Approximate analysis of production systems operated by a CONWIP/finite buffer hybrid control policy*

Asbjoern M. Bonvik† Yves Dallery‡ Stanley B. Gershwin§

August 10, 1999

Abstract

We present a decomposition method for approximate performance analysis of tandem production systems that are controlled by a hybrid control policy, in which the finite buffers of a flow line are augmented by a CONWIP cell enforcing release control to the line. The method decomposes the system into a set of three different types of building blocks. One is the continuous material two-machine line described in the literature; the second is a special case of the first in which the buffer is infinite. The third type of building block is a synchronization stage that is described and analyzed in this paper. Using these building blocks, we represent both the flow of material and the flow of information in the system. We derive new decomposition equations to relate the parameters of the building blocks. An iterative algorithm is constructed to determine the values of the parameters and to estimate the performance measures of the system. It has been observed to be fast and accurate for a large class of systems.

1 Introduction

As "lean manufacturing" has become the minimum requirement for competitiveness in industry (Womack, Jones, and Roos 1990), there has been a surge of interest in control policies for manufacturing systems that combine high service levels with low inventories. This is highly desirable (although hard to achieve) even if a system is unreliable and its demand erratic. One practical control policy that has won wide application is *kanban* control. In its simplest form, kanban control reduces to each machine in the system having a finite output buffer, which the machine attempts to keep full (Berkley 1991). This policy can be implemented using kanbans (tickets in Japanese). Buffers are unlimited in size, but are assigned a limited number of kanbans. When a part moves from a buffer (Buffer B_i) to its downstream machine (Machine M_{i+1}), the buffer sends a kanban to its

*This work was supported in part by the Advanced Research Projects Agency under contract N00174-93-C-0035 and by the Norwegian Research Council under contract 100250/410.

†McKinsey & Co, Inc., Olav V's Gate 5, N-0120 Oslo, Norway

‡Laboratoire d'Informatique de Paris 6 (LIP6-CNRS), Université Pierre et Marie Curie, 4, place Jussieu, 75252 Paris Cedex 05, France

§Laboratory for Manufacturing and Productivity, MIT 35-331, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, gershwin@mit.edu, *to whom correspondence should be addressed.*

upstream machine (M_i). Machine M_i may do an operation if there is a part available in its upstream buffer (B_{i-1}), and if it has at least one kanban. When it does the operation, it sends the part and the kanban to its downstream buffer (B_i). We call this version of kanban the *finite buffer* policy.

Several other versions of kanban have been studied or implemented. In particular, another form of kanban control, sometime referred to as *minimal blocking* policy (Mitra and Mitrani 1990) has been widely analyzed. See Berkley (1992), Buzacott and Shantikumar (1993) and Liberopoulos and Dallery (1997) for a detailed discussion of the different type of kanban policies. Other widely studied control policies are *basestock* control (Kimball 1988) and *CONWIP* control (Spearman, Woodruff, and Hopp 1990). All of these policies control a repetitive make-to-stock manufacturing system to prevent excessive inventories from forming, and they can often be implemented by circulating a limited number of cards, or work orders, or kanbans on the shop floor. The differences among the policies are the different ways these cards are circulated.

A number of recent papers have addressed hybrid control policies that attempt to combine two or more of these policies to exploit their strengths while avoiding the weaknesses of any one policy. One very general framework is the *production authorization card* control (Buzacott and Shantikumar 1992), (Buzacott and Shantikumar 1993). All of the control policies mentioned above can be seen as special cases of this framework. Another generalization is the *extended kanban* control (Dallery and Liberopoulos 1995) that essentially combines basestock and kanban controls. In a recent simulation study by Bonvik, Couch, and Gershwin (1996), several control policies were compared for operating a production line, and a hybrid of CONWIP and finite buffer was found to give the best performance in terms of meeting the service level target with the least inventory. This is the control policy that we analyze in this paper.

These are described as *pull control policies* since the flow of material through the production system is based on actual demands for finished parts. An overview of pull control policies can be found in Liberopoulos and Dallery (1997)

The goal of this paper is to create an analytical (rather than simulation) technique for evaluating tandem systems controlled by the CONWIP/finite buffer hybrid policy. Such a technique would be much faster than simulation, in fact fast enough to do optimization (of both the physical part of the production system and its control system). Our approach is to develop a decomposition for a continuous material approximation of the system. Exact analytic solutions to queuing networks are in general not available for any but the simplest systems. Decomposition has been successful for fast, approximate analysis for a class of manufacturing systems. A starting point for the present work is the algorithm for synchronous manufacturing systems introduced by Gershwin (1987). This was improved by Dallery, David, and Xie (1988), and extended to a homogeneous (one in which all machines have the same operation time) continuous material model by Dallery, David, and Xie (1989). Di Mascolo, David, and Dallery (1991) extended this model to arbitrary homogeneous tree-structured production networks, and Frein, Commault, and Dallery (1996) studied closed production systems where a finite population of pallets limit the amount of material in the system at any instant. Burman (1995) extended the analysis of continuous homogeneous flow lines to inhomogeneous lines. The latter four papers all use an exact solution of a continuous material tandem line of two unreliable machines separated by a finite buffer as the building block. A description of the two-machine line and its solution can be found in Gershwin (1994) (Section 3.10).

There is a large literature on flow line models and their analysis. See Dallery and Gershwin (1992) for a comprehensive review and comparison. Few lines with other control disciplines than the one

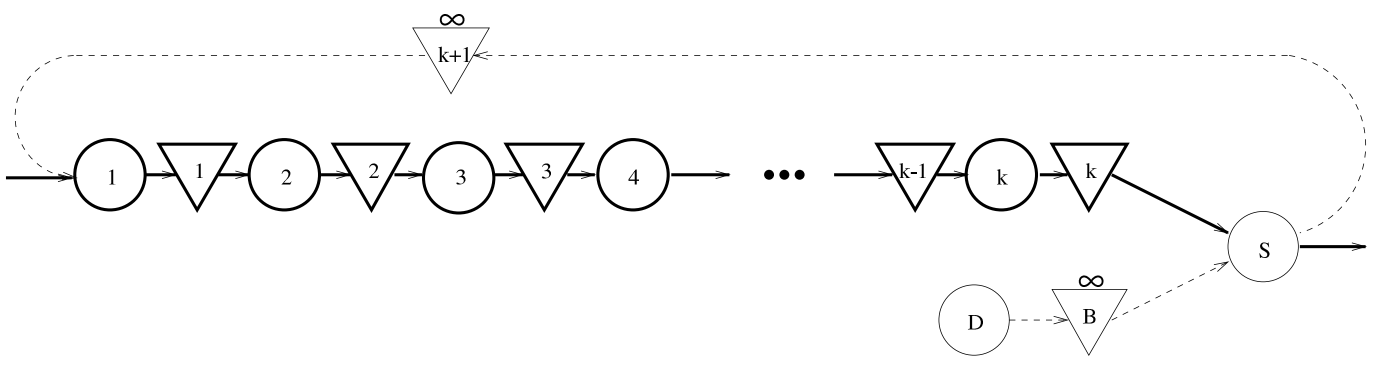

Figure 1: Representation of a k-machine system

imposed by finite buffers have been analyzed; notable exceptions are the papers by Mitra and Mitrani (1990, 1991) and Di Mascolo, Frein, and Dallery (1996) where a kanban control discipline is analyzed, and the paper by Lee and Zipkin (1992) that analyzes the basestock discipline.

We describe the manufacturing system, the control policy and the continuous flow model in Section 2. We present a decomposition method using the synchronization stage and the ordinary two-machine line solution as building blocks in Section 3. The fluid model has a synchronization stage that was not needed in the previous decomposition method, so we analyze this new stage in detail in Section 4. The computational algorithm is presented in Section 5. We show numerical results in Section 6 and discuss possible extensions and improvements in Section 7.

2 The production line

2.1 System

We are interested in analyzing the performance (predicting the production rate and the average in-process and finished goods inventory) of a tandem production line consisting of unreliable machines. Between each pair of machines there is a finite buffer to partially decouple the machines, and the last machine is followed by a finite buffer for finished products. An external demand process removes parts from this buffer. If no part is available when a demand arrives, the demand is entered in a backlog buffer. A k-machine system is depicted in Figure 1, in which the circles marked 1, 2, 3, ..., k represent machines, triangles 1, 2, ..., $k-1$ represent buffers for in-process inventory, triangle k represents a finished goods buffer, circle D represents the demand process, and triangle B is the backlog buffer. (Triangle $k+1$ is part of the control system, described in Section 2.2.) Circle S (*synchronization*) matches demand and production. We treat it like an assembly machine that is infinitely fast and perfectly reliable.

We assume that all machines have equal operation times (that is we restrict our attention to homogeneous lines), and, without loss of generality, we set all operation times to unity. Machine failures are operation dependent (Buzacott and Hanifin 1978), (Dallery and Gershwin 1992), and the failure and repair times are independent. We also assume that the demand is less than the capacity of the system, the rate at which it could produce parts if it were run as a flow line (in which each machine is allowed to operate whenever its upstream buffer is not empty and its downstream buffer is not full).

The first difference between this system and the flow lines previously studied in the decomposition literature is in the demand process and the backlog buffer. In the tandem line, the usual assumption is that the last machine is never blocked and the first machine never starved. In the system we study, the first machine is never starved, but the last machine is sometimes blocked. This occurs when it has caught up with demand and filled the finished goods buffer.

The second difference is the control policy and is described in the next section.

2.2 Control policy

The system, as described so far, uses blocking to match its production rate to demand. It is a kanban controlled system according to the description of Berkley (1991). While buffer 1 is full, no part can enter the system. When all the buffers are full, no part can enter the system until some time after another part leaves. During a machine failure, production may temporarily fall behind demand, but any excess demand is recorded in the backlog buffer and satisfied as soon as possible after the machine is repaired. As long as there is material in the backlog buffer, the system operates at capacity; when the backlog buffer is empty and the demand machine is down, no part can emerge from the system. During such periods, material enters the system (most often at a rate that exceeds the average demand rate) and fills up all available buffer spaces. As a consequence, the average production rate of the system is exactly equal to the demand rate. A further consequence is that the buffers are likely to be full or nearly full most of the time.

One way of improving the performance of the system (to reduce the in-process inventory while still meeting the same demand) is to wrap the entire line in a CONWIP cell, where there is a constraint on the maximal number of parts that can be in the system at any time. This requires two conditions to be satisfied before the first machine (Machine 1 in Figure 1) can load a part: The CONWIP limit N_C must not be reached, and the first buffer (Buffer 1) must not be full. It is helpful to have the finished goods buffer sufficiently large to contain all the inventory permitted in the system. This ensures that the last machine in the line will never be blocked without being simultaneously starved.

