An approximate analytical method for evaluating the performance of closed loop flow systems with unreliable machines and finite buffers — Part II: Large Loops

Stanley B. Gershwin[*] Loren M. Werner[†]

May 8, 2003

Abstract

We present an efficient and accurate approximate analytical decomposition method for evaluating the production rate and distribution of inventory of a closed loop manufacturing system with unreliable machines and finite buffers. It is based on an earlier decomposition for a tandem line; it differs only in accounting for the different sets of machines that could cause blockage or starvation to other machines. The method can be applied to tandem production lines with a limited number of pallets or fixtures; the pallets travel in a loop even thought the parts follow a line. It can also be applied to systems controlled by a conwip policy with infinite or finite buffers since such a policy can be implemented with a limited number of tokens that behave in the same way as pallets.

Contents

[*] Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 USA, gershwin@mit.edu, http://web.mit.edu/manuf-sys.

[†] United States Air Force, Randolph AFB, TX, USA, loren.werner@randolph.af.mil.

1 Introduction

1.1 Problem Statement

In this paper, we present an efficient and accurate analytical decomposition method for evaluating the production rate and distribution of inventory of a closed loop manufacturing system with unreliable machines and finite buffers. This is an extension of the method of Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003) which was only practical for very small systems. The new method, which was first described in Werner (2001), relies on a novel network transformation and can treat systems with a large number of machines and buffers. We focus our attention here on loops with discrete material and with fixed-processing-time machines (the Buzacott model) but the same transformation is applicable to other versions that have been used for lines and assembly/disassembly networks (discrete material and exponential processing time machines; and continuous material).

A closed-loop production system or *loop* is a system in which a constant amount of material (the *loop population*) flows through a single fixed cycle of work stations and storage buffers. This type of system appears frequently in factories. Manufacturing processes which utilize pallets or fixtures can be viewed as loops since the number of pallets/fixtures that are in the system remains constant. Similarly, control policies such as CONWIP and kanban create conceptual loops by imposing a limitation on the number of parts that can be in the system an any given time. Figure 1 represents a K-machine loop.

Decomposition methods have been developed for the performance evaluation of some large systems for which exact analytic methods do not exist and numerical methods are infeasible. These techniques approximate the original large system with a set of smaller systems that satisfy a set of carefully chosen relationships. Networks with finite buffers are the major focus of attention. Early work focused on tandem systems with unreliable machines and finite buffers (Gershwin 1987; Dallery, David, and Xie 1988); more recent papers extended it to assembly/disassembly networks (Mascolo, David, and Dallery 1991; Gershwin 1991). Except for a small number of papers, this work was limited to acyclic or tree-like systems: those whose graphs contain no loops. Literature surveys appear in Dallery and Gershwin (1992) and Gershwin (1994).

In this paper, we make use of and extend a recent decomposition method (Tolio and Matta 1998). We describe the extension of decomposition methods to systems that consist only of the machines and buffers in a single loop. A later paper (Gershwin and Levantesi 2003) will describe the extension to general assembly/disassembly networks with multiple loops. Such networks are of interest because they arise in the analysis of an important class of control policies for manufacturing systems (Gershwin 2000).

Performance measures such as average production rate and the distribution of in-process inventory cannot be expressed in closed form. Simulation provides accurate results for these quantities, but it can be time consuming. Although constant improvements in computer technology steadily reduce simulation time, computation time remains important for optimization, which requires very large numbers of evaluations. Some fast analytical methods have been developed, but they are accurate in a limited class of cases. The purpose of this paper is to describe a more versatile approximate analytical method for evaluating the performance measures of closed-loop production systems.

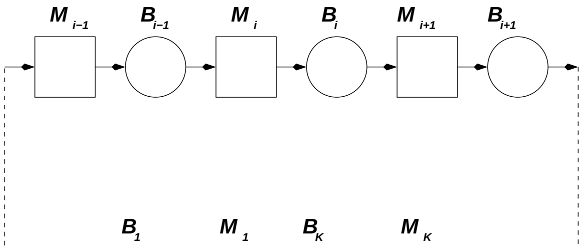


Figure 1: Illustration of a closed-loop production system

1.2 Literature Review

Compared to open production lines, little work has been done on closed-loop production systems with finite buffers and unreliable machines. Onvural and Perros (1990) demonstrated that the production rate of a closed-loop system is a function of the number of parts in the system. In addition, they showed that the throughput versus population curve is symmetric when blocking occurs before service and processing time is exponential. To avoid the complication that finite buffers create in closed-loop systems, Akyildiz (1988) approximated the production rate by reducing the population and evaluating the same system with infinite buffers. Bouhchouch, Frein, and Dallery (1992) used a closed-loop queuing network with finite capacities to model a closed-loop system with finite buffers. For more detailed listings and discussions of previous work dealing with closed-loop finite-buffer systems, see Tolio and Gershwin (1998), Maggio (2000), Maggio, Matta, Gershwin, and Tolio (2003), and Balsamo, de Nitto Personé, and Onvural (2001).

The first approximate analytical method for evaluating the performance of closed-loop systems with finite buffers and unreliable machines was proposed by Frein, Commault, and Dallery (1996). This method was an extension of the Gershwin (1987) decomposition technique. It did not account for the correlation among numbers of parts in each buffer. As a result, the method is only accurate for large loops with populations that are neither too large or too small. A recent method has been presented by Han and Park (2002), but numerical results for only a small number of loops are described.

Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003) present a new decomposition method which does account for the correlation between population and the probability of blocking and starvation. However, the method is too complex to be practical for loops with more than three machines. Here, we simplify and extend these results to loops with any number of machines.

1.3 Outline

We describe the model, the transformation, and the solution technique in Section 2. We describe the algorithm in detail in Section 3. In Section 4 we describe the performance of the technique and in Section 5 we use it to demonstrate some features of loop behavior. We conclude and recommend future research directions in Section 6.

2 Approach

In Section 2.1, we describe the model we are analyzing, review decomposition methods for open lines, present the important features of closed loop systems that are accounted for in our method, and define buffer thresholds and explain their impact on the analysis of these systems. We introduce the transformation that allows us to avoid treating thresholds explicitly in Section 2.2, and define the range of blocking and the range of starvation. In Section 2.3, we present the new decomposition and describe the building block and the decomposition equations. This new decomposition is so similar to Tolio and Matta's method (Tolio and Matta 1998) for an open line, that a program to analyze loops can be written by making a minor modification to a program for evaluating lines. The relationship is illustrated in Section 2.3.3.

2.1 Closed-Loop Production Systems

2.1.1 Basic Model

We extend the deterministic processing time model introduced by Buzacott (1967) and modified by Gershwin (1987) and Tolio and Matta (1998) to closed-loop systems. Processing times for all machines are assumed to be deterministic and identical. We scale time so that the common processing time is one time unit. All operational (i.e. not failed) machines start their operations at the same instant. Parts in the machines are ignored, as is travel time between machines. Machine failure times and repair times are geometrically distributed. We use the version of the model presented by Tolio, Gershwin, and Matta (2002) and Tolio and Matta (1998), which allows machines to fail in more than one mode. This feature is critical to our method and is discussed in detail in Section 2.1.5.

In this paper, M_i refers to Machine i. B_i is its downstream buffer and has capacity N_i. B_{i-1} is the buffer upstream[1] of M_i. A machine is *blocked* if its downstream buffer is full and *starved* if its upstream buffer is empty. When M_i is *working* (operational and neither blocked nor starved) it has a probability p_{ij} of failing in mode j in one time unit. If M_i is down in mode j, it is repaired in a given time unit with probability r_{ij}. By convention, machine failures and repairs take place at the beginnings of time steps and changes in buffer levels occur at the ends of time steps. The *population,* the fixed total number of parts in the system, is N^p. It is convenient to define

$$N^{\textbf{total}} = \sum_{i=1}^{K} N_i,$$

the total buffer space in the loop.

Lower bound on buffer size In this model of a production line, the minimal buffer size is 2. (That is, $N_i \geq 2$ for all i.) This is because of the assumptions on the behavior of empty and full buffers, and because we do not consider the workspace in the machines to be available for storage. Werner (2001) restricted $N_i \geq 2$ for this reason. Tolio, Gershwin, and Matta (2002) required $N_i \geq 3$ because of the structure of their analytical two-machine line solution — if this is not satisfied, there is no n_i such that $2 \leq n_i \leq N_i - 2$, and the solution has a special structure for such n_i.

2.1.2 Tandem Line Decomposition Techniques

It is possible to obtain the exact steady-state probability distribution (and from that the production rate and average buffer level) analytically for a two-machine, one-buffer line. It is not possible for longer lines because of the rapid increase in the state space with the length of the line and the

[1] In an open line, there is no buffer upstream of M_1. In a K-machine closed loop, however, the buffer upstream of M_1 is B_K. The required change of notation is described in Section 2.1.3.

sizes of the buffers. However, accurate approximate decomposition methods have been developed for evaluating the performance of long tandem lines and assembly/disassembly systems (Gershwin 1994). These methods decompose a K-machine tandem line into $K-1$ two-machine lines or *building blocks*. In each building block $L(i)$, the buffer $B(i)$ corresponds to B_i in the original line. The upstream machine $M^u(i)$ represents the collective behavior of the line upstream of B_i and the downstream machine $M^d(i)$ represents the behavior downstream.

