text{ and } M_u(i-1) \textit{ down} \text{ and } M_d(i-1) \textit{ up} \text{ at } t$$
$$|M_u(i) \textit{ up} \text{ and is } n_i < N_i \text{ at } t](1 - r_u(i-1)\delta t - p_d(i-1)\delta t)$$

$$= \frac{\text{prob}\begin{pmatrix} 0 < n_{i-1} \leq \mu_d(i-1)\delta t \text{ and } M_u(i-1) \textit{ down} \text{ and} \\ M_d(i-1) \textit{ up} \text{ and } M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t \end{pmatrix}}{\text{prob}[M_u(i) \textit{ up} \text{ and } n_i < N_i \text{ at } t]} \times$$

$$(1 - r_u(i-1)\delta t - p_d(i-1)\delta t). \tag{2.42}$$

The probability of Stage i being simultaneously blocked and starved is zero (2.33). As an approximation, we assume that the probability that stage i is blocked and almost starved ($0 < n_{i-1} \leq \mu_d(i-1)\delta t$) at the same time is also zero. Therefore, Buffer $i-1$ is almost empty implies Buffer i cannot be full. Therefore, the statement that $M_u(i)$ *is up and* $n_i < N_i$ *at* t is not required in the numerator. If $M_d(i-1)$ *up* and $n_{i-1} > 0$, then M_i must be up and therefore $M_u(i)$ must be up. Therefore, simplification of (2.42) leads to

$$\frac{\text{prob}\left(\begin{array}{c} 0 < n_{i-1} < \mu_d(i-1)\delta t \text{ and} \\ M_u(i-1) \; down \text{ and } M_d(i-1) \; up \text{ at } t \end{array}\right)}{\text{prob}[M_u(i) \; up \text{ and } n_i < N_i \text{ at } t]} (1 - r_u(i-1)\delta t - p_d(i-1)\delta t)$$

$$= \frac{\int_0^{\mu_d(i-1)\delta t} f_{i-1}(x,0,1)dx}{\text{prob}[M_u(i) \; up \text{ and } n_i < N_i \text{ at } t]} \times$$

$$(1 - r_u(i-1)\delta t - p_d(i-1)\delta t)$$

$$= \frac{f_{i-1}(0,0,1)}{\text{prob}[M_u(i) \; up \text{ and } n_i < N_i \text{ at } t]} \mu_d(i-1)\delta t$$

$$= \left(\frac{f_{i-1}(0,0,1)}{E_u(i) - \mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}\right) \mu_d(i-1)\delta t. \tag{2.43}$$

For the third term of (2.41), by a straightforward, almost identical argument to that found in Gershwin (1994), we have

$$\text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \; up \text{ and } M_d(i-1) \; up \text{ at } t$$
$$|M_u(i) \; up \text{ and } n_i < N_i \text{ at } t] p_u(i-1)\delta t$$

$$= \left(\frac{\mathbf{p}_{i-1}(0,1,1)}{E_u(i) - \mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}\right) p_u(i-1)\delta t. \tag{2.44}$$

Equation (2.40) can therefore be written as the sum of (2.43) and (2.44) or

$$\left(\frac{f_{i-1}(0,0,1)}{E_u(i)-\mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}\right)\mu_d(i-1)\delta t+\left(\frac{\mathbf{p}_{i-1}(0,1,1)}{E_u(i)-\mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}\right)p_u(i-1)\delta t. \tag{2.45}$$

The following identity is also derived in Gershwin (1994) (equation (3.205) with 1 replaced by the subscript u and 2 replaced by the subscript d):

$$\mathbf{p}_i(0,0,1)r_u(i)=f_i(0,0,1)\mu_d(i)+\mathbf{p}_i(0,1,1)p_u(i). \tag{2.46}$$

We then combine (2.45) and (2.46) to obtain the following expression that is equivalent to (2.40)

$$\left(\frac{\mathbf{p}_{i-1}(0,0,1)}{E_u(i)-\mathbf{p_i}(N_i,1,1)\frac{\mu_d(i)}{\mu_u(i)}}\right)r_u(i-1)\delta t. \tag{2.47}$$

This can be further refined. By the definition (2.6), we have

$$E_u(i)=\frac{P(i)}{\mu_u(i)}. \tag{2.48}$$

If we substitute (2.48) into (2.47), combine the result with (2.39), and divide both sides by δt the following IOF equation is obtained

$$\begin{aligned} p_u(i) &= \frac{\mu_u(i-1)}{\mu_i} p_i \left(\frac{\mathbf{p}_{i-1}(0,1,1)\mu_u(i)}{P(i) - \mathbf{p_i}(N_i,1,1)\mu_d(i)} \right) + p_i \left(1 - \frac{\mathbf{p}_{i-1}(0,1,1)\mu_u(i)}{P(i) - \mathbf{p_i}(N_i,1,1)\mu_d(i)} \right) + \\ & \left(\frac{\mathbf{p}_{i-1}(0,0,1)\mu_u(i)}{P(i) - \mathbf{p_i}(N_i,1,1)\mu_d(i)} \right) r_u(i-1) \qquad i = 2, \cdots, k-1. \end{aligned} \tag{2.49}$$

With a little algebraic manipulation, (2.49) becomes

$$\begin{aligned} p_u(i) &= p_i \left(1 + \frac{\mathbf{p}_{i-1}(0,1,1)\mu_u(i)}{P(i) - \mathbf{p_i}(N_i,1,1)\mu_d(i)} \left(\frac{\mu_u(i-1)}{\mu_i} - 1 \right) \right) + \\ & \left(\frac{\mathbf{p}_{i-1}(0,0,1)\mu_u(i)}{P(i) - \mathbf{p_i}(N_i,1,1)\mu_d(i)} \right) r_u(i-1) \qquad i = 2, \cdots, k-1. \end{aligned} \tag{2.50}$$

Equation (2.49) is sufficient for algorithmic purposes. However, (2.50) provides a greater insight into the differences between the Gershwin (1989) equations for the exponentially distributed processing times and the new continuous material, deterministic processing time model. When $\mu_u(i-1) > \mu_d(i-1)$, the first term of the equation reduces to p_i, as in Gershwin (1989). This is because it is only when the downstream machine is slow enough to starve the upstream of material that the failure rate is reduced. Since the exponential model only moves discrete parts, the failure rate is never reduced and the other factor is not required. The second term of (2.50) is also almost identical to the exponential model except that μ_i represents the rate of a Poisson process in the exponential model and in our model it is a deterministic production rate.

An almost identical derivation results in the p_d equation:

$$\begin{aligned} p_d(i) &= p_{i+1}\left(1 + \frac{\mathbf{p}_{i+1}(N_{i+1},1,1)\mu_d(i)}{P(i) - \mathbf{p_i}(0,1,1)\mu_u(i)}\left(\frac{\mu_d(i+1)}{\mu_{i+1}} - 1\right)\right) + \\ & \left(\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)\mu_d(i+1)}{P(i) - \mathbf{p_i}(0,1,1)\mu_u(i)}\right) r_d(i+1) \qquad i = 1, \cdots, k-2. \end{aligned} \tag{2.51}$$

When all the μs are the same, it is not difficult to show

$$\mu_i = \mu_j \qquad\qquad i,j = 1,\cdots,k$$

and

$$\mu_i = \mu_u(i) = \mu_d(i) \qquad\qquad i = 1,\cdots,k-1.$$

By substituting μ for all the processing rate variables in (2.50) and (2.51), the entire set of decomposition equations reduces to the Gershwin (1989) equations.

2.3.2 Resumption of Flow Equations

The second of the decomposition equations are the ROF equations that provide the repair rates of $M_u(i)$ and $M_d(i)$. It is important to note that a failure from the perspective of Buffer i could be caused by either Machine i failing or a starvation caused by Buffer $i-1$ being empty simultaneously with $M_u(i-1)$ being down. The ROF equations account for the recovery from both of these conditions. The following derivation is terse version of that found for the exponential long line decomposition in Gershwin (1994) and similar to that suggested in Sevest'yanov (1962). The repair rate of $M_u(i)$ is approximated as a weighted average of the rates of repair as follows:

$$r_u(i)\delta t = A(i-1)X(i) + B(i)X'(i) \qquad i = 2, \cdots, k-1 \tag{2.52}$$

where

$$\begin{aligned} A(i-1) &= \text{prob}[M_i \textit{ up} \text{ and NOT}(n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down}) \text{ at } t + \delta t \\ &\qquad\qquad\qquad\qquad |n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ at } t] \\ &= r_u(i-1)\delta t \end{aligned} \tag{2.53}$$

$$\begin{aligned} B(i) &= \text{prob}[M_i \textit{ up} \text{ and NOT}(n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down}) \text{ at } t + \delta t \\ &\qquad\qquad\qquad\qquad |M_i \textit{ down} \text{ at } t] \\ &= r_i\delta t \end{aligned} \tag{2.54}$$

$$\begin{aligned} X(i) &= \text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ at } t \\ &\qquad |M_i \textit{ down} \text{ or } (n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down}) \text{ at } t] \\ &= \frac{\text{prob}[n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down} \text{ at } t]}{\text{prob}[M_i \textit{ down} \text{ or } \{n_{i-1} = 0 \text{ and } M_u(i-1) \textit{ down}\} \text{ at } t]} \\ &= \frac{\mathbf{p}_{i-1}(0,0,1)}{\text{prob}[M_u(i) \textit{ down}]} \end{aligned} \tag{2.55}$$

$$X'(i) = 1 - X(i). \tag{2.56}$$

Since the rate of transitions M_i makes in and out of the failed state must be equal, we have the following balance equation

$$r_u(i)\text{prob}[M_u(i)\ down] \quad = \quad p_u(i)E_u(i). \tag{2.57}$$

We can then simplify $X(i)$ by solving (2.57) for prob$[M_u(i)\ down]$ and substituting the result into (2.55) to obtain

$$X(i) = \quad \tfrac{\mathbf{p}_{i-1}(0,0,1)r_u(i)}{p_u(i)E_u(i)} \quad = \frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i)}. \tag{2.58}$$

Finally, after substituting (2.53), (2.54) and (2.58) into (2.52) and substituting for $E_u(i)$ using (2.48), we obtain the ROF equation

$$r_u(i) \quad = \quad r_u(i-1)\frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i)} + r_i\left(1 - \frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i)}\right)$$

$$i = 2, \cdots, k-1. \tag{2.59}$$

Equation (2.59 is identical to those found in almost all the other three-parameter decompositions. By a similar derivation

$$r_d(i) \quad = \quad r_d(i+1)\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)r_d(i)\mu_d(i)}{p_d(i)P(i)} + r_{i+1}\left(1 - \frac{\mathbf{p}_{i+1}(N_{i+1},1,0)r_d(i)\mu_d(i)}{p_d(i)P(i)}\right)$$

$$i = 1, \cdots, k-2. \tag{2.60}$$

2.3.3 Processing Rate Equations

The final set of decomposition equations are derived for the $\mu_u(i)$'s (processing rate of the first stage in $L(i)$) and the $\mu_d(i)$'s (processing rate of the second stage in $L(i)$). These equations are derived directly from the FR/IT equations (2.15) and the COF equations (2.11). Since the derivation of the processing rate equations is identical to that found in Gershwin (1994) for the exponential long line model, the following are provided in their final form:

$$\begin{aligned}
\mu_u(i) &= \frac{1}{e_u(i)}\left\{\frac{1}{\frac{1}{P(i)}+\frac{1}{e_i\mu_i}-\frac{1}{e_d(i-1)\mu_d(i-1)}}\right\} i=2,\cdots,k-1 && (2.61)\\
\mu_d(i) &= \frac{1}{e_d(i)}\left\{\frac{1}{\frac{1}{P(i)}+\frac{1}{e_{i+1}\mu_{i+1}}-\frac{1}{e_u(i+1)\mu_u(i+1)}}\right\} i=1,\cdots,k-2. && (2.62)
\end{aligned}$$

2.3.4 Boundary Conditions

So far, we have presented $6(k-2)$ equations. The remaining six boundary equations are also identical in all the decomposition models.

$$\begin{aligned}
r_u(1) &= r_1\\
p_u(1) &= p_1\\
\mu_u(1) &= \mu_1\\
r_d(k-1) &= r_k\\
p_d(k-1) &= p_k\\
\mu_d(k-1) &= \mu_k. && (2.63)
\end{aligned}$$

2.4 Conclusion

In this chapter, we derived the decomposition equations for the continuous material decomposition based on the Gershwin and Schick (1980) ODF two-stage model. Our model is designed to gain an understanding of the behavior of saturated, asynchronous, unreliable, k-stage, finite buffered, deterministic processing time flow lines. We demonstrate an improved method for simultaneously solving these equations in the next chapter. Simulation and numerical experiments are conducted to test our results.

We conclude with the important differences between this work and the work of Glassey and Hong (1993) (Section 2.4.1) and a summary of the equations (Section 2.4.2).

2.4.1 Comparison with Glassey and Hong

The work presented in this chapter deals with a system which is similar to that described in Glassey and Hong (1993). The main differences are

- The interpretation of ODFs.
- The formulation of the Markov chain.
- The solution technique.

Operation Dependent Failures One of the most important differences between the two approaches is in the treatment of operation dependent failures. Glassey and

Hong did not adjust the failure rates of individual stages in proportion to reductions in processing speeds caused by slower up- or downstream stages. The first term of our IOF equation (2.39) is a weighted average of p_i and a failure rate which is reduced in proportion with reductions in the processing rate of M_i. Not adjusting the failure rate would lead to a lower throughput estimate by the Glassey and Hong model.

We claim the Glassey and Hong model misrepresents the physical system. The continuous material approximation is a representation of discrete part production. When a machine's processing rate is reduced due to a slower machine up- or downstream, idle time is created between the processing of parts. Suppose the processing rate of a machine is reduced from its normal rate μ to μ'. For every T time units, the machine only operates for $T(\mu'/\mu)$ time units. Since failures are operation-dependent, p should be reduced by the fraction of time the machine is processing divided by the amount of time it is actual running $(p\mu'/\mu)$. That is, failures should be a function of the number of pieces produced, not the machine runtime.

Markov Process Discrepancy There is an error in the Glassey and Hong IOF equations. A failure of $M_u(i)$ can be caused by one of three events: (1) M_i can fail; (2) B_{i-1} can be empty because M_i is processing faster than M_{i-1} and $M_u(i-1)$ fails; or (3) $M_u(i-1)$ can already be *down* but B_{i-1} still has some material and B_{i-1} empties out. Glassey and Hong did not include failures of type (3) in the formulation of their IOF equation. This is equivalent to leaving out (2.41) of the formulation. This error could potentially bias their estimate of throughput in a positive direction.

Solution Technique With the exception of the μ_u and μ_d equations, the Glassey and Hong equations are similar to ours. However, their equations were not written in a way to take advantage of the DDX algorithm Dallery *et. al.* (1988), so they needed to use a standard non-linear equation solver to determine their steady state solution. In Chapters 3, we demonstrate the advantages of being able to use the DDX algorithm in terms of speed and robustness of solving the decomposition equations.

2.4.2 Summary of Equations

Upstream Equations

$$\begin{aligned}
p_u(i) &= p_i\left(1+\frac{\mathbf{p}_{i-1}(0,1,1)\mu_u(i)}{P(i)-\mathbf{p_i}(N_i,1,1)\mu_d(i)}\left(\frac{\mu_u(i-1)}{\mu_i}-1\right)\right)+\\
&\quad \left(\frac{\mathbf{p}_{i-1}(0,0,1)\mu_u(i)}{P(i)-\mathbf{p_i}(N_i,1,1)\mu_d(i)}\right)r_u(i-1)\\
r_u(i) &= r_u(i-1)\frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i)}+r_i\left(1-\frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i)}\right)\\
\mu_u(i) &= \frac{1}{e_u(i)}\left\{\frac{1}{\frac{1}{P(i)}+\frac{1}{e_i\mu_i}-\frac{1}{e_d(i-1)\mu_d(i-1)}}\right\} i=2,\cdots,k-1 \qquad (2.64)
\end{aligned}$$

Downstream Equations

$$\begin{aligned}
p_d(i) &= p_{i+1}\left(1+\frac{\mathbf{p}_{i+1}(N_{i+1},1,1)\mu_d(i)}{P(i)-\mathbf{p_i}(0,1,1)\mu_u(i)}\left(\frac{\mu_d(i+1)}{\mu_{i+1}}-1)\right)\right)+ \\
&\quad \left(\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)\mu_d(i+1)}{P(i)-\mathbf{p_i}(0,1,1)\mu_u(i)}\right)r_d(i+1)
\end{aligned}$$

$$r_d(i) = r_d(i+1)\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)r_d(i)\mu_d(i)}{p_d(i)P(i)}+r_{i+1}\left(1-\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)r_d(i)\mu_d(i)}{p_d(i)P(i)}\right)$$

$$\mu_d(i) = \frac{1}{e_d(i)}\left\{\frac{1}{\frac{1}{P(i)}+\frac{1}{e_{i+1}\mu_{i+1}}-\frac{1}{e_u(i+1)\mu_u(i+1)}}\right\} i=1,\cdots,k-2 \tag{2.65}$$

Boundary Equations

$$\begin{aligned}
r_u(1) &= r_1 \\
p_u(1) &= p_1 \\
\mu_u(1) &= \mu_1 \\
r_d(k-1) &= r_k \\
p_d(k-1) &= p_k \\
\mu_d(k-1) &= \mu_k
\end{aligned} \tag{2.66}$$

Chapter 3

The Extended Dallery-David-Xie Algorithm

In Chapter 2, we developed a new set of decomposition equations for determining the steady state behavior of non-homogeneous, finite buffered flow lines with unreliable machines, deterministic processing times and operation-dependent failures. The method used to solve these equations is as important as the equations themselves. The Dallery-David-Xie Algorithm (DDX, Dallery et al; 1988) is currently the quickest and most reliably convergent algorithm for solving decomposition-type equations. Gershwin (1989) used a version of the algorithm to vastly improve the results of his decomposition of flow lines with exponentially distributed processing times and discrete material (Choong and Gershwin, 1987). Our equations have also been structured to exploit the benefits of the DDX algorithm. In this chapter, we introduce a new technique derived from the DDX method that is more robust and as much as 10 times faster than the original algorithm.

A description of the DDX algorithm adapted for the equations developed in Chapter 2 is provided in Section 3.1. Section 3.2 presents a new Accelerated DDX ($ADDX$) algorithm. Numerical results are presented in Section 3.3. Section 3.4 provides other methods for improving the original DDX method. Concluding remarks are given in Section 3.5.

3.1 Extended Algorithm

We formulated our equations to take advantage of the DDX algorithm. To improve the algorithm, we make a number of substitutions. The first is based on the Conservation of Flow assumption (2.11). $P(i-1)$ is substituted for $P(i)$ in the all the upstream equations and $P(i+1)$ for $P(i)$ in all of the downstream equations. Without this substitution in the processing rate equations (2.61) and (2.62) there would not be enough equations to solve for all the unknowns because each upstream processing rate equation would have an equivalent downstream equation. The replacements in the other equations are done to improve the convergence of the algorithm.

Unfortunately, the DDX and all the other algorithms we tried still did not efficiently solve the decomposition equations developed in the previous chapter. In order to get the algorithm to converge, we made the following adjustments to equation (2.50):

$$\mu_i \implies \mu_d(i-1) \tag{3.1}$$

$$P(i) - \mathbf{p_i}(N_i, 1, 1)\mu_d(i) \implies P(i), \tag{3.2}$$

and similarly we made the following substitutions in (2.51)

$$\mu_{i+2} \implies \mu_d(i+1) \tag{3.3}$$

$$P(i) - \mathbf{p_i}(0,1,1)\mu_u(i) \implies P(i). \tag{3.4}$$

Conceptually, $\mu_u(i)$ is weighted average of all the upstream production rates. In automated production lines, the rates of production of all the work centers are usually relatively close to one another. Therefore, μ_i and $\mu_u(i)$ should be very close in value and (3.1) should only significantly impact the output of the decomposition technique in flow lines with drastically different production rates. However, we did do a few spot check experiments on short lines on which we varied the production rates from one machine to the next by as much as a factor of two (Appendix C, Experimental Flow Lines - Case 38) and the throughput and WIP estimates appeared to be as accurate as in cases with identical production rates. A similar argument holds for (3.3).

Substitutions (3.2) and (3.4) are critical to achieving a high frequency of convergence of the algorithm. We believe the reason for this is that the current values of $\mathbf{p_i}(N_i$,1,1) and $\mathbf{p_i}$(0,1,1) are used to estimate the values of $p_u(i)$ and $p_d(i)$ at each step of the algorithm. However, $p_u(i)$ and $p_d(i)$ are also required to estimate $\mathbf{p_i}(N_i$,1,1) and $\mathbf{p_i}$(0,1,1). The effect of this is that one is forced to use values $\mathbf{p_i}(N_i$,1,1) and $\mathbf{p_i}$(0,1,1) from a previous iteration of the algorithm to obtain the current estimate of $p_u(i)$ and $p_d(i)$, while most of the other values used to estimate $p_u(i)$ and $p_d(i)$ are from the current iteration. In fact, in the ADDX algorithm, we are able to eliminate all parameter estimates from previous iterations of the algorithm in the current estimates of $p_u(i)$ and $p_d(i)$. We believe this is one of the primary reasons why the ADDX converges so fast and so reliably.

Unfortunately, the values of $\mathbf{p_i}(N_i,1,1)\mu_d(i)$ and $\mathbf{p_i}(0,1,1)\mu_u(i)$ can be substantial, especially in cases with machines with significantly different production rates. We again did some spot checks on simple cases with drastically different production rates and the results still appeared to be as accurate as when the production rates on different machines were very close. We believe that the biases in the experiments on long lines at the end of this section are at least partially due to this substitution. However, the numerical experiments on randomly generated cases in Section 3.3 demonstrate the method still performed well.

Finally after all these changes, (3.6)-(3.11) result. We then employ the DDX algorithm as shown below. This is very similar to the DDX algorithm for long lines with exponentially distributed processing times as it is presented in Gershwin (1994). The primary difference is that we use the Gershwin and Schick (1980) model for continuous material rather than Gershwin and Berman (1981) model for discrete material for solving for the steady-state behavior of two-stage systems ($L(i)$'s) at each step of the algorithm.

1. Initialization.

   Provide the following initial guesses for the parameters of each two-stage line:

$$\begin{aligned}
p_u(i) &= p_i \\
r_u(i) &= r_i \\
\mu_u(i) &= \mu_i \\
p_d(i) &= p_{i+1} \\
r_d(i) &= r_{i+1} \\
\mu_d(i) &= \mu_{i+1} \qquad\qquad i = 1, \ldots, k-1
\end{aligned} \tag{3.5}$$

2. Perform Step 1 and Step 2 until the Termination Condition (3.12) is satisfied.

Step 1 Let i range over values from 2 to $k-1$. Evaluate $L(i-1)$ using the Gershwin and Schick (1980) two-stage model with the most recent values of $r_d(i-1)$, $p_d(i-1)$, $\mu_d(i-1)$, $r_u(i-1)$, $p_u(i-1)$, $\mu_u(i-1)$. Then substitute these parameters and the resulting $P(i-1)$ into the upstream decomposition equations (3.6) - (3.8), in that order.

$$p_u(i) = p_i\left(1+\frac{\mathbf{p}_{i-1}(0,1,1)\mu_u(i)}{P(i-1)}\left(\frac{\mu_u(i-1)}{\mu_d(i-1)}-1\right)\right)+ \left(\frac{\mathbf{p}_{i-1}(0,0,1)\mu_u(i)}{P(i-1)}\right)r_u(i-1) \quad (3.6)$$

$$r_u(i) = r_u(i-1)\frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i-1)}+ r_i\left(1-\frac{\mathbf{p}_{i-1}(0,0,1)r_u(i)\mu_u(i)}{p_u(i)P(i-1)}\right) \quad (3.7)$$

$$\mu_u(i) = \frac{1}{e_u(i)}\left\{\frac{1}{\frac{1}{P(i-1)}+\frac{1}{e_i\mu_i}-\frac{1}{e_d(i-1)\mu_d(i-1)}}.\right\} \quad (3.8)$$

Step 2 Let i range over values from 1 to k-2. Evaluate $L(i+1)$ using the Gershwin and Schick (1980) two-stage model with the most recent values of $r_d(i+1)$, $p_d(i+1)$, $\mu_d(i+1)$, $r_u(i+1)$, $p_u(i+1)$ and $\mu_u(i+1)$. Then substitute these parameters and the resulting $P(i+1)$ into the downstream decomposition equations (3.9) - (3.11), in that order.

$$p_d(i) = p_{i+1}\left(1+\frac{\mathbf{p}_{i+1}(N_{i+1},1,1)\mu_d(i)}{P(i+1)}\left(\frac{\mu_d(i+1)}{\mu_u(i+1)}-1\right)\right)+ \left(\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)\mu_d(i+1)}{P(i+1)}\right)r_d(i+1) \tag{3.9}$$

$$r_d(i) = r_d(i+1)\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)r_d(i)\mu_d(i)}{p_d(i)P(i+1)}+ r_{i+1}\left(1-\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)r_d(i)\mu_d(i)}{p_d(i)P(i+1)}\right) \tag{3.10}$$

$$\mu_d(i) = \frac{1}{e_d(i)}\left\{\frac{1}{\frac{1}{P(i+1)}+\frac{1}{e_{i+1}\mu_{i+1}}-\frac{1}{e_u(i+1)\mu_u(i+1)}}\right\}. \tag{3.11}$$

3. **Termination Condition** Terminate the algorithm when the greatest value of:

$$\|P(i)-P(1)\| \tag{3.12}$$

is less than some specified value, for $i = 2, ..., k-1$.

Each equation is evaluated with the most recent value of each parameter, and this makes the order of evaluation very important. For example, the value of $\mu_u(i)$ on the right side of (3.6) is obtained from the previous estimate of its value. However, once (3.6) has been evaluated, the new value for $p_u(i)$ on the right side of (3.7) comes from the current evaluation of (3.6), while the value of $\mu_u(i)$ is still from the previous iteration. In (3.8), all right side values are current including $e_u(i)$, which is calculated from the current estimates of $p_u(i)$ from (3.6) and $r_u(i)$ from (3.7). Similarly, in Step 2, each equation is evaluated with the most recent value of each parameter.

Homogeneous Long Lines Test

A *homogeneous line* is one in which all the machines and buffers have exactly the same p_i, r_i and μ_i parameters. In Section 3.3, we perform detailed experiments which demonstrate the accuracy and speed of the new ADDX algorithm on a large set of randomly generated cases. First, as a simple check of the continuous material decomposition, we compared the average throughput and WIP estimates of the discrete material models with the continuous material decomposition for number of homogeneous lines with failure rate p_i for all machines of 0.01/minute (*i.e.*, a mean time between failures of 100 minutes), a repair rate r_i for all machines of 0.1/minute (*i.e.*, a mean time to repair of 10 minutes), a processing rate μ_i for all machines of 1/minute (*i.e.*, a processing time of 1 minute), and all buffers had capacities of ten units. The efficiency of each machine was 90.9%.

We started with a homogeneous line with five work centers ($k = 5$). In addition, we tested homogeneous lines for $k = 10$, 15, 20, 25, 30, 35, 40, 45, and 50. A manufacturing system that runs 24 hours per day, seven days per week, runs approximately 40,000 minutes per month. We therefore ran each case 100 times for 80,000 time units and collected data on the second half (the last 40,000 time units) for each case. We used a discrete material simulator programmed by A. Bonvik (Appendix 2.1). We chose this simulator for a number of reasons. First, we wanted to test the continuous material model against the more realistic assumption of discrete part movements. Second, the discrete model is more generally tested in the rest of the literature. Fi-

# of Stages	DDX	Sim.	95% C.I.	% Error
5	0.783	0.780	0.778,0.782	0.38
10	0.741	0.728	0.727,0.730	1.79
15	0.726	0.705	0.704,0.706	2.98
20	0.719	0.693	0.692,0.694	3.75
25	0.715	0.683	0.682,0.685	4.69
30	0.712	0.677	0.676,0.678	5.17
35	0.711	0.674	0.673,0.675	5.49
40	0.710	0.669	0.668,0.670	6.13
45	0.709	0.666	0.665,0.667	6.46
50	0.708	0.665	0.664,0.666	6.47

Table 3.1: Results of DDX Long Line Experiments

nally, this simulator was more robust for longer lines than our continuous material simulator (Appendix 2.2).