If the first machine were perfectly reliable, and the first buffer were infinite, the total amount of inventory in the system would be constant. In that case, N_C is the amount of inventory, not the upper bound on the amount of inventory.

This control policy can be implemented as a very simple modification of a kanban line: Instead of passing kanbans from the finished goods buffer to the last machine, they are passed to the *first* machine instead, and follow the parts back to the finished goods buffer. The other kanban cells operate as before. Buffer $k+1$ in Figure 1 holds the kanbans that are passed from the end of the line to the beginning.

A system operated by this policy, which we call a *hybrid control*, will be idle when all the inventory is in finished goods and all internal buffers are empty. This is different from a kanban system, which is idle when all buffers, both finished goods and internal, are filled. In a simulation study of a production line at a Toyota assembly factory, Bonvik, Couch, and Gershwin (1996) demonstrated that this hybrid policy achieves very high service level targets with significantly less inventory than the tandem buffer and minimal blocking models of kanban control.

The stability condition of this system is that the average demand rate must be less than the production rate of the closed loop system defined by machines $1, \ldots, k$, the finite buffers between these machines, the infinite buffer leading back to machine 1, and the total population of parts and kanban. Closed loop systems with finite buffers were studied by Frein, Commault, and Dallery (1996), and the single infinite buffer can be handled by a simple extension of their method.

2.3 Model

For analytical tractability, we represent the system as one in which material is a continuous fluid. The difference from an asynchronous model is that in the continuous case, the machines do not have operation times. Instead, they pass material through instantly at a constant rate when they are operational. The failure and repair processes turn the machine off and on, not unlike a randomly controlled faucet. This approximation is common in the analysis of queuing systems, and has been shown to be reasonably accurate for the analysis of asynchronous, unreliable manufacturing systems with deterministic operation times if the average times to failures and times to repair are significantly larger than the operation times, which is very often the case in real applications (Alvarez Vargas, Dallery, and David 1992).

We represent the demand process as another machine in the system (D in Figure 1). This machine produces demand that passes through a buffer (B) before it is assembled with equal amounts of finished products by a reliable and infinitely fast synchronization machine (S). This operation causes the product to depart the system and an equal amount of release authorizations to flow back to a buffer feeding the first machine. Note that this implies that buffer k and buffer B can never be non-empty simultaneously.

The solid arrows in the Figure 1 represent material flow, and the dashed arrows represent information flow. We model the demand process by choosing machine failure and repair parameters for machine D so that its average production rate equals the demand rate. The variability of the demand process can also be taken into account. Indeed, the failure and repair parameters can be chosen so as to match the first and second moments of the time between successive demands. Buffer B stores backlogged demand waiting for finished products, and buffer k holds authorizations for machine 1 to release new parts to the system. In this paper, we require these two buffers to be infinite.

We now need to introduce some notation. Let k be the number of machines in the line, not including the demand and synchronization machines. We define

Parameters

r_i to be the repair rate of machine i, for $i = 1, 2, \ldots, k, D$.

p_i to be the failure rate of machine i, for $i = 1, 2, \ldots, k, D$.

N_i to be the capacity of buffer i, for $i = 1, 2, \ldots, k$.

N_C to be the CONWIP limit, the maximal number of parts permitted into the system simultaneously.

States

$\alpha_i(\mathrm{t})$ to be the repair state of machine i, $\alpha_i(t) \in \{0, 1\}$, for $i = 1, 2, \ldots, k, D$.

$x_i(t)$ to be the level of buffer i, $x(t) \in [0, N_i]$, for $i = 1, 2, \ldots, k$.

$x_B(t)$ to be the amount of backlogged demand, $x_B(t) \geq 0$.

The repair state $\alpha_i = 0$ when a machine is down, and $\alpha_i = 1$ when it is functional. Since we have a processing rate of unity at each machine, the instantaneous production rate of machine i cannot exceed α_i.

It is convenient to define the *isolated efficiency* (which is, in this case, the *isolated production rate*) of machine i as

$$e_i = \frac{r_i}{r_i + p_i}.$$

3 The decomposition method

3.1 Overall approach

Decomposition analyzes a system like that in Figure 1 by replacing it with a set of simpler systems, as illustrated in Figure 2. The small systems, or components, are all analytically tractable. They have buffers which are the same as the corresponding buffers in the original system. Ideally, the machines (called *pseudo-machines*) have parameters which are chosen so that an observer in a buffer in the original system would see material arriving and departing in a way that is statistically indistinguishable from what he would see if he were in the corresponding buffer of the corresponding small system. It is not possible to realize this ideal, but, because of the excellent agreement between numerical and simulation results in many cases (Section 6.2), we believe that we achieve a good approximation.

In this figure, there are k two-machine lines and one three-machine assembly system. The two-machine lines are are identified with their buffers, and the upstream and downstream pseudo-machines of line i are called $u(i)$ and $d(i)$. In the three-machine line, the upstream machines are called $u(k)$ and $u(B)$, and the downstream machine is called $d(k)$. However, because buffer B is infinite and machine S is infinitely fast and perfectly reliable, machine $u(B)$ is identical to machine D in Figure 1, and machine $d(k)$ is identical to machine S. In the following, we will sometimes refer to each machine by either notation.

We use essentially the same decomposition approach as in Dallery, David, and Xie (1989). A new feature is that there are three kinds of components: tandem two-machine lines with a finite buffer (as in all previous decompositions), tandem two-machine lines with an infinite buffer, and the three-machine synchronization stage. In each component, we represent everything upstream of the corresponding buffer in the original system by the failure and repair process of the upstream pseudo-machine. We represent everything downstream of the corresponding buffer in the original system by the failure and repair process of the downstream pseudo-machine.

To perform the decomposition, we need two things: the analysis of the components, and a set of equations that relate the parameters of the components with one another and with the original system. This is already available for the entire system except the synchronization stage. For the analysis of the two-machine lines, we use Gershwin (1994) (Section 3.10). For the equations that relate the parameters of two-machine components, we use Dallery, David, and Xie (1989). For the CONWIP population constraint, we use the method of Frein, Commault, and Dallery (1996).

In this section, we derive the equations that describe how the synchronization stage affects its neighboring components, in the same way as was done for pure tandem lines in Dallery, David, and Xie (1989). We analyze the synchronization stage in Section 4.2.

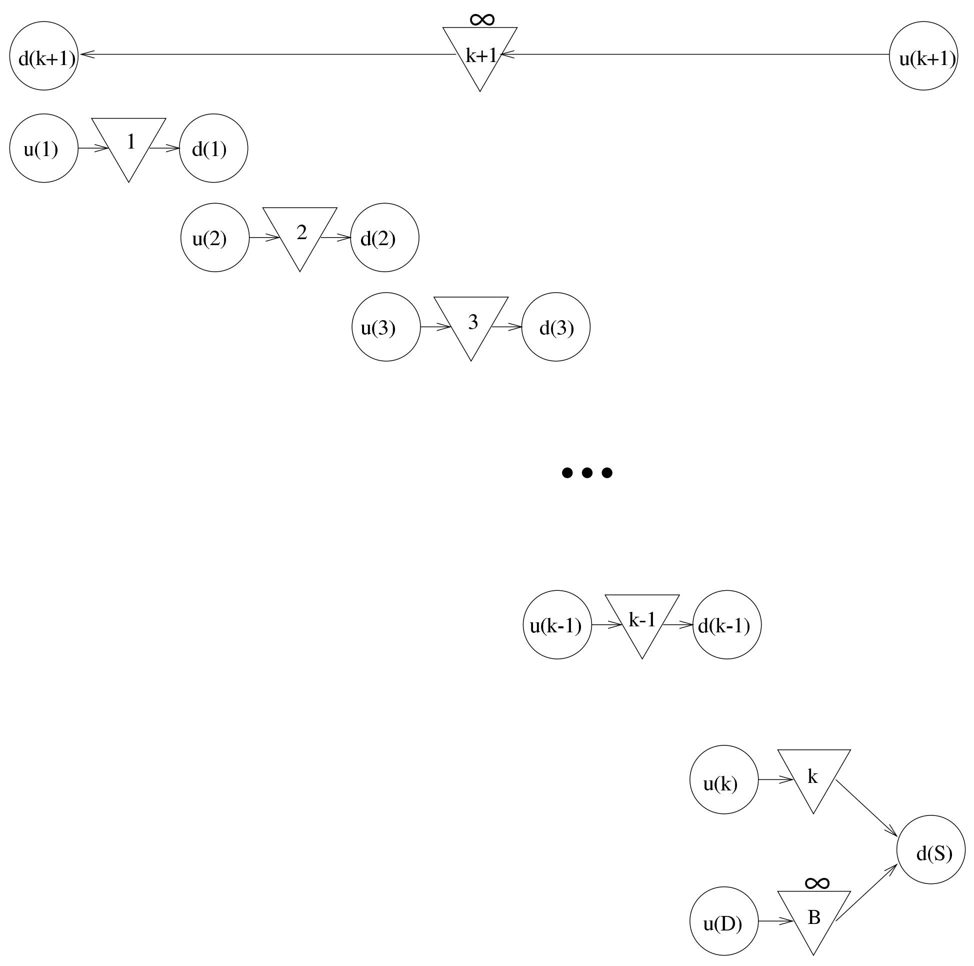

Figure 2: The system decomposed to elementary components

We need more notation. In the following, the parameters are quantities that must be determined from the relations among the components. (These are inputs to the synchronization stage analysis.) The single-machine quantities are simple functions of those parameters. The performance measures are the outputs of the synchronization stage analysis. We define

Parameters and single-machine quantities

$r_u(i)$ to be the repair rate of the upstream machine in component i.

$p_u(i)$ to be its failure rate.

$e_u(i)$ to be its isolated efficiency, $e_u(i) = r_u(i)/[r_u(i) + p_u(i)]$.

$I_u(i)$ to be defined as $I_u(i) = p_u(i)/r_u(i)$.

$r_d(i)$ to be the repair rate of the downstream machine in two-machine line i.

$p_d(i)$ to be its failure rate.

$e_d(i)$ to be its isolated efficiency, $e_d(i) = r_d(i)/[r_d(i) + p_d(i)]$.

$I_d(i)$ to be defined as $I_d(i) = p_d(i)/r_d(i)$.