Consider an observer in B_i who is only allowed to see the arrivals and departures of material from that buffer. Suppose he is told that he is observing the buffer of a two-machine line. The upstream and downstream machines of that line are called $M^u(i)$ and $M^d(i)$. To that observer (whose buffer is referred to as $B(i)$ for consistency), $M^u(i)$ appears to be down when M_i is either down or starved by some upstream machine. In Tolio and Matta (1998), $M^u(i)$ has *local* failure modes corresponding to those of M_i and *remote* failure modes corresponding to each of the upstream machines[2]. Likewise, $M^d(i)$ has local failure modes corresponding to those of M_{i+1} and remote failure modes corresponding to each of the downstream machines. Tolio and Matta (1998) show how to obtain the failure and repair probabilities for these modes. These quantities are functions of all the machines and buffers in the original line, and are therefore related to similar parameters in the fictional two-machine lines monitored by other fictional observers. To obtain these parameters, Tolio and Matta (1998) develop a set of *decomposition equations* that involve the analysis of the two-machine lines to obtain production rates and probabilities of starvation and blockage.

This is illustrated in Figure 2, which focuses on the view of the observer in the buffer B_3, who believes that he is in the buffer of the two-machine line consisting of $M^u(3)$, $B(3)$, and $M^d(3)$. Failure modes are indicated in the machines, so Machine 1 fails in modes 1 and 2, Machine 2 fails in mode 3, etc. The local modes, as seen by the observer in Buffer 3 are 4, 5, 6, and 7. The rest are remote.

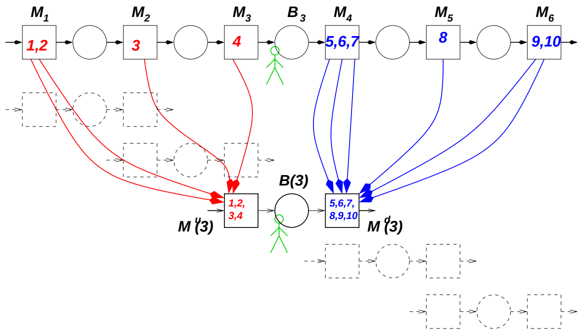


Figure 2: Tolio decomposition

The goal of the decomposition method is to choose the parameters of $M^u(i)$ and $M^d(i)$ such that the flow of parts through $B(i)$ mimics the flow through B_i. Accomplishing this for all building blocks gives approximate values for throughput and average buffer levels in the original tandem line.

2.1.3 Special Characteristics of Closed-Loop Systems

In a tandem line, each machine can block every upstream machine and it can starve every downstream machine. That is, each machine can cause every upstream buffer to become full, if it stays down long enough, and it can empty every downstream buffer if it is down for a sufficiently long time.

The situation is different in loops. Whether or not a machine can be starved or blocked by the failure of another machine depends on the number of parts in the system and the total buffer space between the two machines. We define (i,j) as the set of indices of the machines between M_i and

[2] In papers by Tolio and his co-authors, the local and remote modes were called *real* and *virtual* modes.

M_j inclusive. For ease of notation, we define all subscripts to be modulo K. Then

$$(i,j) = \begin{cases} (i, i+1, ..., j) & \text{if } i < j \\ (i, i+1, ..., K, 1, ..., j) & \text{if } i > j \end{cases} \tag{1}$$

We define $\Psi(v,w)$ as the total buffer space between M_v and M_w in the direction of flow (Maggio 2000; Maggio, Matta, Gershwin, and Tolio 2003). More formally,

$$\Psi(v,w) = \begin{cases} \sum_{z=v}^{w-1} N_z & \text{if } v \neq w \\ 0 & \text{if } v = w \end{cases} \tag{2}$$

The total buffer space in the line is $N^{\textbf{total}} = \Psi(v,w) + \Psi(w,v)$ (for any $w \neq v$) and the population must satisfy

$$0 \leq N^p \leq N^{\textbf{total}}.$$

If $N^p < \Psi(v,w)$, then the failure of M_w can never cause M_v to become blocked because there are not enough parts in the system to fill all buffers between M_v and M_w simultaneously. Conversely, if $N^p > \Psi(v,w)$, M_w cannot starve M_v because there are too many parts in the system to allow all the buffers between M_w and M_v to be empty.

2.1.4 Thresholds

The issue of blocking and starvation is more complicated still. In some cases, whether or not a machine can ever be starved or blocked by the failure of another machine depends on the number of parts in an adjacent buffer. This is the concept of *buffer thresholds* introduced in Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003).

Consider the loop in Figure 3 where 7 parts are traveling through a three-machine loop with three buffers of size 5. If M_2 fails, parts begin to build up in B_1 and eventually M_1 becomes blocked. However, we know that M_1 cannot be blocked if the number of parts in its upstream buffer, B_3, is greater than 2. If there are more than 2 parts in B_3, there must be fewer than 5 parts in B_1 since there are only 7 parts in the system. Therefore, M_1 cannot be blocked.

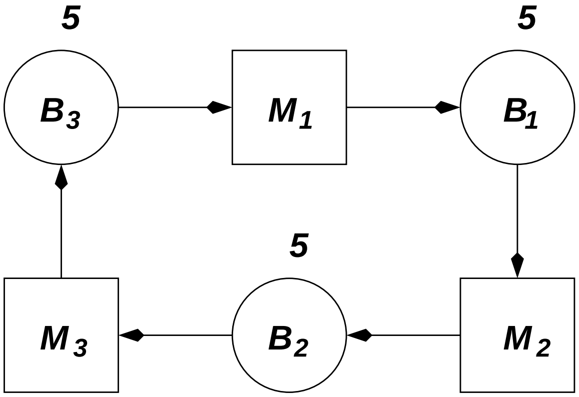


Figure 3: Example of a loop with thresholds. $N^p = 7$.

Conversely, we know that if the number of parts in B_3 is less than 2 then the number of parts in B_2 must be greater than zero and M_3 cannot become starved.[3] Therefore, we say that B_3 has a threshold of 2.

In general, we define the threshold $l_j(i)$ to be the maximum level of B_i such that all buffers between M_{i+1} and M_j can become full at the same time. That is, $l_j(i)$ is the maximum level of B_i such that the failure of M_j could cause M_{i+1} to become blocked. It is

[3] Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003) show that blocking thresholds and starving thresholds are the same.

$$l_j(i) = N^p - \Psi(i+1, j) \tag{3}$$

Note that $l_j(i)$ can assume values ranging from less than zero to greater than N_i depending on the population and buffer sizes. We only need to consider cases where $0 < l_j(i) < N_i$. To deal with these cases, Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003) propose a more detailed building block and a new set of decomposition equations. This approach is accurate and has been implemented for three-machine loops with certain restrictions on population and buffer sizes. However, this building block can only take a single threshold into account. It would be possible to extend the method to larger loops, but the building block would have to become very complex to deal with multiple thresholds.

2.1.5 Ranges of Starvation and Blockage

Here, we define the range of starvation and range of blocking of a machine. The *range of starvation of* M_i is the set of machines M_j such that if M_j is failed for a sufficiently long time and all other machines are operational, M_i will be starved; that is, B_{i-1} will be empty. It is the contiguous set $\{M_{s(i)}, M_{s(i)+1}, ..., M_{i-1}\}$, where $M_{s(i)}$ is the furthest machine upstream that can cause M_i to be starved if it is failed for a long period of time and all the other machines are operational. Similarly, the *range of blocking of* M_i is the contiguous set $\{M_{i+1}, M_{i+2}, ..., M_{b(i)}\}$, where $M_{b(i)}$ is the machine farthest downstream which can cause M_i to be blocked if it is down for a sufficiently long time and all other machine are up. These ranges are illustrated in Figure 4 for machine M_1 in a 6-machine loop in which the sizes of all the buffers are 10 and the population is 35. If M_4 fails and stays down for a very long time, and no other machines are down, M_1 will be blocked since B_1, B_2, and B_3 will be full. (The remaining 5 parts will be in B_6.) M_2 and M_3 can also cause B_1 to be full and therefore M_1 to be blocked.

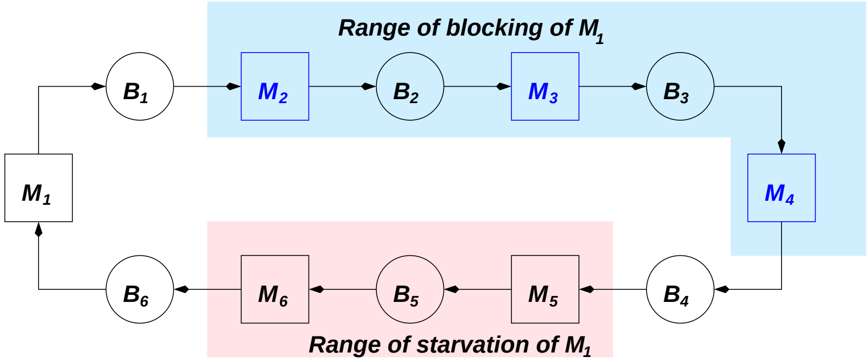


Figure 4: Ranges of blocking and starvation

If M_5 fails for a long time while all other machines are operational, the 35 parts will be found in B_4, B_3, B_2 and B_1. (B_1 will have 5 parts and the others will be full.) Therefore, B_5 and B_6 will be empty and M_1 will be starved.

(Note that if the population were 30 in this case, M_4 would be in both the range of starvation and the range of blocking of M_1. This situation must be considered for the transformation in Section 2.2.)