Table 3.1 provides the results of these experiments. The second column (DDX) gives the analytic estimate of system throughput, the third column the simulated estimate (Sim.) and the fourth column provides a 95% confidence interval (C.I.) calculated by the method described in Appendix 2.4. Column 5 lists the percentage errors from the analytic estimate. We calculate this as follows:

$$\%\text{Error} = \frac{100((\text{DDX Estimate}) - (\text{Simulated Estimate}))}{\text{Simulated Estimate}} \tag{3.13}$$

One can observe from both this table and Figure 3-1, which graphically displays this same information, that the percentage error for this particular type of homogeneous lines with up to 15 machines in length is quite small (under 3%). Strictly, this observation only holds for the lines tested. However, it does give a relative magnitude of error and more importantly, one may observe that as the line approaches a length of 50 machines, the percentage error seems to converge to about 6.5 %. As the line

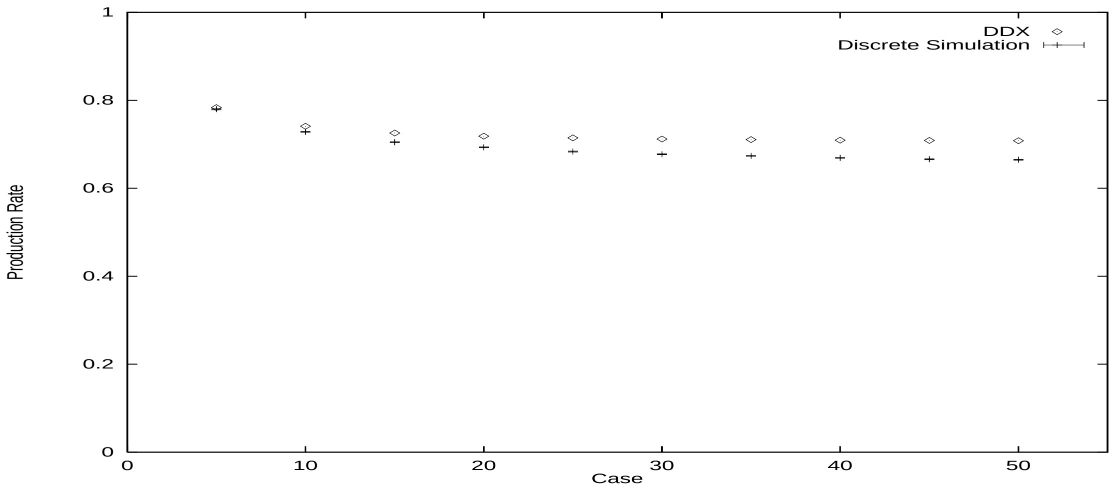

Figure 3-1: Efficiency Estimates of Homogeneous Long Lines

reaches a certain length, events at one end of the line rarely propagate to the other end of the line. This limits the effects of adding additional unreliable machines to the line. Although inconclusive, this may indicate that these models may by useful for estimating the efficiencies of even longer lines. However, we were unable to simulate these lines to test the results.

Figure 3-2 illustrates the estimates from the DDX method and simulation for the average buffer levels for each of the 49 buffers in the 50 stage line. The percentage errors in these estimates are larger than for the efficiency estimates. The observation that the efficiency estimates are better than average buffer levels is consistent with the rest of the queueing literature. However, the shape of the buffer distribution is reasonable. We do not pursue this issue because practitioners applying the model to real systems seem more interested in efficiency and are willing to accept less accuracy in the average inventory estimates.

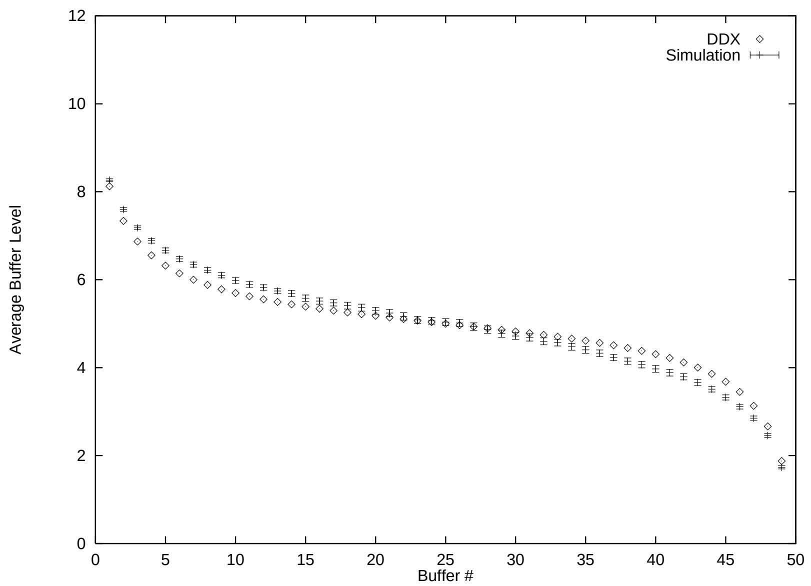

Figure 3-2: Buffer Level Estimates of 50 Stage Homogeneous Line

3.2 Accelerated DDX Algorithm

In the previous section, we modified the DDX algorithm to solve the continuous material decomposition equations. In this section, we present a more advanced method that we call the Accelerated DDX ($ADDX$). While simple in structure, the method is faster than the original DDX by as much as ten times and has a reliability of convergence of nearly 100% for all cases tested in Section 3.3.

In analyzing equations (3.6) - (3.8), one observes a dependency among the equations. For example, the right side of (3.6) contains a $\mu_u(i)$; the right side of (3.7) contains not only $\mu_u(i)$ and $p_u(i)$, but also $r_u(i)$ itself; and (3.8) contains $e_u(i)$ on the right side, where $e_u(i)$ is a function of both $r_u(i)$ and $p_u(i)$. In other words, (3.6) -

(3.8) is a set of three non-linear equations in three unknowns which can be represented as

$$\begin{aligned} p_u(i) &= f(p_u(i), r_u(i), \mu_u(i)) \\ r_u(i) &= f'(p_u(i), r_u(i), \mu_u(i)) \\ \mu_u(i) &= f''(p_u(i), r_u(i), \mu_u(i)). \end{aligned} \tag{3.14}$$

Equations (3.9) - (3.9) have a an equivalent representation for the downstream parameters

$$\begin{aligned} p_d(i) &= f^*(p_d(i), r_d(i), \mu_d(i)) \\ r_d(i) &= f'^*(p_d(i), r_d(i), \mu_d(i)) \\ \mu_d(i) &= f''^*(p_d(i), r_d(i), \mu_d(i)). \end{aligned} \tag{3.15}$$

In this section, we present a closed form solution to (3.14) for $p_u(i)$, $r_u(i)$, and $\mu_u(i)$ and to (3.15) for $p_d(i)$, $r_d(i)$, and $\mu_d(i)$. In our numerical experiments, we will show how this change between calls to the two-stage program can improve the entire algorithm.

Equations (3.6) - (3.8) can be written in the following form:

$$\begin{aligned} p_u(i) &= \mu_u(i)K_1 + p_i \\ r_u(i) &= \frac{r_u(i)\mu_u(i)}{p_u(i)}K_2 + r_i \\ \mu_u(i) &= \frac{p_u(i)}{r_u(i)}K_3 + K_3 \end{aligned} \tag{3.16}$$

where

$$K_1 = p_i \left(\frac{\mathbf{p}_{i-1}(0,1,1)}{P(i-1)} \left(\frac{\mu_u(i-1)}{\mu_d(i-1)} - 1 \right) \right) + \left(\frac{\mathbf{p}_{i-1}(0,0,1)}{P(i-1)} \right) r_u(i-1) \tag{3.17}$$

$$K_2 = (r_u(i-1) - r_i) \left(\frac{\mathbf{p}_{i-1}(0,0,1)}{P(i-1)} \right) \tag{3.18}$$

$$K_3 = \frac{1}{\frac{1}{P(i-1)} + \frac{1}{e_i \mu_i} - \frac{1}{e_d(i-1)\mu_d(i-1)}}. \tag{3.19}$$

It is important that K_1, K_2 and K_3 are not functions of $p_u(i)$, $r_u(i)$ or $\mu_u(i)$. We obtain the following from (3.16):

$$p_u(i) = \frac{p_i K_2 K_3 + r_i p_i + r_i K_1 K_3}{r_i + K_2 K_3 - K_1 K_3} \tag{3.20}$$

$$r_u(i) = \frac{p_i K_2 K_3 + r_i p_i + r_i K_1 K_3}{p_i + K_1 K_3 - K_2 K_3} \tag{3.21}$$

$$\mu_u(i) = \frac{K_3(p_i + r_i)}{r_i + K_2 K_3 - K_1 K_3}. \tag{3.22}$$

A similar procedure can be used to generate:

$$p_d(i) = \frac{p_{i+1}K_5K_6 + r_{i+1}p_{i+1} + r_{i+1}K_4K_6}{r_{i+1} + K_5K_6 - K_4K_6} \tag{3.23}$$

$$r_d(i) = \frac{p_{i+1}K_5K_6 + r_{i+1}p_{i+1} + r_{i+1}K_4K_6}{p_{i+1} + K_4K_6 - K_5K_6} \tag{3.24}$$

$$\mu_d(i) = \frac{K_6(p_{i+1} + r_{i+1})}{r_{i+1} + K_5K_6 - K_4K_6}, \tag{3.25}$$

where

$$\begin{aligned} K_4 &= p_{i+1}\left(\frac{\mathbf{p}_{i+1}(N_{i+1},1,1)}{P(i+1)}\left(\frac{\mu_d(i+1)}{\mu_u(i+1)} - 1\right)\right) + \\ &\quad \left(\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)}{P(i+1)}\right) r_d(i+1) \end{aligned} \tag{3.26}$$

$$K_5 = (r_d(i+1) - r_{i+1})\left(\frac{\mathbf{p}_{i+1}(N_{i+1},1,0)}{P(i+1)}\right) \tag{3.27}$$

$$K_6 = \frac{1}{\frac{1}{P(i+1)} + \frac{1}{e_{i+1}\mu_{i+1}} - \frac{1}{e_u(i+1)\mu_u(i+1)}}. \tag{3.28}$$

Together (3.20) - (3.25) form a new set of decomposition equations. We replace (3.6) - (3.11) with (3.20) - (3.25) in the DDX algorithm described in Section 3.1 to obtain the new algorithm. We will show in the next section how this modification improved the speed of the algorithm by up to 10 times and the reliability of convergence to 100% for all important cases tested.

3.3 Numerical Tests

In this section, we conduct numerous experiments to test the speed, accuracy and convergence properties of the new decomposition equations and solution technique. We begin in Section 3.3.1 by describing the all the models and solution techniques we tested. We continue in Section 3.3.2 to describe the flow lines we analyzed. Finally, in Section 3.3.3, we present the results of the experiments.

3.3.1 Models and Solution Methods

In subsequent sections, we test and compare the following models and/or solution techniques: 1) the continuous material model of Chapter 2 solved by DDX; 2) the same continuous material model solved by ADDX; 3) the Glassey and Hong continuous material model; and 4) the Gershwin discrete material/exponential processing time model. The following is a brief description of each of these:

DDX We presented a continuous material model in Chapter 2. DDX refers to the method presented in Section 3.1 for solving the decomposition equations used to approximate the dynamics this model.

ADDX *ADDX* (Accelerated Dallery-David-Xie) refers to the new solution technique developed in Section 3.2 which replaces the DDX method.

GHDDX Glassey and Hong (1993) developed a continuous material model that is very similar to our model. However, since the solution procedure they used was not

Case	GH E	GHDDX E	GH $\overline{n_1}$	GHDDX $\overline{n_1}$	GH $\overline{n_2}$	GHDDX $\overline{n_2}$
21	0.8624	0.8624	29.0834	29.0457	20.9166	20.9540
22	0.8525	0.8525	10.4648	10.4481	24.2521	24.2383
23	0.7446	0.7446	32.1940	32.1721	40.2251	40.2245
24	0.7454	0.7454	17.1786	17.1844	54.7310	54.6911
25	0.8622	0.8622	29.3945	29.4233	26.6445	24.6471
26	0.8572	0.8572	46.0242	46.1150	15.5393	15.5308
27	0.8356	0.8341	32.7639	30.6040	22.7787	22.2472
28	0.8217	0.8211	53.8960	49.6663	14.3962	14.0315
29	0.8965	0.8966	6.9036	6.3030	12.2344	12.3999
30	0.8811	0.8828	3.3571	2.9915	12.5303	12.8837
31	0.9014	0.9014	42.0521	42.4365	37.9090	38.4114
32	0.9043	0.9036	24.0526	24.3344	52.2070	53.0832

Table 3.2: GHDDX Accuracy Comparison

made explicitly clear in their paper, we develop a set of equations to emulate their model. We then modify the ADDX to solve these equations. We call the resulting algorithm GHDDX. In Table 3.2, we present the estimates (GH) from Glassey and Hong (1993) and from our implementation of the GHDDX for the expected throughput rate (E) and average buffer levels ($\overline{n_i}$, $i = 1, 2$) for Cases 21 - 32 listed in Appendix 3.1. The difference between the estimates of E from the GHDDX method and the estimates originally reported in Glassey and Hong (1993) is less than 10^{-2}. The buffer estimates differ by a slightly greater amount. However, we feel the GHDDX is a good substitute for the original Glassey and Hong model for the experiment later in this section that compares the results of our procedure with Glassey and Hong's (1993) results.

EXPDDX $EXPDDX$ (Exponential—Dallery-David-Xie) refers to both the model and solution technique presented in Gershwin (1989) for estimating throughput for discrete material flow lines with exponentially distributed service times.

3.3.2 Experiments

We ran many experiments to test the models. The following is a description of both special and random cases tested.

Special Cases In Appendix C, we provide the parameters for a number of specific cases designed to test various aspects of the different models and solution techniques. The following is a description of each of these cases:

Cases 1 - 32 These are the cases presented in Glassey and Hong (1993). We use these to test the accuracy and speed of ADDX. We tested each of these cases with DDX, ADDX and GHDDX[1].

Case 33 This is a three-stage homogeneous line that we use as a starting point for our analysis. All the machines have average failure times of 100 time units, average repair times of ten units, and processing times of 1 time unit. All buffers have capacities of 10. We use this case to demonstrate typical results obtained when comparing the continuous material model simulations, discrete material model simulations, ADDX and EXP.

Cases 34 - 37 To test the accuracy and reversibility properties of our model, we created four cases. Each case altered one of the critical parameters of the third machine in Case 33. In Case 34, we change the repair rate of machine 3 by

[1] All algorithms were employed using a convergence criterion of 10^{-5}.

an order of magnitude from 10 time units to 100 time units. In Case 35, we change the capacity of Buffer 2 from 10 to 5 units. In Case 36, we change the failure rate of machine 3 by an order of magnitude from 100 time units to 10 time units. In Case 37, we change the processing time of machine 3 from 1 time unit to 0.5 time units.

Case 38 In Chapter 2, we stress the fact that Glassey and Hong (1993) does not account for the change in the failure rate when a machine is partially blocked or starved. We made the assertion that their model potentially assumes a greater than actual failure rate for each machine. This should bias their throughput estimate in the negative direction. To test this hypothesis, we designed a line with two completely reliable machines feeding a third unreliable machine. To compensate for third machine's failures, it has a speed that is twice as much as the first two machines. In this way, Buffer 2 will usually remain empty and Machines 1 and 2 will be partially protected from a failure of Machine 3, at least until the buffer fills up.

Cases 39 - 42 We present some very simple homogeneous lines of 3 and 10 machines for testing known results for flow lines with zero and infinite buffers with ADDX. In Cases 39 and 40, each machine has an average failure time of 100 time units and an average failure time of 10 time units. For Cases 41 and 42, both average failure and average repair times are 100 time units for each machine. Since the ADDX cannot handle neither infinite nor zero buffers, we approximate these buffer sizes in each case by replacing zero with 0.0001 and infinity with 100000. The fact that ADDX would accept such values is an indication of its versatility.

Case 43 This is a 17-stage flow line. It was randomly generated using the procedure described below. We use this case in the next section to demonstrate the advantages of an iteration technique for improving the performance of the modified DDX algorithm applied to the continuous material model described in Chapter 2.

Case 44 This is a 9-stage randomly generated case. We use this case in the next section to demonstrate the advantages of a technique for improving the convergence reliability of EXP.

Random Lines The convergence properties are among the most important, yet least examined aspects of the decomposition algorithms. One of the real advantages of the ADDX is its superior convergence reliability. Since the number of possible cases that one could examine is inordinate, we felt it was important to generate random flow lines and test the method on these cases in addition to the specific cases discussed above. We used the standard C language functions rand() and srand() to generate a uniformly distributed random variable between 0 and 1. We refer below to a random variable generated in this fashion as RAN. Each reference to RAN is a separate call to the random number generator and represents a different random variable.

Generating meaningful random lines was not straightforward. Invariably, different procedures for generating random lines create bottlenecks which strongly influence the throughput of the line. These bottleneck cases are easy to solve and thus do not usefully test the robustness of the algorithms. Therefore, we develop a procedure for generating random lines. The goal of the procedure was to obtain random, yet realistic lines. It is important to remember that each time we use the variable RAN, an additional call to the random number generator is made.

In the case of random length lines, we generate k as follows:

$$k = 3 + Int(16RAN_1). \tag{3.29}$$

The Int() function finds the greatest integer less than the argument of the function. Equation (3.29) generates values between 3 and 18, each value with equal probability.

To keep the lines realistic, we wanted to keep some consistency among the processing rates (μ_i) of the different machines. However, we want to test lines with machines with different μ_i. We use

$$PROD = 0.1 + RAN_2 \tag{3.30}$$

to generate a random variable $PROD$ which we use to keep production rates at sequential machines relatively close. We use $PROD$ to generate random production rates as follows

$$\mu_i = PROD(3.6 + 0.8 * RAN_{3i}) \qquad i = 1, \cdots, k. \tag{3.31}$$

We would like to test different values of r_i values while maintaining some consistency in their order of magnitude. We first generate

$$x = 1 + (9RAN_4) \tag{3.32}$$

and then

$$y_i = -(1 + RAN_{5i}) \tag{3.33}$$

for each stage. We then substituted x and y_i into

$$r_i = x^{y_i} \qquad i = 1, \cdots, k. \tag{3.34}$$

It is desirable to generate p_i values which are on average an order of magnitude less than their corresponding r_i value. This results in machines that are about 90% efficient. The following equation keeps p_i around one order of magnitude less than r_i but allows it to range from values 2 orders of magnitude less than r_i to values greater than r_i.

$$p_i = r_i * 10^{-(0.66RAN_{6i}+0.66RAN_{7i}+0.66RAN_{8i})}. \tag{3.35}$$

Remember that each of RAN values are different. This lets the efficiency e_i of each machine range from about 50% to 99%.

Finally, we chose the buffer sizes to be roughly proportional to the amount of material produced during an average failure by using the following:

$$N_i = MAX[1, \frac{\mu_i}{r_{i+1}}, \frac{\mu_{i+1}}{r_i}]3RAN_{9i}. \tag{3.36}$$

This chooses buffers which hold between zero and three times the amount of material produced during a failure of average duration of the adjacent machines. It also sets a minimum buffer size of 1.

We generated many different cases using the above procedure and simulated them using the continuous material simulator (Appendix 2.2) with random numbers generated using a publicly available set of random number generating routines called mrandom3.0 (Appendix 2.3).

3.3.3 Results

In this section, we present the results of our experiments. We compare the continuous and the discrete model. We examine issues such as accuracy, speed, and reliability of convergence. We also do some reality checks of the model by checking reversibility properties and extreme cases with zero and infinite buffers.

Continuous, Discrete and Exponential Model Comparison

One motivating factor for creating this model was the claim that the variability of the exponentially distributed processing times assumed in the decomposition of Choong and Gershwin (1987) substantially increased the amount of time the buffer was blocked and starved and would thus negatively bias the throughput estimate. In Case 33, we examined a very simple 3 stage homogeneous line. We compared ADDX and EXP estimates with a discrete simulation. We ran the simulation 100 times to reduce the confidence intervals to a very small size. Table 3.3 illustrates that while the EXP model was off by over 10% for the throughput estimate, the ADDX method was off by less than 1%. Results for average buffer level were also better for the ADDX method, but not quite as significant.

Case 33			
	Throughput Estimate	$\overline{n}_1$	$\overline{n}_2$
Discrete Simulation	0.823	6.135	3.942
ADDX	0.825	6.202	3.798
EXP	0.7125	5.752	4.248

Table 3.3: Comparison of Model Performance Estimates

Accuracy Tests

In this section, we examine the accuracy of the ADDX method by testing it on a number of special and random cases and comparing the results with simulation and other analytic models.

Testing The Effects of Changing Parameters As described in the previous section, Cases 34 through 37 were designed to test the robustness of the algorithm by changing the parameters of the third machine from Case 33. We changed r_3 in 34, N_2 in 35, p_3 in 36 and μ_3 in 37. For each case, we ran ADDX and compared this with both a continuous (Appendix 2.2) and discrete (Appendix 2.1) simulation. Both simulations were run 100 times. Each time, the system was run for a period of 40,000 time units before any statistics were collected and then run for an additional period of 40,000 time units during which the data displayed in Table 3.4 was gathered. The confidence intervals were small due to the large number of trials (*e.g.*, less than 10^{-3} for throughput estimates and 10^{-1} for average buffer levels) and are not included so that we can focus on the important aspects of the results.

	Case 34	Case 35	Case 36	Case 37
ADDX E	0.479	0.815	0.492	0.848
C. Simulation E	0.477	0.814	0.492	0.848
D. Simulation E	0.477	0.816	0.492	0.848
ADDX Buffer 1	8.473	6.470	9.352	5.442
C. Simulation $\overline{n}_1$	8.308	6.404	9.274	5.443
D. Simulation $\overline{n}_1$	8.311	6.436	9.293	5.427
ADDX Buffer 2	7.148	1.945	9.181	0.367
C. Simulation $\overline{n}_2$	7.173	1.986	9.178	0.366
D. Simulation $\overline{n}_2$	7.208	2.007	9.194	0.360

Table 3.4: Accuracy of ADDX for Experiment Cases 34 - 37

Table 3.4 presents our results for Cases 34 through 37. For each case, the following data is provided: the ADDX estimate for expected throughput (ADDX E), the expected buffer level for Buffer 1 (ADDX Buffer 1) and for Buffer 2 (ADDX Buffer 2). The same data is also provided from each of the simulations. Data related to the continuous simulation is proceeded by "C. Simulation" and data related to the discrete simulation is proceeded by "D. Simulation".

We can draw two important conclusions from Table 3.4. First, the ADDX method predicted expected throughput within 10^{-2} of both simulations in all cases and the buffer estimates were within about 10^{-1}. Second, the estimates of the performance measures from the discrete and continuous material simulations were as close to each other as to the ADDX estimate. This indicates that the continuous model is a good approximation for discrete material systems.

Glassey and Hong Model Comparison Since Glassey and Hong (1993) had previously been the only model in the literature for continuous material decompositions of non-homogeneous lines with operation-dependent failures, we created a case that would highlight the potential bias of their model. Case 38 represents a three-machine system of two reliable machines feeding an unreliable machine that processes material at twice the rate of the first two machines when operational.

Case 38			
	Throughput	$\overline{n}_1$	$\overline{n}_2$
ADDX	0.800	9.996	4.000
C. Sim.	0.799	9.996	3.998
D. Sim.	0.805	9.996	4.013
GHDDX	0.750	10.000	5.000

Table 3.5: Glassey and Hong Bias Demonstration

We ran both continuous and discrete material simulations for 100 repetitions in the same manner as with Cases 34 through 37. The results are in Table 3.5. The ADDX estimate of throughput was within 1% while the Glassey and Hong estimate (GHDDX) was off by more than 6.5%. The GHDDX estimate of $\overline{n}_2$ was off by about 20% while the ADDX estimate was off by less than 1%. This is because the GHDDX method did not compensate for the difference in the $\mu's$ in determining the $p_u's$ and the $p_d's$.

Random Cases In addition to the specific cases discussed above, we generated 300 random flow lines according the procedure described in the previous section to test the accuracy of the model. The lines ranged in length from 3 machines to 18 machines. We evaluated each line with ADDX and the continuous material simulation for 30 trials. Each trial was run for 40,000 time units as in the other experiments. We then sorted the cases according to the throughput estimate of the ADDX method. The results are displayed in Figure 3-3. For each of the random cases, the ADDX results and the simulation with a 95% confidence interval are plotted. The results are so close that we provide an additional plot in Figure 3-4 which displays the corresponding percentage error of each case. While there was one case with a 5% error, the average percentage error was 1.32%. Experience with the model has also demonstrated that such cases usually have some unrealistic characteristic such as drastically different $r's$ or $\mu's$. The method seems to consistently produce $p's$ which are too large. This is consistent with the anticipated effects of approximations (3.1) - (3.4) because these approximations would tend to estimate values for p_d and p_u.

While the throughput estimates appeared to be consistently good, the results for average buffer levels were not as promising. We ran an additional 400 random cases. Each flow line was nine stages long. Figures 3-5 and 3-6 illustrate the errors between the ADDX estimate and simulation as a percentage of buffer capacity for

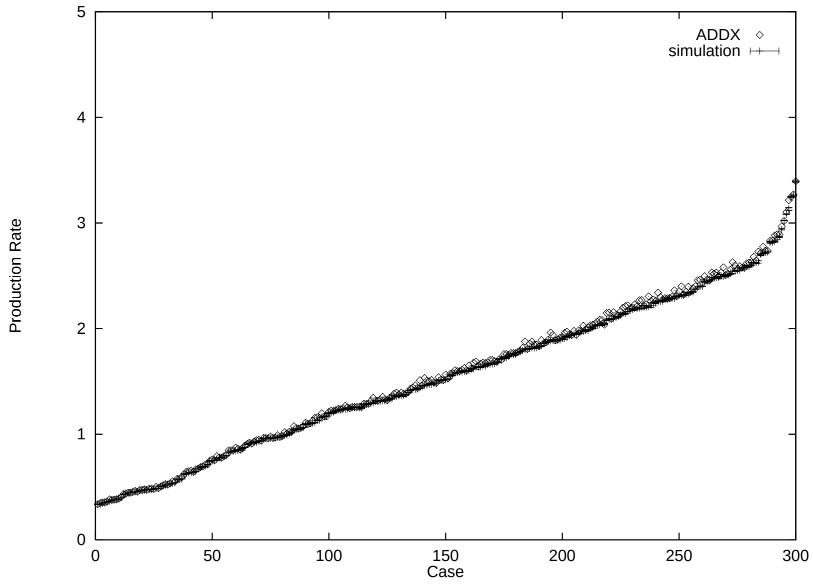

Figure 3-3: ADDX vs. Simulated Estimate of Throughput for Random Flow Lines

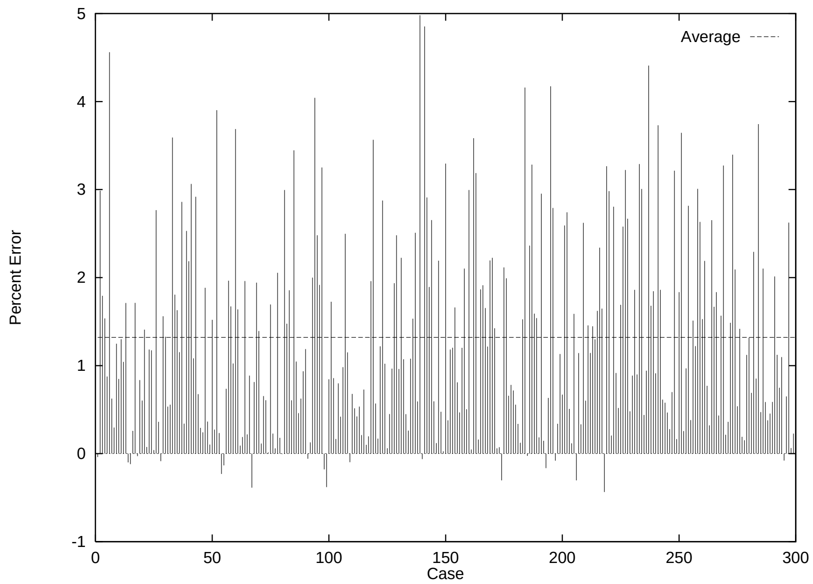

Figure 3-4: Percentage Error of ADDX Throughput Estimate for Random Flow Lines

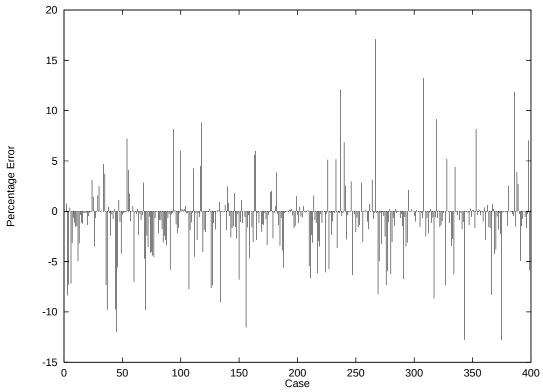

Figure 3-5: Average Buffer Level Error for Buffer 1

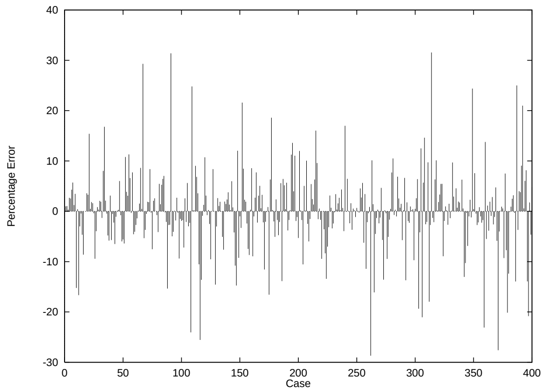

Figure 3-6: Average Buffer Level Error for Buffer 5

the first and fifth buffers, respectively. Although uncommon, errors of as much as 30% were observed in the estimate of average buffer level for Buffer 5. However, as with expected throughput, careful examination of the cases where errors were very large usually had some type of aberration in the line specification. In addition, the error may also be due to the simulation which often had large confidence intervals for average buffer levels. These greater errors in buffer level estimates seem consistent with casual observations of the rest of the flow line literature.