Performance Measures

$E(i)$ to be the production rate of two-machine line i.

$\bar{x}(i)$ to be the average level of buffer i.

$p_s(i)$ to be the starvation probability of the downstream machine in component i.

$p_b(i)$ to be the blocking probability of the upstream machine in component i.

The unknown quantities to be determined are $r_u(i)$, $p_u(i)$, $r_d(i)$, and $p_d(i)$ for all components i, except for the synchronization stage k, where only $r_u(k)$ and $p_u(k)$ must be determined. Note that the upstream machine(s) in each individual component, whether it is a two-machine line or the synchronization stage, is never starved, and the downstream machine never blocked.

The usual approach to developing decomposition equations is to define what it means for an observer to believe that a pseudo-machine is up or down, then to define what really happens when an observer thinks he sees a failure or a repair, and then to use these definitions and other statements to develop equations for the failure and repair rate parameters listed above. Equations based on the definitions of $r_u(i)$ or $r_d(i)$ are called upstream or downstream *resumption of flow* equations. Other relationships that are used include *conservation of flow* and *the flow rate–idle time relationship.* In the following, we discuss in detail only those equations that are new because of the new network structure analyzed here.

3.2 Decomposition equations

Conservation of Flow

Conservation of flow asserts that the flow rate through all the components are the same, or

$$E(1) = E(2) = \ldots = E(k-1) = E(k).$$

Downstream Resumption of Flow in Line $k-1$

Suppose an observer in buffer $k-1$ of the line sees no flow out of the buffer. This can be caused by a failure of machine k or by a blockage of machine k. A blockage occurs when the buffer downstream of a machine is full, and there is no flow out of that buffer. The machine is then forced to stop processing. If machine k is failed, it will be repaired at rate r_k. On the other hand, if it is blocked, the blockage can only be caused by a failure of the demand process D. This is because buffer $k+1$ is infinite, so the synchronization machine S is never blocked. The synchronization machine is also perfectly reliable, so it can never fail. The only remaining way that machine k can be blocked is if buffer k is full, buffer D is empty and the demand process D is failed. The blockage will therefore be cleared at rate r_D.

Consider the time from flow out of buffer $k-1$ is interrupted until it resumes. In the original line, this interruption will either be cleared at rate r_k if it is due to a failure of machine k, or at rate r_D if it is due to a blockage. In the decomposition, we approximate this time to resumption of flow by the exponentially distributed repair process of the downstream machine in two-machine line $k-1$, which has repair rate $r_d(k-1)$. Let $X_d(k-1)$ be the fraction of all interruptions of flow out of buffer $k-1$ that are caused by blockages. Then the mean time to repair pseudo-machine $d(k-1)$ is

$$\frac{1}{r_d(k-1)} = X_d(k-1)\frac{1}{r_D} + [1 - X_d(k-1)]\frac{1}{r_k} \tag{1}$$

This requires the mean time to repair of the downstream machine in the decomposition to match the average duration of an interruption of flow out of buffer $k-1$ in the original system. The fraction $X_d(k-1)$ is

$$X_d(k-1) = \frac{\bar{N}_{bk}}{\bar{N}_{fk}}$$

where $\bar{N}_{bk}$ is the average number of blockages of machine k occurring per time unit, and $\bar{N}_{fk}$ the average number of flow interruptions at machine k per time unit. In the decomposition, the blockages are represented in the synchronization stage k and the flow disruptions out of buffer $k-1$ in two-machine line $k-1$. We can therefore write

$$X_d(k-1) = \frac{\bar{N}_b(k)}{\bar{N}_{fd}(k-1)}$$

where $\bar{N}_b(k)$ is the average number of blockages of the upstream machine in the synchronization stage per unit of time, and $\bar{N}_{fd}(k-1)$ is the average number of interruptions of flow out of buffer $k-1$ per unit of time. Since the average duration of a blockage in the synchronization stage k is $1/r_D$, the fraction of time in which the machine is blocked must be

$$p_b(k) = \bar{N}_b(k)\frac{1}{r_D}$$

so

$$\bar{N}_b(k) = p_b(k)\, r_D$$

Similarly

$$\bar{N}_{fd}(k-1) = p_{fd}(k-1)\, r_d(k-1)$$

where $p_{fd}(k-1)$ is the fraction of time the downstream machine in two-machine line $k-1$ is down.

Since the machines will be processing at rate 1.0 whenever possible, the production rate $E(k-1)$ is the fraction of time the downstream machine in line $k-1$ is working (operational and not starved). Failures are operation dependent, so this is also the fraction of time it is exposed to failures, which occur at rate $p_d(k-1)$. The average number of failures per unit of time is then $p_d(k-1)E(k-1)$. Similarly, the number of repairs per unit of time is $r_d(k-1)p_{fd}(k-1)$. The number of failures must equal the number of repairs, or

$$p_d(k-1)\, E(k-1) = r_d(k-1)\, p_{fd}(k-1)$$

Solving for $p_{fd}(k-1)$, we have

$$p_{fd}(k-1) = \frac{p_d(k-1)}{r_d(k-1)}E(k-1) = I_d(k-1)E(k-1) = I_d(k-1)E(k)$$

where the last equality is due to conservation of flow. Then

$$X_d(k-1) = \frac{p_b(k)\, r_D}{p_{fd}(k-1)\, r_d(k-1)} = \frac{p_b(k)\, r_D}{I_d(k-1)\, E(k)\, r_d(k-1)} = A_d(k-1)\frac{r_D}{r_d(k-1)}$$

where $A_d(k-1) = p_b(k)/(I_d(k-1)E(k))$. We can now rewrite equation (1) as

$$\frac{1}{r_d(k-1)} = A_d(k-1)\frac{1}{r_d(k-1)} + \left(1 - A_d(k-1)\frac{r_D}{r_d(k-1)}\right)\frac{1}{r_k}$$

which after a little algebra becomes

$$r_d(k-1) = A_d(k-1)\, r_D + [1 - A_d(k-1)]\, r_k \tag{2}$$

From the definition of $I_d(k-1)$, we have

$$p_d(k-1) = I_d(k-1)\, r_d(k-1) \tag{3}$$

Flow Rate–Idle Time at Machine M_k

We must now find $I_d(k-1)$ to complete the description of how the synchronization stage affects two-machine line $k-1$. There is a general relationship between the flow rate and idle time of any machine: the production rate equals the isolated efficiency times the fraction of time the machine is not starved or blocked (Gershwin 1994), or

$$E_i = e_i(1 - p_{si} - p_{bi}) \tag{4}$$

where p_{si} and p_{bi} are the probabilities of machine i being starved or blocked. In this context, (4) can be approximated as:

$$\begin{aligned} E(k-1) &= e_d(k-1)\,[1 - p_s(k-1)] && (5)\\ E(k) &= e_u(k)\,[1 - p_b(k)] && (6) \end{aligned}$$

Also,

$$E(k-1) = E(k) \tag{7}$$

where the last relationship is due to the conservation of flow in the system. We substitute $p_s(k-1)$ for p_{si}, $p_b(k)$ for p_{bi}, and $E(k)$ for E_i in equation (4), and use equations (5), (6), and (7), as well as the definitions of $e_u(k)$, $e_d(k-1)$, $I_u(k)$, and $I_d(k-1)$. After some algebra, we arrive at the equation

$$I_d(k-1) = \frac{1}{e_k} + \frac{1}{E(k)} - I_u(k) - 2 \tag{8}$$

This completes the description of the downstream machine in two-machine line $k-1$.

Upstream Resumption of Flow in Line $k+1$

Now suppose there is no flow entering buffer $k+1$ of the original system. The synchronization machine is perfectly reliable, so the interruption of flow can either be caused by a failure or starvation of machine k, or by a failure of the demand machine D.

We must be careful about what causes an interruption of flow into buffer $k+1$. It is possible to be in a state such that both machines k and D are down, which certainly interrupts the flow. We say that the interruption is due to machine k if the flow immediately resumes when that machine is repaired, and similarly, that the interruption is due to machine D if a repair of that machine immediately restores flow. This means that a failure can be invisible to machine S if there is a sufficient amount of finished goods (or backlogged demand) for the machine to recover, even if a failure of the other machine is starving S at the same time. Example: Suppose that the finished goods buffer is empty, there is some backlog, and both machines k and D are working. Suppose D fails. Machine k will continue working to service the backlog, maintaining flow into buffer $k+1$. Now suppose machine k also fails, interrupting the flow. If D is repaired first, it will not restore the flow, but merely cause more backlog to be accumulated. On the other hand, if machine k is repaired first, flow will resume immediately. We therefore say that this interruption is due to machine k.

An interruption of flow into buffer $k+1$ is represented by a failure of the upstream machine in two-machine line $k+1$ in the decomposition. This machine is repaired at rate $r_u(k+1)$. Let $X_u(k+1)$ be the fraction of interruptions that are caused by the upstream machine of the synchronization stage. Then

$$\frac{1}{r_u(k+1)} = X_u(k+1)\frac{1}{r_u(k)} + [1 - X_u(k+1)]\frac{1}{r_D} \tag{9}$$

Let $p_{su}(k)$ be the fraction of time the synchronization machine S in stage k is starved due to the upstream machine. By reasoning like that in the previous case, the fraction of all interruptions that are due to this machine are

$$X_u(k+1) = \frac{p_{su}(k)\, r_u(k)}{p_s(k)\, r_u(k+1)}$$

We can solve equation (9) for $r_u(k+1)$ and get

$$r_u(k+1) = A_u(k+1)r_u(k) + [1 - A_u(k+1)]r_D \tag{10}$$

where

$$A_u(k+1) = \frac{p_{su}(k)}{p_s(k)}$$

Again from the definitions, we have

$$p_u(k+1) = I_u(k+1)\, r_u(k+1) \tag{11}$$

Assuming that the system is stable, $E(k) = e_D$, since the demand machine is never blocked or starved, and the synchronization stage is never failed or blocked. Then $e_u(k+1) = e_D$, or

$$I_u(k+1) = I_D \tag{12}$$

which completes the decomposition equations for the synchronization stage.