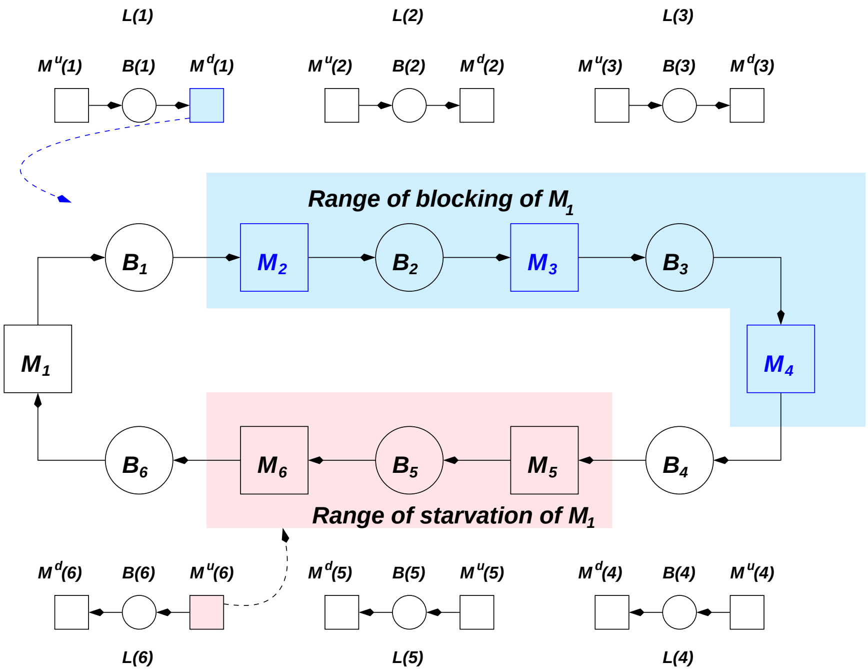


Figure 5: Loop decomposition

We calculate[4] $s(i)$ and $b(i)$ as follows:[5]

$$\begin{aligned} s(i) &= \max_j \{j+1 | \Psi(i,j) < N^p\} \quad (4) \\ b(i) &= \min_j \{j | \Psi(i,j+1) > N^p\} \quad (5) \end{aligned}$$

2.1.6 Loop Decomposition

We are now in a position to describe the loop decomposition: it is exactly the same as the tandem line decomposition but with one exception. In the tandem line, the range of starvation of machine M_i is simply the set of machines upstream of M_i, and the range of blocking is the set of machines downstream of M_i. In the loop the ranges of blocking of starvation and blockage are described instead by (4)–(5). This is represented in Figure 5, which is the decomposition of the loop in Figure 4.

In building block $L(i)$, the failure modes of $M^u(i)$ are the same as those of the machines in the range of starvation of M_{i+1}: machines $M_{s(i+1)}$ through M_i. The local modes are those of M_i and the remote modes are those of $M_{s(i+1)}$ through M_{i-1}. $M^d(i)$ has failure modes which are the same

[4] Here, "min" means "travel downstream from M_i until you find the first machine for which the condition is satisfied," and "max" means "travel downstream from M_i until you find the last machine for which the condition is satisfied."

[5] Note that the inequalities are strict. We use this convention to deal with the situation of simultaneous blocking and starvation. See Section 2.2.2.

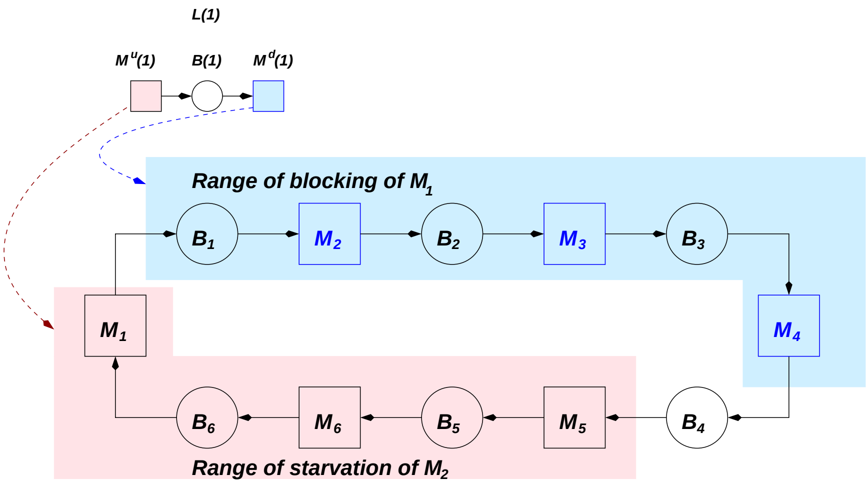


Figure 6: Failure modes in $L(1)$

as those of the machines in the range of blocking of M_i, the modes of M_{i+1} through $M_{b(i+1)}$. The remote modes are those of M_{i+2} through $M_{b(i+1)}$ and the local modes are those of M_{i+1}.

For example, consider Figure 5 with $N^p = 35$. $B(1)$ in building block $L(1)$ being full is an event that should occur in almost the same way as (ie, almost statistically indistinguishably from) how B_1 being full in the original loop occurs. Since B_1 can become full when and only when M_2, M_3, or M_4 fail, then the failure modes of $M^d(1)$ (whose failure causes the filling of $B(1)$) should be the same as those of M_2, M_3, and M_4. In this case, $i = 1$ and $b(1) = 4$.

Similarly, $B(6)$ in building block $L(6)$ being empty is an event that should occur in almost the same way as (ie, almost statistically indistinguishable from) how B_6 being empty in the original loop occurs. Since B_6 can become empty when and only when M_5 or M_6 fail, then the failure modes of $M^u(6)$ (whose failure causes the emptying of $B(6)$) should be the same as those of M_5 and M_6. Here, $i = 6$ and $i + 1 = 7 = 1 \bmod 6$ so $s(1) = 5$.

Figure 6 shows the failure modes of the machines of $L(1)$. We have already described the failure modes of $M^d(1)$. The failure modes of $M^u(1)$ are those of the machines in the range of starvation of M_2

Recall that the sets of machines that are in each range of blocking and starvation depend on the population N^p (as well as the sizes of the buffers). The method takes N^p into account by including in the building blocks only those failure modes from machines within the appropriate ranges of blocking and starvation.

2.2 Loop Transformation

2.2.1 Elimination of Thresholds

It is possible to eliminate the complications in the two-machine building blocks due to thresholds by transforming the loop. The transformation allows us to evaluate much larger loops for a wider range of population levels and buffer sizes than is possible using the method presented in Maggio

(2000) and Maggio, Matta, Gershwin, and Tolio (2003). Instead of dealing with the thresholds directly, we transform the loop into one without thresholds that behaves in almost the same way. The resulting loop is relatively easy to analyze.

Consider again the three-machine loop of Figure 3. Into each of the three buffers, we insert a perfectly reliable machine so that the buffer of size 5 is replaced by an upstream buffer of size 3 and a downstream buffer of size 2. See Figure 7. The performance of this new six-machine loop is approximately the same[6] as the original three-machine loop, and we have eliminated all thresholds between 0 and N_i.

We can extend this approach to any K-machine loop. For each threshold $0 < l_k(i) < N_i$, we insert a perfectly reliable machine M_{k^*} into buffer B_i such that $\Psi(k^*, k) = N^p$. B_i is now represented as a buffer of size $N_i - l_k(i)$ followed by M_{k^*} followed by a buffer of size $l_k(i)$. Since each unreliable machine can generate at most one threshold between zero and N_i, the transformed loop will consist of at most $2K$ machines. Although the loop is larger, we can now use the same building block that is used in Tolio's tandem line decomposition (Tolio, Gershwin, and Matta 2002, Tolio and Matta 1998). Furthermore, the computational complexity does not increase with the addition of the new machines because no new failure modes are introduced.

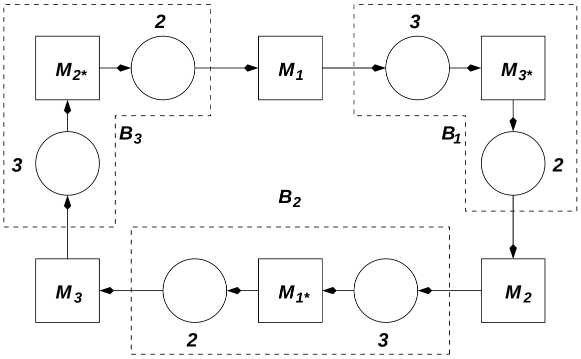


Figure 7: Illustration of a loop which is transformed so that thresholds are eliminated

Unfortunately, this transformation creates a problem for discrete-material models. It may create a buffer that violates the minimal buffer size condition of Section 2.1.1. In the algorithm of Section 3, we suggest increasing such buffers to the minimal size. This is crude, but it appears not to create large errors.

2.2.2 Simultaneous Blocking and Starvation

If $\Psi(v, w) = N^p$ then machine M_v can become simultaneously blocked and starved when M_w is down for a long period of time. In this case, the threshold $l_w(v-1) = 0$ and $l_w(v) = N_v$. In transformed loops, this situation can occur at each reliable machine M_{k^*} when M_k fails since the buffer sum between the two machines $\Psi(k^*, k) = N^p$ by construction.

The two-machine building block developed in Tolio and Gershwin (1996), Tolio, Gershwin, and Matta (2002) does not account for the states where both machines are down and the buffer level is either zero or full. However, this is not a problem. Although a real machine may be simultaneously blocked and starved, the pseudo-machines in the two-machine lines are never both down due the failure of the same real machine.

Figure 8 shows the ranges of blockage and starvation of M_1 in the six-machine loop where all the buffer sizes are 10 and the population is 30. Note that M_4 is in both ranges. However, the ranges of blockage and starvation of the machines in $L(1)$ are the same as in Figure 6. If M_4 fails for a long time and is the only machine that is down, buffers B_1, B_2, and B_3 will be full. In $L(1)$, buffer

[6] This is approximate because breaking up a buffer by inserting a machine adds one time unit of delay to the time it takes a part to traverse the buffer.