Reversibility Test

We tested the ADDX method on the reverse the lines in Cases 34 through 37. The estimates of throughput and average buffer levels are presented for both the original and reversed in Table 3.6. A more detailed description of reversibility can be found in Section 4.4.1 and Muth (1979). The two primary properties we want to check are: 1) that the efficiency of the reversed line is the same as that of the original line and 2) that the sum of the average buffer level estimate for Buffer i of the original line plus the average buffer level estimate for Buffer $k - i$ of the reversed line is equal to the capacity of Buffer i. Table 3.6 illustrates this to be true.

	Case 34	Case 35	Case 36	Case 37
Original Throughput	0.479	0.815	0.492	0.848
Reverse Line Throughput	0.479	0.815	0.492	0.848
Original Buffer 1	8.473	6.470	9.352	5.442
Reverse Line Buffer 2	1.527	3.530	0.648	4.558
Original Buffer 2	7.148	1.945	9.181	0.367
Reverse Line Buffer 1	2.852	3.055	0.819	9.633

Table 3.6: Check for Reversibility Properties

Zero and Infinite Buffer Tests

As a second reality check of the model, we test the limiting cases of zero and infinite buffers for homogeneous lines. Buzacott (1967a) presents the following two results for homogeneous lines: 1) the throughput of an infinite buffered flow line is equal to the minimum isolated throughput of all the machines in the line or

$$E_\infty = \min_{i=1,\cdots,k} \left[\frac{r_i}{r_i + p_i} \right], \tag{3.37}$$

and 2) the throughput of a zero buffer line is:

$$E_0 = \frac{1}{1 + \sum_{i=1}^{k} \frac{p_i}{r_i}} \tag{3.38}$$

Using (3.37) and (3.38) we estimated the expected throughput for homogeneous lines represented by Cases 39 through 42. The results illustrated in Table 3.7 for the ADDX and Buzacott estimates are almost identical for all cases.

	Case 39	Case 40	Case 41	Case 42
Buzacott Zero Buffer Estimate	0.7692	0.5000	0.2500	0.0909
ADDX Zero Buffer Estimate	0.7692	0.5000	0.2500	0.0909
Buzacott Infinite Buffer Estimate	0.9091	0.9091	0.5000	0.5000
ADDX Infinite Buffer Estimate	0.9091	0.9091	0.5000	0.4994

Table 3.7: Infinite and Zero Buffer Tests

Efficiency Estimates						
Case #	G & H Est.	DDX Est.	ADDX Est.	G & H # Calls	DDX # Calls	ADDX # Calls
1	0.4680	0.4680	0.4680	20	7	7
2	0.9026	0.9026	0.9026	20	5	5
3	0.3207	0.3207	0.3207	34	7	7
4	0.3558	0.3588	0.3588	22	9	9
5	0.7604	0.7602	0.7604	24	7	7
6	0.3015	0.3015	0.3015	441	232	232
7	0.6351	0.6351	0.6351	572	230	170
8	0.2315	0.2315	0.2315	640	675	645
9	0.2296	0.2296	0.2296	3819	990	990
10	0.2927	0.2927	0.2927	1976	1314	1350
11	0.8356	0.8341	0.8341	28	9	9
12	0.8588	0.8567	0.8567	46	9	7
13	0.7280	0.7278	0.7278	46	9	9
14	0.8170	0.8170	0.8170	50	11	7
15	0.8754	0.8748	0.8748	28	29	19
16	0.8352	0.8258	0.8257	402	26	26
17	0.8056	0.8000	0.8000	495	14	18
18	0.7476	0.7473	0.7473	102	34	26
19	0.8256	0.8321	0.8321	116	45	45
20	0.8323	0.8326	0.8326	112	57	57

Table 3.8: Algorithm Speed Comparison

Speed Test

The only existing cases that we can compare the speed of our algorithm to are Cases 1 through 20 in Glassey and Hong (1993). The parameters for these cases can be found in Appendix 3.1. We took the Glassey and Hong results directly from their paper and tested both the DDX and the ADDX methods on each case. All these results are presented in Table 3.8. In the cases that Glassey and Hong tested, the production rates were relatively close on all the machines in each flow line. When the rates are identical at every machine in a given line, Glassey and Hong's procedure is essentially the same as ours. This explains why their estimates of expected throughput (G& H Est.) and our estimates (DDX Est.) are the same in cases 1 through 10 (when the production rates in all machines are identical) and very close for the rest of the cases. Notice also that the DDX estimate is equal to the ADDX estimate, as we would expect.

Columns 5 through 7 provide the number of calls to the two-stage sub-model required for the algorithm to converge to a solution for the given case. The amount of time it takes decomposition-type algorithms to converge is roughly proportional to this number of calls. The DDX method was substantially faster than the Glassey and Hong method in all cases except for Case 8. In Case 16, the DDX method was over 10 times faster. The ADDX algorithm did approximately the same as the original DDX method. This is because the advantages of the ADDX method only become apparent for longer lines and lines with more different parameters than Glassey and Hong tested. We demonstrate this in Section 3.4.

Convergence Reliability

In addition to speed and accuracy, we are concerned with how reliably the algorithms converge. This is an important problem that is not often discussed in the literature. We generated 100 5-stage, 10-stage, 25-stage, and 100-stage flow lines using the procedure presented in Section 3.3.2. For each length of line, we rank ordered each case according to the number of calls to the two-stage model the ADDX algorithm required to converge. We then charted the number of two-stage model calls for the ADDX algorithm (diamonds) and the DDX algorithm (crosses). A value of zero was used in cases when the algorithm did not converge. The results appear in Figures 3-7 through 3-10.

The ADDX algorithm converged over 99.9% of the time in the cases we tested. In all cases, the ADDX was noticeably faster than the DDX algorithm. It is important to observe that for 5-stage lines, the DDX algorithm converged only about 90% of the time. For the 10-stage lines, the convergence rate for the DDX was down to 50%, for the 25-stage lines the convergence rate was lower than 5% and for the 100-stage line, the DDX never converged.

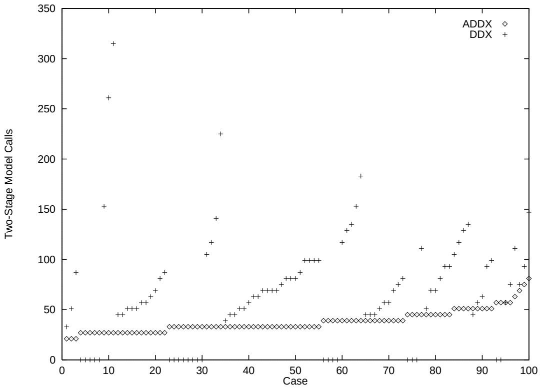

Figure 3-7: 5 Stage Systems

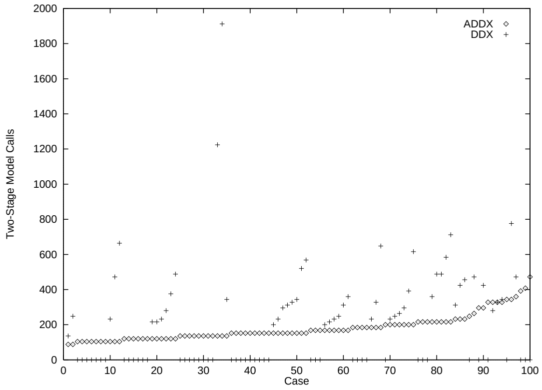

Figure 3-8: 10 Stage Systems

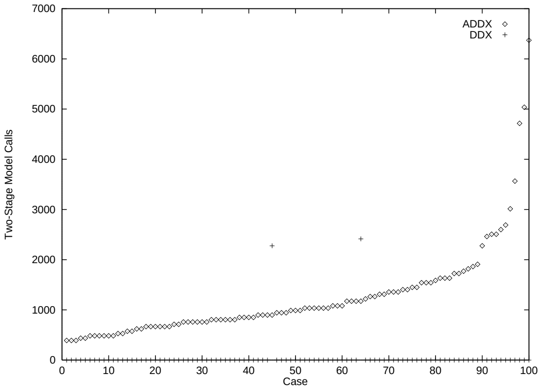

Figure 3-9: 25 Stage Systems

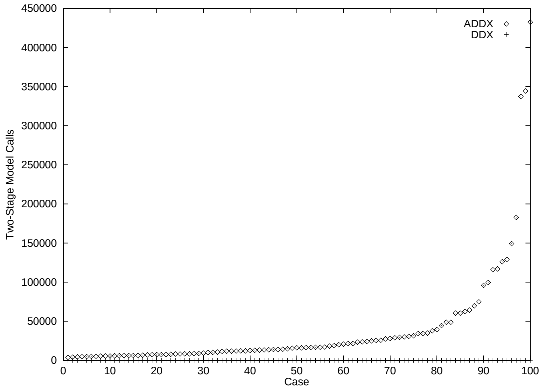

Figure 3-10: 100 Stage Systems

3.4 Other Improvement Techniques

We studied three other techniques that make the DDX algorithm converge more reliably and/or more quickly. All three methods work in a similar fashion to the method described in Section 3.2. Each modifies the estimates of $p_u(i)$, $r_u(i)$, $\mu_u(i)$, $p_d(i)$, $r_d(i)$, and $\mu_d(i)$ between each call to the two-stage model. The closed-form solution technique described in Section 3.2 worked better than all these methods for our continuous material model. However, these methods may be valuable in improving other decomposition algorithms when a closed form solution for the $r_u's$,

$p'_u s$, $\mu'_u s$, $r'_d s$, $p'_d s$ and $\mu'_d s$ such as the one presented Section 3.2 is not available. They can be used independently or in conjunction with one another. We provide only a cursory description of these methods with few examples.

3.4.1 Multiple Iterations

In the original *DDX* algorithm in Section 3.1, equations (3.6) - (3.11) make one estimate for $p_u(i)$, $r_u(i)$ and $\mu_u(i)$ between each call to the two-stage model. Suppose we substitute our original estimate for $p_u(i)$, $r_u(i)$ and $\mu_u(i)$ into the right side of (3.6)-(3.8) to obtain $p_u^*(i)$, $r_u^*(i)$ and $\mu_u^*(i)$ as illustrated here

$$\begin{aligned} p_u^*(i) &= f(p_u(i), r_u(i), \mu_u(i)) \\ r_u^*(i) &= f'(p_u(i), r_u(i), \mu_u(i)) \\ \mu_u^*(i) &= f''(p_u(i), r_u(i), \mu_u(i)). \end{aligned} \tag{3.39}$$

The left side of (3.39) can be substituted into the right side any number of times between calls to the two-stage program. Essentially, this is getting closer and closer to the fixed point found in Section 3.2. The same technique can be used with the downstream equations.

Sim.	95% CI	DDX Iter.	2 Iter.	5 Iter.	ADDX	Anal. Est.
1.206	1.194,1.218	∞	4275	2235	405	1.257

Table 3.9: Improvement From Reiteration Procedure

Table 3.9 displays the output of experiments with a randomly generated 17 Stage line (Case 43 - Appendix 3.2). The line was evaluated using simulation, the original DDX, DDX with (3.39) iterated twice, DDX with (3.39) iterated five times times and finally the ADDX. Each evaluation method arrived at the same estimate of 1.257 (Table 3.9) for the average throughput and average buffer levels (results displayed Appendix 3.2) were also identical to three decimal places. As illustrated in (Table 3.9, the original DDX method did not even converge. As the number of iterations of (3.39) was increased from 2 to 5, the number of calls the the two-stage sub-model required to get convergence reduced from 4275 to 2235. However, notice that the ADDX procedure only required 405 calls (greater than a 90% reduction from 4275).

This iterative method substantially improved the frequency of convergence for the DDX algorithm while usually reducing the number of calls to the two-stage subproblem algorithm. We believe that this procedure may ultimately help find a solution to the original equations developed in Chapter 2.

3.4.2 Convex Combinations

Our next technique uses a weighted average of the new estimate for $p_u(i)$, $r_u(i)$ and $\mu_u(i)$ from (3.6) - (3.11) ($p_u^c(i)$, $r_u^c(i)$ and $\mu_u^c(i)$) and the previous estimate of these parameters ($p_u^p(i)$, $r_u^p(i)$ and $\mu_u^p(i)$) as the next value for $p_u(i)$, $r_u(i)$ and $\mu_u(i)$. For some $1 > \epsilon > 0$, we use the following to generate $p_u^*(i)$, $r_u^*(i)$ and $\mu_u^*(i)$

$$p_u^*(i) = \epsilon p_u^p(i) + (1-\epsilon)p_u^c(i) \tag{3.40}$$

$$r_u^*(i) = \epsilon r_u^p(i) + (1-\epsilon)r_u^c(i) \tag{3.41}$$

$$\mu_u^*(i) = \epsilon \mu_u^p(i) + (1-\epsilon)\mu_u^c(i). \tag{3.42}$$

Similar equations are established for the downstream parameters. In rare cases when the fixed point solution of Section 3.2 did not converge, this method could usually find a solution by choosing ϵ small enough. The method takes smaller, more accurate steps towards a solution. However, the number of times the two-stage model must be solved to achieve the convergence in these cases is usually very large. One way of improving the algorithm is to lower ϵ as the difference between successive estimates for P falls below various thresholds. No rigorous testing of this concept was conducted. However, spot checks using the method drastically improved the exponential model of Gershwin (1989). For example, Figure 3-11 is a graph of the error (3.12) observed after each iteration of the EXP algorithm using different ϵ values for a randomly generated 9-stage line (Case 44 - Appendix 3.2).

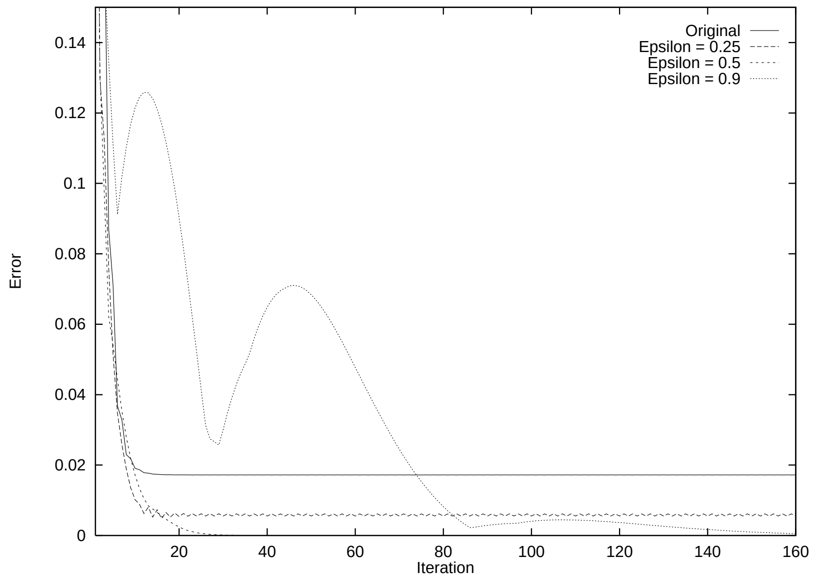

Figure 3-11: Results of Using Convex Combinations with EXP

From Figure 3-11, one can observe that the original method did not converge. When we set ϵ to 0.25, it still did not converge. When we set ϵ to 0.5, the method converged in less than 40 iterations. However, when we set ϵ to 0.9, the method still converge, but much less smoothly and less quickly. We have also experimented with this method to improve the converge of the ADDX and DDX when applied to the original equations developed in Chapter 2. The initial results looked promising.

3.4.3 Hard Constraints

In the numerical experiments from Section 3.3, we frequently observed the DDX making poor estimates for the $r'_u s$, $p'_u s$, $\mu'_u s$, $r'_d s$, $p'_d s$ and $\mu'_d s$. For example, resumption of flow equation would sometimes generate a negative value for r. When this happened, the algorithm would often not converge. The following hard constraints were imposed after each call to (3.6)-(3.8) to eliminate this situation. After generating initial estimates for $p_u(i)$, $r_u(i)$ and $\mu_u(i)$, we use the following constraints to generate $p^*_u(i)$, $r^*_u(i)$ and $\mu^*_u(i)$, the values actually used in the next iteration of the algorithm.

p **Constraints** $p_u(i)$ is an approximation for the failure rate of $M_u(i)$. It must be positive. Similarly, $p_d(i)$ must be positive. Therefore, we have the following constraints

$$\begin{aligned} p_u(i) &\geq 0 \\ p_d(i) &\geq 0. \end{aligned} \tag{3.43}$$

We enforce these by using:

$$\begin{aligned} p_u^*(i) &= MAX(0, p_u(i)) \\ p_d^*(i) &= MAX(0, p_d(i)). \end{aligned} \tag{3.44}$$

r **Constraints** $r_u(i)$ is an approximation for the repair rate of $M_u(i)$. It is a convex combination of the actual rate at which M_i is repaired and the rate at which $M_u(i-1)$ is repaired when it is down and buffer $i-1$ is empty. Therefore, $r_u(i)$ must be between r_i and $r_u(i-1)$. Similarly, $r_u(i)$ must be between r_{i+1} and $r_d(i+1)$. Therefore, we have the following constraints

$$\begin{aligned} MIN(r_i, r_u(i-1)) \leq \quad & r_u(i) \quad \leq MAX(r_i, r_u(i-1)) \\ MIN(r_{i+1}, r_d(i+1)) \leq \quad & r_d(i) \quad \leq MAX(r_{i+1}, r_d(i+1)) \end{aligned} \tag{3.45}$$

which we enforce by using:

$$\begin{aligned} x &= MAX(r_i, r_u(i-1)) \\ y &= MIN(r_i, r_u(i-1)) \\ z &= MIN(r_u(i), x) \\ r_u^*(i) &= MAX(y, z) \\ x' &= MAX(r_{i+1}, r_d(i+1)) \\ y' &= MIN(r_{i+1}, r_d(i+1)) \\ z' &= MIN(r_d(i+1), x') \\ r_d^*(i) &= MAX(y', z'). \end{aligned} \tag{3.46}$$

μ **Constraints** $\mu_u(i)$ is an approximation for the production rate of $M_u(i)$ that accounts for upstream machines. Therefore, the modified production rate of $M_u(i)$ must be less than the actual production rate of M_i. Similarly,the modified production rate of $M_d(i)$ must be less than the actual production rate of M_{i+1} Therefore, we have the following constraints

$$\begin{aligned} 0 \leq \mu_u(i) \leq \mu_i \\ 0 \leq \mu_d(i) \leq \mu_{i+1} \end{aligned} \tag{3.47}$$

which we enforce by using:

$$\begin{aligned} x &= MAX(0, \mu_u(i)) \\ \mu_u^*(i) &= MIN(x, \mu_i) \\ x' &= MAX(0, \mu_d(i)) \\ \mu_d^*(i) &= MIN(x', \mu_{i+1}). \end{aligned} \tag{3.48}$$

By replacing $p_u(i)$, $r_u(i)$, $\mu_u(i)$, $p_d(i)$, $r_d(i)$ and $\mu_d(i)$ with $p_u^*(i)$, $r_u^*(i)$, $\mu_u^*(i)$, $p_d^*(i)$, $r_d^*(i)$, and $\mu_d^*(i)$ the reliability of convergence of the algorithm improved noticeably. In the infrequent cases when the ADDX did not converge, this method only rarely helped.

3.5 Conclusion

Most of the decomposition literature does not discuss the difficulty in obtaining an algorithm that consistently converges. We slightly modified the decomposition equations developed in Chapter 2 and used the original DDX algorithm to solve these equations. We demonstrated through numerical experimentation that the method had a high level of accuracy compared with simulation, but still had significant convergence difficulties as the flow lines became longer. To overcome this, we developed the Accelerated DDX Algorithm (ADDX). For a large number of random flow lines with up to 100 stages, the method converged over 99.9% of the time with a high level of accuracy and speed.

Chapter 4

Unreliable Buffers

The effect of interstage buffer storage on the output of a production line has long and frequently been discussed in the literature. The buffers in this literature are often treated as available space between consecutive processes in which material may be easily stored and retrieved. In reality, these buffers may be subject to failures like any other mechanism in the system. Buffer failures are a significant phenomenon. Lipset *et al.* (1994a) reports statistics from a U.S. car manufacturing facility in which 27.7% of all system down time was due to buffer failures. This resulted in a direct loss of 10.3% of total capacity and a revenue loss of $40.9 million over an 8 week period. In this chapter, we extend one of the basic analytic models for the performance evaluation of two-machine production lines to account for finite buffers with operation-dependent failures.

Buffer failures may be caused by a number of significant factors. Most material handling systems have a mechanism that loads and unloads parts from the processing machinery. This mechanism can fail. However, since this type of failure does not necessarily shut down the buffer or the processing machine at the other end of the buffer, this case can be treated by assuming that there exists an additional machine between the buffer and the processing machine. We are more interested in failures that simultaneously prevent the buffer from releasing and accepting material. This usually occurs when an accumulator shuts down because of a mechanical failure or a conveyor system jams. Casual observations of real systems indicated that this type of failure was nearly as common as processing machinery failures. These are ODFs.

Recent work of Lipset *et al.* (1994a, 1994b) provides heuristics for analyzing time-dependent buffer failures that are based on the Schick and Gershwin (1978) two-parameter, ODF, synchronous, reliable buffer, discrete time, discrete material model. Also extending the Schick and Gershwin model, we describe a generalized birth-death process for the failing buffer case in Section 4.1. This model is then analyzed in a similar manner to that suggested by Schick and Gershwin. By first calculating internal state probabilities, and then determining the boundary state probabilities, an analytic solution to an operation-dependent model is developed in Section 4.2. The model is analyzed in Section 4.3. Numerical experiments are presented in Section 4.4. Two heuristics are suggested in Section 4.5. Finally, concluding remarks are provided in Section 4.6.

4.1 Problem Description

Material or parts flow from outside the system to the first machine (M_1), then to the buffer (B), then to the second machine (M_2), and finally out of the system (Figure 4-1). The machines and the buffer are all unreliable.

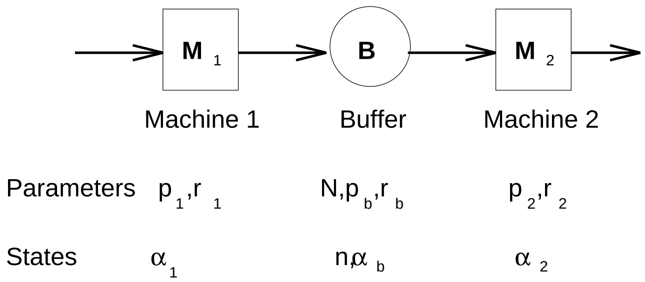

Figure 4-1: The Failing Buffer System

4.1.1 Model Assumptions

In this chapter, we depart from the continuous material model presented in Chapter 2. Our starting point is the Schick and Gershwin (1978) synchronous, ODF, discrete material model. Machines are assumed to have geometrically distributed failures and repairs. Perfect yield and saturation are still assumed. However, we add the additional phenomenon of operation-dependent, geometrically distributed buffer failures and repairs. The nature of the failures and repairs, the order in which events are assumed to occur, and the movement of material are all very important and we investigate each of these issues below.

Failures Failures are operation dependent. The times between failures of the machines are assumed to be geometrically distributed, and uncorrelated. Buffers are often mechanisms themselves that may fail. Therefore, the times between failures of the buffer are also assumed to be geometrically distributed, and uncorrelated. We assume the buffer failures are operation dependent. This makes sense given that the failures of importance are due to mechanical errors and part jams. We assume that the probability of failure is unaffected by the number of pieces in the buffer because the likelihood of failure is linked only to the number of operations the buffer must complete. We assume the buffer cannot fail at any time when it is not working. For instance, it cannot fail if it has no material. It also will not fail if one of the machines is down and the other is either blocked or starved.

Order of Events The transformations of states in discrete time Markov processes are indexed by integers. A *cycle* is the time between changes in the index. Figure 4-2 illustrates a typical cycle. To make the Markov process precise, we need to specify the assumed order of events within a given cycle. Failures, when they occur, occur immediately at the beginning of the cycle, with buffer failures occurring before machine failures. In other words, if the buffer fails, then neither of the two machines may operate or fail during the same cycle. Repairs, if any occur immediately after failures. However, the repair must be of a failure that occurred in the previous time step or before. Changes in the buffer level occur at the end of the cycle. If a machine is working in a given time step, it transfers a part at the end of that time step.

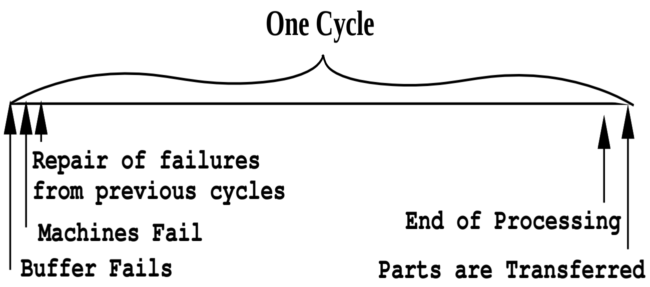

Figure 4-2: Event Points.

Repairs It is assumed that there are unlimited repair personnel. Therefore, repair times are uncorrelated. Repairs are also assumed to be geometrically distributed. The probability of any failed machine or buffer being repaired in a given time step is independent of the state of the rest of the system.

Movements of Material The *blocking mechanism* is the method by which material from M_1 is blocked when the buffer is full. As discussed in Dallery and Gershwin (1992), there are a number of different blocking mechanisms that may be assumed in transfer-line models. We assume M_1 will always operate unless the buffer is either full or failed. This type of blocking mechanism is often referred to as *communication blocking* or *BBS* (blocking before service). For example, if the buffer is full at the end of period t, M_1 must wait at least until the end of period $t+1$ until the buffer empties of one part and can accept another. Consequently, M_1 cannot transfer a part until at least the end of period $t+2$. We also assume the transportation delay of the buffer is negligible compared to the service times of the machines.

4.1.2 Model Notation

The system is modeled as a Markov chain. The state of the system is represented by

$$s(t) = (n(t), \alpha_1(t), \alpha_2(t), \alpha_b(t)).$$

We calculate

$$\mathbf{p}(s) = \text{steady-state probability of state } s.$$

Here n is an integer representing the buffer level ($0 \leq n \leq N$), α_i is the repair state of Machine i ($i = 1, 2$) and α_b is the repair state of the buffer. For each component of the system, $\alpha_j = 0$ if the machine or buffer is down and $\alpha_j = 1$ if it is up ($j = 1, 2, b$). The following is the definition of the failure and repair probabilities of machine i and for the repair and failure probabilities of buffer b:

$$p_i = \begin{cases} \text{probability that Machine } i \text{ will fail in the next time unit} \\ \text{given that it is currently up and not starved or blocked} \\ \text{by another machine or buffer.} \end{cases}$$

$$r_i = \begin{cases} \text{probability that Machine } i \text{ will be repaired in the next} \\ \text{time unit given that it is currently down.} \end{cases}$$

$$p_b = \begin{cases} \text{probability that the buffer will fail in the next time unit} \\ \text{given that it is currently up, not full, not empty and} \\ \text{either } M_1 \text{ or } M_2 \text{ is } up. \end{cases}$$

$$r_b = \begin{cases} \text{probability that the buffer will be repaired in the next} \\ \text{time unit given that it is currently down.} \end{cases}$$

The following indicator functions are also defined:

$$I_1(t) = \begin{cases} 1 & \text{if } \alpha_1(t+1) = 1,\ \alpha_b(t+1) = 1 \text{ and } n(t) < N \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$I_2(t) = \begin{cases} 1 & \text{if } \alpha_2(t+1) = 1,\ \alpha_b(t+1) = 1 \text{ and } n(t) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

That is, $I_j = 1$ indicates that a machine can actually transfer a part, while $I_j = 0$ indicates that the machine cannot ($j = 1, 2$). It accounts simultaneously for both machine and buffer states.

4.1.3 Performance Measures

The purpose of this research is to relate system throughput and average buffer level to buffer size and reliability. The production rate or throughput of Machine i in parts per time unit is E_i and is determined as follows:

$$\begin{aligned} E_1 &= \text{prob } (\alpha_1 = 1, n < N \text{ and } \alpha_b = 1) \\ E_2 &= \text{prob } (\alpha_2 = 1, n > 0 \text{ and } \alpha_b = 1). \end{aligned} \tag{4.3}$$

Since there is neither the creation nor destruction of material, flow is conserved and E_1 should equal E_2.