Equations relating other components

The corresponding equations for the ordinary two-machine line components are derived in Dallery, David, and Xie (1989), and are

$$
\begin{aligned}
I_u(i) &= \frac{1}{e_i} + \frac{1}{E(i-1)} - I_d(i-1) - 2 && (13)\\
A_u(i) &= \frac{p_s(i-1)}{I_u(i)\,E(i-1)} && (14)\\
r_u(i) &= A_u(i)\,r_u(i-1) + [1 - A_u(i)]\,r_i && (15)\\
p_u(i) &= I_u(i)\,r_u(i) && (16)\\
I_d(i) &= \frac{1}{e_{i+1}} + \frac{1}{E(i+1)} - I_u(i+1) - 2 && (17)\\
A_d(i) &= \frac{p_b(i+1)}{I_d(i)\,E(i+1)} && (18)\\
r_d(i) &= A_d(i)\,r_d(i+1) + [1 - A_d(i)]\,r_{i+1} && (19)\\
p_d(i) &= I_d(i)\,r_d(i) && (20)
\end{aligned}
$$

Equations (13) through (16) hold for $i = 2, \ldots, k-1$, and equations (17) through (20) hold for $i = 1, \ldots, k-2$. As special cases, the upstream machine in line 1 is described similarly to equations (13) through (16) where $i-1$ is replaced by $k+1$; and the downstream machine in line $k+1$ is described by equations similar to (17) through (20) where $i+1$ is replaced by 1.

Equations (12), (10), (11), (8), (2), and (3) together with equations (13) through (20) give $4k-2$ equations in as many unknowns, but these equations are not independent. There are only $4k-3$ independent equations, since the conservation of flow equations (7) can be seen to introduce a dependency. Written out, these equations are

$$
\begin{aligned}
E(1) &= E(2)\\
E(2) &= E(3)\\
&\vdots\\
E(k) &= E(1)
\end{aligned}
$$

Adding these up, we have $E(1) = E(1)$, so we must therefore find one more equation.

CONWIP constraint

So far, we have not considered the CONWIP character of the control policy at all. This requires the sum of all buffer levels (except backlog) to equal the CONWIP limit at all times. This constraint cannot be expressed in the framework already created. We instead use a relaxed form of this equation that only requires the sum of the *average* buffer levels to equal the constraint, as proposed by Frein, Commault, and Dallery (1996). This equation can be written

$$N_C = \sum_{i=1}^{k+1} \bar{x}(i)$$

where $\bar{x}(i)$ is evaluated from each component model.

Before considering an algorithm to solve the resulting system of equations, we must look more closely at the individual components.

4 Analysis of the subsystems

As it appears, the decomposition method relies on the solution of the subsystems. Three types of subsystems are encountered in this decomposition method: two machine lines with a finite intermediate buffer (subsystems $1, \ldots, k-1$); two machine lines with an infinite intermediate buffer (subsystem $k+1$), and the synchronization station (subsystem k). In this section, we analyze the second and third types of subsystems. The solution of the first type, i.e. the classical two-machine one-finite-buffer subsystem, can be found in Dallery, David, and Xie (1989) and Gershwin (1994).

4.1 Two-Machine line with infinite buffer

Consider a two-machine line with an infinite intermediate buffer. Let r_u, p_u and r_d, p_d be the repair and failure rates of the upstream and downstream machines, respectively. The solution of this subsystem can easily be obtained from that of the two-machine one-finite-buffer subsystem by letting the capacity of the buffer go to infinity. The performance parameters can then simply be expressed as follows:

$$E = e_u \tag{21}$$

$$p_s = 1 - e_u/e_d \tag{22}$$

$$p_b = 0 \tag{23}$$

$$\bar{x} = e_u \frac{I_d}{I_u - I_d} \frac{r_u + r_d}{r_u r_d} \tag{24}$$

This of course requires that the system be stable. The stability condition is that the average production rate of the upstream machine in isolation is less than that of the downstream machine, which can be expressed as:

$$e_u = \frac{r_u}{r_u + p_u} < e_d = \frac{r_d}{r_d + p_d} \tag{25}$$

4.2 Synchronization stage solution

Consider the synchronization stage consisting of machine k, the demand process D, the synchronization machine S and the intermediate buffers. We now look at this three machine sub-system in isolation, as indicated in Figure 3.

When analyzing the synchronization stage in isolation, we refer to the finished goods buffer as buffer FG and the machine upstream of it as machine M. There are three key properties of this system:

- The synchronization machine S is infinitely fast and perfectly reliable.
- The synchronization machine S is never blocked.
- The demand process D and the machine M are never starved.

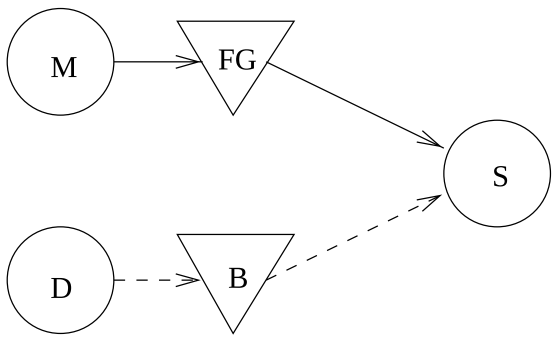

Figure 3: The synchronization stage in isolation

Because of these properties, x_{FG} and x_B cannot both be positive at the same time. Define the state variable $x = x_{FG} - x_B$, $x \in (-\infty, N_{FG}]$. This is the *production surplus* of the machine, which is used as a state variable in optimal control formulations for this system (Bielecki and Kumar 1988; Van Ryzin 1987). Notice that this variable captures the information contained in both x_{FG} and x_B, since $x > 0$ indicates that there is some amount of finished goods (and no backlog), while $x < 0$ indicates some backlog (and no finished goods). We can therefore represent the state of the synchronization stage by $s = (x, \alpha_M, \alpha_D)$.

The stability condition of the isolated synchronization stage is that the average demand rate is less than the average production rate, or

$$e_D = \frac{r_D}{r_D + p_D} < e_M = \frac{r_M}{r_M + p_M} \tag{26}$$

We now consider the state dynamics of this system. Consider some state $(x, 0, 1)$, where $x < N_{FG}$. The system can enter this state from the state $(x+\delta, 0, 1)$ if machine M is not repaired and the demand process D does not fail during an interval of length δ, from the state $(x, 1, 1)$ if machine M fails during the interval, or from the state $(x, 0, 0)$ if the demand process D resumes. The system can leave the state $(x, 0, 1)$ through either a repair of machine M at rate r_M or through a failure of the demand process D at rate p_D. Writing this out to the first order, we have the development of the density in time, which is

$$\begin{aligned} f(x,0,1,t+\delta) &= (1-(r_M+p_D)\delta)\, f(x+\delta,0,1,t) \\ &\quad + p_M\delta f(x,1,1,t) + r_D\delta f(x,0,0,t) + o(\delta) \end{aligned}$$

We now do a first-order Taylor expansion of the $f(x+\delta,0,1,t)$, and get

$$\begin{aligned} f(x,0,1,t+\delta) &= (1-(r_M+p_D)\delta)\left(f(x,0,1,t) + \delta\frac{\partial}{\partial x}f(x,0,1,t)\right) \\ &\quad + p_M\delta f(x,1,1,t) + r_D\delta f(x,0,0,t) + o(\delta) \end{aligned}$$

Rearranging terms, we have that

$$\begin{aligned} \frac{f(x,0,1,t+\delta) - f(x,0,1,t)}{\delta} &= -(r_M+p_D)\, f(x,0,1,t) + \frac{\partial}{\partial x}f(x,0,1,t) \\ &\quad + p_M f(x,1,1,t) + r_D f(x,0,0,t) + \frac{o(\delta)}{\delta} \end{aligned}$$

Letting δ approach zero, we get the partial differential equation

$$\begin{aligned}\frac{\partial}{\partial t}f(x,0,1,t) &= -(r_M+p_D)\,f(x,0,1,t)+\frac{\partial}{\partial x}f(x,0,1,t)\\ &\quad +p_M f(x,1,1,t)+r_D f(x,0,0,t)\end{aligned}$$

In steady state, the time derivative of the density is zero. Dropping the time argument, we have that

$$\frac{\partial}{\partial x}f(x,0,1)-(r_M+p_D)\,f(x,0,1)+p_M f(x,1,1)+r_D f(x,0,0)=0 \tag{27}$$

Similarly, we get the other three differential equations

$$-(r_M+r_D)f(x,0,0)+p_M f(x,1,0)+p_D f(x,0,1)=0 \tag{28}$$

$$-\frac{\partial}{\partial x}f(x,1,0)-(p_M+r_D)\,f(x,1,0)+r_M f(x,0,0)+p_D f(x,1,1)=0 \tag{29}$$

$$-(p_M+p_D)\,f(x,1,1)+r_M f(x,0,1)+r_D f(x,1,0)=0 \tag{30}$$

We now turn to the boundary conditions at $x=N_{FG}$. There are four point probabilities at this point, and we refer to them as $\mathbf{p}(N_{FG},\alpha_M,\alpha_D)$, where the α's can take on the same values as before. First, we note that

$$\mathbf{p}(N_{FG},0,1)=0 \tag{31}$$

because the buffer level x will change at rate -1 in this repair state, so the system will depart from this state immediately. Furthermore, in the state $(N_{FG},1,0)$, machine M cannot fail since its processing rate is zero and we assumed operation dependent failures. Then the state $(N_{FG},0,0)$ is not reachable from any other state, and

$$\mathbf{p}(N_{FG},0,0)=0 \tag{32}$$

Consider the state $(N_{FG},1,0)$. The system can persist in this state as long as the demand process D does not resume. It can also arrive at this state from $(N_{FG},1,1)$ by having a failure of the demand process, or from a state $(N_{FG}-\delta,1,0)$ if the demand process does not resume and machine M does not fail during a time interval of length δ. This point probability then develops in time as

$$\begin{aligned}\mathbf{p}(N_{FG},1,0,t+\delta) &= (1-r_D\delta)\,\mathbf{p}(N_{FG},1,0,t)+p_D\delta\,\mathbf{p}(N_{FG},1,1,t)\\ &\quad +\delta\, f(N_{FG}-\delta,1,0,t)+o(\delta)\end{aligned}$$

Doing the same manipulations as for the density f, we get that

$$\frac{\partial}{\partial t}\mathbf{p}(N_{FG},1,0,t)=-r_D\,\mathbf{p}(N_{FG},1,0,t)+p_D\,\mathbf{p}(N_{FG},1,1,t)+f(N_{FG},1,0,t)$$

In steady state, this yields the boundary condition

$$r_D\,\mathbf{p}(N_{FG},1,0)=p_D\,\mathbf{p}(N_{FG},1,1)+f(N_{FG},1,0) \tag{33}$$