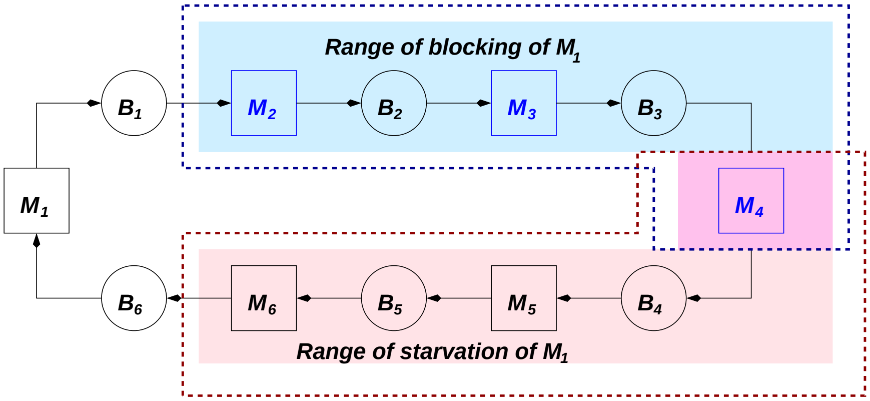


Figure 8: Ranges of blocking and starvation of M_1 when $N^p = 30$

$B(1)$ is full and $M^d(1)$ is down. Since machines M_5, M_6, and M_1 are up, $M^u(1)$ is up. $M^u(1)$ can be down only due to a long failure of M_5, M_6, or M_1. Consequently, we can never be in a situation in which $M^u(1)$ and $M^d(1)$ appear to fail simultaneously due to a failure of the same real machine.

Figure 9 shows which real machines contribute failure modes to the pseudo-machines of line $L(6)$, when the buffer sizes are all 10 and the population is 30. Since M_4 is in the range of starvation of M_1, its modes are present in $M^u(6)$. However, it is not in the range of blockage of M_6, so its modes do not appear in $M^d(6)$.

2.2.3 Reduction of Large Buffers

Consider a loop in which one buffer is the same size as the population. That is, there is one buffer B_i such that $N_i = N^p$. We claim that if N_i is increased by $m > 0$ spaces to $N^p + m$, the performance of the loop is not affected. This can be seen by running two loops which differ only in the size of

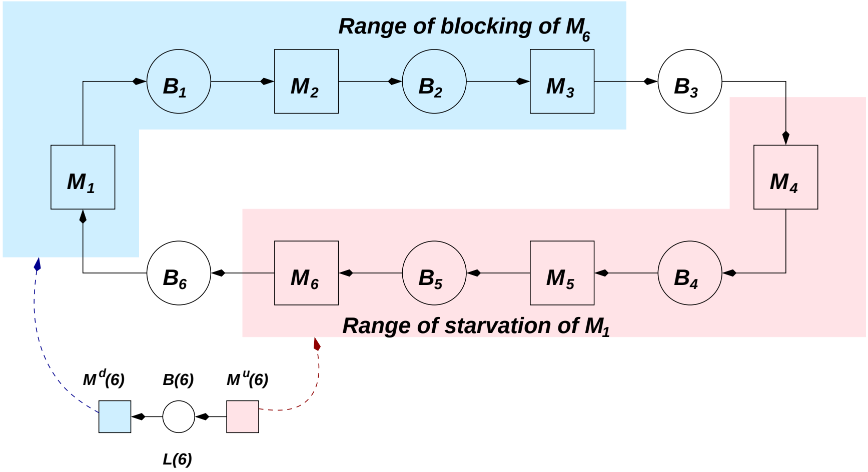


Figure 9: Failure modes in $L(6)$

N_i from the same initial condition (ie, distribution of parts and repair/failure states of machines) with the same sample (ie, the same schedule of failures and repairs of machines). Assume that in the initial condition, there are fewer than N^p parts in B_i.

Until the first time that the smaller B_i is full, the histories of all the corresponding $n_j(t)$ are the same in the two loops because both sets of trajectories are subject to the same events and constraints. At the time when both B_i hold N^p parts, machine M_i is blocked in one of the loops and not in the other. However, it is starved in both, and is thus unable to operate or fail in both. Therefore the states of both loops remain the same. When a part leaves B_i in both loops (which will happen simultaneously), the situation reverts to that of the initial condition: the distribution of inventory is the same in the two loops and the the repair/failure states are the same in the two loops. Again the buffer trajectories remain the same at least until the next time B_i holds N^p parts, but the two loops stay the same when that happens as well. To summarize: there are no events than can cause the states of the two loops to differ, so they remain the same for all time.

As a consequence, we can reduce the size of any buffer which is larger than N^p to exactly N^p without affecting the performance of the loop. In the rest of Section 2 and in Section 3, we assume $N_i \leq N^p$ for all i. In the algorithm, we reduce the size of any buffer larger than N^p to N^p.

Equivalence (Ammar and Gershwin 1989) can be invoked to reduce large buffers when the population of the loop is very large. We can define a *hole* to be a space in a buffer. If, at time t, the are $n(t)$ parts in a buffer of size N, then there are $N - n(t)$ holes in that buffer. Therefore, there are $N^{\textbf{total}} - N^p$ holes in a closed loop, and the number of holes stays constant. Every time a part moves from B_{i-1} through M_i to B_i, a hole simultaneously moves from B_i through M_i to B_{i-1}. Therefore, the flow of holes is the same as the flow of parts through a loop which is made up of the same machines and buffers but in reverse order. The throughput rates will be the same, and the average number of parts in a buffer of the original loop is the complement of the number of holes in the corresponding buffer of the reversed system.

Consider a loop in which $N^{\textbf{total}} - N^p$ is smaller than the largest buffer. This loop will be equivalent to a reversed loop with population $N^{p\prime} = N^{\textbf{total}} - N^p$. As above, we can reduce the size of any buffer in the reversed loop which is larger than $N^{p\prime}$ to $N^{p\prime}$ without changing the throughput of the loop or the average number of parts in any buffer of the reversed loop. We can then evaluate the reversed loop with reduced buffers and use its results to evaluate the original loop: the production rate of the original loop will be the same, and the average inventory in B_i of the original loop will be $N_i - \bar{n}_i{}'$, where $\bar{n}_i{}'$ is the average number of parts in the reversed loop with reduced buffers.

2.3 Decomposition of Loops

To develop a decomposition for closed loop systems, we must establish a building block model and find a way to relate the building blocks to one another. This section discusses the parameters of the building blocks and the equations used to find them. Since, according to Section 2.2, it is always possible to transform a loop into one in which buffer thresholds need not be treated explicitly, we restrict our attention to loops without thresholds.

2.3.1 The Building Block Parameters

As in the Tolio tandem line decomposition (Tolio and Matta 1998), we evaluate the loop by approximating its behavior with a set of two-machine building blocks. Each building block $L(i)$ is associated with the buffer B_i in the original loop. The upstream machine $M^u(i)$ has local failure modes corresponding to those of M_i and remote failure modes corresponding to those of machines $M_{s(i)}$ through M_{i-1}. We use the symbol λ_{kj} to represent a failure mode of $M^u(i)$: the mode corresponding to mode j of Machine M_k, $k = s(i), ..., i-1$. Similarly, $M^d(i)$ has local failure modes corresponding to those of M_{i+1} and remote failure modes corresponding to those of machines M_{i+2} through $M_{b(i+1)}$. Here, λ_{kj} represents mode j of Machine M_k, $k = i+1, ..., b(i)$.

When we have the parameters of two-machine line $L(i)$, we can use the results of Tolio, Gershwin, and Matta (2002) to evaluate its production rate $E(i)$ and the probabilities of starvation $P^{st}_{kj}(i)$ due to mode λ_{kj} and blockage $P^{bl}_{k'j'}(i)$ due to mode $\lambda_{k'j'}(i)$. These quantities are needed in the decomposition equations (Section 2.3.2).

As shown in Tolio and Matta (1998) and Maggio, Matta, Gershwin, and Tolio (2000), the failure and repair probabilities for the local failure modes are equal to the probabilities of the corresponding modes of the machines in the loop. That is,

$$p^u_{ij}(i) = p_{ij} \tag{6}$$

$$r^u_{ij}(i) = r_{ij} \tag{7}$$

$$p^d_{i+1,j}(i) = p_{i+1,j} \tag{8}$$

$$r^d_{i+1,j}(i) = r_{i+1,j} \tag{9}$$

In addition, we know that the probability of repair when a machine is down in remote failure mode λ_{kj} is equal to the probability that machine M_k is repaired when it is down in failure mode j. This gives us

$$r^u_{kj}(i) = r_{kj} \tag{10}$$

$$r^d_{kj}(i) = r_{kj} \tag{11}$$

To evaluate the performance measure of the loop, we must find the remote failure probabilities $p^u_{kj}(i)$ and $p^d_{kj}(i)$ for each $L(i)$. This is the objective of solving the decomposition equations.

2.3.2 Decomposition Equations for a Line

Here, we review the decomposition equations for the tandem line derived in Tolio and Matta (1998). We define $P^{st}_{kj}(i)$ as the probability that $B(i)$ is empty due to $M^u(i)$ being down in remote failure mode λ_{kj}. Likewise, $P^{bl}_{kj}(i)$ is the probability that $B(i)$ is full due to $M^d(i)$ being down in remote failure mode λ_{kj}. Finally, we define $E(i)$ to be the average throughput of building block $L(i)$. Using this notation, we write the decomposition equations. For all $2 \leq i \leq K-1$ and all j, Tolio and Matta (1998) show that

$$p^u_{kj}(i) = \frac{P^{st}_{kj}(i-1)}{E(i)} r_{kj} \text{ for } k = 1, ..., i-1 \tag{12}$$

and

$$p^d_{kj}(i) = \frac{P^{bl}_{kj}(i+1)}{E(i)} r_{kj} \text{ for } k = i+2, ..., K \tag{13}$$

We know the values of the local p parameters and all the r parameters in the line from (6)–(11). With the quantities determined in (12) and (13), we have enough information to evaluate the two-machine line of Tolio, Gershwin, and Matta (2002) to determine $P^{st}_{kj}(i-1)$ and $P^{bl}_{kj}(i+1)$ in (12) and (13). The decomposition equations (12) and (13) thus form a system of $2K$ independent equations in $2K$ unknowns.