The average buffer level $\overline{n}$ can be calculated as follows:

$$\overline{n} = \sum_{n=0}^{N} n \sum_{\substack{\alpha_1=0,1 \\ \alpha_2=0,1 \\ \alpha_b=0,1}} \mathbf{p}(n, \alpha_1, \alpha_2, \alpha_b). \tag{4.4}$$

4.1.4 System Dynamics

The following is a formal definition of the transition probabilities:

$$\text{prob } [\alpha_1(t+1) = 1 | \alpha_1(t) = 0] = r_1 \tag{4.5}$$
$$\text{prob } [\alpha_1(t+1) = 0 | \alpha_1(t) = 0] = 1 - r_1 \tag{4.6}$$
$$\text{prob } [\alpha_2(t+1) = 1 | \alpha_2(t) = 0] = r_2 \tag{4.7}$$
$$\text{prob } [\alpha_2(t+1) = 0 | \alpha_2(t) = 0] = 1 - r_2 \tag{4.8}$$
$$\text{prob } [\alpha_b(t+1) = 1 | \alpha_b(t) = 0] = r_b \tag{4.9}$$
$$\text{prob } [\alpha_b(t+1) = 0 | \alpha_b(t) = 0] = 1 - r_b \tag{4.10}$$

$$\begin{aligned}
\text{prob }[\alpha_b(t+1)=0|\alpha_b(t)=1,\ 0<n<N \text{ and } \alpha_1(t)+\alpha_2(t)>0] &= p_b && (4.11)\\
\text{prob }[\alpha_b(t+1)=1|\alpha_b(t)=1,\ 0<n<N \text{ and } \alpha_1(t)+\alpha_2(t)>0] &= 1-p_b && (4.12)\\
\text{prob }[\alpha_1(t+1)=0|\alpha_b(t+1)=1,\ \alpha_1(t)=1,\ \alpha_b(t)=1 \text{ and } n<N] &= p_1 && (4.13)\\
\text{prob }[\alpha_1(t+1)=1|\alpha_b(t+1)=1,\ \alpha_1(t)=1,\ \alpha_b(t)=1 \text{ and } n<N] &= 1-p_1 && (4.14)\\
\text{prob }[\alpha_2(t+1)=0|\alpha_b(t+1)=1,\ \alpha_2(t)=1,\ \alpha_b(t)=1 \text{ and } n>0] &= p_2 && (4.15)\\
\text{prob }[\alpha_2(t+1)=1|\alpha_b(t+1)=1,\ \alpha_2(t)=1,\ \alpha_b(t)=1 \text{ and } n>0] &= 1-p_2 && (4.16)\\
\text{prob }[\alpha_b(t+1)=1|\alpha_b(t)=1 \text{ and } (\alpha_1(t)+\alpha_2(t)=0 \text{ or } n=N)] &= 1 && (4.17)\\
\text{prob }[\alpha_b(t+1)=0|\alpha_b(t)=1 \text{ and } (\alpha_1(t)+\alpha_2(t)=0 \text{ or } n=N)] &= 0 && (4.18)\\
\text{prob }[\alpha_b(t+1)=1|\alpha_b(t)=1 \text{ and } (\alpha_1(t)+\alpha_2(t)=0 \text{ or } n=0)] &= 1 && (4.19)\\
\text{prob }[\alpha_b(t+1)=0|\alpha_b(t)=1 \text{ and } (\alpha_1(t)+\alpha_2(t)=0 \text{ or } n=0)] &= 0 && (4.20)\\
\text{prob }[\alpha_1(t+1)=1|\alpha_1(t)=1 \text{ and } (\alpha_b(t)=0 \text{ or } n=N)] &= 1 && (4.21)\\
\text{prob }[\alpha_1(t+1)=0|\alpha_1(t)=1 \text{ and } (\alpha_b(t)=0 \text{ or } n=N)] &= 0 && (4.22)\\
\text{prob }[\alpha_2(t+1)=1|\alpha_2(t)=1 \text{ and } (\alpha_b(t)=0 \text{ or } n=0)] &= 1 && (4.23)\\
\text{prob }[\alpha_2(t+1)=0|\alpha_2(t)=1 \text{ and } (\alpha_b(t)=0 \text{ or } n=0)] &= 0. && (4.24)
\end{aligned}$$

For example, (4.13) says that if the buffer is not full (not blocking machine 1), M_1 is up and the buffer is working at time t and does not immediately fail, the probability that M_1 will be down at time $t+1$ is p_1. (4.18) and (4.20) indicate that neither machine can fail if the buffer is failed.

We also define the following dynamics of buffer level using I_j $(j=1,2)$ as defined in (4.1) and (4.2):

$$n(t+1) \;=\; n(t)+I_1(t)-I_2(t).$$

4.2 Model Development

In this section, transition equations are established to determine the steady state probabilities of the system. The desired performance measures (4.3) - (4.4) may then be derived from the steady state probabilities.

4.2.1 Transient States

A *transient state* in a Markov process has zero steady state probability. A transient state may only be reached from itself, another transient state, or it might not have any possible predecessor. The steady state analysis of this system can be simplified by ignoring all transient states. The following discusses which states in the system are transient and why. In most cases, we argue that for the system to be in a certain state at time $t+1$, it must have been in the same state or another state that is known to be transient at time t. In some cases, we show that there is no state that could have preceded the state under examination.

- All states of the form $(n, 0, 0, 0)$ are transient. If $\alpha_b(t) = 0$, neither of the machines can fail. If $\alpha_b(t) = 1$ and $(\alpha_1(t) = 1$ or $\alpha_2(t) = 1)$, the convention that the buffer fails before the machines means that any machine which was operational at time t remains operational at time $t + 1$. If $\alpha_1(t) = 0$, $\alpha_2(t) = 0$, and $\alpha_b(t) = 1$ the buffer cannot fail. Therefore, the only state from which $(n, 0, 0, 0)$ can be reached is itself and is thus transient.

- The state $(0, 1, 0, 1)$ is transient because it cannot be reached from any state. If $\alpha_1(t + 1) = 1$ and $\alpha_2(t + 1) = 0$, then $n(t + 1) = n(t) + 1 \geq 1$, since, where

$n(t) \geq 0$.

- The state $(0,1,0,0)$ is transient because it can only be reached from itself, or the transient states $(0,0,0,0)$ and $(0,1,0,1)$. Since $\alpha_b(t+1) = 0$, then $n(t+1) = n(t)$ and only states of the form $(0, \alpha_1, \alpha_2, \alpha_b)$ need to be considered. If $\alpha_2(t) = 1$ and $\alpha_b(t) = 0$ then $\alpha_2(t+1) = 1$. If $\alpha_1(t) = 0$ and $\alpha_b(t) = 1$, then $\alpha_b(t+1) = 1$. The state $(0,1,0,0)$ cannot be reached from $(0,1,1,1)$ because the buffer and M_2 cannot simultaneously fail.

- The state $(0,0,0,1)$ is transient because it can be reached only from itself, or the transient states $(0,0,0,0)$ or $(0,1,0,1)$. It cannot be reached from $(0,0,1,1)$, $(0,0,1,0)$, $(0,1,1,0)$, or $(0,1,1,1)$ since the second machine may not fail with the buffer empty. If $\alpha_1(t) = 1$, $\alpha_2(t) = 0$, and $\alpha_b(t) = 0$ then $\alpha_1(t+1)$ would have to be 1, which is not the case. Therefore, $(0,1,1,0)$ cannot be the prior state. If $\alpha_1(t) = 0$ and $\alpha_2(t) = 0$, then $n(t) = (t+1) = 0$ because $\alpha_1(t+1) = 0$ and $\alpha_2(t+1) = 0$. Therefore, only states of the form $(0, \alpha_1, \alpha_2, \alpha_b)$ need to be considered as possible previous states.

- The state $(0,0,1,0)$ is transient because it can only be reached from itself, or the transient state $(0,0,0,0)$. Since $\alpha_b(t+1) = 0$, then $n(t+1)$ would have to equal to $n(t)$. If $n(t) = 0$ and $\alpha_b(t) = 1$ then $\alpha_b(t+1)$ would have to be 1. If $\alpha_b(t) = 0$, then it must be the case that $\alpha_1(t) = 0$ and $\alpha_2(t) = 1$.

- The state $(0,1,1,1)$ is transient because if $\alpha_1(t+1) = 1$, $\alpha_2(t+1) = 1$, $\alpha_b(t+1) = 1$ and $n(t) \geq 0$, then it would have to be the case that $n(t+1) \geq 1$.

- The state $(0,1,1,0)$ is transient because it can only be reached from itself and the transient states $(0,0,0,0)$, $(0,0,1,0)$, $(0,1,0,0)$, $(0,1,0,1)$ and $(0,1,1,1)$. Again, since $\alpha_b(t+1) = 0$ implies that $n(t+1) = n(t)$, only states of the form $(0, \alpha_1, \alpha_2, \alpha_b)$ need be considered as possible previous states. If $\alpha_1(t) = 0$ and

$\alpha_b(t) = 1$, then it must be true that $\alpha_b(t+1) = 1$. This eliminates $(0, 0, 1, 1)$ and $(0, 0, 0, 1)$.

- The state $(1, 1, 0, 1)$ is transient because it can only be reached from the transient states $(0, 0, 0, 0)$, $(0, 0, 0, 1)$, $(0, 1, 0, 0)$, $(0, 1, 1, 1)$ and $(0, 1, 0, 1)$. If $\alpha_1(t+1) = 1$, $\alpha_2(t+1) = 0$, $\alpha_b(t+1) = 1$ and $n(t+1) = 1$, then $n(t)$ must equal 0. Therefore, only states of the form $(0, \alpha_1, \alpha_2, \alpha_b)$ need be considered as possible previous states. $(1, 1, 0, 1)$ cannot be directly preceded by $(0, 0, 1, 0)$ or $(0, 1, 1, 0)$ because M_2 cannot fail when the buffer is failed. Finally, $(1, 1, 0, 1)$ cannot be reached from $(0, 0, 1, 1)$ because the machines cannot fail when the buffer fails, and thus all cases are covered.

- The state $(1, 1, 0, 0)$ is transient because it may only be reached from itself and the transient states $(1, 1, 0, 1)$ and $(1, 0, 0, 0)$. This can be shown with a similar argument used for the state $(0, 1, 0, 0)$.

- By symmetric arguments, it can be similarly proven that the following states are also transient: $(N, 0, 0, 1)$, $(N, 0, 1, 0)$, $(N, 0, 1, 1)$, $(N, 1, 0, 0)$, $(N, 1, 1, 0)$, $(N, 1, 1, 1)$, $(N - 1, 0, 1, 0)$ and $(N - 1, 0, 1, 1)$.

4.2.2 Transition Equations

For each buffer level n between 2 and $N-2$, there exists a set of transition equations with a common structure illustrated by the state transition diagram in Figure 4-3. For the special states in which n equals 0, 1 , $N-1$, and N, the transition equations are different due to the boundary effects.

Internal Equations

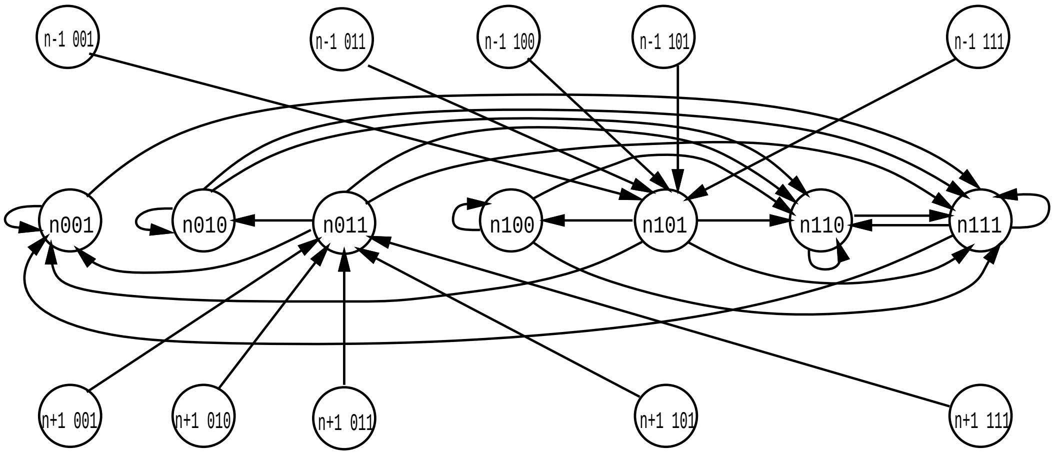

Figure 4-3: Internal State Transitions

Figure 4-3 represents some of the possible state transitions in the system when $2 \leq n \leq N-2$. The figure illustrates those transitions that result in the buffer level being at level n. For example, when the resultant state has both machines down as illustrated by state $(n,0,0,1)$, the buffer level will not change. That is, transitions may only come from previous states where the buffer level is n. Notice, that the system may only make a transition to a state with a different buffer level when the buffer is up.

The set of states in which there are n pieces in the buffer may only be entered from states with more or fewer pieces in the buffer through states $(n,0,1,1)$ or $(n,1,0,1)$. This is because for all other states, M_1 feeds a piece of material into the buffer for every piece that M_2 pulls from it. Each state transition with a positive conditional probability provided in (4.5) through (4.24) is represented by an arrow in Figure 4-3. Using these conditional probabilities, the following state transition equations are generated:

$$\mathbf{p}(n,0,0,1) = (1-r_1)(1-r_2)\mathbf{p}(n,0,0,1) + (1-r_1)p_2(1-p_b)\mathbf{p}(n,0,1,1) + p_1(1-r_2)(1-p_b)\mathbf{p}(n,1,0,1) + p_1p_2(1-p_b)\mathbf{p}(n,1,1,1) \tag{4.25}$$

$$\mathbf{p}(n,0,1,0) = (1-r_1)(1-r_b)\mathbf{p}(n,0,1,0) + (1-r_1)p_b\mathbf{p}(n,0,1,1) \tag{4.26}$$

$$\mathbf{p}(n,0,1,1) = (1-r_1)r_2\mathbf{p}(n+1,0,0,1) + (1-r_1)r_b\mathbf{p}(n+1,0,1,0) + (1-r_1)(1-p_2)(1-p_b)\mathbf{p}(n+1,0,1,1) + p_1r_2(1-p_b)\mathbf{p}(n+1,1,0,1) + p_1(1-p_2)(1-p_b)\mathbf{p}(n+1,1,1,1) \tag{4.27}$$

$$\mathbf{p}(n,1,0,0) = (1-r_2)p_b\mathbf{p}(n,1,0,1) + (1-r2)(1-r_b)\mathbf{p}(n,1,0,0) \tag{4.28}$$

$$\mathbf{p}(n,1,0,1) = r_1(1-r_2)\mathbf{p}(n-1,0,0,1) + r_1p_2(1-p_b)\mathbf{p}(n-1,0,1,1) + (1-r_2)r_b\mathbf{p}(n-1,1,0,0) + (1-p_1)(1-r_2)(1-p_b)\mathbf{p}(n-1,1,0,1) + (1-p_1)p_2(1-p_b)\mathbf{p}(n-1,1,1,1) \tag{4.29}$$

$$\mathbf{p}(n,1,1,0) = r_1(1-r_b)\mathbf{p}(n,0,1,0) + r_1p_b\mathbf{p}(n,0,1,1) + r_2(1-r_b)\mathbf{p}(n,1,0,0) + r_2p_b\mathbf{p}(n,1,0,1) + (1-r_b)\mathbf{p}(n,1,1,0) + p_b\mathbf{p}(n,1,1,1) \tag{4.30}$$

$$\mathbf{p}(n,1,1,1) = r_1r_2\mathbf{p}(n,0,0,1) + r_1r_b\mathbf{p}(n,0,1,0) + r_1(1-p_2)(1-p_b)\mathbf{p}(n,0,1,1) + r_2r_b\mathbf{p}(n,1,0,0) + (1-p_1)r_2(1-p_b)\mathbf{p}(n,1,0,1) + r_b\mathbf{p}(n,1,1,0) + (1-p_1)(1-p_2)(1-p_b)\mathbf{p}(n,1,1,1). \tag{4.31}$$

For example, we describe how it is possible to reach state $(n, 0, 0, 1)$ at time $t+1$ as represented in (4.25. In this state, both machines are down, there are n pieces of material in the buffer, and the buffer is operational. First, the system could have been in the same state at time t. Since machine repairs are independent, with probability $(1-r_1)(1-r_2)$ neither machine is repaired, no processing takes place and the system remains in state $(n, 0, 0, 1)$. Second, the system at time t could have been in a state where there were n pieces of material in the buffer, Machine 1 was down, Machine 2 was up and the buffer was also up. Since both machines are down at time $t+1$, the buffer level stays the same. With probability $(1-p_b)$, the buffer does not fail. Given that the buffer did not fail, the probability that Machine 2 fails is p_2. Again, since repairs are independent of system state, the probability that Machine 1 is not repaired is $(1-r_1)$. Therefore, the probability of going from state $(n, 0, 1, 1)$ to state $(n, 0, 0, 1)$ in one time step is $(1-r_1)p_2(1-p_b)$. A third possibility is that the system at time t could have been a state where there were n pieces of material in the buffer, Machine 1 was up, Machine 2 was down and the buffer was also up. By a similar argument as in the second case, the probability of going from state $(n, 1, 0, 1)$ to state $(n, 0, 0, 1)$ is $(1-r_2)p_1(1-p_b)$. Finally, if at time t, all machines and buffers are up and there are n pieces of material in the buffer, the probability that the buffer does not fail is $(1-p_b)$ and the probabilities of Machine 1 and Machine 2 failing given that the buffer has not failed are p_1 and p_2, respectively. Therefore, the probability of going from state $(n, 1, 1, 1)$ to state $(n, 0, 0, 1)$ is $p_1p_2(1-p_b)$.

Lower Boundary Equations

While the internal transition equations explain the behavior for states where $2 \leq n \leq N-2$, a different set of transitions occur for $n = 0, 1, N-1$, and N. The

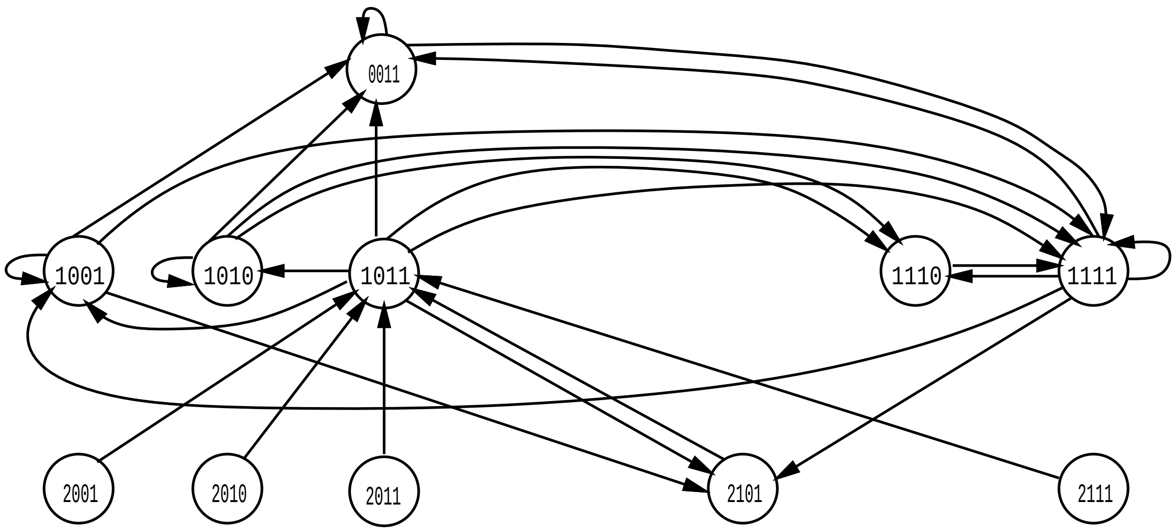

Figure 4-4: Lower Boundary State Transitions

following are the lower boundary equations, again generated by (4.5) through (4.24), and represented in Figure 4-4. Note that the transient states, which have steady state probabilities of zero, are ignored.

$$\begin{aligned}\mathbf{p}(0,0,1,1) \quad = \quad & (1-r_1)\mathbf{p}(0,0,1,1) + (1-r_1)r_2\mathbf{p}(1,0,0,1) + (1-r_1)r_b\mathbf{p}(1,0,1,0) + \\ & (1-r_1)(1-p_2)(1-p_b)\mathbf{p}(1,0,1,1) + \\ & p_1(1-p_2)(1-p_b)\mathbf{p}(1,1,1,1)\end{aligned} \tag{4.32}$$

$$\begin{aligned}\mathbf{p}(1,0,0,1) \quad = \quad & (1-r_1)(1-r_2)\mathbf{p}(1,0,0,1) + (1-r_1)p_2(1-p_b)\mathbf{p}(1,0,1,1) + \\ & p_1p_2(1-p_b)\mathbf{p}(1,1,1,1)\end{aligned} \tag{4.33}$$

$$\mathbf{p}(1,0,1,0) = (1-r_1)(1-r_b)\mathbf{p}(1,0,1,0) + (1-r_1)p_b\mathbf{p}(1,0,1,1) \tag{4.34}$$

$$\begin{aligned}\mathbf{p}(1,0,1,1) = \ & (1-r_1)r_2\mathbf{p}(2,0,0,1) + (1-r_1)r_b\mathbf{p}(2,0,1,0) + \\ & (1-r_1)(1-p_2)(1-p_b)\mathbf{p}(2,0,1,1) + p_1r_2(1-p_b)\mathbf{p}(2,1,0,1) + \\ & p_1(1-p_2)(1-p_b)\mathbf{p}(2,1,1,1)\end{aligned} \tag{4.35}$$

$$\begin{aligned}\mathbf{p}(1,1,1,0) = \ & r_1(1-r_b)\mathbf{p}(1,0,1,0) + r_1p_b\mathbf{p}(1,0,1,1) + \\ & (1-r_b)\mathbf{p}(1,1,1,0) + p_b\mathbf{p}(1,1,1,1)\end{aligned} \tag{4.36}$$

$$\begin{aligned}\mathbf{p}(1,1,1,1) = \ & r_1\mathbf{p}(0,0,1,1) + r_1r_2\mathbf{p}(1,0,0,1) + r_1r_b\mathbf{p}(1,0,1,0) + \\ & r_1(1-p_2)(1-p_b)\mathbf{p}(1,0,1,1) + r_b\mathbf{p}(1,1,1,0) + \\ & (1-p_1)(1-p_2)(1-p_b)\mathbf{p}(1,1,1,1)\end{aligned} \tag{4.37}$$

$$\begin{aligned}\mathbf{p}(2,1,0,1) = \ & r_1(1-r_2)\mathbf{p}(1,0,0,1) + r_1p_2(1-p_b)\mathbf{p}(1,0,1,1) + \\ & (1-p_1)p_2(1-p_b)\mathbf{p}(1,1,1,1).\end{aligned} \tag{4.38}$$

Upper Boundary Equations

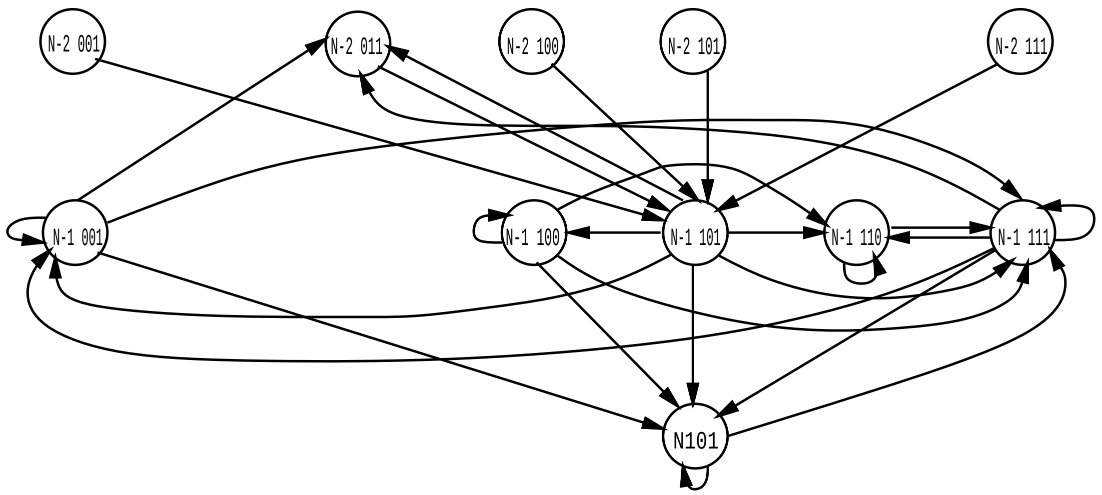

Figure 4-5: Upper Boundary State Transitions

Figure 4-5 is a state transition diagram for the upper boundary states. Figure 4-4 and Figure 4-5 reveal a symmetry between the upper and lower boundary transitions. The *BBS* assumption (Section 4.1.2) guarantees this symmetry. The following are the upper boundary equations representing the transitions in Figure 4-5:

$$\begin{aligned}\mathbf{p}(N,1,0,1) &= (1-r_2)\mathbf{p}(N,1,0,1) + r_1(1-r_2)\mathbf{p}(N-1,0,0,1) + \\ &\quad (1-r_2)r_b\mathbf{p}(N-1,1,0,0) + \\ &\quad (1-p_1)(1-r_2)(1-p_b)\mathbf{p}(N-1,1,0,1) + \\ &\quad (1-p_1)p_2(1-p_b)\mathbf{p}(N-1,1,1,1)\end{aligned} \tag{4.39}$$

$$\begin{aligned}\mathbf{p}(N-1,0,0,1) &= (1-r_1)(1-r_2)\mathbf{p}(N-1,0,0,1) + \\ &\quad p_1(1-r_2)(1-p_b)\mathbf{p}(N-1,1,0,1) + \\ &\quad p_1p_2(1-p_b)\mathbf{p}(N-1,1,1,1)\end{aligned} \tag{4.40}$$

$$\begin{aligned}\mathbf{p}(N-1,1,0,0) &= (1-r_2)(1-r_b)\mathbf{p}(N-1,1,0,0)+ \\ &\quad (1-r_2)p_b\mathbf{p}(N-1,1,0,1)\end{aligned} \tag{4.41}$$

$$\begin{aligned}\mathbf{p}(N-1,1,0,1) &= r_1(1-r_2)\mathbf{p}(N-2,0,0,1)+r_1p_2(1-p_b)\mathbf{p}(N-2,0,1,1)+ \\ &\quad (1-r_2)r_b\mathbf{p}(N-2,1,0,0)+ \\ &\quad (1-p_1)(1-r_2)(1-p_b)\mathbf{p}(N-2,1,0,1)+ \\ &\quad (1-p_1)p_2(1-p_b)\mathbf{p}(N-2,1,1,1)\end{aligned} \tag{4.42}$$

$$\begin{aligned}\mathbf{p}(N-1,1,1,0) &= r_2(1-r_b)\mathbf{p}(N-1,1,0,0)+r_2p_b\mathbf{p}(N-1,1,0,1)+ \\ &\quad (1-r_b)\mathbf{p}(N-1,1,1,0)+p_b\mathbf{p}(N-1,1,1,1)\end{aligned} \tag{4.43}$$

$$\begin{aligned}\mathbf{p}(N-1,1,1,1) &= r_2\mathbf{p}(N,0,1,1)+r_1r_2\mathbf{p}(N-1,0,0,1)+r_2r_b\mathbf{p}(N-1,1,0,0)+ \\ &\quad (1-p_1)r_2(1-p_b)\mathbf{p}(N-1,1,0,1)+r_b\mathbf{p}(N-1,1,1,0)+ \\ &\quad (1-p_1)(1-p_2)(1-p_b)\mathbf{p}(N-1,1,1,1)\end{aligned} \tag{4.44}$$

$$\begin{aligned}\mathbf{p}(N-2,0,1,1) &= (1-r_1)r_2\mathbf{p}(N-1,0,0,1)+p_1r_2(1-p_b)\mathbf{p}(N-1,1,0,1)+ \\ &\quad p_1(1-p_2)(1-p_b)\mathbf{p}(N-1,1,1,1).\end{aligned} \tag{4.45}$$

The reader interested in a more complete explanation of the equations above is directed to the end of the section on the internal transition equations.

4.3 Solution Technique

Using the transition equations of Section 4.2, steady state probabilities can be generated. First, unnormalized probabilities for the internal states are calculated, followed by the unnormalized probabilities for the lower boundary and upper boundary states. Then the unnormalized probabilities are normalized. It is important to notice that when p_b is set to zero, the resultant equations are identical to the reliable buffer case detailed in Gershwin (1994). This new model and proposed new method of solution may be viewed as a generalization of this previous work.

4.3.1 Internal States

There are seven states with potentially positive probabilities represented by (4.25) - (4.31) when there are n pieces of material in the buffer. Since the buffer level may only increase or decrease by 1 from one time period to the next, the number of transitions from $n+1$ to n equals the number of transitions from n to $n+1$ for all n. This is a property of general birth-death processes (Ross, 1983). This observation substantially reduces the complexity of solving the transition equations because the ratio of the probabilities for different machine and buffer states is constant for all n.