Similarly, we can derive the other boundary condition

$$(p_M+p_D)\,\mathbf{p}(N_{FG},1,1)=r_D\,\mathbf{p}(N_{FG},1,0) \tag{34}$$

In addition, we have the normalization equation

$$\sum_{\alpha_1=0}^{1}\sum_{\alpha_2=0}^{1}\left[\mathbf{p}(N_{FG},\alpha_1,\alpha_2)+\int_{-\infty}^{N_{FG}} f(x,\alpha_1,\alpha_2)\,dx\right]=1 \tag{35}$$

A reasonable guess at the solution is of the form

$$f(x,\alpha_1,\alpha_2)=Ce^{\lambda x}Y_1^{\alpha_1}Y_2^{\alpha_2}$$

Plugging this into the differential equations, we have that

$$\begin{aligned}
-(r_M+r_D)+p_MY_1+p_DY_2&=0 \quad \text{from eqn (28)}\\
\lambda Y_2-(r_M+r_D)Y_2+p_MY_1Y_2+r_D&=0 \quad \text{from eqn (27)}\\
-\lambda Y_1-(p_M+r_D)Y_1+p_DY_1Y_2+r_M&=0 \quad \text{from eqn (29)}\\
-(p_M+p_D)Y_1Y_2+r_MY_2+r_DY_1&=0 \quad \text{from eqn (30)}
\end{aligned}$$

Doing some algebra, we can transform these equations to the *parametric equations*

$$p_MY_1-r_M+p_DY_2-r_D=0 \tag{36}$$

$$-\lambda=(p_MY_1-r_M)\,\frac{1+Y_1}{Y_1} \tag{37}$$

$$\lambda=(p_DY_2-r_D)\,\frac{1+Y_2}{Y_2} \tag{38}$$

By some more algebra, we can solve these equations and get

$$Y_1=Y_2=Y \;=\; \frac{r_M+r_D}{p_M+p_D} \tag{39}$$

$$\lambda \;=\; (r_Mp_D-p_Mr_D)\left(\frac{1}{p_M+p_D}+\frac{1}{r_M+r_D}\right) \tag{40}$$

We can now use equations (39) and (40) together with the boundary conditions to get the complete solution, yielding

$$\mathbf{p}(N_{FG},1,0) \;=\; C\,\frac{r_M+r_D}{p_Mr_D}\,e^{\lambda N_{FG}} \tag{41}$$

$$\mathbf{p}(N_{FG},1,1) \;=\; C\,\frac{r_M+r_D}{p_M(p_M+p_D)}\,e^{\lambda N_{FG}} \tag{42}$$

The normalization constant C can now be chosen to satisfy equation (35).

Comparing these equations with equations (3.260) – (3.262) in Gershwin (1994), we see that this system is very closely related to the traditional two-machine line: The densities are the same, with machine M acting as the first machine in the tandem line and the demand process D as the second, and where both processing rates $\mu_M=\mu_D=1$. This is because the demand process draws material from the buffer in the same manner as a downstream machine in a tandem line.

From this solution, we can derive the performance measures of interest. The production rate $E(k)$ is known a priori to be equal to the isolated production rate of the slowest machine, since we have an infinite buffer decoupling the machines. For stability, this must be the demand machine, whose production rate is $r_D/(r_D+p_D)$. In addition, let

p_s be the probability of starving the synchronization machine S

p_{su} be the probability of the upstream machine starving S

p_{sd} be the probability of the demand process starving S

p_b be the probability of blocking machine M

$\bar{x}_{FG}$ be the average finished goods inventory

$\bar{x}_B$ be the average backlog

We have

$$
\begin{aligned}
p_b &= \mathbf{p}(N_{FG}, 1, 0) && (43)\\
p_{su} &= \int_{-\infty}^{0} f(x,0,1)\,dx + \int_{-\infty}^{0} f(x,0,0)\,dx && (44)\\
p_{sd} &= \mathbf{p}(N_{FG}, 1, 0) + \int_{0}^{N_{FG}} f(x,1,0)\,dx + \int_{0}^{N_{FG}} f(x,0,0)\,dx && (45)\\
p_s &= p_{su} + p_{sd} && (46)\\
\bar{x}_{FG} &= \int_{0}^{N_{FG}} x \sum_{\alpha_M=0}^{1} \sum_{\alpha_D=0}^{1} f(x,\alpha_1,\alpha_2)\,dx \\
&\quad + N\left[\mathbf{p}(N_{FG},1,1) + \mathbf{p}(N_{FG},1,0)\right] && (47)\\
\bar{x}_B &= \int_{-\infty}^{0} -x \sum_{\alpha_M=0}^{1} \sum_{\alpha_D=0}^{1} f(x,\alpha_1,\alpha_2)\,dx && (48)
\end{aligned}
$$

Note that these performance measures are different from those of the tandem line, since we must now account for negative buffer levels, which are not permitted in a tandem line.

5 Solution algorithm

We now have everything we need to solve the decomposed system. We use an algorithm that consists of an outer search loop and a fixed point iteration.

The fixed point iteration is similar to the solution algorithm for tandem lines proposed by Dallery, David, and Xie (1989), but we use the three different kinds of components described in the previous section. In addition, when the algorithm has completed a sweep downstream, we need to update the parameters of the upstream machine in the first component, since this machine represents the arrival process of material at the first buffer in the system. This is regulated by the demand process. Similarly, at the end of an upstream sweep, the downstream machine of two-machine line $k+1$ is updated.

The outer loop is a search in $I_u(j)$ for some two-machine line j. As proposed by Frein, Commault, and Dallery (1996), we fix the ratio $I_u(j)$ to an initial value and run the fixed point iteration until it converges. We then calculate the resulting average loop population Q by summing the average levels of buffers 1 through $k+1$. The result is not likely to match the loop population prescribed by N_C.

An important empirical observation is that Q is monotonically decreasing in $I_u(j)$ for all choices of j. We can therefore use a binary search in $I_u(j)$ to find a value that makes $Q = N_C \pm \epsilon$, where

$$\epsilon = \left| \frac{Q - N_C}{N_C} \right| \tag{49}$$

The search is terminated when ϵ is sufficiently small.

To perform the binary search, we need an initial upper and lower bound on $I_u(j)$. A reasonable choice for the lower bound is $I_{\min} = p_j/r_j$, since starvations propagating from upstream will cause the apparent down time of the upstream machine to be larger than what is indicated by these values. The upper bound is harder to guess, so we successively try values $I_u(j) = 2^a I_{\min}$ for $a = 1, 2, \ldots$. For each value, we perform the inner loop of the algorithm given below to get the corresponding loop population estimate. We set $I_{\max}$ to the first value of $I_u(j)$ encountered such that the estimate Q is less than the CONWIP limit N_C.

For each value of $I_u(j)$, the inner loop, a fixed point iteration is performed until convergence. We measure the degree of convergence by calculating the maximal relative change in any unknown variable between iterations of the inner loop, and consider the inner loop to have converged if this is sufficiently small. In the results reported below, we used convergence criteria of 10^{-5} for the inner loop and 10^{-3} for the outer loop.

We can now state the complete algorithm:

```
Read parameters $N_C$, $r_D$, $p_D$, $r_i$, $p_i$, $N_i$ for $i = 1, \ldots, k$;          *Initialization*
**for** $i = 1, \ldots, k$ **begin**
    $e_i := r_i/(r_i + p_i)$;
    $r_u(i) := r_i$; $p_u(i) = p_i$; $I_u(i) = p_i/r_i$;
    $r_d(i) := r_{i+1}$; $p_d(i) = p_{i+1}$; $I_d(i) := p_{i+1}/r_{i+1}$;
**end**
Designate a special machine $j$;
Find initial estimates of $I_{\min}$ and $I_{\max}$;

**repeat**                                                                  *Outer loop*
    $I_u(j) := 0.5 * (I_{\min} + I_{\max})$;
    $p_u(j) := I_u(j) * r_u(j)$;

    **repeat**                                                              *Inner loop*
        **for** $i = 2, \ldots, k-1$ **begin**                                  *Downstream sweep*
            Evaluate two-machine line $i - 1$;
            **if** $(i \neq j)$ **begin**
                Evaluate $I_u(i)$ from equation (13);
            **end**
            Evaluate $r_u(i)$ from equation (15);
            Evaluate $p_u(i)$ from equation (16);
        **end**

        Evaluate synchronization stage $k$;
        Evaluate $I_u(k+1)$ from equation (12);
        Evaluate $r_u(k+1)$ from equation (10);
        Evaluate $p_u(k+1)$ from equation (11);

        Evaluate $I_d(k+1)$ from equation (8);                    *Wrapping around*
        Evaluate $r_d(k+1)$ from equation (2);
        Evaluate $p_d(k+1)$ from equation (3);

        Evaluate line $k+1$;                                      *Upstream sweep*
        Evaluate synchronization stage $k$;
        Evaluate $I_d(k-1)$ from equation (8);
        Evaluate $r_d(k-1)$ from equation (2);
        Evaluate $p_d(k-1)$ from equation (3);

        **for** $i = k-2, \ldots, 1$ **begin**
            Evaluate two-machine line $i+1$;
            Evaluate $I_d(i)$ from equation (17);
            Evaluate $r_d(i)$ from equation (19);
            Evaluate $p_d(i)$ from equation (20);
        **end**

        **if** $(j \neq 1)$ **begin**                             *Wrapping around again*
            Evaluate $I_u(1)$ from equation (13);
        **end**
        Evaluate $r_u(1)$ from equation (15);
        Evaluate $p_u(1)$ from equation (16);

    **until** convergence of $r_u$, $p_u$, $r_d$ and $p_d$;       *End of inner loop*

    $Q := \sum_{i=1}^{k+1} \bar{x}(i);$
    **if** $(Q \geq N_C)$ $I_{\min} := I_u(j)$; **else** $I_{\max} := I_u(j)$;   *Update search bounds*

**until** $Q$ is sufficiently close to $N_C$;                     *End of outer loop*
```

6 Results

The decomposition we have presented does not require all machines in the line to be identical, but it is easier to interpret results if the machines are the same. We therefore concentrate on lines with identical machines in the following. Similar results are obtained for lines where the machines are not identical in Bonvik (1996).