2.3.3 Decomposition Equations for a Loop

The decomposition equations for a loop are nearly identical to those of the tandem line decomposition. In fact, we need only modify the limits of the indices in (12) and (13) to account for the range of blocking and starvation and the fact that loops contain as many buffers as machines.

In the line, all machines upstream of a machine can cause its starvation, and all downstream machines can cause it to be blocked. Here, however, we need to determine which machines can cause starvation and blockage to each given machine. Recall from Section 2.1.5 that $s(i)$ is the index of the machine furthest upstream which falls within the range of starvation of machine M_i. Similarly, $b(i)$ is the index of the machine furthest downstream which falls within the range of blocking of M_i. Then, for all i and j,

$$p^u_{kj}(i) = \frac{P^{st}_{kj}(i-1)}{E(i)} r_{kj} \text{ for } k = s(i+1), ..., i-1 \tag{14}$$

and

$$p^d_{kj}(i) = \frac{P^{bl}_{kj}(i+1)}{E(i)} r_{kj} \text{ for } k = i+2, ..., b(i) \tag{15}$$

Again we know the values of the local p parameters and all the r parameters, and we use the decomposition equations (14) and (15) to determine the rest of the p parameters. The decomposition equations (14) and (15) represent a system of $2K'$ independent equations in $2K'$ unknowns. We present an algorithm, based on the DDX algorithm (Dallery, David, and Xie 1988) for a line, in Section 3.

2.3.4 Additional comments

Conditions on parameters

- We must require that for each machine M_i,

$$\sum_j p_{ij} < 1 \tag{16}$$

  This is because Tolio, Gershwin, and Matta (2002) have that requirement for both machines of their two-machine line. In fact, to satisfy those requirements, we must actually satisfy the stronger condition

$$\sum_{j,k} p^u_{kj}(i) \quad < \quad 1 \tag{17}$$

$$\sum_{j,k} p^d_{kj}(i) \quad < \quad 1 \tag{18}$$

  where the sums are taken over all the failure modes of all the machines in the range of starvation and the range of blocking, respectively. However, we cannot specify this at the initiation of the algorithm.

- To use the method of Tolio, Gershwin, and Matta (2002) for the two-machine line, we must have a minimal buffer size of 3. Werner (2001) solved the two-machine line by iterated matrix multiplication, so he was able to have a minimal buffer size of 2.

- The method will not work well when the population is very small and the size of the loop is very large. Consider a loop made up of 100 perfectly reliable machines where the population $N^p = 3$. Assume all 100 buffer sizes are larger than 3. The production rate of this system will be 3/100 because it takes 100 time steps for each part to circumnavigate the loop. On the other hand, our algorithm will calculate a production rate of 1.

  The reason for this discrepancy is that we do not consider the single time unit that a part spends in a machine for its operation. When this time is large compared to downtime due to failures, the algorithm is inaccurate.

Non-equality of production rates Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003) observed that when the algorithm converged, the production rates were not exactly equal. We observe the same. This is undesirable, but the errors never appear to be large. We have not determined the cause of this discrepancy. In our experience, the maximum $E(i)$ has proved to be the most accurate estimate of production rate when compared with simulation.

No specification of inventory sum It is noteworthy that we do not have an explicit requirement in the method that specifies the sum of the buffer levels. This is in contrast with the method of Frein, Commault, and Dallery (1996), which explicitly requires that

$$\sum_i \bar{n}(i) = N^p$$

3 Implementing the Loop Transformation and Decomposition

This section provides a step-by-step procedure for evaluating a loop. In Section 2.2 we transform an arbitrary loop into one that we can analyze. In Section 3.2 we describe an algorithm for solving the decomposition equations which is a slight modification of that of Tolio and Matta (1998).

3.1 The Transformation Algorithm

1. For all $N_i > N^p$, set $N_i = N^p$. (See Section 2.2.3.)

2. Insert a perfectly reliable machine M_{i*} for every unreliable machine M_i such that $\Psi(i*, i) = N^p$ (unless a machine M_j such that $\Psi(j, i) = N^p$ already exists). The new loop consists of K' machines separated by K' buffers (where $K \leq K' \leq 2K$).

3. Re-number the machines and buffers from 1 to K'.

4. Calculate the range of starvation and range of blocking for each M_i using (4) and (5).

5. For all N_i less than the minimal buffer size, set N_i to the minimal buffer size.

Note that some loops will not be changed at all by this transformation.

3.2 The Decomposition Algorithm

This algorithm is designed for loops that are the result of the transformation of Section 3.1.

1. For each line $L(i)$, the set of failure modes of $M^u(i)$ (ie, the set of $\lambda^u_{kj}(i)$) is the set of failure modes in the range of starvation of M_{i+1}. The set of failure modes of $M^d(i)$ (ie, the set of $\lambda^d_{kj}(i)$) is the set of failure modes in the range of blocking of M_i.

2. Initialize $p^u_{ij}(i)$, $r^u_{ij}(i)$, $p^d_{i+1,j}(i)$, $r^d_{i+1,j}(i)$, $r^u_{kj}(i)$, and $r^d_{kj}(i)$ for all valid failure modes using equations (6)–(11). Set $p^u_{kj}(i) = p_{kj}$ and $p^d_{kj}(i) = p_{kj}$.

3. For $i = 1$ to K':
   – Calculate $E(i)$ and $P^{st}_{kj}(i)$.
   – Update $p^u_{kj}(i+1)$ using (14).

4. For $i = K'$ to 1:
   – Calculate $E(i)$ and $P^{bl}_{kj}(i)$.
   – Update $p^{d}_{kj}(i-1)$ using (15).

5. Repeat (c) and (d) until the parameters converge to an acceptable tolerance.

6. Record performance measures.

   - Use $\max_i E(i)$ as an estimate of production rate E.
   - Calculate the average buffer level $\overline{n}(i)$. Form the appropriate sums of the average levels of the buffers created in the transformation of Section 3.1 to obtain the average levels of the buffer of the original loop.

4 Performance of the Method

In this section, we examine the accuracy, convergence reliability, and speed of the method. The method was tested extensively on three- to ten-machine loops with machine parameters and buffer sizes generated randomly. Repair probabilities for each machine in the loop where drawn from a uniform distribution between 0.05 and 0.2 and were required to be on the same order of magnitude. Failure probabilities were randomly generated from a uniform distribution such that the isolated efficiency $r/(r+p)$ of each machine in the loop was between 75 and 99 percent. Buffer sizes were drawn from a uniform distribution between $1/5r$ and $5/r$. For each loop, the decomposition and simulation were performed for all possible population levels. Here, we summarize the accuracy and convergence reliability. Details can be found in Werner (2001).

4.1 Accuracy

In this section, we compare the analytical results to those obtained through simulation. The measures of interest are average throughput and average buffer levels.

The production rate errors are calculated according to

$$\epsilon_E = \frac{E_{\textbf{decomposition}} - E_{\textbf{simulation}}}{E_{\textbf{simulation}}}$$

Buffer level errors were calculated according to

$$\epsilon_{\bar{n}(i)} = 2\left|\frac{\bar{n}(i)_{\textbf{decomposition}} - \bar{n}(i)_{\textbf{simulation}}}{N_i}\right|$$

We calculate buffer level errors in this way so as not to magnify the errors in small buffer levels while, at the same time, reducing the apparent errors in large buffer levels. Consider the reported error in the buffer level for a buffer of size 100 where the decomposition estimates the level to be 89.9 and the simulation calculates 90.0. Using the usual calculation of error (as we do for the

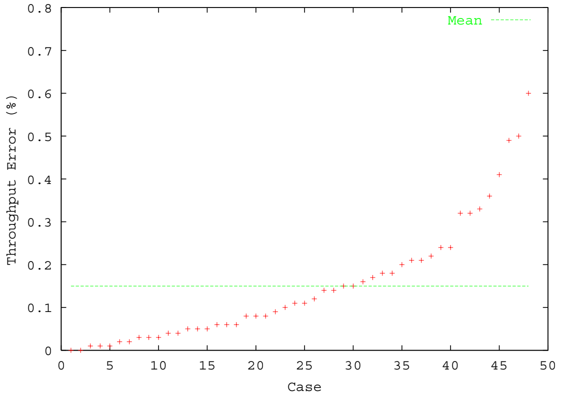


Figure 10: Throughput error (three-machine loops)

production rate), the error would appear to be about -0.1%. However, if we reversed the direction of flow in the system, we would have an equivalent system with the same production rate but with *complementary* buffer levels (Ammar 1980; Ammar and Gershwin 1989; Gershwin 1994). In that case, the decomposition would calculate a buffer level of 1.1 and the simulation would calculate 1.0. The apparent error would then be 10%. This is unreasonable, since the two systems are equivalent and the decomposition calculations are exactly the same. Using the formula above, we find $\epsilon_{\bar{n}} = .002$ for both cases.

4.1.1 Three-machine Loops

The method was observed to be accurate in evaluating three-machine loops. In all cases where the population was within the specified range, the relative throughput error is less than 1%. See Figure 10[7].

The mean for buffer level error for the 48 three-machine cases is 2.9%, but the error reaches as high as 9.58%. See Figure 11.

4.1.2 Six-machine Loops

For six-machine loops, the mean throughput error increased to 1.1% with a maximum of 2.7% for the 287 cases tested. The mean buffer level error was 5.04%. Although the maximum error is 21.0%, 87.3% of the cases have buffer level errors of less than 10.0%. Figures 12, 13, and 14 give a graphical representation of the results.

[7] In all of the error graphs, cases are sorted in order of ascending error.