According to theinternal transition equations, the system may only enter into a state in which there are n pieces from a state in which there are $n+1$ pieces via the state $(n, 0, 1, 1)$. In addition, the system may only enter into a state in which there are n pieces from a state in which there are $n-1$ pieces via the state $(n, 1, 0, 1)$. (4.27) gives the total expected number of transitions per time unit from a state with a buffer level of n+1 to a state with a buffer level of n. By substituting n+1 for n, (4.29) gives the total expected number of transitions per time unit from a state with a buffer level of n to to a state with a buffer level of $n+1$ for $2 \leq n \leq N-2$. Equation (4.27) provides the expected number of transitions per time unit in the opposite direction. Therefore, since the rate of transitions in both directions must be equal in a steady state birth-death process we have

$$\mathbf{p}(n+1,1,0,1) = \mathbf{p}(n,0,1,1). \tag{4.46}$$

Equation (4.46) can be used in conjunction with (4.27) and (4.29) to obtain

$$\begin{aligned}
\mathbf{p}(n,0,1,1) &= r_1(1-r_2)\mathbf{p}(n,0,0,1) + r_1p_2(1-p_b)\mathbf{p}(n,0,1,1) + \\
&\quad (1-r_2)r_b\mathbf{p}(n,1,0,0) + (1-p_1)(1-r_2)(1-p_b)\mathbf{p}(n,1,0,1) + \\
&\quad (1-p_1)p_2(1-p_b)\mathbf{p}(n,1,1,1) && (4.47)\\
\mathbf{p}(n,1,0,1) &= (1-r_1)r_2\mathbf{p}(n,0,0,1) + (1-r_1)r_b\mathbf{p}(n,0,1,0) + \\
&\quad (1-r_1)(1-p_2)(1-p_b)\mathbf{p}(n,0,1,1) + p_1r_2(1-p_b)\mathbf{p}(n,1,0,1) + \\
&\quad p_1(1-p_2)(1-p_b)\mathbf{p}(n,1,1,1). && (4.48)
\end{aligned}$$

There are now seven equations: (4.25), (4.26), (4.28), (4.30) (4.31), (4.47) and (4.48). All these equations have the same buffer level on both sides. These seven equations have seven unknown functions of n. By manipulating (4.26) and (4.28), the following equations emerge

$$\mathbf{p}(n,0,1,0) = \left(\frac{(1-r_1)p_b}{1-(1-r_1)(1-r_b)}\right)\mathbf{p}(n,0,1,1) \tag{4.49}$$

$$\mathbf{p}(n,1,0,0) = \left(\frac{(1-r_2)p_b}{1-(1-r_2)(1-r_b)}\right)\mathbf{p}(n,1,0,1). \tag{4.50}$$

By substituting (4.49) and (4.50) into (4.25), (4.47), and (4.48), we obtain (4.51), (4.52) and (4.53). We then add (4.30) and (4.31) to eliminate **p**(n,1,1,0) and substitute (4.49) and (4.50) to eliminate **p**(n,0,1,0) and **p**(n,1,0,0) and the result is (4.54). The following four equation, four unknown system is obtained:

$$\begin{aligned}\mathbf{p}(n,0,0,1) &= (1-r_1)(1-r_2)\mathbf{p}(n,0,0,1) + (1-r_1)p_2(1-p_b)\mathbf{p}(n,0,1,1) + \\ &\quad p_1(1-r_2)(1-p_b)\mathbf{p}(n,1,0,1) + p_1p_2(1-p_b)\mathbf{p}(n,1,1,1)\end{aligned} \tag{4.51}$$

$$\begin{aligned}\mathbf{p}(n,0,1,1) &= r_1(1-r_2)\mathbf{p}(n,0,0,1) + r_1p_2(1-p_b)\mathbf{p}(n,0,1,1) + \\ &\quad \left(\frac{(1-r_2)^2 r_b p_b}{1-(1-r_2)(1-r_b)} + (1-p_1)(1-r_2)(1-p_b)\right)\mathbf{p}(n,1,0,1) + \\ &\quad (1-p_1)p_2(1-p_b)\mathbf{p}(n,1,1,1)\end{aligned} \tag{4.52}$$

$$\begin{aligned}\mathbf{p}(n,1,0,1) &= (1-r_1)r_2\mathbf{p}(n,0,0,1) + \\ &\quad \left(\frac{(1-r_1)^2 r_b p_b}{1-(1-r_1)(1-r_b)} + (1-r_1)(1-p_2)(1-p_b)\right)\mathbf{p}(n,0,1,1) \\ &\quad p_1r_2(1-p_b)\mathbf{p}(n,1,0,1) + p_1(1-p_2)(1-p_b)\mathbf{p}(n,1,1,1)\end{aligned} \tag{4.53}$$

$$\begin{aligned}\mathbf{p}(n,1,1,1) &= r_1r_2\mathbf{p}(n,0,0,1) + \\ &\quad \left(\frac{r_1(1-r_1)p_b}{1-(1-r_1)(1-r_b)} + r_1(1-p_2)(1-p_b) + r_1p_b\right)\mathbf{p}(n,0,1,1) + \\ &\quad \left(\frac{r_2(1-r_2)p_b}{1-(1-r_2)(1-r_b)} + (1-p_1)r_2(1-p_b) + r_2p_b\right)\mathbf{p}(n,1,0,1) + \\ &\quad \left((1-p_1)(1-p_2)(1-p_b) + p_b\right)\mathbf{p}(n,1,1,1).\end{aligned} \tag{4.54}$$

We assume that this system of equations (which is independent of n) has a non-unique solution. We obtain this solution using a standard matrix inversion software algorithm (Press *et al.*; 1988). Therefore, the seven probabilities at each level of buffer must have the form:

$$\begin{aligned}
\phi_n &= (\mathbf{p}(n,0,0,1), \mathbf{p}(n,0,1,0), \mathbf{p}(n,0,1,1), \mathbf{p}(n,1,0,0), \mathbf{p}(n,1,0,1), \\
&\quad \mathbf{p}(n,1,1,0), \mathbf{p}(n,1,1,1)) \\
&= K_n(\phi(0,0,1), \phi(0,1,0), \phi(0,1,1), \phi(1,0,0), \phi(1,0,1), \\
&\quad \phi(1,1,0), \phi(1,1,1)) \qquad (4.55) \\
&= K_n \phi
\end{aligned}$$

where K_n is a scalar function of n and ϕ is the vector ($\phi(0,0,1)$, $\phi(0,1,0)$, $\phi(0,1,1)$, $\phi(1,0,0)$, $\phi(1,0,1)$, $\phi(1,1,0)$, $\phi(1,1,1)$). From (4.27),

$$\begin{aligned}
K_n\phi(0,1,1) &= (1-r_1)r_2K_{n+1}\phi(0,0,1) + (1-r_1)r_bK_{n+1}\phi(0,1,0) + \\
&\quad (1-r_1)(1-p_2)(1-p_b)K_{n+1}\phi(0,1,1) + \\
&\quad p_1r_2(1-p_b)K_{n+1}\phi(1,0,1) + \\
&\quad p_1(1-p_2)(1-p_b)K_{n+1}\phi(1,1,1). \qquad (4.56)
\end{aligned}$$

Let X_n be the ratio of K_n to K_{n+1} so that:

$$\begin{aligned}
X_n &= \frac{K_n}{K_{n+1}} \\
&= \frac{(1-r_1)r_2\phi(0,0,1) + (1-r_1)r_b\phi(0,1,0)}{\phi(0,1,1)} + \\
&\quad \frac{(1-r_1)(1-p_2)(1-p_b)\phi(0,1,1) + p_1r_2(1-p_b)\phi(1,0,1)}{\phi(0,1,1)} + \\
&\quad \frac{p_1(1-p_2)(1-p_b)\phi(1,1,1)}{\phi(0,1,1)}. \qquad (4.57)
\end{aligned}$$

X_n could be similarly calculated from (4.28). The only assumption on n is that it is between 2 and $N-2$. It is important to see that $X_n = X$ is independent of n. Therefore:

$$\phi_n = \phi_{n+1} X$$

or

$$\phi_n = X^{-n}\phi. \tag{4.58}$$

4.3.2 Boundary Solutions

In the previous section, an unnormalized solution to the internal states is derived. With this result, determining the unnormalized boundary state probabilities is straightforward. Starting with the lower boundary equations (4.32)-(4.38), $\mathbf{p}$(2,1,0,1) and $\mathbf{p}$(1,0,1,1) can be treated as internal states so that:

$$\mathbf{p}(2,1,0,1) = \phi(1,0,1)X^{-2} \tag{4.59}$$

$$\mathbf{p}(1,0,1,1) = \phi(0,1,1)X^{-1} \tag{4.60}$$

By manipulating (4.34) , it is easy to show that:

$$\begin{aligned} \mathbf{p}(1,0,1,0) &= \left(\frac{(1-r_1)p_b}{1-(1-r_1)(1-r_b)}\right)\mathbf{p}(1,0,1,1) \\ &= \left(\frac{(1-r_1)p_b}{1-(1-r_1)(1-r_b)}\right)\phi(0,1,1)X. \end{aligned} \tag{4.61}$$

Four lower boundary equations (4.32), (4.33), (4.36) and (4.37) in four unknowns ($\mathbf{p}$(0,0,1,1), $\mathbf{p}$(1,0,0,1), $\mathbf{p}$(1,1,1,0) and $\mathbf{p}$(1,1,1,1)) remain. These can all be solved for in terms of the internal state probabilities. A similar analysis may used to determine the probabilities associated with the upper boundary states in terms of the internal state probabilities.

4.3.3 Normalization

Finally, these equations must satisfy

$$1 = \sum_{n=0}^{N} \sum_{\substack{\alpha_1=0,1 \\ \alpha_2=0,1 \\ \alpha_b=0,1}} \mathbf{p}(n, \alpha_1, \alpha_2, \alpha_b). \tag{4.62}$$

Once the probabilities have been normalized, average system throughput and buffer level may be calculated from (4.3) and (4.4).

4.4 Numerical Tests

We wrote a computer program in C to test the analytic technique developed in the previous sections for generating efficiency E and average buffer level $\overline{n}$. Four cases are discussed in Section 4.4.1. The effects of buffer reliability on E are presented in Section 4.4.2, and the effects on $\overline{n}$ are presented in Section 4.4.3.

4.4.1 Cases

A large number of experiments were run for a diversity of parameter values, and the results were similar in all cases. Four specific cases (Table 4.1) were chosen for demonstrating the phenomena observed. For each of the four cases, the following information is provided: isolated failure probability of M_i (p_i), isolated mean time between failures of M_i ($1/p_i$), repair probability of M_i (r_i), mean repair time of M_i ($1/r_i$), the isolated efficiency of M_i (e_i), operation-dependent failure probability of the buffer (p_b), isolated mean time between failures of the buffer ($1/p_b$), repair probability of the buffer (r_b) and mean repair time of the buffer ($1/b_i$). Table 4.1 provides a list of parameter chosen for the cases based on actual observations of manufacturing systems. A repair rate for the buffer (r_b) of 0.008 (average repair time of 125 time units) was chosen for all cases. The values of the parameter p_b were chosen over a broad range of values (including unrealistic values) to check the robustness of the methodology. We tested buffer sizes for all integer values between 0 and 300.

	Case 1	Case 2	Case 3	Case 4
p_1	0.005	0.001	0.005	0.005
r_1	0.050	0.050	0.050	0.050
e_1	0.909	0.980	0.909	0.909
p_2	0.001	0.005	0.005	0.005
r_2	0.050	0.050	0.050	0.007
e_2	0.980	0.909	0.909	0.569
p_b	*varied*	*varied*	*varied*	*varied*
r_b	0.008	0.008	0.008	0.008

Table 4.1: Unreliable Experiment Case Parameters

The following is a description of the different cases:

Case 1

Case 1 is a line with two relatively reliable machines where the first machine is approximately 10 percent less efficient than the second.

Case 2

Case 2 is identical to Case 1, except that the machines have been switched.

Case 3

Case 3 is a line with two identical machines.

Case 4

Case 4 is a reliable machine followed by an unreliable machine. This case becomes important when the two-stage models are used for decomposition analyses of longer lines. This is because, in decomposition, the two-stage models are used to evaluate pseudo-machines that represent the combined behavior of all machines up- and downstream of a buffer. The effects of aggregating machines can often lead to two-machine lines in which the machines are very different in their efficiencies and reliability behavior.

4.4.2 Effects on Line Efficiency

Efficiency (E) is the expected number of units produced in T time units divided by T when the system is in steady state. In this section, we look at the effects of changes in system parameters on E. Each Case 1 curve in Figure 4-6 approaches a limit as the buffer size increased from 0 to 300. This figure also illustrates that as the value of p_b was varied, the reliability of the buffer (e_b) was increased from 95% to 100%, and the E associated with each buffer size approached the graphs obtained from experiments

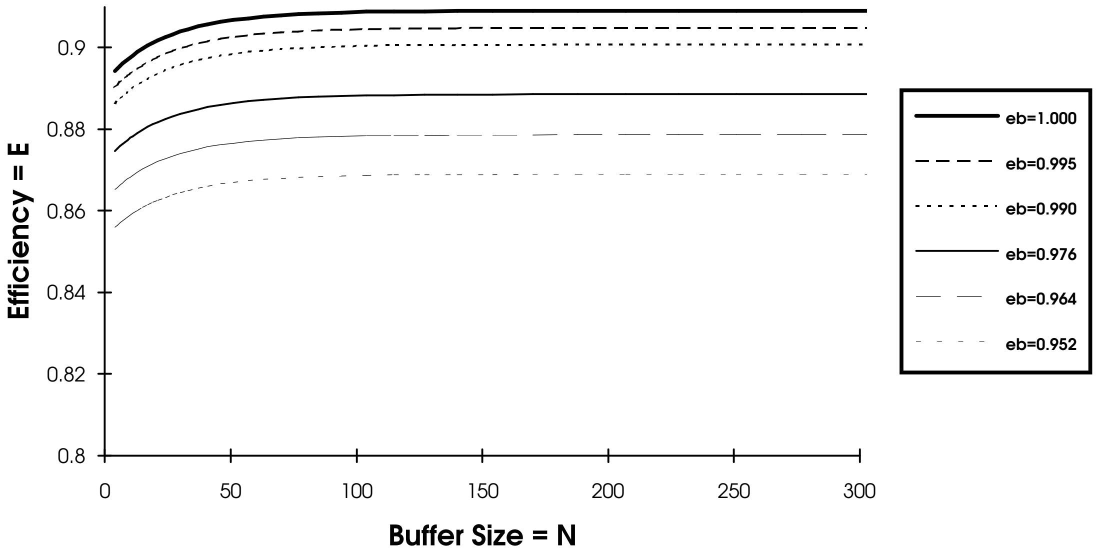

Figure 4-6: Failing Buffer Case 1 and 2

with two-machine models (illustrated by the darkest solid black curve), that assume reliable buffers (Gershwin, 1994). These curves indicate the detrimental impact of inefficient buffers on overall line efficiency.

In a *reversed flow line*, each machine i is replaced by machine $k+i-1$ and each Buffer i is replaced by Buffer $k-i$. Case 2 is the *reverse* of Case 1. However, both can be illustrated by Figure 4-6 because the machine order did not have an impact on E. This phenomenon of flow line models, called *reversibility*, was observed in all the cases we tested. This is a theoretical result which can be proven by comparing the Markov chains of the reversed systems (Ammar and Gershwin, 1989).

In the experiments in which the differences between isolated machine efficiencies were small (Case 1 and 2), the system efficiency levels approached their limit at relatively low ($\approx$ 100) buffer capacities. However, Figure 4-7 illustrates the results

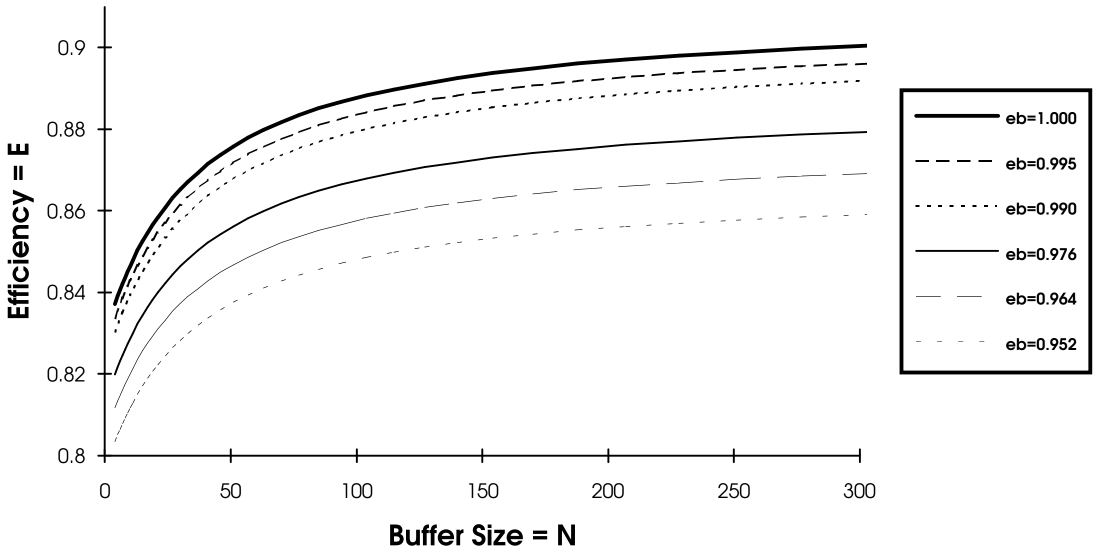

Figure 4-7: Case 3 - Identical Machines

from Case 3 in which the machines were identical. It is drawn on the same scale as Figure 4-6 to illustrate the steeper slope in system efficiency at the lower levels of buffer capacity. The slopes of the efficiency curves remain significant even when buffer capacities exceeded 300. The graphs indicate that buffer size has a greater impact on system performance when a line is highly balanced. We compare Cases 1 and 2 with Case 3 to illustrate that when faced with the task of increasing throughput, increasing the efficiency of one machine in the line may have a similar effect on E as increasing the buffer capacity. However, in Section 4.4.3 we demonstrate that if the second machine is made more efficient (Case 1) the associated WIP is much lower than if the buffer size is increased or if the first machine efficiency is increased.

Case 4 was chosen to illustrate the effect of a very inefficient machine (with a 57% isolated efficiency) on the system. Individual machines would usually not have

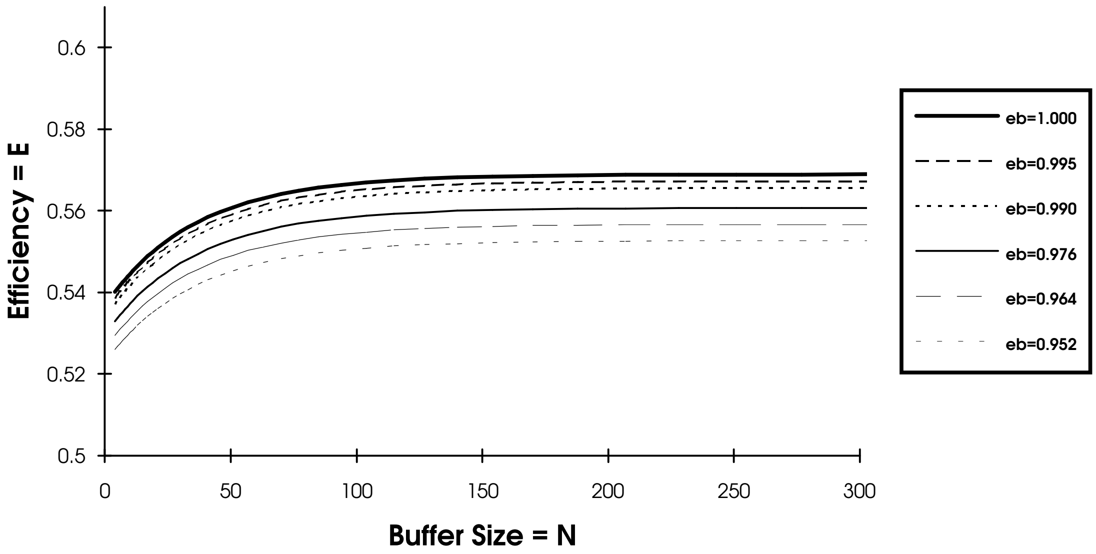

Figure 4-8: Case 4 - Low Efficiency Test

such poor performance. However, the decomposition techniques for dealing with the longer flow lines discussed in Chapters 2 and 3 approximate the behavior of the entire system up- and down-stream of each buffer as one machine. These two machines and the buffer between them are evaluated using two-machine models similar to the model discussed here. These pseudo-machines may often be inefficient. Experiments with Case 4 demonstrated that even systems with very inefficient machines demonstrated reversibility. In addition, the results obtained as the buffer became more efficient again approached the graphs for reliable buffer system ($e_b = 1$) presented in Gershwin (1994).

4.4.3 Effects on Average Buffer Level

The effects of isolated buffer efficiency on average buffer level were minimal in the numerical tests. As shown in Figure 4-9, the difference for Case 1 for varying values of e_b between 61% and 100% is less than 1 unit of material. This demonstration of the relatively low effect of buffer efficiency on average buffer level is consistent with the results of all of our experiments. However, Figure 4-9 also illustrates that as isolated buffer efficiency approaches 1, the results again approach those of the two-machine reliable buffer models (Gershwin, 1973, 1994) represented by the darkest sold black curve.

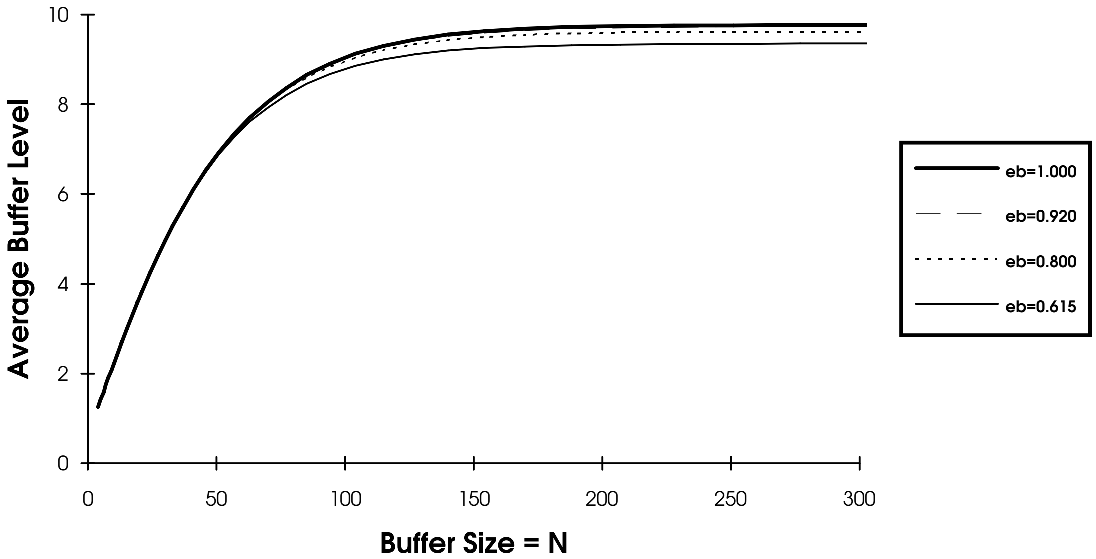

Figure 4-9: Case 1 - Average Buffer Level

The following is a possible explanation for why buffer efficiency had such a low impact on average buffer levels. Buffer failures have a low correlation to the buffer level. Machines cannot fail when the buffer is failed. Consequently, the buffer level

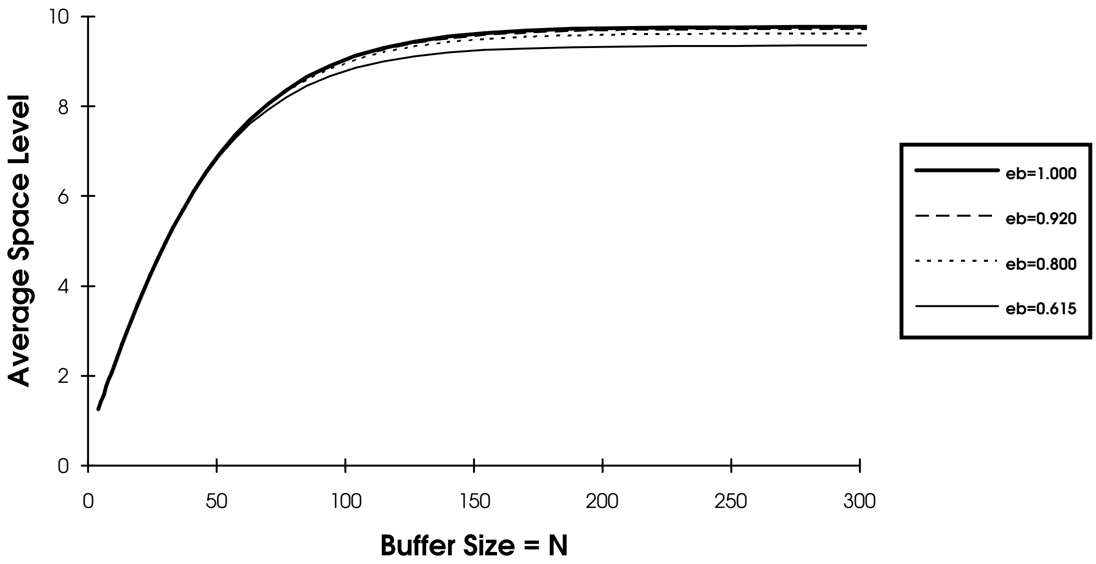

Figure 4-10: Case 2 - Space Processing System

distribution when the buffer is down is very similar to the distribution when the buffer is up. The distributions are not exactly the same, because a machine that was down when the buffer failed may be repaired during the buffer failure. However, when the buffer fails when both machines are up, the buffer level stays constant until the buffer is repaired.

We briefly discuss reversability as it relates to efficiency in the previous section. Average buffer levels are also affected by the reversibility properties. When a production line is reversed, one may look at empty spaces in the buffer as pieces. These empty spaces behave the same as parts would through a system that is the exact reverse of the original system. Spaces and units of material in the buffer must always sum to the size of the buffer. Therefore, the average amount of material in the buffer in the reversed system must be equal to the number of spaces in the original system.

We observed these results in all of our experiments.

Figure 4-9 illustrates the average buffer levels for Case 1 while Figure 4-10 illustrates the average number of spaces in the buffer for Case 2 (which is the reverse of Case 1.) These graphs are identical, thus demonstrating this point. As further proof of this concept, the average buffer size for Case 3 (not shown) was always exactly half of the total buffer capacity for all values of N. This is a consequence of the first argument. We observed this reversible behavior in all of our experiments.

4.5 Proposed Heuristics

The structure of the graphs suggested two heuristics for estimating the system efficiency of the two-machine, failing buffer system. A simple extension of 0-buffer models (Buzacott, 1967) is discussed in Section 4.5.1. A second heuristic is discussed in Section 4.5.2.

4.5.1 Zero-Buffer Heuristic

Buzacott (1967) provides a formula for determining the efficiency of a long line, synchronous processing time, zero-buffer system. For a two-machine line his formula reduces to:

$$E = \frac{1}{1 + \frac{p_1}{r_1} + \frac{p_2}{r_2}}. \tag{4.63}$$

Imagine a two-machine, reliable buffer system. Now imagine the same system with an unreliable buffer. Whenever the buffer goes down it shuts down the original system. When the original system is not operating it effectively shuts down the buffer. This is the same type of behavior observed in two-stage systems with no buffer in which we treat the entire two machine, reliable buffer system as the first machine in his model and the buffer failures as the second machine. Let $E_2^R(r_1, p_1, r_2, p_2, N)$ as the efficiency of a two-machine line with a reliable buffer. Treat this system as a single machine with a mean time to fail of $1/p_R$ and a mean time to repair of $1/r_R$. Then

$$\begin{aligned} E_2^R(r_1, p_1, r_2, p_2, N) &= \frac{r_R}{r_R + p_R} \\ &= \frac{1}{1 + \frac{p_R}{r_R}}. \end{aligned} \tag{4.64}$$

Thus, we have

$$\frac{p_R}{r_R} = \frac{1}{E_2^R(r_1, p_1, r_2, p_2, N)} - 1. \tag{4.65}$$

Assume that the buffer failures act as the other machine. Equation (4.63) determines the failing-buffer system efficiency ($E_F^{H_1}$) as follows:

$$E_F^{H_1} = \frac{1}{\frac{p_b}{r_b} + \frac{1}{E_2^R(r_1, p_1, r_2, p_2, N)}}. \tag{4.66}$$

Figure 4-11 illustrates the comparison between the analytic estimate and this 0-buffer heuristic for Case 4 when the buffer capacity (N) is set at 4. This particular case was chosen for illustration purposes, because in all the experiments, both this 0-buffer heuristic and the linear heuristic in the next section performed the worst under this scenario. This scenario also emphasized the effects of reducing e_b.

Numerical experiments using (4.66) invariably estimated efficiency within one-tenth of a percent of the analytical model in all cases tested. Notice that this difference is indistinguishable in Figure 4-11.