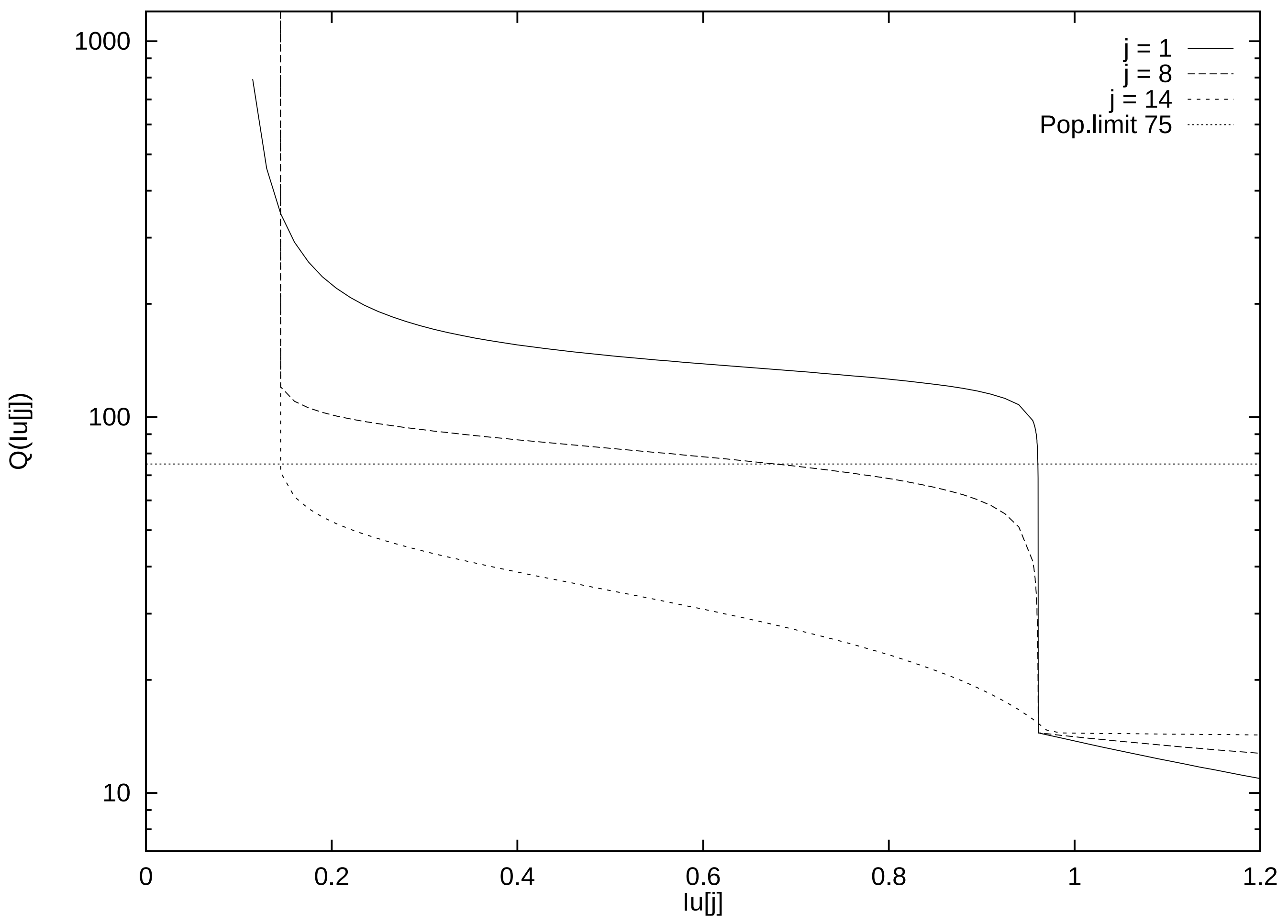

Figure 4: Behavior of $Q(I_u(j))$ for different j.

6.1 Convergence

The key to the algorithm's behavior is the function $Q(I_u(j))$. In the loop decomposition studied by Frein, Commault, and Dallery, this is experimentally determined to be a decreasing function of $I_u(1)$. Our case is messier. For any machine in the line, if we increase I_u of this machine far enough, the machine will have a lower production rate than the demand rate, and the system will be unstable. We can therefore expect some complications for a value of $I_u(j)$ such that instability is about to occur.

It turns out that the choice of the special machine j is very important for the convergence properties of the algorithm. Poor choices of j lead to slow or no convergence. Figure 4 illustrates this for a 15 machine line where the demand rate is .5, all machines have $r = .1$ and $p = .01$. The CONWIP limit is 75, and the demand process is modeled as a machine with $r_D = p_D = .1$. All buffers are size 10.

From Figure 4, we see that for this case, choosing $j = 8$ will lead to much faster convergence of the binary search procedure than either $j = 1$ or $j = 14$. Computational results bear this out: Using $j = 8$, the algorithm requires about .3 seconds, while for $j = 1$ it takes 42 seconds, and for $j = 14$ it takes 10.7 seconds. Different values of the population constraint will lead to different behavior.

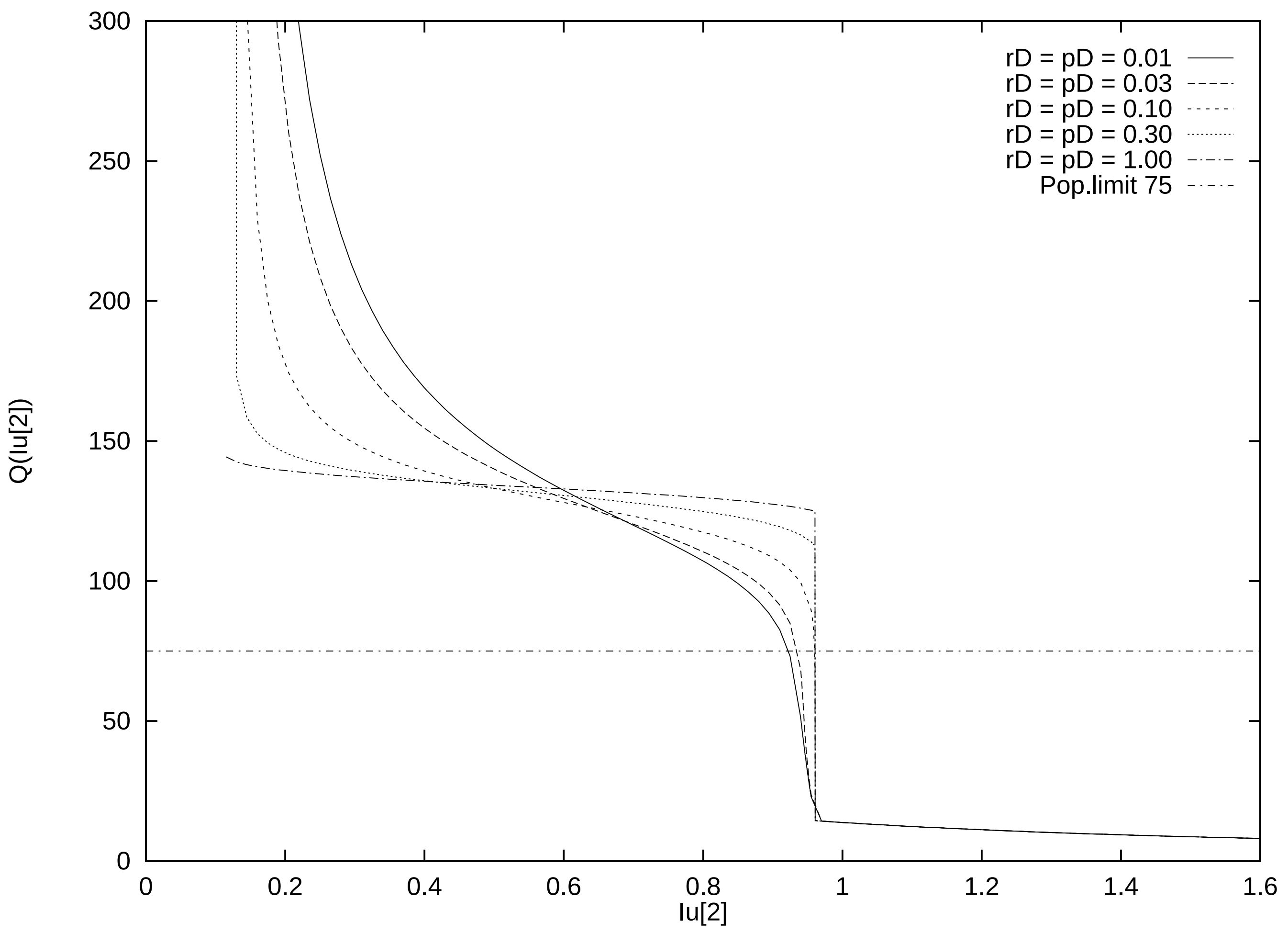

Figure 5: Behavior of $Q(I_u(2))$ for different demand representations.

The shape of $Q(I_u(j))$ depends on the system parameters. In figure 5, we have plotted this function for different representations of the demand machine in the same system as above, using $j = 2$. In all cases, the demand rate is .5, but we change the failure and repair rates of the demand machine simultaneously. The discontinuity at the onset of instability becomes taller as the failure and repair rates of the demand machine increase. For this choice of j, the algorithm does not converge when the failure and repair rates exceed $r_D = p_D = .2$.

For the remainder of the paper, we use $j = \lfloor k/2 \rfloor$, choosing a machine in the middle of the line as the special one.