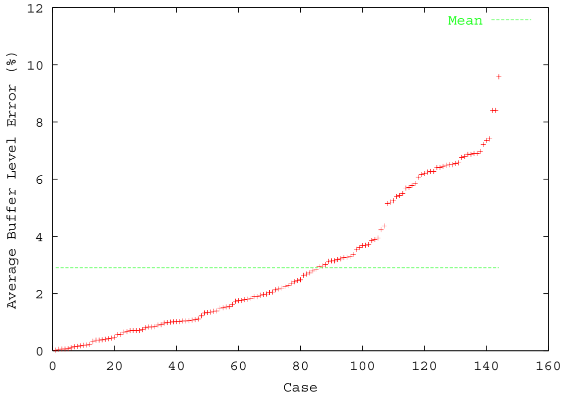


Figure 11: Average buffer level error (three-machine loops)

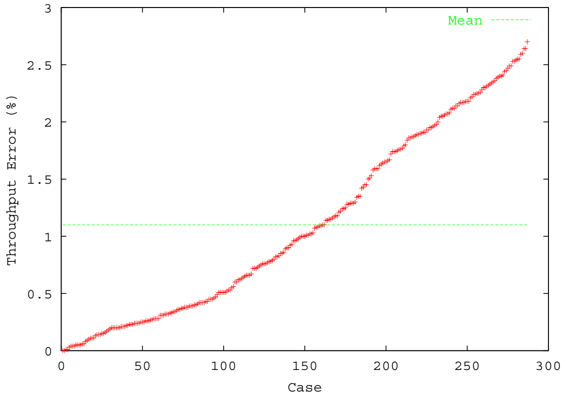


Figure 12: Throughput error (six-machine loops)

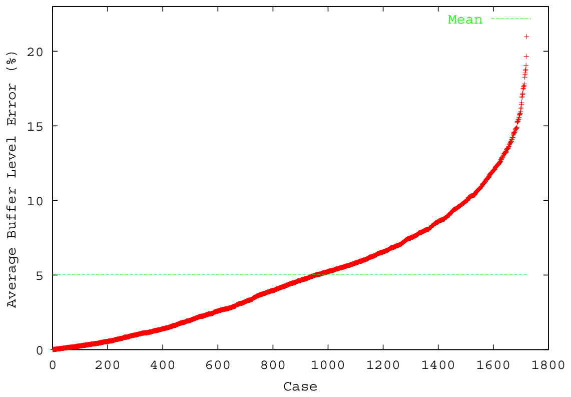


Figure 13: Average buffer level error (six-machine loops)

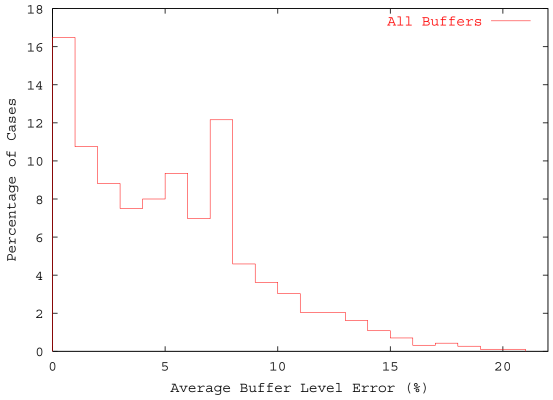


Figure 14: Distribution of average buffer level error (six-machine loops)

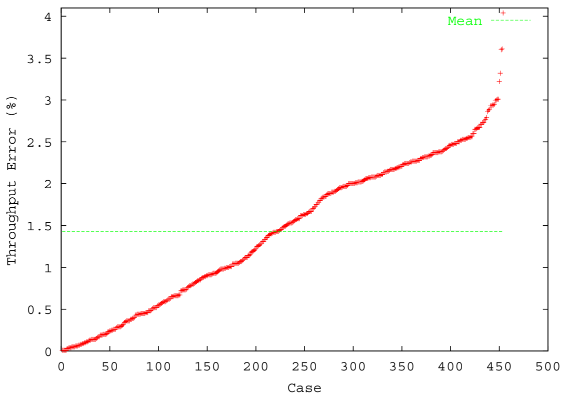


Figure 15: Throughput error (ten-machine loops)

4.1.3 Ten-machine Loops

In the ten-machine loops studied, the errors are slightly larger. For the 454 cases tested, the mean throughput error is 1.43% and the maximum is 4.04% (see Figure 15). Average buffer level errors range from 0.0% to 43.8% with a mean of 6.27%. 80.2% of the mean buffer errors are less than 10.0%.

4.2 Convergence Reliability

In nearly all cases studied, the decomposition algorithm converged. The criterion used for convergence was that difference in the value of all p^u and p^d between successive iterations be less than the specified tolerance of 10^{-6}. The few cases where the algorithm did not converge were ten-machine loops that exceeded the maximum number of iterations before satisfying the convergence criterion. Even though the algorithm did not converge to the tolerance, the errors in throughput and buffer levels are still very small.

As in Maggio (2000) and Maggio, Matta, Gershwin, and Tolio (2003), our decomposition al-

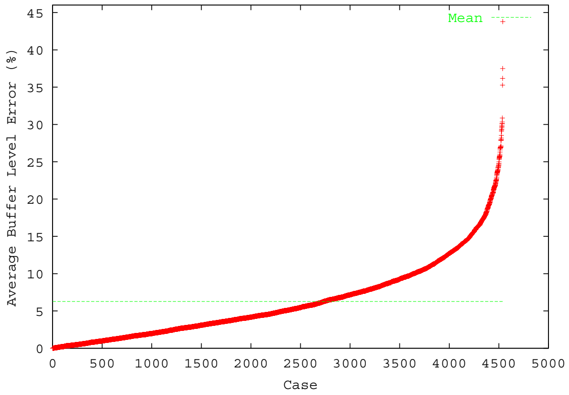


Figure 16: Average buffer level error (ten-machine loops)

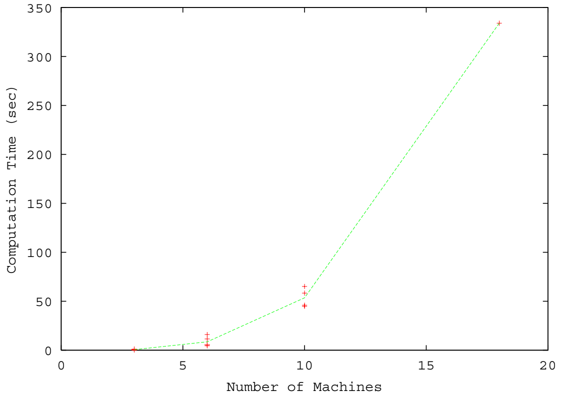


Figure 17: Computation time versus loop size

gorithm does not exactly satisfy conservation of flow even though Tolio's equations imply that it should hold. However, the differences between the throughputs of the building blocks are generally very small in the cases we examined.

4.3 Speed

The speed of the algorithm was tested by finding the computation time for five randomly generated loops of size three, six, and ten. Cases were run on a 650 MHz Pentium III PC with 128 MB of RAM. The mean run times were 0.61, 8.6, and 53.33 seconds for three-, six-, and ten-machine loops, respectively. Figure 17 shows the results. From the graph, we see that computation time increases rapidly as the number of machines in the loop increases.

In addition, one case of an 18-machine loop was tested. This was the largest loop evaluated. Each of the machines had $r = 0.1$ and $p = 0.01$. All of the buffers were size 10. The case was run with 100 parts in the system and required 334 seconds of computation time.

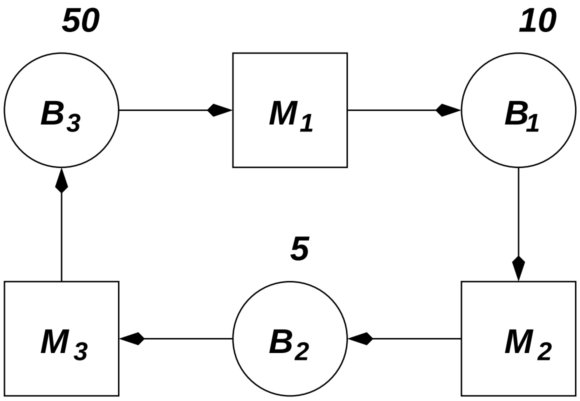


Figure 18: Example of loop with transfer line flatness

5 Observations on Loop Behavior

Consider a three-machine loop with buffers of size 10, 5, and 50 (Figure 18). The production rate and average buffer levels are shown in Figures 19 and 20. Note the symmetry and flatness of Figure 19.

In this section we discuss some of the loop phenomenon we observed while developing and testing the method.

5.1 Flatness

The most interesting observation we have made using the algorithm has to do with the relationship between the loop parameters, population, and throughput. Specifically, we observed a characteristic we call *flatness*, which refers to the shape of the throughput versus population curves of closed-loop systems.

5.1.1 Transfer Line Flatness

This special type of flatness occurs in loops where the capacity of the largest buffer is greater than the sum of the capacities of the other buffers. For all population levels N^p such that $N^{\mathbf{total}} - N^{max} < N^p < N^{max}$, the throughput is constant.

To illustrate the concept of transfer line flatness, we consider a three-machine loop with buffers of size 10, 5, and 50 (see Figure 18). When there are 16 parts in the system, it is possible for buffers B_1 and B_2 to be both full and empty. However, B_3 can never become full or empty. This means that machine M_1 can never be starved and M_3 can never be blocked. If we ignore B_3, the system has the same production rate and average buffer levels as a transfer line consisting of M_1, B_1, M_2, B_2, and M_3. This behavior remains the same for populations up to 50 because in each of these cases M_1 is never starved and M_3 is never blocked. In this population range, the average throughput and average buffer levels of B_1 and B_2 remain the same (Figures 19 and 20). Only the average buffer level of B_3 varies with the population.