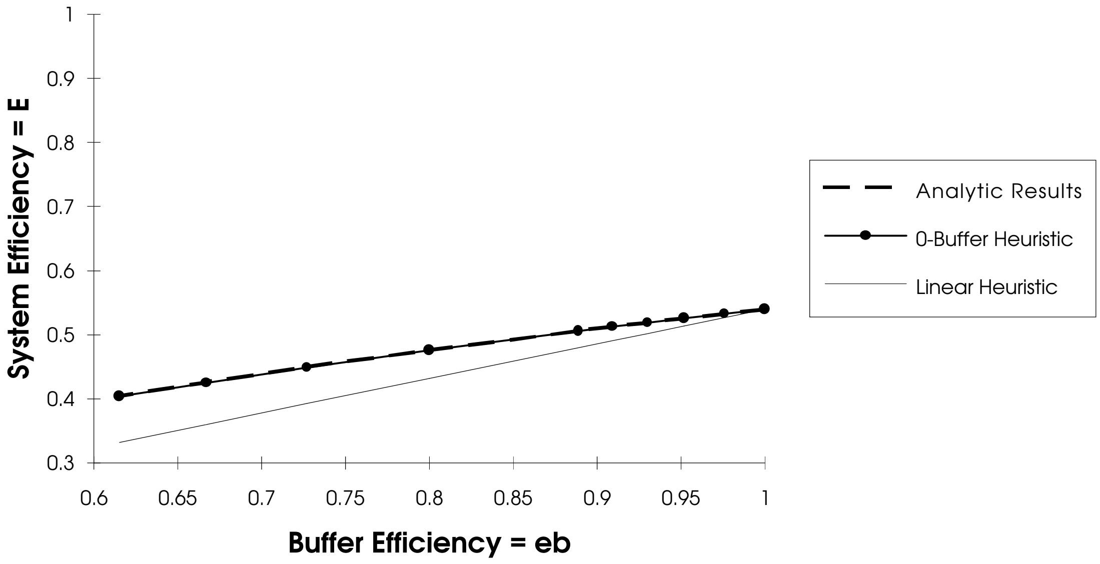

Figure 4-11: Case 4 - Analytic Results and Heuristics

4.5.2 Linear Heuristic

A second and simpler heuristic is proposed that linearizes the effect of the buffer efficiency. This is identical to the approximation of the time-dependent solution proposed in Lipset *et al.* (1994a). Using $E_2^R(r_1, p_1, r_2, p_2, N)$ as defined in (4.64), the linear heuristic efficiency estimate ($E_F^{H_2}$) is

$$E_F^{H_2}(r_1, p_1, r_2, p_2, N) = e_b E_2^R(r_1, p_1, r_2, p_2, N). \tag{4.67}$$

We experimented with (4.67) for all previous cases. Typical results obtained when using this linear heuristic are illustrated in Figure 4-11. For e_b above 0.99, this linear heuristic came within one percent of the analytic estimate. However, as e_b decreased, the estimate became consistently worse. Observations at a number of manufacturing facilities indicate that high buffer efficiencies are actually quite common. Therefore, for practical purposes, this method seems adequate for most systems. However, as indicated in Figure 4-11, (4.66) performed better than (4.67) under a much wide set of conditions.

4.6 Conclusion

We have shown how to extend previous two-machine, reliable buffer models to account for failing buffers. Two heuristics also demonstrated a performance close to the analytic model. In the future, we will try to use these results in the other two-stage and decomposition models described in Gershwin (1994).

Chapter 5

Series–Parallel Flow Lines

This chapter extends our analysis to more complicated flow lines. It is less mathematically rigorous than the previous chapters and is presented in a more exploratory manner. We complicate the flow line by letting stages consist of parallel machines. The technique proposed in this chapter approximates each stage of the original flow line as a single unreliable machine by matching the moments of the amount produced in a specified time interval. The system is reduced to a single machine per stage flow line of the type discussed in Chapter 2. Reasonable estimates of expected throughput and average buffer levels can then be quickly obtained using the decomposition procedure. Each stage in the flow line now represents a set of parallel machines as illustrated in Figure 5-1. Machine M_{ij} represents the jth machine at stage i of J_i machines. We call this new system a *series-parallel* flow line.

Multiple machine stages are quite common in industry. Technological limitations often prevent machines from performing effectively beyond a specific speed that is significantly slower than the other stages in the line. In such cases, multiple machines

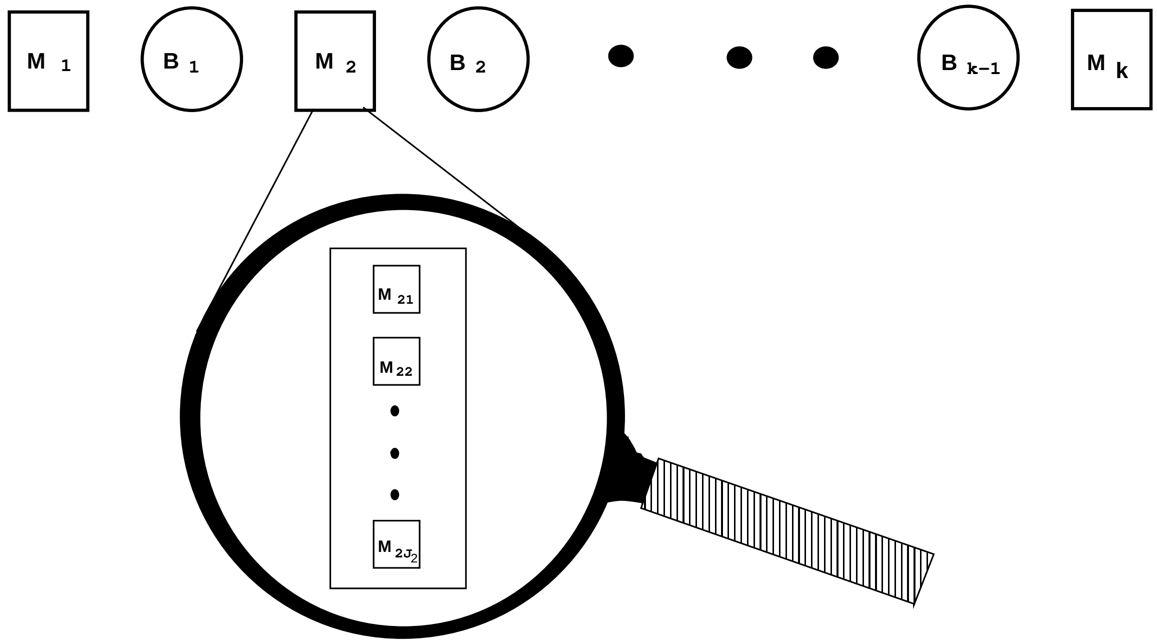

Figure 5-1: Buffered Multi-Machine Stage Flowline

are used to increase the capacity of the work center. Some machines may also be subject to significant failures. So, while a machine might have a sufficient expected throughput to meet demand, it may fail for considerable periods of time. This can wreak havoc on delivery schedules and customer satisfaction. Redundant machinery may then be employed to reduce processing rate variability. Even when machines have sufficient capacity and are relatively reliable, there are still other reasons for stacking capacity. For example, the machine might be up- or downstream of an expensive system bottleneck for which it is desirable to avoid starvation or blockage.

While a wide range of literature exists on the subject of transfer lines as discussed in the main literature review in Section 1.3, very little work has been produced in the area of flowlines with stages consisting of banks of parallel machines. Ignall and

Silver (1977) estimated the performance of a two-stage discrete material, synchronous flow line with multiple identical machines at each stage. The method made use of Buzacott's (1977) observation that the efficiency of a two-stage flow line with buffer capacity N can be written as a convex combination of E_0 (expected throughput of the same machine configuration with zero buffer capacity) and E_∞ (expected throughput of the same machine configuration with infinite buffer capacity).

Elasyed and Hwang (1984) considered a similar two-stage system. However, they assumed that only one machine was required to meet demand at each stage and that the additional machines operated only when the primary machine went down. Forestier (1980) and Mitra (1986) both formulated series parallel generalizations of the continuous material two-stage models (Zimmern, 1956; Wijngaard, 1979; Gershwin and Schick, 1980).

Iyama and Ito (1987) analyzed a line with parallel machines and exponential processing times. They chose a Markov chain approach that became impractical for solving for lines with more than three stages. Alves (1990) was able to overcome the line length limitation by formulating the problem similarly to Iyama and Ito, but with the additional restriction of zero buffer capacities. These models effectively reduced each set of parallel machines to an equivalent single machine that operated according to a complicated Markov process. The primary limitation to this method is the restrictions on the complexity of solvable cases. The model presented in this chapter follows the assumptions of the continuous material models, allows for buffers of non-zero capacity and significantly reduces the complexity of obtaining a solution. This is achieved by reducing each set of parallel machines to an equivalent single machine that operates according to a simple Markov process. The result is a single machine per stage flow line for which we have already established a solution.

A brief description of our model is provided in Section 5.1. A general description of a method for establishing a single machine equivalent to a set of parallel machines is described in Section 5.2. Specific techniques for estimating the parameters for the single machine equivalent are developed in Section 5.3. Numerical experiments are provided in Section 5.4. Finally, conclusions and future research directions are discussed in Section 5.5.

5.1 The Series–Parallel Model

Most assumptions about the series-parallel system are same as those for the continuous material model in Chapter 2. We model the flow line as a continuous time, mixed state Markov process. We assume machine processing rates are non-homogeneous, machines are unreliable with ODFs, yield is 100%, and failure and repair times are independent. Work Center i (M_i) has J_i machines.

As in the single machine per stage flow line, a machine or work center produces at its maximum rate when it (all machines in work center) is up, and not blocked or starved.

We define

$$\mu_{ij}\delta t = \begin{cases} \text{the amount of material processed by } M_{ij} \text{ in } (t, t+\delta t) \text{ for} \\ \text{small } \delta t \text{ when it is not blocked or starved} \end{cases}$$

$$\begin{aligned} \mu_i\delta t &= \begin{cases} \text{the amount of material processed by } M_i \text{ in } (t, t+\delta t) \text{ for} \\ \text{small } \delta t \text{ when it is not blocked or starved} \end{cases} \\ &= \sum_{i=1}^{J_i} \mu_{ij}\delta t. \end{aligned} \tag{5.1}$$

For each Machine ij (M_{ij}) in M_i, we define:

$$\begin{aligned}
\mu_{ij}(t)\delta t &= \begin{cases} \text{the amount of material processed by } M_{ij} \text{ in } (t, t+\delta t) \text{ for} \\ \text{small } \delta t \end{cases} \\
\mu_{i}(t)\delta t &= \begin{cases} \text{the amount of material processed by } M_{i} \text{ in } (t, t+\delta t) \text{ for} \\ \text{small } \delta t. \end{cases}
\end{aligned} \tag{5.2}$$

Each machine in the system is considered to be unreliable and goes up and down randomly. Failures are operation dependent. Formally, we define

$$\begin{aligned}
p_{ij}\delta t &= \begin{cases} \text{the probability that } M_{ij} \text{ goes from } \textit{up} \text{ to } \textit{down} \\ \text{during } (t, t+\delta t) \text{ when it is not blocked or starved,} \\ \text{for small } \delta t \end{cases} \\
r_{ij}\delta t &= \begin{cases} \text{the probability that } M_{ij} \text{ goes from } \textit{down} \text{ to } \textit{up} \\ \text{during } (t, t+\delta t) \text{, for small } \delta t. \end{cases}
\end{aligned} \tag{5.3}$$

As in Chapter 2, failure rates are proportional to the speed a machine is operating relative to its maximum isolated production rate.

Each Machine ij in Work Center i (M_{ij}) is capable of processing material at a maximum rate of μ_{ij}. It will process material at this rate unless impeded by a failure, blockage or a starvation of raw material. A detailed description of the processing behavior of a single machine is presented in Chapter 2. When a work center is either partially blocked or partially starved of material, each machine produces material at a rate proportional to its production capacity as a percentage of total capacity of the work center. For example, if Work Center i is processing material at rate $\mu_i(t)$ at time t, where $\mu_i(t) \le \mu_i$, then M_{ij} processes at the rate

$$\mu_{ij}(t) = \mu_i(t)\frac{\mu_{ij}}{\mu_i} \qquad \text{at time } t. \tag{5.4}$$

Buffer B_i can hold a maximum amount of material equal to N_i and behaves in the following identical manner to that presented in Chapter 2

$$\dot{n}_i(t) = \mu_i(t) - \mu_{i+1}(t). \tag{5.5}$$

Now that the system has been described, we will proceed in the next section to demonstrate how to approximate the performance of this system using the decomposition technique.

5.2 Solution Description

We represent the flow line by a Markov process. The long-term average behavior of the flow line can then be determined by the steady state probabilities of the Markov process. The state space of a series-parallel line grows large with the complexity of the system, and even more so than the single machine per stage systems. For example, a simple 4-stage line with 3 machines at each stage and buffer with capacity of 100 would have $2^{4*3} * 100^3$ or approximately 4 billion states. Typical manufacturing processes may have as many 10, 20 or possibly many more stages. In high volume consumer product production, buffers with capacities on the order of 1000 part types are quite common. To overcome this immense complexity, we attempt to represent each stage by an equivalent single machine (M_i') which is established independently of the rest of the machines and buffers in the line as illustrated in Figure 5-2.

It is at this point that our approach fundamentally differs from previous work. The decomposition technique developed in earlier chapters is extremely efficient in

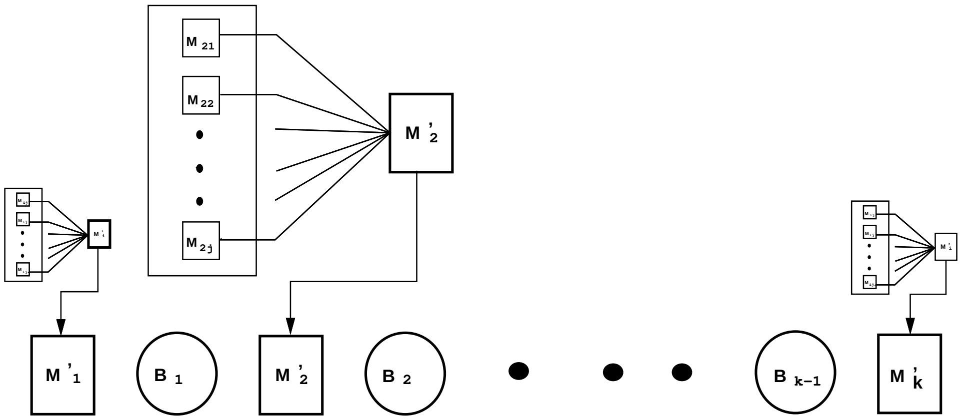

Figure 5-2: Approximated Flowline

estimating critical performance characteristics of traditional flowlines. However, the method requires that each stage be described in terms of the three parameters p_i, r_i and μ_i (rate of failure, rate of repair and processing rate). If the description of the $M_i's$ could be limited to these three parameters, analyzing the new system is reduced in complexity to that of the existing long-line decomposition techniques. Therefore, unlike the methods discussed in the introduction to this chapter that approximate the $M_i's$ as some type of complicated Markov process, our approach limits the description of each M_i' to an estimate of a new p_i', r_i' and μ_i'.

5.3 Determining p_i', r_i' and μ_i'

For each set of parallel machines at Work Center i, an estimate of the three parameters, p_i', r_i' and μ_i' must be generated. Three equations are developed in this section

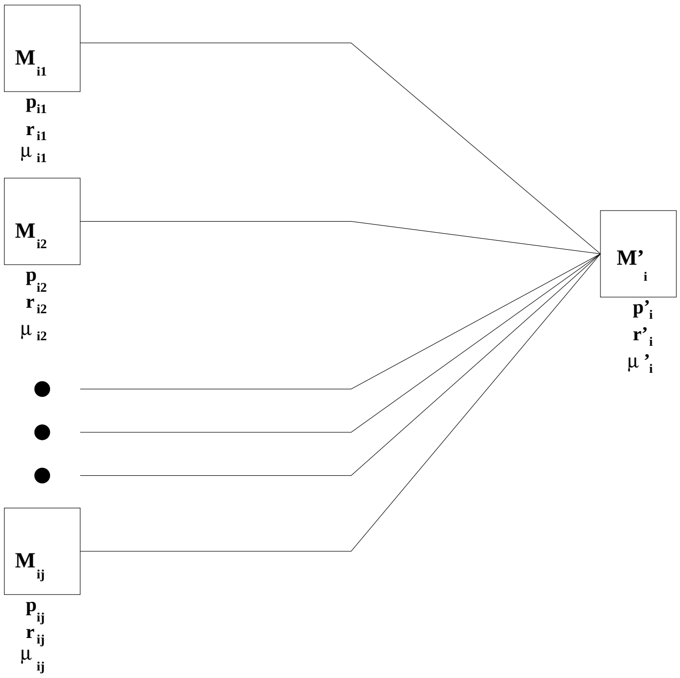

Figure 5-3: Single Machine Approximation of Parallel Machines

to estimate these three unknowns. The first equation matches the variability of the output of M_i' with that of the original set of parallel machines at Stage i. This equation provides the most important contribution of the three equations. The second equation matches expected throughput and the third equation matches the maximum production rate.

5.3.1 Equation 1 — Variance of Output

We begin with the observations that lead to the first of the three equations. Figure 5-4 illustrates the simplest non-trivial system that involves a multi-machine station. Stage 1 represents a set of J_1 parallel unreliable machines feeding a reliable machine at Stage 2. The stages are separated by a buffer with a capacity of 20. Each machine at Stage 1 has a mean time to repair and a mean time to fail of 100 time units (and therefore a 50% efficiency); and each machine in Stage 1 runs at a constant rate of $\mu_{1j} = 2/J_1$ units of material per time unit while it is operational. The Stage 2 machine runs at a constant rate of $\mu_2 = 1$. The isolated expected throughput of each stage is therefore 1 unit of material per time unit.

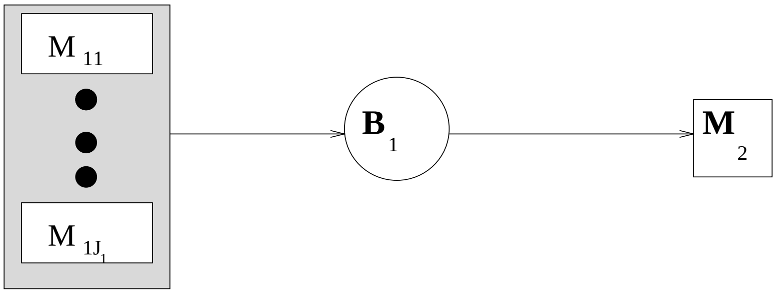

Figure 5-4: Simulated System

We simulated this system using the continuous material simulator (Appendix 2.2) with a random number generator from mrandom3.0 (Appendix 2.3.) for values of J_1 of 1, 2, 10 and 50. For all cases, $r_{1j}'s$ and $p_{1j}'s$ remain the same for all Stage 1 machines. The goal of the experiment was to study the effect of varying the number of Stage 1 machines while holding maximum throughput, expected throughput, and mean time to repair in the system constant. Before collecting data, we let the simulation run for 10,000 time units to reach steady state. We then collected statistics on average buffer level and throughput for the next 25,000 time units. We ran each simulation 30 times.

Table 5.1 indicates that the expected throughput of the simulated system when $J_1 = 1$ was only 0.7145 units of material per time unit. Figure 5-5 helps explain where the capacity was lost by illustrating the sample path of the buffer level between the

J_i	Throughput	95% CI	$\overline{n}$	95% CI
1	0.7145	(0.6403,0.7887)	13.1	(11.5,14.7)
2	0.8365	(0.7876,0.8854)	12.0	(10.6,13.4)
10	0.9349	(0.9167,0.9531)	11.5	(9.8,13.1)
50	0.9792	(0.9778,0.9805)	11.4	(11.1,11.8)

Table 5.1: Simulation Results

times of 10000 and 12000 of the simulation. Although the average buffer level is 13.1 units of material, notice how often the path remains on the boundaries ($n = 0$ and $n = 20$). This indicates the starvation and blockage in the system that results in lost capacity.

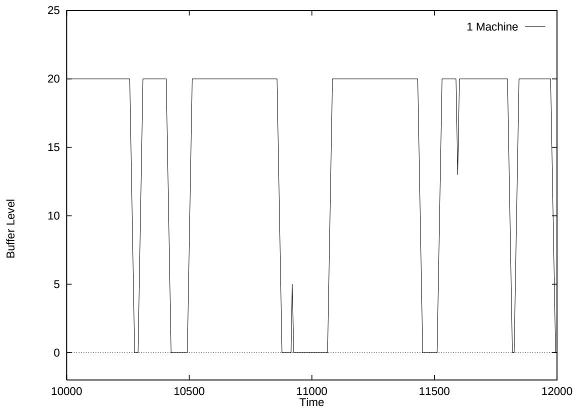

Figure 5-5: Sample Path of Buffer Level with 1 Stage 1 Machine

95% confidence intervals[1] for each statistic were calculated. As J_1 increased, the confidence interval decreased, indicating a reduction in the variability of the system. Figure 5-6 illustrates the sample paths of the amount of material in the buffer for the simulations when J_1 was 2, 10 and 50. With each increase in the number of machines, one can observe a reduction in the amount of variability of the buffer level. This might explain why the confidence intervals decreased as J_1 increased. The

[1]See Appendix 2.4 for description of their calculation.

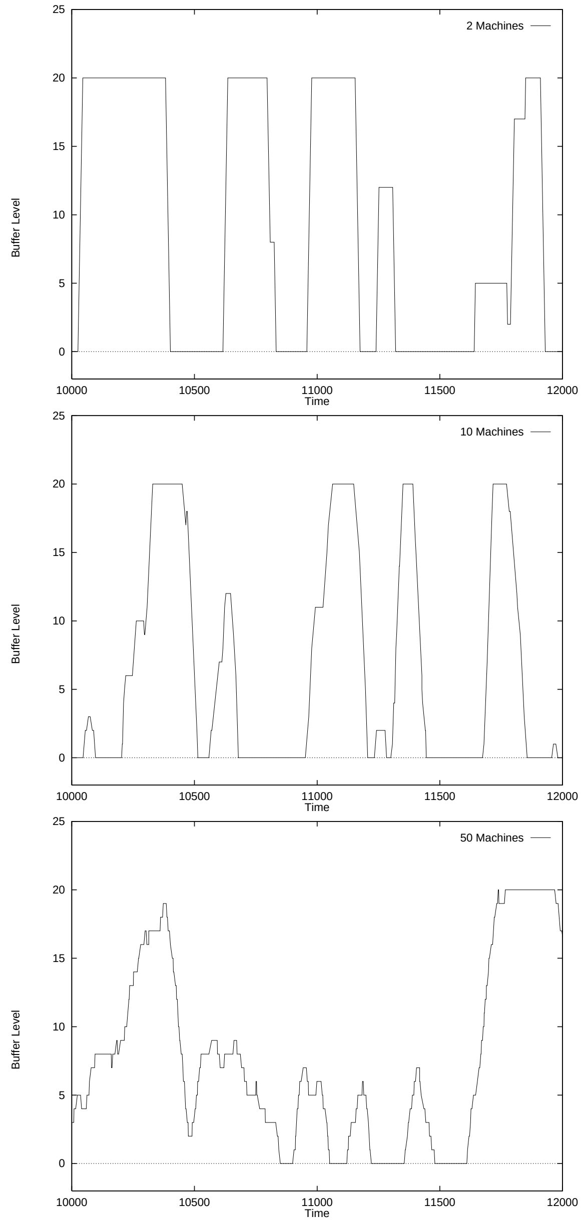

Figure 5-6: Sample Path of Buffer Level with 2, 10 and 50 Stage 1 Machines

increased variability in the buffer level corresponded to a reduction in the variability in the processing rate at Stage 1. This ultimately resulted in lower blockage and starvation and an increase in the expected throughput from 0.7145 to 0.9792.

It appeared that the tendency to fluctuate (the amount of variability) in the production rate directly affected the throughput of the system when isolated expected throughput and maximum throughput per stage were held constant. Table 5.1 also indicates that as J_1 increases, the average buffer level went down from 13.1 to 11.4. However, we have no explanation for this trend.

Having made these observations about the variability, we ran three additional simulations. The first new simulated system was identical to the single-machine-per-stage case of Figure 5-4. However, this time, instead of increasing the number of Stage 1 machines, we reduced the variability of the system by increasing the rates of repair (r_1) and failure (p_1) by 2, 10 and 50 times while holding expected throughput and maximum throughput rate of both work centers constant. The results displayed in Table 5.2 indicate a similar behavior to that observed by the first set of systems. As variability decreased, throughput increased. The sample buffer level paths illustrated in Figure 5.2 further confirmed these results.

J_i	Throughput	95% CI	$\overline{n}$	95% CI
1	0.7145	(0.6403,0.7887)	13.1	(11.5,14.7)
2	0.7344	(0.6790,0.7898	12.6	(11.5,13.7)
10	0.8584	(0.8353,0.8815)	11.5	(10.6,12.3)
50	0.9560	(0.9456,0.9663)	10.4	(9.6,11.2)

Table 5.2: Second Simulation Results

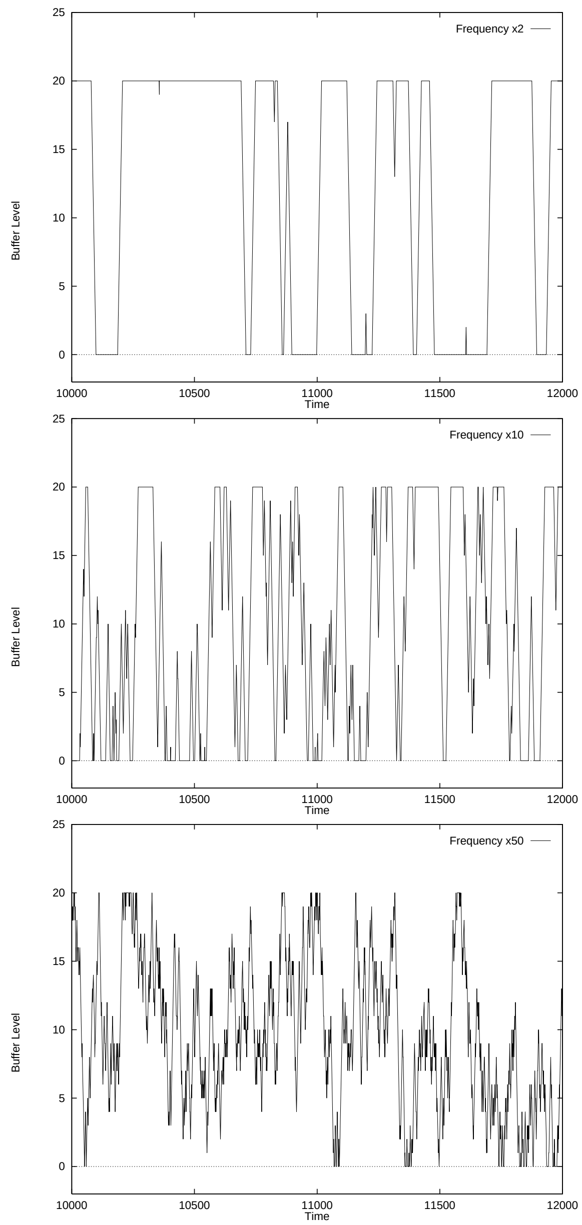

Figure 5-7: Frequency of Events Multiplied by 2, 10 and 50

Although the sample buffer level paths for the single machine with increased parameters values do not exactly match the behavior of the multi-machine simulations, they do resemble each other in one important way. The amount of time spent in the blocked and starved state is similar, which implies a similar throughput. This appears to be caused by the variability in the buffer level which is caused by the variability in the production rate. We can approximate the variability of output of the set of parallel machines over t time units by first predicting the variance of output of each machine M_{ij} (derived in Appendix A):

$$V_{ij}(t) = \mu_{ij}^2 \left((e^{-(r_{ij}+p_{ij})t} - 1)\frac{2r_{ij}p_{ij}}{(r_{ij}+p_{ij})^4} + \frac{2r_{ij}p_{ij}}{(r_{ij}+p_{ij})^3}t \right).$$

Assumption 1 *When the set of machines operates in isolation, a failure of one machine has no effect on other machines in the same set because the supply and demand at the individual machine remain constantly in a saturated state. Therefore, we assume that each of the machines in a set of parallel machines operates independently.*

This is only an assumption because a failure of a machine in a work center that is either partially blocked or partially starved affects the processing rate, and in turn the failure rate of other machines in that work center.

The variance of the sum of a set of independent random variables is the sum of the variances (Drake, 1988). Therefore, the variance of all parallel machines combined is

$$\begin{aligned} V_i'(t) &= \text{Variance of combined set of parallel machines at Stage } i \\ &= \sum_{j=1}^{J_i} \mu_{ij}^2 \left((e^{-(r_{ij}+p_{ij})t} - 1)\frac{2r_{ij}p_{ij}}{(r_{ij}+p_{ij})^4} + \frac{2r_{ij}p_{ij}}{(r_{ij}+p_{ij})^3}t \right). \end{aligned} \tag{5.6}$$

Since our simulations indicated that there exists a correlation between the variability in the production rate of Stage 1 and the expected throughput of the whole system, we propose that the first equation match the variance of the production output of a single machine to the variance of the set of parallel machines over some appropriate length of time that we call t^*. We must select r'_i, p'_i and μ'_i such that M'_i has the correct variance. This is represented by the following equation

$$\mu_i'^2 \left((e^{-(r'_i+p'_i)t^*} - 1)\frac{2r'_ip'_i}{(r'_i+p'_i)^4} + \frac{2r'_ip'_i}{(r'_i+p'_i)^3}t^* \right) =$$

$$\sum_{j=1}^{J_i} \mu_{ij}^2 \left((e^{-(r_{ij}+p_{ij})t^*} - 1)\frac{2r_{ij}p_{ij}}{(r_{ij}+p_{ij})^4} + \frac{2r_{ij}p_{ij}}{(r_{ij}+p_{ij})^3}t^* \right). \quad (5.7)$$

Unfortunately, at this point, we do not have a definitive way of choosing t^*. However, after extensive experimentation, it appears that t^* loosely correlates to the size of the smaller of the two buffers adjacent to the machine. This approximation breaks down when the capacity of adjacent buffers is too small to hold as much material as is produced during an average failure.

We suggest two reasons why this might be so. First, the independence-of-machines assumption used in determining the variance of the set of parallel machines may be incorrect when buffers are small. For example, suppose that the buffer is very small, and that the upstream supply is far less than the capacity of the set of parallel machines but greater than the capacity of a single machine. Next, suppose that all but one of the parallel machines fail. At this point, the remaining operating machine must work at full capacity and thus, has a greater likelihood to fail. When buffers are small, the effects of up- or downstream machines are more significant, which causes a stronger dependence amongst all the machines in the set of parallel machines. Also, when adjacent buffers to M_i are very small, neighboring buffers may serve as the primary buffering mechanism for M_i. This adversely affects the estimate of $V'(i)$.