6.2 Accuracy

Our base case is the same 15-machine line as in the previous section. Figure 6 shows how the estimated buffer levels from the decomposition compare with a continuous material simulation of the same system. In the figure, buffer 15 is the finished goods buffer, buffer 16 is the buffer of free kanbans or production authorizations waiting at machine 1, and buffer 17 is the system backlog. To get tight confidence intervals, we averaged across 15 replications of the simulation, each one 10^7 time units

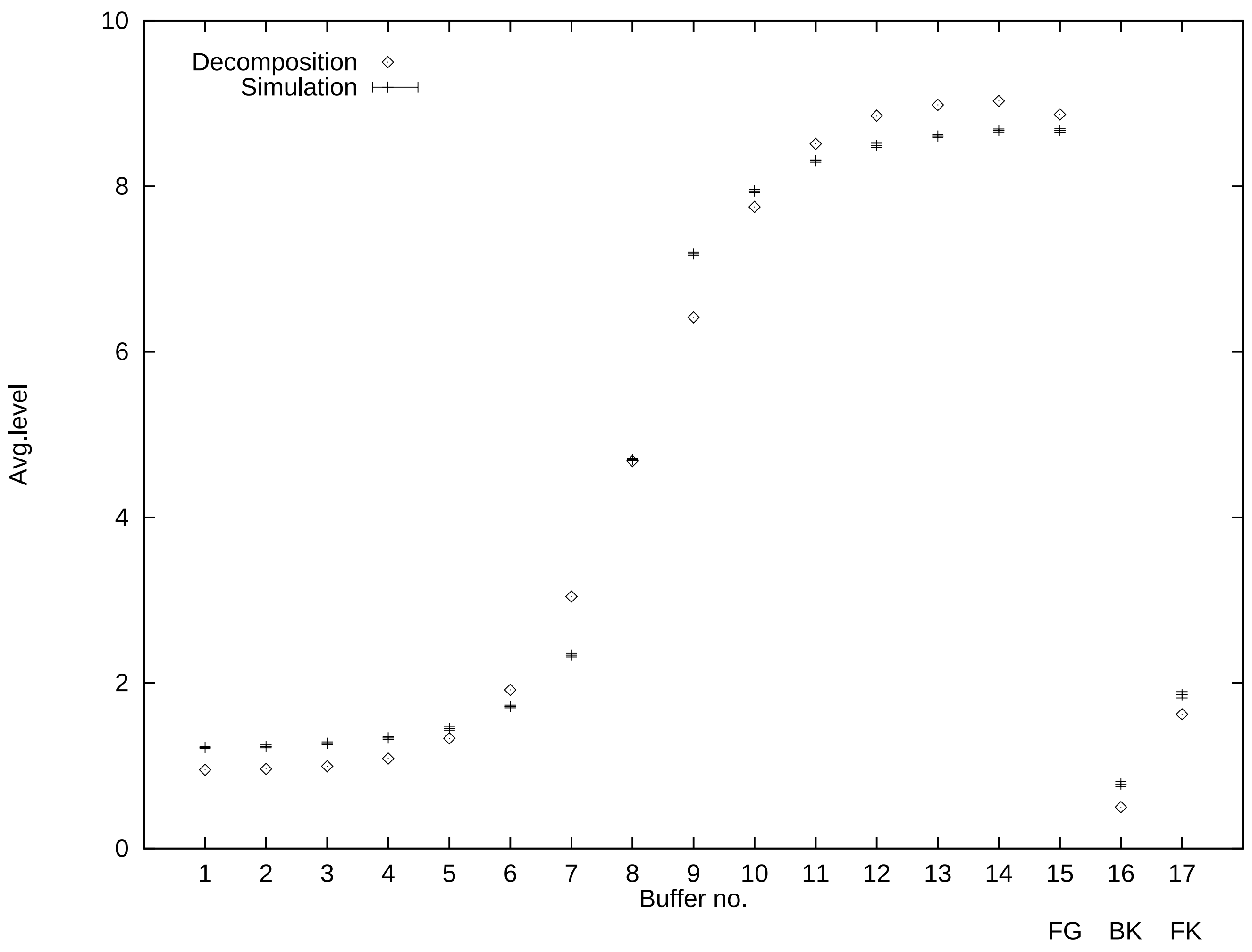

Figure 6: Accuracy of estimated average buffer levels for 15 machine line

long with an initial warm-up period of 10^5 before statistics were collected. The confidence intervals were calculated from a Student's t-distribution with 14 degrees of freedom (Arnold 1990).

The accuracy depends strongly on the system size. In figure 7 we have plotted the average level of the finished goods buffer, as estimated by decomposition and simulation, for lines from 2 to 25 machines. All buffers were size 10, and the CONWIP limit was $N_C = 5k$ where k is the number of machines. For these systems, the accuracy is good for $k \geq 8$, where the error is less than 2%. The error was larger for shorter lines. For two-machine lines, the estimated finished goods level was off by a factor of almost three. The poor accuracy for small system was also observed by Frein, Commault, and Dallery, and is most likely due to the stronger correlations between buffer levels in a small system. For instance, in a two machine line, there are three buffers, and if we know the levels of any two, we can determine the level of the third. In large systems, knowing the levels of a few buffers gives much less information about the remaining ones.

Similarly, we have observed that the accuracy in estimates of performance measures for a given system become better as the CONWIP limit of the system increases. This is also due to correlation effects: If the limit is one job in the system at any instant, knowing that the level of some buffer is one also determines the levels of all other buffers.

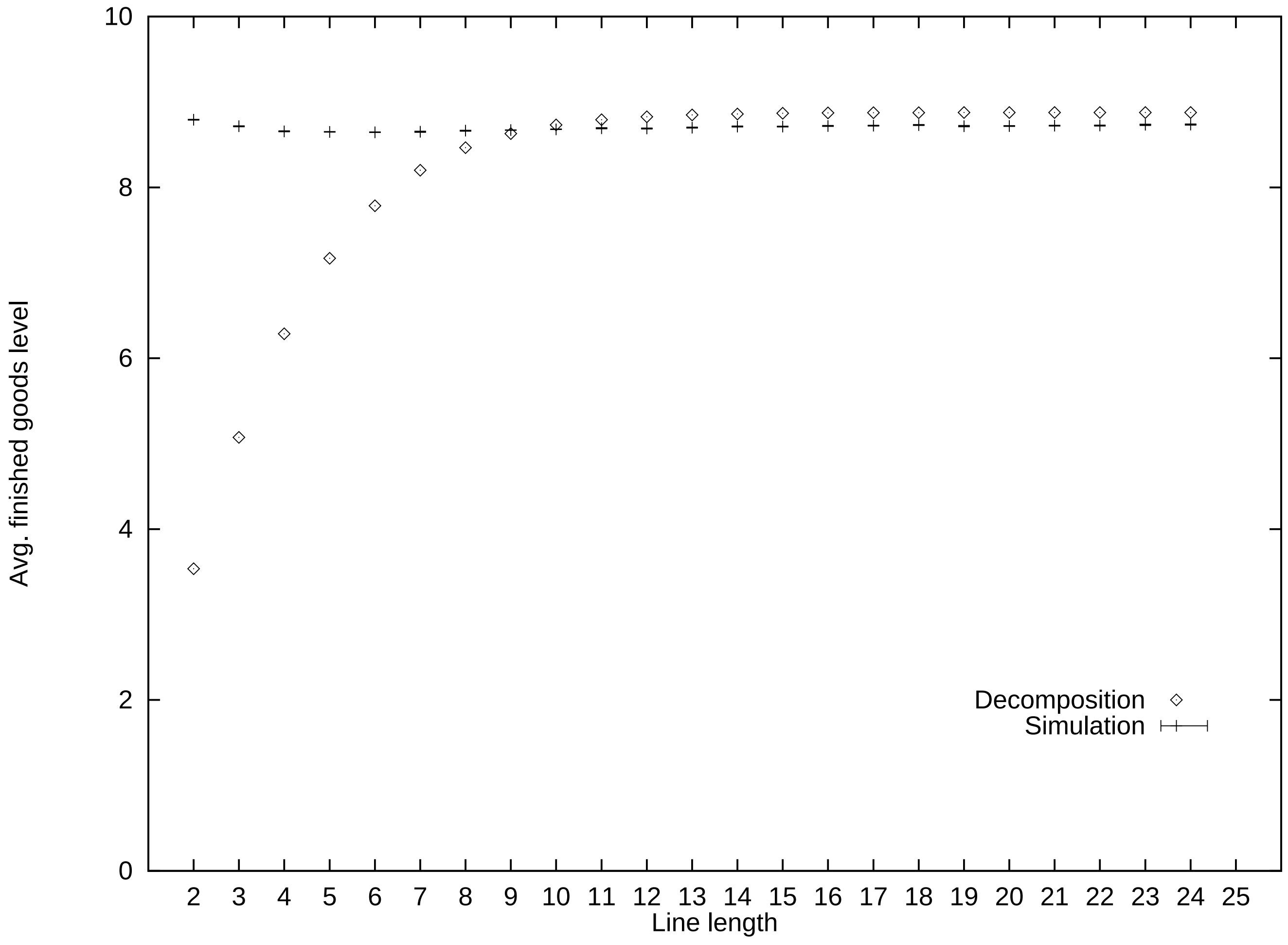

Figure 7: Accuracy of estimated average finished goods level as a function of line length

We can define the *utilization* of the system as the ratio of the demand rate to the production rate of the same system in saturating demand. For instance, our 15 machine base case has a simulated throughput of .698 in saturating demand. The .5 demand rate imposed on the system is then a utilization of .5/.698 = .716.

The accuracy of buffer level estimates from the decomposition, especially the system backlog, is poor for highly utilized systems. One reason for these errors is that the saturated production rate estimated by the decomposition may be slightly different from the simulation value. For the same target production rate, this leads to different apparent system utilizations in the decomposition and simulation. Example: If the decomposition gives a saturated production rate of .69 and the simulation value is .70, a demand rate of .68 seems to be a utilization of .986 in the decomposition, but just .971 in the simulation. Worse still, a demand rate of .689 seems to be a utilization of .999 to the decomposition, but will really be just .984. This can lead to significant differences, especially in the estimate of average system backlog.

This effect is shown for the 15 machine line in figure 8. We have measured utilization as a fraction of the throughput of a simulation of the line in saturating demand. The simulation reports a throughput of .698 and the decomposition .710. This is a difference of 1.72%. As the curves marked

"Simulation" and "Decomposition" show, this can lead to a very significant difference in estimated backlog when the system approaches capacity. From basic queuing theory, we expect the backlog to grow as $\rho/(1-\rho)$, where ρ is the utilization (Kleinrock 1975).