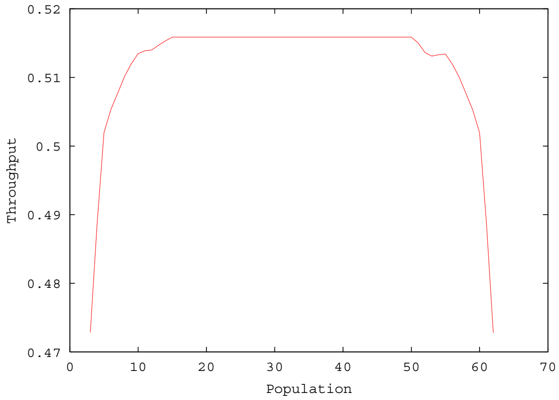


Figure 19: Analytical throughput as a function of population

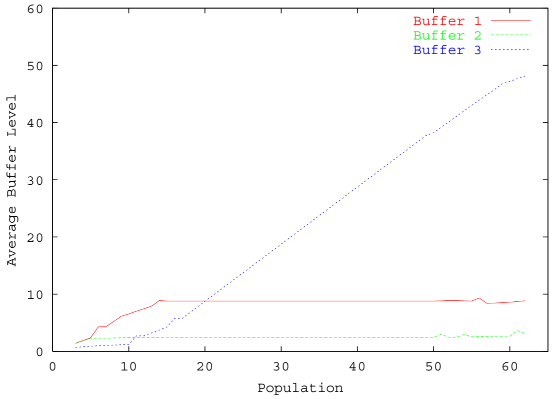


Figure 20: Analytical average buffer level as a function of population

5.1.2 Near Flatness and Non-Flatness

We also observed a phenomenon that we call *near flatness*. It occurs in loops that do not meet the requirements for transfer line flatness, but have population ranges where the throughput is nearly constant.

In the cases we studied, *symmetrical* loops, in which the machines are identical and the buffer capacities are the same, did not seem to exhibit near flatness. Loops which were very asymmetrical did exhibit near flatness. The degree of flatness seemed to increase with the degree of asymmetry in the loop.

Specifically, the standard deviation in the buffer capacities $\sigma_{\mathbf{buffers}}$ seems to give a good indication of how flat the throughput versus population curve will be for a given loop. We calculate $\sigma_{\mathbf{buffers}}$ for a K-machine loop as follows:

$$\sigma_{\mathbf{buffers}} = \sqrt{\frac{K \sum_i N_i - (\sum_i N_i)^2}{K^2}} \tag{19}$$

Figures[8] 21, 22, and 23 illustrate how the degree of flatness increases as $\sigma_{\mathbf{buffers}}$ increases from 7.97 to 19.12.

5.2 Sensitivity to Parameters

In this section, we explore how changes in machine parameters and buffer sizes affect average throughput and buffer levels. The basic loop used for these tests is the symmetrical three-machine loop with $r = 0.1$, $p = 0.01$ and $N = 10$ shown in Figure 24. For all of the tests, the loop population is held constant at $N^p = 15$.

5.2.1 Machine Parameters

In this section, we examine the effect of varying one or all the repair probabilities on the performance of the loop.

Only r_1 varied First, we examine the effect of varying r_1 between 0.0 and 1.0. Figures 25 and 26 give average throughput E and average buffer levels $\overline{n}(i)$ as a function of r_1. As r_1 approaches 0.0, throughput goes to 0.0. In addition, we see that $\overline{n}(1)$ approaches 0.0, $\overline{n}(2)$ approaches 5.0, and $\overline{n}(3)$ approaches 10.0. This result is consistent with intuition. When machine M_1 fails, parts begin to build up buffer B_3 until it reaches its capacity of ten. This causes M_3 to become blocked and parts begin to build up in B_2 until all of the five remaining parts are in B_2. Since $r_1 = 0.0$, the system can never leave this state and throughput is zero.

As r_1 increases, M_1 spends less time down. M_2 is starved less frequently and M_3 is blocked less frequently. This translates into an increase in throughput and $\overline{n}(1)$ and a decrease in $\overline{n}(3)$.

When $r_1 = 0.1$, the loop is symmetrical and all of the average buffer levels are equal to 5.0 since the 15 parts are distributed evenly between the three buffers.

[8] The parameters for these loops can be found in Werner (2001).

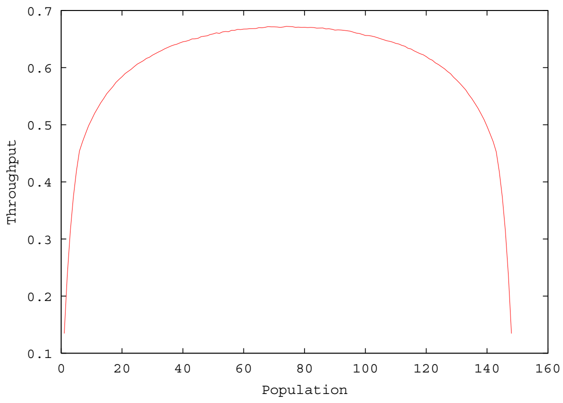


Figure 21: $\sigma_{\mathbf{buffers}} = 7.97$.

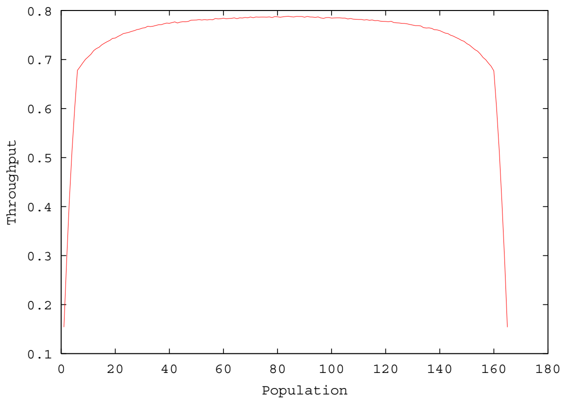


Figure 22: $\sigma_{\mathbf{buffers}} = 11.80$.

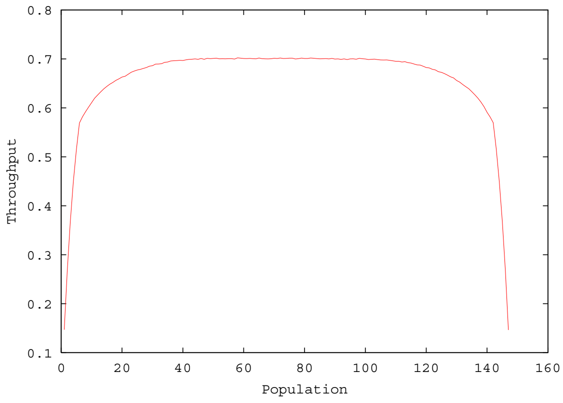


Figure 23: $\sigma_{\mathbf{buffers}} = 19.12$.

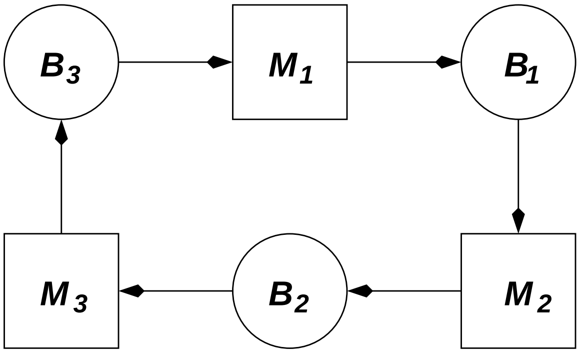
As r_1 approaches 1.0, the probability that M_2 will be starved approaches 0.0, as does the probability that M_3 will be blocked. By performing what is essentially the inverse of our loop transformation, we can view the system as a two-machine loop where M_1^{new} represents M_2, B_1^{new} represents B_2, M_2^{new} represents M_3, and B_2^{new} represents B_3, M_1, B_1. Since $N_1^{new} = 10$, $N_2^{new} = 20$, and $N^p = 15$, the system acts essentially like a two-machine transfer line made up of the original M_2, B_2, and M_3. When we compare the throughput of the loop with that of the line, we find that they are nearly identical. As r_1 approaches 1.0, the average throughput of the loop approaches 0.8535. The average throughput of the corresponding two-machine transfer line is 0.8561.

All r_i varied together Next, we study the effect of varying all of the rs together between 0.0 and 1.0. Since the loop is symmetrical in all cases here, the average buffer levels remain unchanged (i.e. $\overline{n}(i) = 5.0$). However, it is interesting to look at average throughput as a function of both r and the isolated efficiency $e = \frac{r}{r+p}$ of the machines. Figures 27 and 28 illustrate the relationship. We observe that throughput is equal to 0.0 when $r = 0$ and approaches 1.0 asymptotically as r increases to 1.0. In addition, we see that throughput increases hyper-linearly as a function of isolated efficiency.

5.2.2 Buffer Size

Here, we consider the effect that changing the buffer sizes has on average throughput. To do this, we use our standard symmetrical three-machine loop but set $N^p = 28$. Figure 29 shows how throughput changes as we vary the buffer size N between 10 and 35.

When $N = 10$, the probability of blocking P^{bl} is very high, causing throughput to be relatively low. At this point, the probability of starvation P^{st} is zero because the number of holes is less than N. As N increases to 14, P^{bl} decreases but P^{st} remains zero, resulting in an increase in throughput. For $N > 13$, P^{st} is no longer zero. In the range $13 < N < 28$, the decrease in P^{bl} is greater than the increase in P^{st} so there is a net increase in throughput. However, for $N \geq 28$, $P^{bl} = 0.0$, P^{st} is constant, and throughput is constant. All of the parts can fit in any of the buffers so no machine can ever become blocked. Furthermore, increasing the buffer size beyond N^p does not increase the probability that one of the buffers can become empty.