Second, experimentation has shown that as the buffer gets small, the sensitivity of t^* to buffer size is very high. Therefore, when buffers get small, a more precise method than we have been able to develop is required to choose t^*. For the remainder of this chapter, we focus on situations in which each buffer is capable of holding as much material as is produced during an average failure of the machines adjacent to it.

Given that buffers are at least of certain size, we can assume that t^* is relatively large. When t^* is large, the first term of (5.6) becomes insignificant compared with the second. After removing this term and dividing both sides by t^*, (5.7) becomes

$$\mu_i'^2 \left(\frac{2r_i'p_i'}{(r_i' + p_i')^3} \right) = \sum_{j=1}^{J_i} \mu_{ij}^2 \left(\frac{2r_{ij}p_{ij}}{(r_{ij} + p_{ij})^3} \right). \tag{5.8}$$

Thus, we have the first of our required three equations.

5.3.2 Choosing the Isolated Production Rate and μ'

The second and third equations are derived in a much more straightforward manner.

Equation 2 — Isolated Production Rate It seems reasonable that the expected production rate of the equivalent machine (M_i') equal that of the set of parallel machines. Therefore, the second proposed equation matches the expected long term isolated throughput rate of M_i' with the long term isolated throughput rate of the combined set of parallel machines at each stage i. We define

$$\begin{aligned} E_{ij}(t) &= \begin{cases} \text{The expected amount of material processed by } M_{ij} \text{ in } t \\ \text{time units, when it is not limited by other machines or} \\ \text{buffers.} \end{cases} \\ &= \mu_{ij} \left(\frac{r_{ij}}{r_{ij} + p_{ij}} \right) t. \end{aligned} \tag{5.9}$$

The expected value of the sum of a set of random variable is the sum of the expected values (Drake, 1988). Therefore, we have

$$
\begin{aligned}
E_i(t) &= \begin{cases} \text{The expected amount of material processed by a set of} \\ J_i \text{ parallel machines in } t \text{ time units in isolation} \end{cases} \\
&= \sum_{j=1}^{J_i} E_{ij}(t) \\
&= \sum_{j=1}^{J_i} \mu_i \left(\frac{r_{ij}}{r_{ij} + p_{ij}} \right) t. \qquad (5.10)
\end{aligned}
$$

For the new equivalent machine M_i' we define

$$E_i'(t) = \mu_i \left(\frac{r_i'}{r_i' + p_i'} \right) t. \qquad (5.11)$$

By equating the expected production of M_i' with that of the original set of parallel machine (5.11) and (5.10) and dividing by t, we obtain

$$\mu_i \left(\frac{r_i'}{r_i' + p_i'} \right) = \sum_{j=1}^{J_i} \mu_i \left(\frac{r_{ij}}{r_{ij} + p_{ij}} \right). \qquad (5.12)$$

Equation 3 — Maximum Processing Rate We define the *maximum isolated processing rate* as the maximum possible rate at which a stage can produce material. When all of the parallel machines are working, the maximum isolated processing rate of the set is simply the sum of all the individual isolated processing rates. Since the single machine equivalent is assumed only to be in one of two states (up or down), its maximum throughput rate is μ_i'. The third equation matches the maximum isolated processing rates of M_i' with that of the set of parallel machines, or

$$\mu_i' = \sum_{j=1}^{J_i} \mu_{ij}. \qquad (5.13)$$

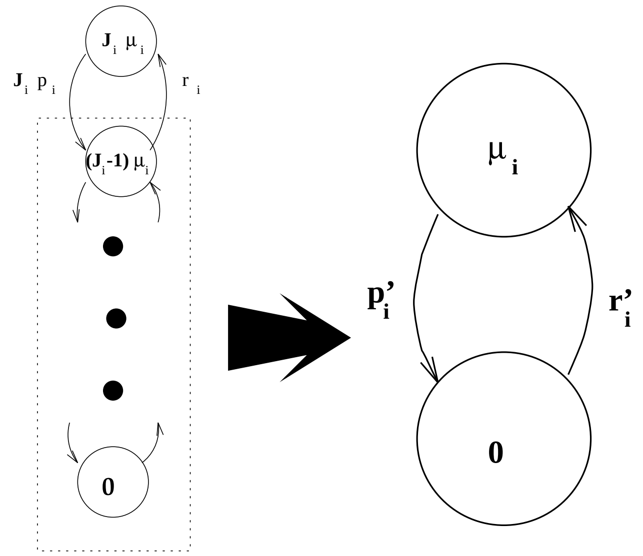

Figure 5-8: Dallery Parallel Machine Approximation

The r_i', p_i', and μ_i' that satisfy (5.8), (5.12) and (5.13) worked well in the experiments presented in Section 5.4. However, there are many other possibilities and more work is required to see if a different set of relationships might work better.

5.3.3 Heuristic

For the rest of this chapter, we will only look at the special case when all the machines at Work Center i are identical. This case is important for two reasons. First, it is a common situation in reality. Second, if the machines are not identical, a control issue exists. When, a work center is partially starved or blocked, work can usually be divided among the different machines at the work center in more than one way. Different policies for allocating this production may result in different estimates of production capacity. We will not deal with this issue in this thesis.

In casual discussions, Yves Dallery (1995) presented a different view of the problem when all the machines are identical. He approximated all the states that occur when the set of machines are operating into one state as illustrated in Figure 5-8. He, like us, assumed the rate of processing of the single machine equivalent is the sum of all the parallel machines ($J_i\mu$). He also showed that the number of transitions out of the Machine *up* state is equivalent to the number of transitions out of the all-machines-*up* state in the original system. He then showed that because of the balance of flow in the Markov chain, the rate of repair of the equivalent machine must be J_ir in order to maintain the correct isolated production rate. In summary, Dallery's approximation turns out to be

$$\begin{aligned} p_i' &= J_i p_{i1} & i &= 1, \cdots, k \\ r_i' &= J_i r_{i1} & i &= 1, \cdots, k \\ \mu_i' &= J_i \mu_{i1} & i &= 1, \cdots, k. \end{aligned} \tag{5.14}$$

Interestingly, (5.14) is the solution to equations (5.8), (5.12) and (5.13) when all the machines in a given work center are the same. That is, it is the same as our methods when t^* is very large.

This led us to the following proposed heuristic when machines within the set of parallel machines are fairly similar:

$$\begin{aligned} p_i' &= \sum_j^{J_i} p_{ij} & i &= 1, \cdots 1, k \\ r_i' &= \sum_j^{J_i} r_{ij} & i &= 1, \cdots 1, k \\ \mu_i' &= \sum_j^{J_i} \mu_{ij} & i &= 1, \cdots 1, k. \end{aligned} \tag{5.15}$$

Equation (5.15) will not satisfy (5.8), (5.12) and (5.13) in general. However, we believe it is a good approximation when machines are similar. We will show in Section 5.4 that this heuristic works reasonable well in certain environments. However, we will also show that when buffers are small or the machines are very different, this heuristic is inaccurate.

5.4 Numerical Experiments

For series-parallel flow lines, the number of alternative line designs is inordinate. We have only chosen a select few specific systems to experiment with that are discussed in Section 5.4.1. These examples highlight some of the fundamental observations made in all of our experiments. In addition in Section 5.4.2, we report on the results of some experiments using randomly generated series-parallel flow lines. In every case in this section, all the machines within a given work center have identical $r's$, $p's$, and $\mu's$.

5.4.1 Basic Experiments

In this section, we would like to highlight the following three observations we made in a large number of our experiments:

1. Matching variance is important when approximating multiple machines by a single equivalent machine.

2. Small buffers reduce the accuracy of our method.

3. The method works well in some cases but not all cases.

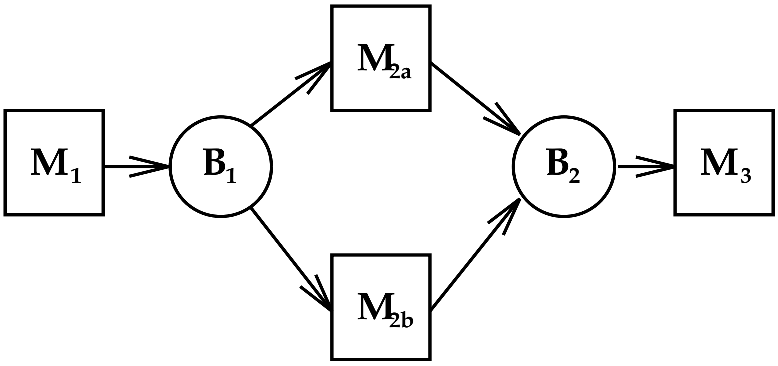

Figure 5-9: Simplest Non-trivial Series-Parallel System

We examine three fundamental series-parallel systems. Each system has three stages. Each has one machine feeding two machines feeding one machine (Figure 5-9). The average time between failures on the first and third machines is 100 time units, the average repair times on these machines are 10 time units and each has a deterministic processing rate $\mu = 1$. Each has a different second work center design to illustrate a different flow line attribute. We experiment with each system twice. First, we set all the buffer capacities to 10 and then set them to 1. A full description of the parameters for each all the cases referred to in this section can be found in Appendix 3.3. Each case was simulated using the continuous material simulation (Appendix 2.2) with a transient period of 40000 time units, a data collection period of 40000 time units and each system simulated 100 times. We have omitted the confidence intervals from our tables of results so that we can focus on the more qualitative aspects.

Redundant Machinery Our first primary examples (Case 45 and Case 48) highlight a system in which both Stage 2 machines are identical in all respects to the first and third machines. In these cases, we consider the second Stage 2 machine redundant. For each case, we approximate Stage 2 with one approximately equivalent machine. We first ignore variance in the calculation of parameters of the equivalent machine by only using Equations (5.12) and (5.13), and setting $p_2' = p_{2a}$. In other words we match the maximum production rate and the expected isolated production rates and failure rate with one of the machines. We do not treat variance explicitly. We refer to series-parallel lines approximated ion this manner as *estimates without variance* (*EWOV*). We also test the method described in Section 5.3.3 and refer to this as the *estimate with variance* (*EWV*). In Table 5.3, we report the estimates of throughput rate (P), $\overline{n}_1$ and $\overline{n}_2$ obtained from the continuous material simulation of the original system, the EWOV flow line and the EWV flow line.

	Buffers of 10			Buffers of 1		
	Original	EWOV	EWV	Original	EWOV	EWV
	Case 45	Case 46	Case 47	Case 48	Case 49	Case 50
P	0.870	0.853	0.861	0.838	0.809	0.811
$\overline{n}_1$	3.724	3.572	3.437	0.469	0.425	0.381
$\overline{n}_2$	6.246	6.451	6.543	0.528	0.573	0.618

Table 5.3: Redundant Machinery Experiment Results

For all these cases, the confidence intervals (not shown) were on the order of 10^{-2} for the P estimates and about 5% of the buffer sizes for the average buffer level estimates. The results indicate an insignificant advantage of including variance when choosing parameters for the equivalent Stage 2 machine in the estimating of P and a slightly more significant disadvantage in estimating the average buffer levels.

Slow Machinery Our second primary examples (Case 51 and Case 54) highlight a system in which both Stage 2 machines are identical in all respects to the first and third machines except that each machine operates at half the speed of the first and third machines. In these cases, the second Stage 2 machines are so slow that, although there two of them, the line is still balanced. For each case, we approximate Stage 2 with one approximately equivalent machine, first with the EWOV procedure and then with EWV procedure. In Table 5.4, we report the estimates of throughput rate (P), $\overline{n}_1$ and $\overline{n}_2$ obtained from the continuous material simulation of the original system, the EWOV flow line and the EWV flow line.

	Buffers of 10			Buffers of 1		
	Original	EWOV	EWV	Original	EWOV	EWV
	Case 51	Case 52	Case 53	Case 54	Case 55	Case 56
P	0.831	0.824	0.830	0.781	0.778	0.778
$\overline{n}_1$	6.619	6.124	6.603	0.726	0.657	0.720
$\overline{n}_2$	3.407	3.871	3.397	0.275	0.343	0.280

Table 5.4: Slow Machinery Experiment Results

For all these cases, the confidence intervals were about the same as in the previous examples. The results indicate an slight but still insignificant advantage of including variance when choosing parameters for the equivalent Stage 2 machine in the estimating of P and a slightly more significant advantage in estimating the average buffer levels.

Unreliable Machinery Our final primary examples (Case 57 and Case 60) highlight a system in which both Stage 2 machines are identical in all respects to the first and third machine except that we have increased the failure rate so that each Stage 2 machine is half as reliable as the first and third machines. In these cases, the unreliable Stage 2 machines results in a balanced line. For each case, we again approximate the Stage 2 with one approximately equivalent machine, first with the EWOV procedure and then with EWV procedure. In Table 5.5, we report the estimates of throughput rate (P), $\overline{n}_1$ and $\overline{n}_2$ obtained from the continuous material simulation of the original system, the EWOV flow line and the EWV flow line.

	Buffers of 10			Buffers of 1		
	Original	EWOV	EWV	Original	EWOV	EWV
	Case 57	Case 58	Case 59	Case 60	Case 61	Case 62
P	0.756	0.682	0.728	0.676	0.575	0.590
$\overline{n}_1$	5.803	5.179	5.393	0.554	0.469	0.452
$\overline{n}_2$	4.215	4.826	4.598	0.447	0.532	0.549

Table 5.5: Unreliable Machinery Experiment Results

For all these cases, the confidence intervals were about the same as in the previous examples. This time, the results indicate a significant advantage of including variance when choosing parameters for the equivalent Stage 2 machine in the estimating of P and in estimating the average buffer levels. However, both methods did poorly when the buffer capacities were set to 1. We believe a better method is needed to estimate the variance of machine production when buffers are small.

5.4.2 Random Examples

The examples in the previous section are only a very small fraction of the total possible series-parallel systems. To try to get a better understanding of how the method suggested in Section 5.3.3 performs, we generated random cases on which we could test the procedure. We generated the random cases using the method described in Section 3.3.2 with two modifications. First, the number of machines at each stage was randomly generated. We chose either one, two, or three machines at each stage, each with a probability of one-third. Second, we made sure that the capacity of each buffer was at least big as the amount of material produced by the machines adjacent to the buffer during a period of time equal to the average failure duration of the

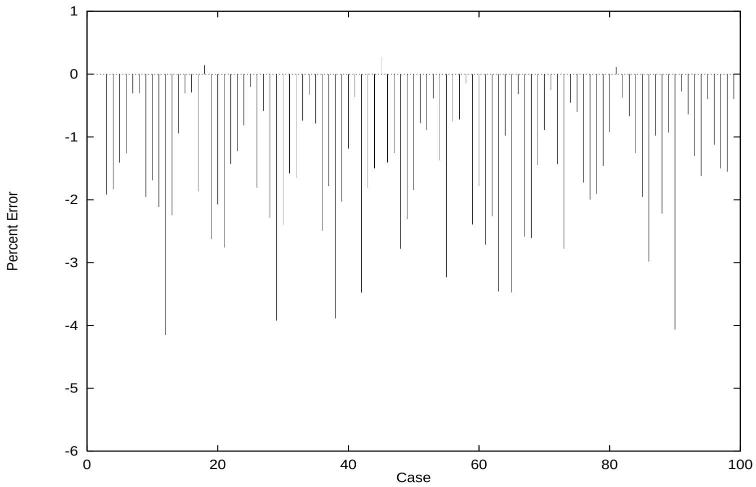

Figure 5-10: P Error - Approximation not Accounting for Variance

machines on the opposite side of the buffer.

We simulated each line three times again using the continuous material simulation. First, we simulated the original line to obtain an estimate for all the performance characteristics. Second, we simulated a single-machine-per-stage line determined using the EWOV procedure which did not explicitly account for the variance of the production rate. Finally, we simulated a another single-machine-per-stage line that was obtained using the EWV procedure that does explicitly account for the variability of the production rate. We generated 100 random cases each for lines of lengths 3, 5, 10, 15, 20 and 25. However, since the results were very similar, we only report on the results from the five-stage lines.

In Figure 5-10, we illustrate the errors associated with the P obtained from the single machine-per-stage line approximation of the series-parallel line that did not account for variance. The average error was about 1.5%. However, when we did account for variance for the same cases, the average error reduced to less than 0.3% as illustrated in in Figure 5-11. Employing variance seemed to only make a small

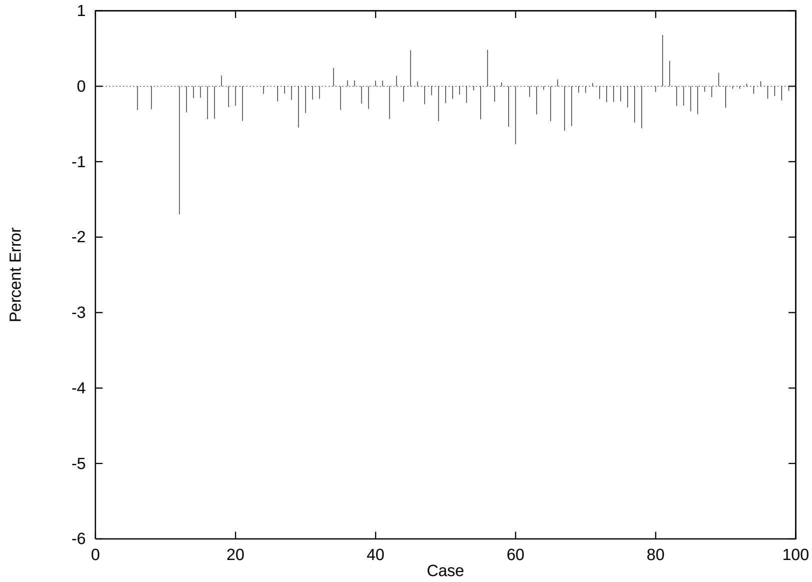

Figure 5-11: P Error - Approximation Accounting for Variance

improvement in the estimation of average buffer level. Therefore, we only report the error associated with the estimate of the average buffer level for Buffer 3 in Figure 5-12. The average error was about 1% in magnitude. It is important to remember that we calculate the buffer level percentage error as a fraction of total buffer size.

5.5 Conclusion

The existing analytic performance estimation techniques for series-parallel systems are only tractable for lines with a maximum of two work centers. In this chapter, we present a method for reducing each set of parallel machine at a given work center to one approximately equivalent machine. We achieve this by matching the transient moments of the equivalent machine with those of the original set of parallel machines. Once each work center is represented by a single machine, we can then employ the

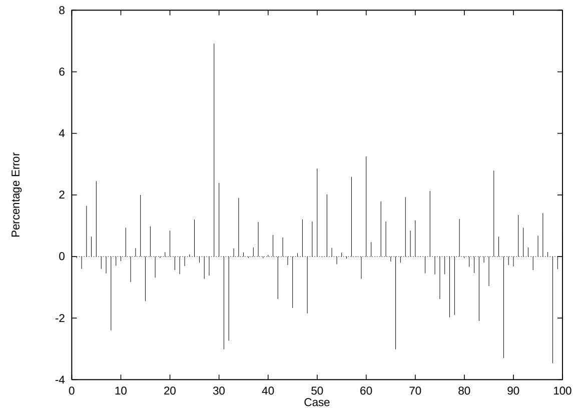

Figure 5-12: Buffer 3 - Average Buffer Level Estimate Error

decomposition technique developed in Chapters 2 and 3 for estimating the performance of the single machine per stage flow lines. The method is very quick and and very accurate for systems with reasonably sized buffers (i.e., large enough to hold the amount of material produced during a failure of average duration by either machine adjacent to the buffer). For small buffers, the method is less accurate and more work is required to improve the results in certain cases.

Chapter 6

Conclusion

In this thesis, we have made four substantial technological advances. First in Chapter 2, we formulated the first complete set of decomposition equations for non-homogeneous flow lines with unreliable machines, deterministic processing times and operation-dependent failures. Second, we developed the ADDX algorithm in Chapter 3, the fastest, most reliable convergent method known to date for solving these decomposition equations. Third, in Chapter 4 we introduced the first analytical model for addressing the issue of flow lines with failing buffers. Finally, we established an approximation technique for analyzing series parallel flow lines in Chapter 5. The results of these advances are presented in Sections 6.1 through 6.4. Summary remarks are provided in Section 6.5.

6.1 Continuous Material Flow Line Decomposition

Continuous material models are the only way known to date for analytically estimating performance characteristics of flow lines in which 1) each machine has a constant production time and where 2) the rate of production may differ from machine to machine. Decomposition is also currently the most accurate and practical technique for estimating the throughput of these systems. In Chapter 2, we present a complete derivation of decomposition equations for continuous material flow lines with operation dependent failures. The equations were developed to exploit the solution technique known as the DDX algorithm.

6.2 Accelerated DDX Algorithm

Most of the decomposition literature does not discuss the difficulty in obtaining an algorithm that consistently converges. In Chapter 3, we slightly modified the decomposition equations developed in Chapter 2 and use the original DDX algorithm to solve these equations. We demonstrated through numerical experimentation that the method had a high level of accuracy compared with simulation, but still had significant convergence difficulties as the flow lines became longer. To overcome this convergence problem, we developed the Accelerated DDX Algorithm (ADDX). For a large number of random flow lines with up to 100 stages, the method converged over 99.9% of the time with a high level of accuracy and speed.

6.3 Failing Buffer Model

Buffers in automated manufacturing facilities are often conveyor systems and accumulators which is machinery that can fail. The current flow line literature always treats the buffer as a completely reliable entity. In Chapter 4, we presented an analytical model for estimating the throughput and average buffer level of a simple two-stage synchronous flow line with an unreliable buffer. We also develop two heuristics to approximate this result.

6.4 Parallel Machine Approximation

It is often necessary to have redundant or extra processing equipment at a particularly unreliable or slow part of the flow line. To date, the existing analytic performance estimation techniques for series-parallel systems are only tractable for lines with a maximum of two work centers. In Chapter 4, we presented a method for reducing each set of parallel machines at a given work center to one approximately equivalent machine. We achieve this by matching the moments of the amount of material produced by the equivalent machine with that of the original set of parallel machines. Once each work center is represented by a single machine, we can then employ the decomposition technique developed in Chapters 2 and 3 for estimating the performance of the single machine per stage flow lines. The method is very quick and and very accurate for systems with reasonably sized buffer (*i.e.*, large enough to hold the amount of material produced during a failure of average duration by either machine adjacent to the buffer). For small buffers, the method is less accurate and more work is required to improve the results.

6.5 Final Remarks

As described in Section 1.1, this research was motivated by the actual needs of the process engineering department of the Vancouver Division of Hewlett Packard. By spending considerable time at the Vancouver facility, constantly keeping HP abreast of the developments in the research and focusing on their requirements, we were able to develop the above tools which of immense financial benefit to HP. HP plans in the future to regularly employ our decomposition technique as the first part of the system design process. We are currently in the process of verifying the model. Johnson and Johnson's Critikon division has also implemented the algorithm and added financial information for the redesign of their production facilities. We are currently talking to General Motors and Motorola about the possibility of applications in their facilities.

As we begin to employ the model, it will be required to account for more phenomena. Based on our observations of industry, the following are the most important next advances required:

1. Efficiently solve original equations derived in Chapter 2. We feel the orginal equations may get rid of the positive efficiency estimate bias observed in the analysis of longer lines.

2. Incorporate yield into the model. While HP did not have a yield problem, badly produced parts can often significantly impact line efficiency. For example, if yield upstream is very low, it is important to reduce the throughput to the downstream portions of the line. It may show that less capacity is required at the end of the line if throughput from upstream is significantly curtailed.

3. Estimate the variance of production of the flow line. Not only is average

throughput important, but the variability of the production rate is very important when setting safety stock and capacity strategies to meet demand with a certin level of confidence.

4. Extend the continuous model to include assembly/disassembly. Once the decomposition equations have been solve, it is relatively straightforward to include assembly and disassembly components to the model. This is particularly valuable to mass manufacturers like HP.

5. Extend the multiple machine method to handle smaller buffers. Our assumption of buffers of a certain minimum size is simply not valid in a large number of important cases.

Appendix A

Derivation of Variance Equation

In this appendix, we derive the variance of the amount of continuous material produced during the time interval [0,t] by an unreliable machine operating with exponentially distributed failures and repairs with rates p and r respectively, and a constant processing rate of μ. We begin by determining the expected amount of material produced during [0,t] when $\mu = 1$. We use this result in obtaining the variance when $\mu = 1$. We follow a similar derivation to that presented for discrete material in Gershwin (1993). Finally, we use this result to obtain the variance of the amount of material produced during [0,t] for any μ.

We define:

$$\pi(x, \alpha, t)\delta x + o(\delta x) \quad = \quad \begin{cases} \text{probability of being in machine state } \alpha \text{ and having pro-} \\ \text{cessed between } x \text{ and } x + \delta x \text{ material by time } t. \end{cases}$$

The system operates according to the following dynamics:

$$\pi(n,0,t+\delta t) = \pi(n,0,t)(1-r\delta t)+\pi(n,1,t)p\delta t+o(\delta t) \tag{1.1}$$

$$\pi(n,1,t+\delta t) = \pi(n-\delta t,0,t)r\delta t+\pi(n-\delta t,1,t)(1-p\delta t)+o(\delta t) \tag{1.2}$$

with initial conditions:

$$\pi(0,0,0) = \frac{p}{r+p} \tag{1.3}$$

$$\pi(0,1,0) = \frac{r}{r+p} \tag{1.4}$$

We add the $o(\delta x)$ and $o(\delta t)$ terms because the above expressions are approximations for Taylor expansions. However, these terms have no impact and they are dropped for the remainder of the derivation.

Lemma 1 *Assuming a constant processing rate of $\mu = 1$, if we let:*

$$E_\alpha(t) = \int_0^t n\pi(n,\alpha,t)dn. \tag{1.5}$$

Then,

$$\begin{aligned} E_0(t) &= \frac{-rp}{(r+p)^3}+\frac{rp}{(r+p)^2}t+\frac{rp}{(r+p)^3}e^{-(r+p)t} \\ E_1(t) &= \frac{rp}{(r+p)^3}+\frac{r^2}{(r+p)^2}t-\frac{rp}{(r+p)^3}e^{-(r+p)t} \end{aligned} \tag{1.6}$$

Proof:

$$E_0(t) = \int_0^t n\pi(n,0,t)dn \tag{1.7}$$
$$E_1(t) = \int_0^t n\pi(n,1,t)dn \tag{1.8}$$

We substitute $t + \delta t$ for t into (1.7) manipulate the result to obtain

$$\begin{aligned}
E_0(t+\delta t) &= \int_0^{t+\delta t} n\pi(n,0,t+\delta t)dn \\
&= \int_0^{t+\delta t} n\left[\pi(n,0,t)(1-r\delta t)+\pi(n,1,t)p\delta t\right]dn \\
&= \int_0^t n\pi(n,0,t)(1-r\delta t)dn + \int_0^t n\pi(n,1,t)p\delta tdn + \\
&\quad \int_t^{t+\delta t} n\pi(n,0,t)(1-r\delta t)dn + \int_t^{t+\delta t} n\pi(n,1,t)p\delta tdn
\end{aligned} \tag{1.9}$$

Since the machine cannot produce more than n units of material in t time units (since $\mu = 1$), we have the following:

$$\int_t^{t+\delta t} n\pi(n,0,t)(1-r\delta t)dn + \int_t^{t+\delta t} n\pi(n,1,t)p\delta tdn = 0. \tag{1.10}$$

And therefore, we have

$$\begin{aligned}
E_0(t+\delta t) &= \int_0^t n\pi(n,0,t)(1-r\delta t)dn + \int_0^t n\pi(n,1,t)p\delta tdn \\
&= (1-r\delta t)E_0(t) + p\delta tE_1(t)
\end{aligned} \tag{1.11}$$

which results in the following differential equation

$$\frac{dE_0(t)}{dt} = -rE_0(t) + pE_1(t). \tag{1.12}$$

In as similar fashion, we can obtain the following differential equation, This results in a second differential equation:

$$\frac{dE_1(t)}{dt} = rE_0(t) - pE_1(t) + \frac{r}{r+p}. \tag{1.13}$$

We assume the initial conditions:

$$\begin{aligned} E_0(0) &= 0 \\ E_1(0) &= 0. \end{aligned} \tag{1.14}$$

Equations (1.12) and (1.13) are two differential equations in $E_0(t)$ and $E_1(t)$ with initial conditions (1.14). They have the following solution:

$$\begin{aligned} E_0(t) &= \frac{-rp}{(r+p)^3} + \frac{rp}{(r+p)^2}t + \frac{rp}{(r+p)^3}e^{-(r+p)t} \\ E_1(t) &= \frac{rp}{(r+p)^3} + \frac{r^2}{(r+p)^2}t - \frac{rp}{(r+p)^3}e^{-(r+p)t}. \end{aligned} \tag{1.15}$$

Theorem 1 *The variance of the amount of material produced during [0,t] is*

$$V(t) = 2(e^{-(r+p)t} - 1)\frac{\mu^2 rp}{(r+p)^4} + \frac{\mu^2 rp}{(r+p)^3}t. \tag{1.16}$$

Proof: We begin by determining the variance of $n(t)$ when $\mu = 1$. Let

$$\sigma_n^2 = E[n(t)^2] - E[n(t)]^2 \tag{1.17}$$

where

$$E[n(t)] \quad = \quad S(t) = \int_0^t n^2 \left(\pi(n, 0, t) + \pi(n, 0, t)\right) dn. \tag{1.18}$$

We substitute $t + \delta t$ into (1.18) and manipulate the result to obtain

$$\begin{aligned}
S(t + \delta t) \quad &= \quad \int_0^{t+\delta t} n^2 \left(\pi(n, 0, t + \delta t) + \pi(n, 0, t + \delta t)\right) dn \\
&= \quad \int_0^{t+\delta t} n^2 \left(\pi(n, 0, t)(t - r\delta t) + \pi(n, 1, t)p\delta t + \pi(n - \delta t, 0, t)r\delta t + \right. \\
&\quad\quad \left. \pi(n - \delta t, 1, t)(1 - p\delta t)\right) dn \\
&= \quad \int_0^{t} n^2 \left(\pi(n, 0, t)(t - r\delta t) + \pi(n, 1, t)p\delta t\right) dn + \\
&\quad\quad \int_0^{t} n^2 \left(\pi(n, 0, t)r\delta t + \pi(n - \delta t, 1, t)(1 - p\delta t)\right) dn + \\
&\quad\quad 2\int_0^t n\pi(n, 0, t)r\delta t dn\delta t + 2\int_0^t \pi(n, 1, t)(1 - p\delta t)dn\delta t \\
&= \quad S(t) + 2E_0(t)r\delta t^2 + 2E_1(t)\delta t - 2E_1(t)\delta tp\delta t.
\end{aligned} \tag{1.19}$$

After removing the δt^2 terms, (1.19) becomes

$$S(t + \delta t) = S(t) + 2E_1(t)\delta t. \tag{1.20}$$

This results in the following differential equation:

$$\begin{aligned} \frac{dS(t)}{dt} &= 2E_1(t) \\ &= \frac{2rp}{(r+p)^3} + \frac{2r^2}{(r+p)^2}t - \frac{2rp}{(r+p)^3}e^{-(r+p)t} \end{aligned} \tag{1.21}$$

with the initial condition

$$S(0) = 0. \tag{1.22}$$

By simultaneously solving (1.21) and (1.22) the following results are obtained

$$S(t) = (e^{-(r+p)t} - 1)\frac{2rp}{(r+p)^4} + \frac{2rp}{(r+p)^3}t + \frac{r^2}{(r+p)^2}t^2. \tag{1.23}$$

Then we substitute (1.23) into (1.17) to get

$$\begin{aligned} \sigma_n^2 &= S(t) - (E_0(t) + E_1(t))^2 \\ &= S(t) - \frac{r^2}{(r+p)^2}t^2 \\ &= (e^{-(r+p)t} - 1)\frac{2rp}{(r+p)^4} + \frac{2rp}{(r+p)^3}t. \end{aligned}$$

This provides σ_n^2. However, we are interested $V(t)$ when $\mu \neq 1$. Since the machine processes at a constant rate μ when it is up and it is up for $n(t)$ time units when it is operated for a total of t time units, we can obtain total material produced in t time units by multiplying these two quantities or

$$\text{The amount of material processed in } t \text{ time units} \quad = \quad \mu n(t). \tag{1.24}$$

Since σ_n^2 is the variance of the random variable $n(t)$, μ is constant and $\mu n(t)$ is a random variable equivalent to the total amount of material processed in t time units, the variance of the total amount of material processed in t time units is:

$$
\begin{aligned}
V(t) &= \mu^2 \sigma_n^2 \\
&= 2(e^{-(r+p)t} - 1)\frac{\mu^2 rp}{(r+p)^4} + \frac{\mu^2 rp}{(r+p)^3} t
\end{aligned}
$$

Appendix B

Simulators

Simulation experiments are an important component of this research. In this appendix, we describe the simulators, random number generators and method for calculating confidence intervals.

