Performance Evaluation of a Make-To-Stock Production Line with a Two-Parameter-per-Machine Policy: the Control Point Policy

Francis de Vericourt* and Stanley B. Gershwin†
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307

September 14, 2000

Abstract

This paper presents a decomposition method for evaluating the performance of a production line controlled by the control point policy (CPP), a policy with two parameters per machine. Policies with two parameters per machine show very good performance while remaining simple to use. However, decomposition methods have not yet been developed to analyze them. We consider a production line with exponentially distributed production, failure, and repair times, controlled by a control point policy. We decompose this line into two-machine CPP lines, which makes the decomposition equations very simple. Furthermore, the information loops are then included in the building blocks, and can be solved numerically. A numerical study shows that the method is accurate.

Key Words: performance evaluation, decomposition, control policy, make-to-stock, control point policy

1 Introduction

One of the main issues in the design of a manufacturing system is the control policy which makes real-time decisions related to the movement of parts into and through the system. The purpose of the policy is to provide good service to customers while minimizing costs such as in-process inventory. The inherent variability in production processes and in demand makes this difficult,

*fvericourt@yahoo.com
†gershwin@mit.edu; http://web.mit.edu/manuf-sys/www

even for a simple production line where a series of machines separated by buffers produces a single part type.

It is possible to formulate this problem as a stochastic optimal control problem (Gershwin, 1994). The objective may be to determine the policy which minimizes the average holding costs under a demand service level constraint. A more tractable alternative consists in minimizing the holding and backorder costs due to waiting demands. Veatch and Wein (1994) characterized the optimal policy for a queuing model of a two-machine line. Unfortunately, optimal policies for these systems generally possess a very complex structure, and their direct application is quite often impossible in practice. However, optimization studies can reveal properties that can be useful in the design of good sub-optimal heuristics. An example is the hedging point concept that appears in the characterization of many optimal policies (Kimemia and Gershwin, 1983; Bielecki and Kumar 1988; Ha 1997; de Vericourt et al. 2000). The hedging point is the maximum desired level for the excess of production over the demand at a machine.

A complementary approach to this problem is to find classes of control policies that provide good (but sub-optimal) performance, and are easier to apply than optimal policies. The objective is to determine the best control policy within such a class. Much research has been devoted to the definition of such classes, and more specifically to pull control policy classes where the production is driven by actual demands. See Liberopoulos and Dallery (1997) for a unified framework of pull control mechanisms. Basestock, kanban, generalized kanban, extended kanban, and CONWIP are some examples.

Other research has been devoted to evaluating the performance of these control policies, in order to optimize their parameters. One approach consists in modeling the production system as a queueing network, and to evaluate its steady state probability using product form approximations. This technique has been developed to analyze kanban systems (Di Mascolo and al., 1996) and generalized kanban systems (Frein and al., 1995). An another approach is the decomposition method introduced by Gershwin (1994), which decomposes the original system into a set of tractable subsystems. Considerable efforts have been devoted to improving the efficiency of this technique (Dallery et al., 1989; Tolio and Gershwin, 1998; Dallery and Lebihan, 1997), or to extending its scope (Burman, 1995; Bonvik, 1996; and many others). Until now, the decomposition method has been used to analyze assembly/disassembly systems and single loop systems controlled by a kanban policy. Bonvik (1995) proposed some extensions to analyze CONWIP policies, or kanban policies where the last machine of the line is controlled by a basestock policy.

The policies that can be evaluated with decomposition techniques are easy to use in practice, but perform poorly in many situations. Karaesmen and Dallery (1999) showed that for a simple two-machine line model, more refined policies with two parameters per machine such as the extended kanban (Dallery and Liberopoulos, 2000) or the generalized kanban (Buzacott and Shanthikumar, 1993), performs better, and seem to provide a good tradeoff between complexity and efficiency. For larger systems, Gzouli (2000) observed, through intensive simulations and preliminary case studies, that in many situations the control point policy (CPP) can achieve high service levels with less on-hand inventories than kanban, CONWIP or basestock. The control point policy, which was recently introduced by Gershwin (2000), is also characterized by two parameters per machine: a hedging point and a buffer capacity. Until now, no efficient tool has been proposed to evaluate policies such as extended kanban or CPP. One of the main reasons is that these policies create multiple information loops, and existing decomposition methods cannot treat them.