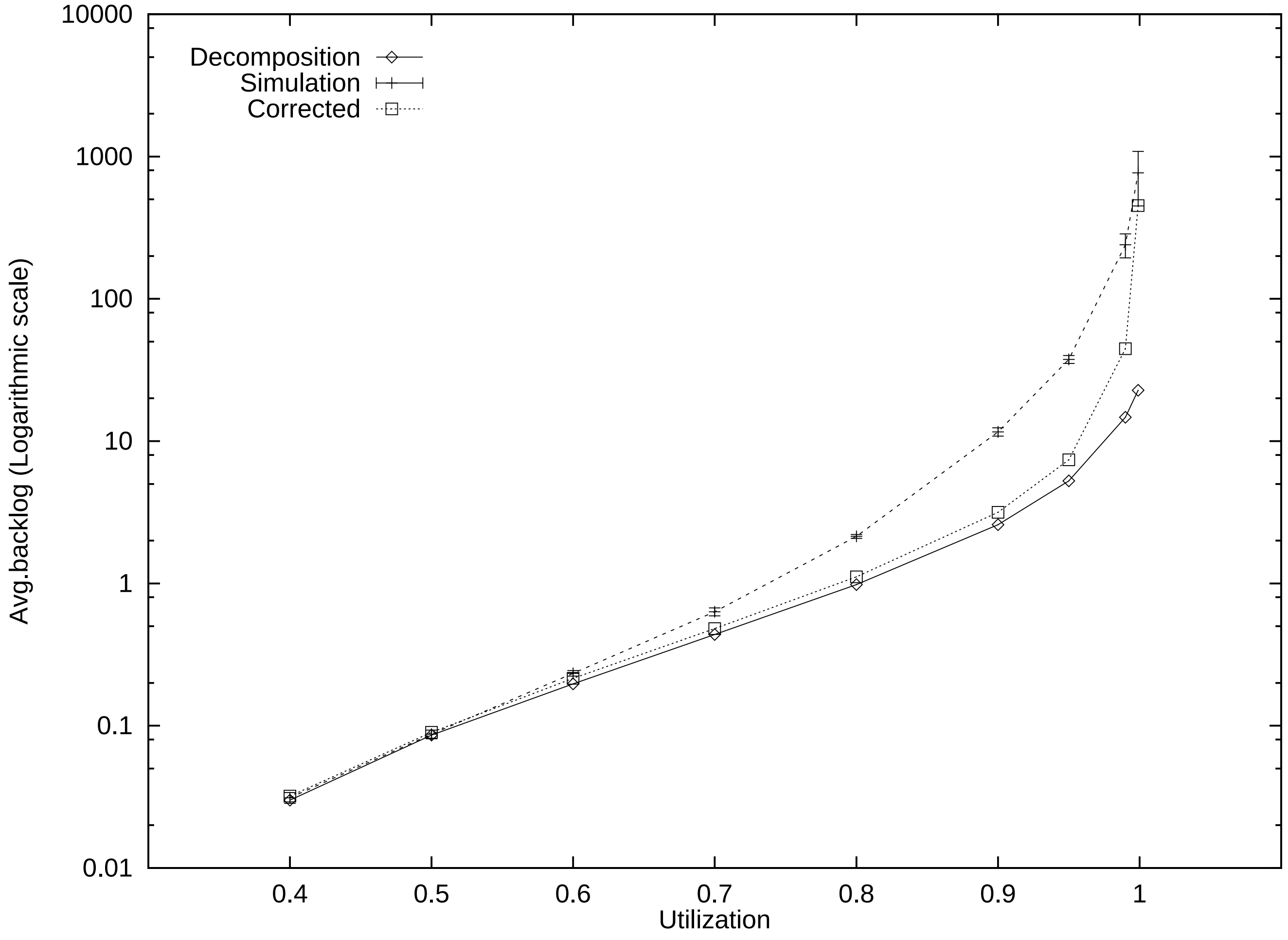

Figure 8: Accuracy of estimated average backlog for different utilizations

We have also plotted a corrected series of decomposition results in Figure 8, where the utilization is measured as a fraction of the saturated throughput estimated by decomposition. This shows that the error in very high utilization is due to the apparently small difference in estimated throughput between the decomposition and simulation. We also see that for utilizations in the .8 to .95 range there are still significant errors in the estimated backlog. We believe this is due to the way the exponential failure process in the upstream machine of the synchronization stage is used to represent all interruptions of flow arriving at the finished goods buffer from anywhere in the production line.

In the line, the time from when flow resumes until the next interruption occurs is not exponentially distributed, since it is a mix of the failure distributions of the various machines and the time to drain the buffers downstream of the failed machine. We have measured coefficients of variability (CV) for these times until the next interruption using simulation. In the 15-machine line studied here, the values range up to about 1.3. Most importantly, the CV of the times with flow into the finished goods buffer is about 1.26. Approximating these times by exponentially distributed (CV 1.0) failure times with the same mean will give too low estimates of system backlog. In contrast, when a machine is repaired, the flow resumes instantly at all starved machines downstream. With identical machines, this gives an exponentially distributed time to flow resumption.

7 Conclusions

We have constructed a fast algorithm for approximate performance analysis of tandem manufacturing systems controlled by a hybrid control policy incorporating CONWIP and finite buffers. These systems are synchronized to an external demand process. Our algorithm gives good accuracy for medium-sized and large systems operating in light to medium utilization.

Some extensions to this algorithm seem worthwhile exploring. We have combined three different building blocks in a decomposition: tandem two-machine lines with finite and infinite buffers, and a synchronization stage. It is possible to model other control disciplines than the one analyzed in the present paper by assembling these building blocks in different ways. This is an area of active research.

Similar methods could be used to analyze systems with multiple loops. This way, reentrant systems could be analyzed, and also control policies like basestock where there are several nested loops of information flow.

Burman (1995) developed a decomposition algorithm for *inhomogeneous systems*, that is, one where the machines can have different processing rates. In our algorithm, a low variability demand process must be modeled by high failure and repair rates at the machine representing the demand process. Extending our algorithm to use the inhomogeneous decomposition equations could alleviate the convergence problems encountered when doing this by combining a low demand rate with rare failures and repairs of the demand machine.

Another promising development is the decomposition approach developed by Dallery and Le Bihan (1997) using *generalized exponential* distributions (Dallery 1994) for the down-time distributions of the individual building blocks in the decomposition. The decomposition can then match two moments of the down-times rather than just the mean. This has been shown to be an advantage if the repair rates of adjacent machines differ by several orders of magnitude.

We demonstrated that the distribution of up-times can also lead to significant approximation errors. It would be very worthwhile to develop a set of decomposition equations that matched two moments of the *uptime* distributions, for example by using generalized exponential distributions.

Some recent work (Schor 1995), (Gershwin and Schor 1997) has shown that the qualitative properties of production rate and average buffer levels in production lines are such that efficient optimization of buffer space distributions is possible. Such work should be extended to machine selection, and to hybrid-controlled systems of the type studied here.

References

Alvarez Vargas, R., Y. Dallery, and R. David (1992). A study of the continuous flow model of production lines with unreliable machines and finite buffers. *Journal of Manufacturing Systems 13*(3), 221–234.

Arnold, S. F. (1990). *Mathematical Statistics*. Englewood Cliffs, New Jersey: Prentice Hall.

Berkley, B. J. (1991). Tandem queues and kanban-controlled lines. *International Journal of Production Research 29*(10), 2057–2081.

Berkley, B. J. (1992). A review of the kanban production control research literature. *Production and Operations Management 1*(4), 393–411.

Bielecki, T. and P. R. Kumar (1988). Optimality of zero-inventory policies for unreliable manufacturing systems. *Operations Research 36*(4), 532–541.

Bonvik, A. M. (1996). *Performance Analysis of Manufacturing Systems Under Hybrid Control Policies*. Ph. D. thesis, Massachusetts Institute of Technology.

Bonvik, A. M., C. Couch, and S. B. Gershwin (1996). A comparison of production-line control mechanisms. Accepted for publication in the *International Journal of Production Research.*

Burman, M. H. (1995). *New results in flow line analysis*. Ph. D. thesis, Massachusetts Institute of Technology. Also available as Report LMP-95-007, MIT Laboratory for Manufacturing and Productivity.

Buzacott, J. A. and L. E. Hanifin (1978). Models of automatic transfer lines with inventory banks — a review and comparison. *AIIE Transactions 10*(2), 197–207.

Buzacott, J. A. and J. G. Shantikumar (1992). A general approach for coordinating production in multiple-cell manufacturing systems. *Production and Operations Management 1*(1), 34–52.

Buzacott, J. A. and J. G. Shantikumar (1993). *Stochastic Models of Manufacturing Systems*. Englewood Cliffs, New Jersey: Prentice Hall.

Dallery, Y. (1994). On modeling failure and repair times in stochastic models of manufacturing systems using generalized exponential distributions. *Queuing Systems 15*, 199–209.

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., R. David, and X.-L. Xie (1989). Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on automatic control 34*(9), 943–953.

Dallery, Y. and S. B. Gershwin (1992). Manufacturing flow line systems: A review of models and analytical results. *Queuing Systems Theory and Applications 12*(1-2), 3–94. Special issue on queuing models of manufacturing systems.

Dallery, Y. and H. Le Bihan (1997). An improved decomposition method for the analysis of production lines with unreliable machines and finite buffers. Accepted for publication in *International Journal of Production Research.*

Dallery, Y. and G. Liberopoulos (1995). Extended kanban control system: Combining base-stock and kanban. Accepted for publication in *IIE Transactions*.

Di Mascolo, M., R. David, and Y. Dallery (1991). Modeling and analysis of assembly systems with unreliable machines and finite buffers. *IIE Transactions 23*(4), 315–330.

Di Mascolo, M., Y. Frein, and Y. Dallery (1996). An analytical method for performance evaluation of kanban controlled production systems. *Operations Research 44*(1), 50–64.

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*, 291–305.

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

Gershwin, S. B. and J. E. Schor (1997). Efficient algorithms for buffer space allocation. Accepted for publication in *Annals of Operations Research*.

Kimball, G. (1988). General principles of inventory control. *Journal of Manufacturing and Operations Management 1*(1), 119–130.

Kleinrock, L. (1975). *Queuing Systems, Volume I: Theory.* New York: John Wiley & Sons.

Lee, Y.-J. and P. Zipkin (1992). Tandem queues with planned inventories. *Operations Research 40*(5), 936–947.

Liberopoulos, G. and Y. Dallery (1997). A unified framework for pull control mechanisms in multi-stage manufacturing systems. Unpublished manuscript.

Mitra, D. and I. Mitrani (1990). Analysis of a kanban discipline for cell coordination in production lines, I. *Management Science 36*(12), 1548–1566.

Mitra, D. and I. Mitrani (1991). Analysis of a kanban discipline for cell coordination in production lines, II: Stochastic demands. *Operations Research 39*(5), 807–823.

Schor, J. E. (1995). Efficient algorithms for buffer allocation. Master's thesis, Massachusetts Institute of Technology. Also available as MIT Laboratory for Manufacturing and Productivity report LMP-95-006.

Spearman, M. L., D. L. Woodruff, and W. J. Hopp (1990). CONWIP: a pull alternative to kanban. *International Journal of Production Research 28*(5), 879–894.

Van Ryzin, G. (1987). Control of manufacturing systems with delays. Master's thesis, Massachusetts Institute of Technology. Also available as MIT Laboratory for Information and Decision Systems Report LIDS-TH-1676.

Womack, J. P., D. T. Jones, and D. Roos (1990). *The machine that changed the world : The story of lean production.* Rawson Associates.