Basic Parameters			
r_i	p_i	N_i	N^p
0.1	0.01	10	15

Figure 24: Basic loop

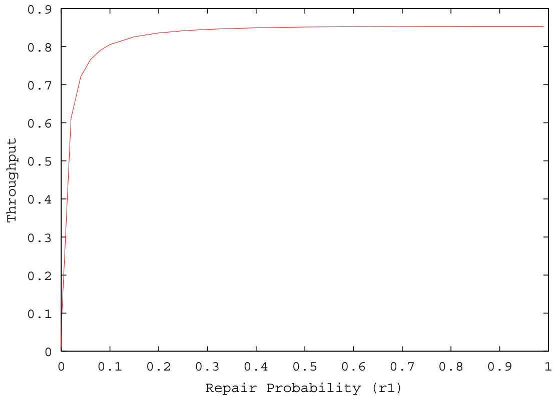


Figure 25: Average Throughput as a Function of r_1

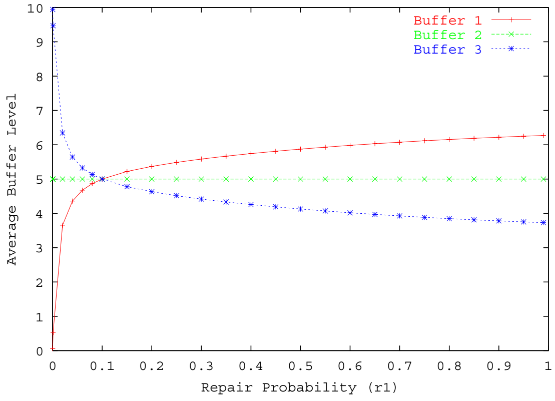


Figure 26: Average Buffer Level as a Function of r_1

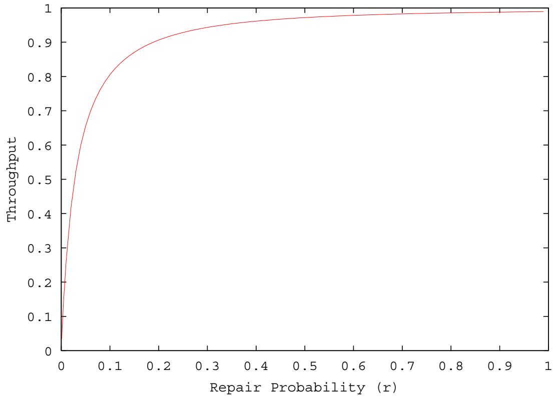


Figure 27: Average Throughput as a Function of r

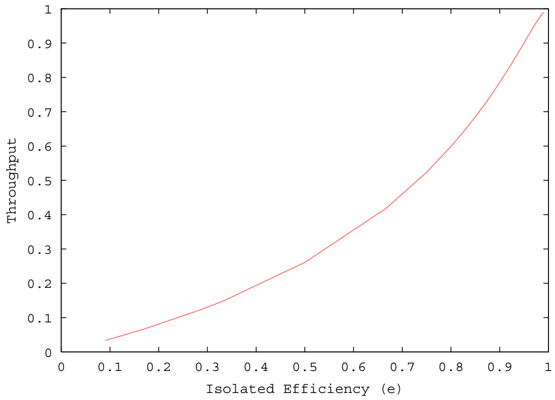


Figure 28: Average Throughput as a Function of e

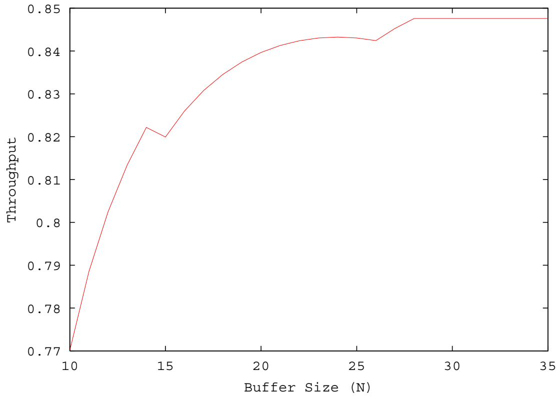


Figure 29: Analytical Throughput as a Function of Buffer Size

6 Conclusions and Future Work

The purpose of this research was to build on earlier work (Maggio 2000; Maggio, Matta, Gershwin, and Tolio 2003) to find a practical general approach to evaluating closed-loop systems. Our transformation algorithm significantly reduces the complexity of large loops by eliminating multiple thresholds. The transformation and decomposition technique described in this paper provide extremely accurate approximations of average production rate.

There are several extensions to the method which would prove useful:

1. The approach described here could be extended to multiple loop systems. (See Levantesi (2001).) This is of particular interest for evaluating the performance of systems operated under token-based control policies (Gershwin 2000).

2. The method could also be modified to deal with closed-loop systems in which multiple part types share a common set of resources. In this type of system, different part types compete for resources and therefore the production of one part interferes with the production of another.

3. Another possibility is the combination of the first two items. The method can be extended to evaluate multiple loops with multiple part types.

4. The method should be extended to other models of production loops, including exponential processing time and continuous material models (Gershwin 1994).

5. Improvements to the performance of the algorithm are possible. One would be to use the method of Tolio, Gershwin, and Matta (2002) for the two-machine lines early in the algorithm, and iterated matrix multiplication (which Werner (2001) did, but for the whole algorithm) in the late stages. This would reduce the computational effort if the initial guess for each two-machine line evaluation was the probability distribution calculated the last time the algorithm evaluated that two-machine line.

Acknowledgments

We are grateful for research support from the Singapore-MIT Alliance, the Lean Aerospace Initiative, and the National Science Foundation. We are grateful for the suggestions of Raniero Levantesi, Youichi Nonaka, and Zhenyu (James) Zhang.

References

Akyildiz, I. (1988). On the exact and approximate throughput analysis of closed queueing networks with blocking. *IEEE Trans on Soft Eng vol. 14*, 62–70.

Ammar, M. H. (1980). Modeling and analysis of unreliable manufacturing assembly networks with finite storages. Master's thesis, Massachusetts Institute of Technology.

Ammar, M. H. and S. B. Gershwin (1989). Equivalence relations in queuing models of fork/join networks. *Performance Evaluation Journal 10*, 233–245. Special Issue on Queueing Networks with Finite Capacities.

Balsamo, S., V. de Nitto Personé, and R. Onvural (2001). *Analysis of Queuing Networks with Blocking*. Kluwer Academic Publishers.

Bouhchouch, A., Y. Frein, and Y. Dallery (1992). Analysis of a closed loop manufacturing system with finite buffers. *CARs and FOF. 8th International Conference on CAD/CAM, Robotics and Factories of the Future vol. 2*, 1695–1710.

Buzacott, J. A. (1967). Automatic transfer lines with buffer stocks. *International Journal of Production Research 5*(3), 183–200.

Dallery, Y., R. David, and X.-L. Xie (1988). An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. *IIE Transactions 20*(3), 280–283.

Dallery, Y. and S. B. Gershwin (1992, December). Manufacturing flow line systems: A review of models and analytical results. *Queueing Systems Theory and Applications, Special Issue on Queueing Models of Manufacturing Systems 12*, 3–94.

Frein, Y., C. Commault, and Y. Dallery (1996). Modeling and analysis of closed-loop production lines with unreliable machines and finite buffers. *IIE Transactions 28*, 545–554.

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

Gershwin, S. B. (1991, December). Assembly/disassembly systems: An efficient decomposition algorithm for tree-structured networks. *IIE Transactions 23*(4), 302–314.

Gershwin, S. B. (1994). *Manufacturing Systems Engineering*. Prentice-Hall. See `http://web.mit.edu/manuf-sys/www/gershwin.errata.html` for corrections.

Gershwin, S. B. (2000). Design and operation of manufacturing systems — the control-point policy. *IIE Transactions 32*(2), 93–103.

Gershwin, S. B. and R. Levantesi (2003). Analysis of general material flow networks with unreliable machines and finite buffers. MIT report in preparation.

Han, M.-S. and D.-J. Park (2002). Performance analysis and optimization of cyclic production lines. *IIE Transactions 34*(4), 411–422.

Levantesi, R. (2001). *Analysis of Multiple Loop Assembly/Disassembly Networks*. Ph. D. thesis, Politecnico di Milano.

Maggio, N. (2000). An analytical method for evaluating the performance of closed loop production lines with unreliable machines and finite buffer. Master's thesis, Politecnico di Milano.

Maggio, N., A. Matta, S. Gershwin, and T. Tolio (2000). An analytical method for the performance evaluation of closed loop production lines with multiple unreliable machines and finite buffers. will be published.

Maggio, N., A. Matta, S. B. Gershwin, and T. Tolio (2003). An approximate analytical method for evaluating the performance of closed loop flow systems with unreliable machines and finite buffers — Part I: Small loops. Unpublished manuscript.

Mascolo, M. D., R. David, and Y. Dallery (1991, December). Modeling and analysis of assembly systems with unreliable machines and finite buffers. *IIE Transactions 23*(4), 315–331.

Onvural, R. and H. Perros (1990). Throughput analysis of cyclic queueing networks with blocking. Technical report, CS Dept, North Carolina State University Raleigh.

Tolio, T. and S. B. Gershwin (1996). Deterministic two-machine lines with multiple failure modes. Technical report, Politicnico di Milano — Dipartimento di Meccanica.

Tolio, T. and S. B. Gershwin (1998, March). Throughput estimation in cyclic queueing networks with blocking. *Annals of Operations Research 79*, 207–229.

Tolio, T., S. B. Gershwin, and A. Matta (2002). Analysis of two-machine lines with multiple failure modes. *IIE Transactions 34*(1), 51–62.

Tolio, T. and A. Matta (1998). A method for performance evaluation of automated flow lines. *Annals of the CIRP 47/1*, 373–376.

Werner, L. (2001). Analysis and design of closed loop manufacturing systems. Master's thesis, MIT OR Center.