2.1 Discrete Material Simulator

For all of our discrete material simulations, we used a program called *genlinesim* developed by Asbjoern Bonvik[1]. To run the program, one must specify r_i, p_i, and μ_i for each Machine i and all buffer capacities. The parameters r_i and p_i are interpreted in the same way as in the description of the continuous material model (Section 2.2). However, μ_i represents the reciprocal of a constant production time. This is because this simulation models the processing of individual parts rather than continuous material.

[1] PhD student at the Operations Research Center at the Massachusetts Institute of Technology.

The simulation gathers data for a specified period of time after a transient period during which no data is collected. It reports total production, average production, and average buffer levels. When more than one simulation of the same system is run, the program calculates 95% confidence intervals and maximum and minimum values for each of the performance metrics estimated for the individual trials. For all of our experiments, we ran either 30 or 100 trials with transient periods of 40,000 time units followed by 40,000 time units during which we collected data.

The program employs standard C random number generators rand() and srand().

2.2 Continuous Material Simulator

For all of our continuous material simulations, we used a program called $ContSim$ we developed with the help of Alan Kaufman[2]. To run the program, one specifies r_i, p_i, and μ_i for each Machine i and all buffer capacities. The simulation was designed to exactly represent our continuous material model (Section 2.2).

The simulation gathers data for a specified period of time after a transient period during which no data is collected. It reports total production, average production, and average buffer levels. When more than one simulation of the same system is run, the program calculates 95% confidence intervals based on the calculation presented below in Section 2.4. For all of our experiments, we ran either 30 or 100 trials with transient periods of 40000 time units followed by 40000 time units during which we collected data.

The program has some other important features. The program employs an advanced set of random generator routines called mrandom3.0 described below in Sec-

[2]PhD student at the Operations Research Center at the Massachusetts Institute of Technology.

tion 2.3. In addition to single machine per stage flow lines, the simulation could also simulate series-parallel system and failing buffers. This was required for Chapters 4 and 5. The main limitation of the program is that it is currently designed only to model lines up to lengths of 25 work centers.

2.3 Random Number Generators

For the continuous material simulation, we used a special set of C routines referred to as mrandom3.0. The current version of mrandom3.0 used in this thesis (version 3.0) is a major rewrite of mrandom version 2.1, released by Clark Thomberson in July 1992. It is available via anonymous **ftp** from **theory.lcs.mit.edu**. The package provides extensive facilities for generating random numbers.

2.4 Confidence Intervals

To calculate a confidence interval for a given variable, we first calculate the sample expected value and sample variance for the quantity of interest. For each trial i of simulation of a given line simulated n times, we obtain an estimate for a given parameter x that we will call x_i. For example, x might be the throughput rate for a line simulated for 40000 time units. We first calculate, for random variable x

$$\begin{aligned}
\sigma^2 &= E[x^2] - E[x]^2 \\
S_x^2 &= \frac{\sum_{i=1}^{n} x_i^2 - [\sum_{i=1}^{n} x_i]^2}{n-1}.
\end{aligned}$$

However, we are interested in the confidence interval for $\overline{x}$. We estimate the expected value of $\overline{x}$ with the following:

$$\hat{\overline{x}} = \frac{\sum_{i=1}^{n} x_i}{n}$$

and the sample variance of $\overline{x}$ is

$$\begin{aligned} S_{\overline{x}}^2 &= \frac{nS_x^2}{n^2} \\ &= \frac{S_x^2}{n} \\ &= \frac{\sum_{i=1}^{n} x_i^2 - [\sum_{i=1}^{n} x_i]^2}{n(n-1)}, \end{aligned}$$

which leads to a sample standard deviation of $\overline{x}$ of

$$S_{\overline{x}}^2 = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - [\sum_{i=1}^{n} x_i]^2}{n(n-1)}}.$$

To determine the confidence interval, we invoke the Law of Large Numbers (Drake, 1988). Since $N > 30$, we make the common assumption that x follows a normal distribution. Therefore, we can use the following to estimate the confidence interval around this value:

$$95\% \text{ Confidence Interval} = \hat{\overline{x}} \pm 1.96\sqrt{\frac{\sum_{i=1}^{n} x_i^2 - [\sum_{i=1}^{n} x_i]^2}{n(n-1)}}.$$

Appendix C

Experimental Flow Lines

We performed a number of experiments to test the models in Chapters 3 and 5. In Section 3.1, we list the examples provided in Glassey and Hong (1993). In Section 3.2, we list more examples that we created to test the robustness of the algorithm for single-machine-per-stage systems. Finally, in Section 3.3, the parameters for a number of series-parallel systems are provided.

3.1 Glassey and Hong Cases

The following is a listing of each of the test cases provided in Glassey and Hong (1993):

Case 1					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-2	0.100	0.050	1.000	0.667	5
3	0.100	0.050	1.000	0.667	n/a

Case 2					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-2	0.050	0.002	1.000	0.962	5
3	0.050	0.002	1.000	0.962	n/a

Case 3					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-2	0.100	0.100	1.000	0.500	5
3	0.100	0.100	1.000	0.500	n/a

Case 4					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-2	0.100	0.100	1.000	0.500	10
3	0.100	0.100	1.000	0.500	n/a

Case 5					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.070	0.010	1.000	0.875	10
2	0.100	0.013	1.000	0.885	10
3	0.050	0.007	1.000	0.877	n/a

Case 6					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-9	0.100	0.100	1.000	0.500	10
10	0.100	0.100	1.000	0.500	n/a

Case 7					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1,4,7,10	0.070	0.010	1.000	0.875	10
2,5,8,11	0.100	0.013	1.000	0.885	10
3,6,9	0.050	0.007	1.000	0.877	10
12	0.050	0.007	1.000	0.877	n/a

Case 8					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-16	0.100	0.100	1.000	0.500	5
17	0.100	0.100	1.000	0.500	n/a

Case 9					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-19	0.100	0.100	1.000	0.500	5
20	0.100	0.100	1.000	0.500	n/a

Case 10					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-19	0.100	0.100	1.000	0.500	10
20	0.100	0.100	1.000	0.500	n/a

Case 11					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	50
2	0.090	0.020	1.100	0.818	50
3	0.090	0.030	1.200	0.750	n/a

Case 12					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.020	1.100	0.833	50
2	0.080	0.010	1.000	0.889	50
3	0.100	0.010	1.000	0.909	n/a

Case 13					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.050	1.500	0.667	30
2	0.080	0.020	1.000	0.800	70
3	0.070	0.030	1.100	0.700	n/a

Case 14					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.080	0.010	1.000	0.889	70
2	0.200	0.020	1.000	0.909	30
3	0.500	0.050	0.900	0.909	n/a

Case 15					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.050	0.005	1.000	0.909	50
2	0.050	0.005	1.100	0.909	50
3	0.050	0.005	1.000	0.909	n/a

Case 16					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.080	0.010	1.000	0.889	50
2	0.090	0.020	1.100	0.818	50
3	0.090	0.020	1.100	0.818	50
4	0.080	0.010	1.000	0.800	n/a

Case 17					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	50
2	0.070	0.010	1.000	0.875	50
3	0.100	0.010	0.900	0.909	50
4	0.080	0.010	1.000	0.889	n/a

Case 18					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.080	0.020	1.100	0.800	40
2	0.070	0.010	1.000	0.875	50
3	0.090	0.010	1.000	0.900	60
4	0.050	0.010	0.900	0.833	n/a

Case 19					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1,3	0.090	0.010	1.000	0.900	50
2,4	0.090	0.020	1.100	0.818	50
5	0.090	0.010	1.000	0.900	n/a

Case 20					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	30.000	0.900	0.090
2	0.005	0.900	40.000	0.0055249	0.090
3	0.012	1.100	50.000	0.0107914	0.090
4	0.015	1.200	60.000	0.0123457	0.090
5	0.010	1.000	44.000	0.0099010	n/a

Case 21					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1-2	0.090	0.010	1.000	0.900	50
3	0.090	0.010	1.000	0.900	n/a

Case 22					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	20
2	0.090	0.010	1.000	0.900	80
3	0.090	0.010	1.000	0.900	n/a

Case 23					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.020	1.000	0.818	50
2	0.090	0.010	1.000	0.900	50
3	0.090	0.030	1.000	0.750	n/a

Case 24					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.020	1.000	0.818	30
2	0.090	0.010	1.000	0.900	70
3	0.090	0.030	1.000	0.750	n/a

Case 25					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	50
2	0.100	0.010	1.000	0.909	50
3	0.080	0.010	1.000	0.889	n/a

Case 26					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	70
2	0.100	0.010	1.000	0.909	30
3	0.080	0.010	1.000	0.889	n/a

Case 27					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	50
2	0.090	0.020	1.100	0.818	50
3	0.090	0.030	1.200	0.750	n/a

Case 28					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	70
2	0.090	0.020	1.100	0.818	30
3	0.090	0.030	1.200	0.750	n/a

Case 29					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	50
2	0.090	0.010	1.200	0.900	50
3	0.090	0.010	1.100	0.900	n/a

Case 30					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.090	0.010	1.000	0.900	20
2	0.090	0.010	1.200	0.900	80
3	0.090	0.010	1.100	0.900	n/a

Case 31					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.080	0.010	1.200	0.889	50
2	0.090	0.010	1.100	0.900	50
3	0.100	0.010	1.000	0.909	n/a

Case 32					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.080	0.010	1.200	0.889	30
2	0.090	0.010	1.100	0.900	70
3	0.100	0.010	1.000	0.909	n/a

3.2 Other Single Machine per Stage Cases

In addition to the 32 Glassey and Hong (1993) experimental cases, we tested a number of other cases in Chapter 3. These are listed below.

Case 33					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.010	1.0	0.909	10
2	0.100	0.010	1.0	0.909	10
3	0.100	0.010	1.0	0.909	n/a

Case 34					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.010	1.000	0.909	10
2	0.100	0.010	1.000	0.909	10
3	0.010	0.010	1.000	0.500	n/a

Case 35					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.010	1.000	0.909	10
2	0.100	0.010	1.000	0.909	5
3	0.100	0.010	1.000	0.909	n/a

Case 36					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.010	1.000	0.909	10
2	0.100	0.010	1.000	0.909	10
3	0.100	0.100	1.000	0.500	n/a

Case 37					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	0.100	0.010	1.000	0.909	10
2	0.100	0.010	1.000	0.909	10
3	0.100	0.010	2.000	0.909	n/a

Case 38					
	Parameters				
i	r_i	p_i	μ_i	e_i	N_i
1	1.000	0.000	1.000	1.000	10
2	1.000	0.000	1.000	1.000	10
3	0.100	0.100	2.000	0.500	n/a

Case 39				
	Parameters			
i	r_i	p_i	μ_i	e_i
1-3	0.100	0.010	1.000	0.909

Case 40				
	Parameters			
i	r_i	p_i	μ_i	e_i
1-10	0.100	0.010	1.000	0.909

Case 41				
	Parameters			
i	r_i	p_i	μ_i	e_i
1-3	0.010	0.010	1.000	0.500

Case 42				
	Parameters			
i	r_i	p_i	μ_i	e_i
1-10	0.010	0.010	1.000	0.500

Case 43							
	Parameters				Buffer Results		
i	r_i	p_i	μ_i	N_i	Sim. $\overline{n}$	95% CI	ADDX$\overline{n}$
1	0.0201	0.0018	3.840	1196	1192.7	1191.8,1193.7	1192.9
2	0.0451	0.0278	4.050	101	90.3	89.5,91.1	91.0
3	0.2343	0.0461	3.540	39	37.3	37.1,37.4	37.7
4	0.3718	0.2156	2.610	13	5.6	5.4,5.7	7.2
5	0.0823	0.0421	3.530	35	17.7	16.9,18.4	28.1
6	0.3740	0.1318	2.710	19	8.5	8.1,8.9	14.8
7	0.6799	0.2630	3.478	11	4.6	4.3,4.8	8.8
8	0.0321	0.0200	3.250	532	341.9	291.9,391.9	518.4
9	0.0490	0.0190	3.470	348	320.7	300.5,341.0	339.7
10	0.6039	0.1575	3.260	30	27.8	26.9,28.6	28.8
11	0.0334	0.0023	2.690	123	118.4	116.0,120.8	120.2
12	0.1351	0.0688	2.990	9	5.7	5.6,5.8	6.5
13	0.1206	0.0120	3.270	69	60.4	59.7,61.1	64.3
14	0.3296	0.1759	3.300	11	8.3	8.3,8.4	8.8
15	0.8293	0.2979	2.870	20	11.7	11.6,11.9	11.5
16	0.0526	0.0550	3.370	30	10.8	10.6,11.1	9.7
17	0.0843	0.0700	3.290	n/a	n/a	n/a	n/a

Case 44				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.0190	0.00027	1.40	226
2	0.1202	0.01400	1.43	111
3	0.0347	0.00831	1.41	193
4	0.0213	0.00027	1.49	146
5	0.1226	0.00078	1.42	18
6	0.0766	0.00645	1.46	79
7	0.0554	0.00224	1.47	96
8	0.0257	0.00210	1.45	157
9	0.1113	0.00385	1.65	1

3.3 Multi-Machine Examples

There are no standard cases to test for long series parallel flow lines. The following 18 examples were chosen to illustrate how the approximation developed in Chapter 5 works under a number of specific conditions:

Case 45				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2a	0.100	0.010	1.000	n/a
2b	0.100	0.010	1.000	10
3	0.100	0.010	1.000	n/a

Case 46				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2a	0.100	0.010	1.000	n/a
2b	0.100	0.010	1.000	1
3	0.100	0.010	1.000	n/a

Case 47				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2	0.100	0.010	2.000	10
3	0.100	0.010	1.000	n/a

Case 48				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2	0.100	0.010	2.000	1
3	0.100	0.010	1.000	n/a

Case 49				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2	0.200	0.020	2.000	10
3	0.100	0.010	1.000	n/a

Case 50				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2	0.200	0.020	2.000	1
3	0.100	0.010	1.000	n/a

Case 51				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2a	0.100	0.010	0.500	n/a
2b	0.100	0.010	0.500	10
3	0.100	0.010	1.000	n/a

Case 52				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2a	0.100	0.010	0.500	n/a
2b	0.100	0.010	0.500	1
3	0.100	0.010	1.000	n/a

Case 53				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2	0.100	0.010	2.000	10
3	0.100	0.010	1.000	n/a

Case 54				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2	0.100	0.010	2.000	1
3	0.100	0.010	1.000	n/a

Case 55				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2	0.200	0.020	2.000	10
3	0.100	0.010	1.000	n/a

Case 56				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2	0.200	0.020	2.000	1
3	0.100	0.010	1.000	n/a

Case 57				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2a	0.100	0.120	1.000	n/a
2b	0.100	0.120	1.000	10
3	0.100	0.010	1.000	n/a

Case 58				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2a	0.100	0.120	1.000	n/a
2b	0.100	0.120	1.000	1
3	0.100	0.010	1.000	n/a

Case 59				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2	0.100	0.120	2.000	10
3	0.100	0.010	1.000	n/a

Case 60				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2	0.100	0.120	2.000	1
3	0.100	0.010	1.000	n/a

Case 61				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	10
2	0.100	0.120	2.000	10
3	0.100	0.010	1.000	n/a

Case 62				
	Parameters			
i	r_i	p_i	μ_i	N_i
1	0.100	0.010	1.000	1
2	0.100	0.120	2.000	1
3	0.100	0.010	1.000	n/a

Bibliography

[1] R. Alves. Performance evaluation of series-parallel systems. *Journal of Manufacturing Operations Management*, 3:224–250, 1990.

[2] M. Ammar and S. Gershwin. Equivalence relations in queueing models of fork/join queueing networks with blocking. *Performance Evaluation*, 10:233–245, 1989.

[3] B. Ancelin and A. Semery. Calcul de la productivité d'une ligne intégré de fabrication: Calif, un logiciel baseé sur une nouvelle heuristique. *APII*, 21:209–238, 1987.

[4] P. Awate and B. Sastry. Analysis and decomposition of transfer and flow lines. *Operations Research*, 24:175–196, 1987.

[5] G. Buxey, N. Slack, and R. Wild. Production flow line system design - A review. *AIIE Transactions*, 5:37–48, 1973.

[6] J. Buzacott. Automatic transfer lines with buffer stocks. *International Journal of Production Research*, pages 183–200, 5 1967a.

[7] J. Buzacott. *Markov Chain Analysis of Automatic Transfer Line with Buffer Stock*. PhD thesis, University of Birmingham, 1967b.

[8] J. Buzacott. Prediction of the efficiency of production systems without internal storage. *International Journal of Production Research*, 6:173–188, 1968.

[9] J. Buzacott. The effect of station breakdowns and random processing times on the capacity of flow lines. *AIIE Transactions*, 4:308–312, 1972.

[10] J. Buzacott. Optimal operating rules for automatic manufacturing systems. *IEEE Transactions Automatic Control*, AC-27:80–86, 1982.

[11] J. Buzacott and L. Hanifin. Models of automatic transfer lines with inventory banks - A review and comparison. *IIE Transactions*, 10:197–207, 1978a.

[12] J. Buzacott and L. Hanifin. Transfer line design and analysis - An overview. In *IIE Conference*, 1978b.

[13] J. Buzacott and J. Shanthikumar. *Stochastic Models of Manufacturing Systems*. Prentice Hall, NJ, USA, 1993.

[14] Y. Choong and S. Gershwin. A decomposition method for the approximate evaluation of capacitated transfer lines with unreliable machines and random processing times. *IIE Transactions*, 19:150–159, 1987.

[15] C. Commault and A. Semery. Taking into ccount delays in buffers for analytical performance evaluation of transfer lines. *IIE Transactions*, 22:133–142, 1990.

[16] R. Dallery, Y. David and X. Xie. An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. *IIE Transactions*, 20:280–283, 1988.

[17] R. Dallery, Y. David and X. Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, AC-34:943–953, 1989.

[18] Y. Dallery, 1995. Private conversation about series-parallel systems.

[19] Y. Dallery and S. Gershwin. Manufacturing flow lines systems: A review of models and analytic results. *Queueing Systems: Theory and Applications*, 12(1-2):3–94, December 1992.

[20] R. David, X. Xie, and Y. Dallery. Properties of continuous models of transfer lines with unreliable machines and finite buffers. *IMA Journal of Math Business Ind.*, 6:281–308, 1990.

[21] M. De Koster. Estimation of line efficiency by aggregation. *International Journal of Production Research*, 25:615–626, 1987.

[22] A. Drake. *Fundamentals of Applied Probability Theory.* McGraw-Hill Publishing Company, Inc., 1988.

[23] D. Dubois and J. Forestier. Productivité et en cours moyen d'un ensemble de deux machines séparées par une zone de stockage. *RAIRO Automatique*, 16, 1982. 105-132.

[24] E. Elsayed and C. Hwang. Analysis of manufacturing systems with buffer storage and redundant machines. Technical report, Dept. of Ind. Eng., Rutgers Univ., 1984.

[25] J. Forestier. Modelisation stochastique et comportement asymptotique d'un system automatise deproduction. *RAIRO Automatique*, 14:127–144, 1980.

[26] Y. Frein, C. Commault, and Y. Dallery. Analytical performance evaluation of closed transfer lines with limited number of pallets. In *IEEE Conference on Robotics and Automation*, 1992.

[27] D. Gaver. A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society*, B24:73–90, 1954.

[28] S. Gershwin. An efficient decomposition algorithm for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research*, 35:291–305, 1987a.

[29] S. Gershwin. Representation analysis of transfer lines with machines that have different processing rates. *Annals of Operations Research*, 9:511–530, 1987b.

[30] S. Gershwin. An efficient decomposition algorithm for unreliable tandem queueing systems with finite buffers. In H. G. Perros and T. Altiok, editors, *Queueing Networks with Blocking*, pages 127–146. North Holland, Amsterdam, 1989.

[31] S. Gershwin. Variance of output of a tandem production system. In R. Onvural and I. Akyildiz, editors, *Queueing Network with Finite Capacity*, pages 291–304. North-Holland, Amsterdam, 1993.

[32] S. Gershwin. *Manufacturing Systems Engineering*. Prentice Hall, Inc., New Jersey, 1994.

[33] S. Gershwin and O. Berman. Analysis of transfer lines consisting of two unreliable machines with random processing times and finite storage buffers. *AIIE Tranportation*, 13, 1981.

[34] S. Gershwin and Y. Goldis. Efficient algorithms for transfer line design. MIT Laboratory for Manufacturing and Productivity working paper in preparation, 1995.

[35] S. Gershwin and I. Schick. Continuous model of an unreliable two-stage material flow system with a finite interstage buffer. Technical Report LIDS-R-1039, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980.

[36] C. Glassey and Y. Hong. The analysis of behavior of an unreliable two-stages automatic transfer line with inter-stage buffer storage. Technical report, University of Berkeley, Berkeley, California, 1986.

[37] C. Glassey and Y. Hong. The analysis behavior of an unreliable n-stage automatic transfer line with $(n-1)$ interger-stage buffer storages. *International Journal of Production research*, 31(3):519–530, 1993.

[38] F. Hillier and R. Boling. Finite queues in series with exponential or Erlang service times - A numerical approach. *Operations Research*, 16:286–303, 1967.

[39] Y. Hong and D. Seong. The analysis of an unreliable n-machine flow-line manufacturing system with random processing times. Pohang Institute of Technology, Dept. of Industrial Engineering, 1989. Technical Report 89-03.

[40] E. Ignall and A. Silver. The output of two-storage system with unreliable machines and limited storage. *AIIE Transactions*, 9:183–188, 1977.

[41] T. Iyama and S. Ito. The maximum production rate for an unbalances multi-server flow line system with finite buffer storage. *International Journal of Production Research*, 25:1157–1170, 1987.

[42] M. Jafari and J. Shanthikumar. An appropriate model of multistage automatic transfer lines with possible scrapping of work pieces. *IIE Transactions*, 19:252–265, 1987.

[43] E. Koenigsberg. Production lines and internal storage - a review. *Management Science*, 5:410–433, 1959.

[44] R. Lipset, S. Sengupta, and R. Van Til. Analytic modeling of the unreliable buffer in the serial transfer line. In *26th IEEE Southeastern Symposium on System Theory, Athens, OH*, 1994a.

[45] R. Lipset, S. Sengupta, and R. Van Til. A decomposition technique for steady-state performance evaluation of serial transfer lines subject to machine and buffer failures. In *33rd IEEE Conference on Decision and Control, Florida*, 1994b.

[46] X. Liu. *Toward Modeling Assembly Systems: Applications of Queuing Networks with Blocking*. PhD thesis, University of Waterloo, Waterloo, Canada, 1990.

[47] D. Mitra. Stochastic theory of a fluid model of multiple failure-susceptible producers and consumers couples by a buffer. *Advanced Applied Probability*, September 1988.

[48] J. Muth. The reversibility property of production lines. *Management Science*, 25:152–158, 1979.

[49] K. Okamura and H. Yamashina. Analysis of the effect of buffer storage capacity in transfer line systems. *AIEE Transactions*, 9:127–135, 1977.

[50] H. G. Perros. Queueing networks with blocking: A bibliography. *Performance Evaluations, Rev. 12*, 8-14, 1986.

[51] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, U.K., 1988.

[52] S. Ransley. A manufacturing cell design tool for the development of transfer lines. Master's thesis, Massachusetts Institute of Technology, 1995.

[53] S. Ross. *Stochastic Processes*. John Wiley and Sons, New York, USA, 1983.

[54] I. Schick and S. Gershwin. Modelling and analysis of tandem queues without intermediate buffers. Technical Report ESL-FR-834-6, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

[55] R. Schonberger. *World Class Manufacturing: The Lessons of Simplicity Applied*. The Free Press, New York, US, 1986.

[56] A. Semery. Modelisation des lif: Integration du modele de dubois et forestier a l'heuristique de gershwin, pour les lignes a temps de cycle differents. *Rapport de recherche AS/619-87*, 1987.

[57] B. Sevast'yanov. Influence of storage bin capacity on the average standstill time of a profuction line teoriya veroyatnostey i ee primeneniya. *Theory Prob. Appl.*, 7:438–447, 1962.

[58] J. Shanthikumar and C. Tien. An algorithmic solution to two-stage transfer lines with possible scrapping of units. *Management Science*, 29:1069–1086, 1983.

[59] T. Smunt and W. Perkins. Stochastic unpaced line design: Review and further experimental results. *Journal of Operations Management*, 5:351–373, 1985.

[60] E. Snodgrass. *Beyond the Basics of Reengineering: Survival Tactics for the 90's*. Quality Resources, White Plains, New York, USA, 1994.

[61] R. Suri and G. Diehl. A variable buffer-size model and its use in analyzing closed queueing networks with blocking. *Management Science*, 32:206–224, 1986.

[62] R. Suri and B. Fu. Using continuous flow models to enable rapid analysis and optimization of discrete production lines - A progress report. In *NSF Grantees Conference on Design and Manufacturing Systems*, volume 19, Charlotte, NC, January 1993.

[63] C. Terracol and R. David. An aggregation method for performance evaluation of transfer lines with unreliable machines and finite buffers. In *International Conference on Robotics and Automation*, 1987.

[64] A. P. Vladzievski. Probabilistic law of operation and internal storage of automatic lines. *Avtomatika i Telemehanika*, 13, 1952. (In Russian).

[65] J. Wijngaard. The effect of interstage buffer storage on the output of two unreliable production units in series with different production rates. *AIIE Transactions*, 11:42–47, 1979.

[66] S. Yeralan and E. Muth. A general model of a production line with intermediate buffer and station breakdown. *IIE Transactions*, 19:201–212, 1975.

[67] B. Zimmern. Etudes de la propagation des arrets aleatoires dans les chaines de production. *Review Statististical Applications*, 4:85–104, 1956.