In this paper, we propose a new approximation technique able to analyze a make-to-stock line controlled by CPP. This is the first decomposition method developed to evaluate a control policy with two parameters per stage. The technique is presented using a model with Poisson demand, and where the machines are subject to exponentially distributed failure and repair times and produce discrete parts according to an exponential production time.

The control point policy and the model of the production line are presented in Section 2. The decomposition technique is provided in Section 3. The numerical solution of the two-machine system is described in Section 4. An algorithm to solve the decomposition equations appears in Section 5. The accuracy of the method is discussed in Section 6, where some numerical examples are shown. We present concluding remarks in Section 7.

2 Model of a Production Line Controlled by the Control Point Policy

The Control Point Policy (CPP) was designed to control a large variety of production systems, including those with multiple part types and re-entrant flows (Gershwin, 2000). In Section 2.1 we describe a make-to-stock tandem production line which produces a single part type. We briefly summarize the CPP in Section 2.2.

2.1 The Make-To-Stock Production Line

Consider a production line with k machines $(M_1, \ldots, M_k)$ separated by $k-1$ intermediate buffers $(B_1, \ldots, B_{k-1})$, and terminated by a finished goods inventory B_k. Parts are discrete and we denote by $n_i(t)$ the number of parts in B_i at time t.

Material flows from M_1 to B_1, then from B_1 to M_2, and so on. Operations take place at the machines, and demands are met with parts in the finished good inventory. An arriving demand is satisfied with a part from B_k if B_k is not empty. If it is empty, the demand is back-ordered. We assume that raw material is always available. That is, we assume that M_1 is never starved.

The demand is modeled by a Poisson process with parameter λ. Each machine is subject to failures that are operation dependent. (That is, a machine can fail only while processing a part.) The time to fail, time to repair, and the production time of Machine M_i are exponentially distributed with parameters p_i, r_i, and μ_i respectively. The *isolated efficiency* e_i of machine M_i is the fraction of time that M_i would be working if it were isolated never starved or blocked:

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

We denote by $\alpha_i(t)$ the state of Machine i M_i at time t. $\alpha_i = 1$ if M_i is up (operational) and $\alpha_i = 0$ if M_i is down (failed, or under repair).

2.2 The Control Point Policy

2.2.1 Surplus-Based Definition

The Control Point Policy controls the flow into each machine. It states that Machine M_i does not produce if the excess of production at M_i over the demand is too large, or if the excess of production at M_i over that at M_{i+1} is too large. The former condition is maintained by the use of a hedging point; the latter by limiting the buffer size. (Of course production is also prevented by starvation.)

We define the *surplus* $x_k(t)$ of Machine M_k such that

- if $n_k(t)$, the number of finished items in B_k at time t, is positive, then $x_k(t) = n_k(t)$; and
- if $n_k(t) = 0$, then $-x_k(t)$ is the number of waiting demands (the backlog) at time t.

In the following, $x_i^+ = \max(0, x_i)$ and $x_i^- = \max(0, -x_i)$.

The CPP says that M_k is allowed to produce at time t if $x_k(t) < Z_k$ and there is a part in B_{k-1} at time t. Z_k is called the *hedging point*, and, as a consequence, x_k is always less than or equal to Z_k.

More generally, we define the *surplus* $x_i(t)$ of Machine M_i recursively by

$$x_i(t) = x_{i+1}(t) + n_i(t). \tag{1}$$

The Control Point Policy is defined by $2k-1$ parameters: k hedging points Z_i $(i \le k)$ and $k-1$ buffer constraints C_i $(i < k)$. The CPP states that Machine M_i may produce at time t if and only if

- for $i = 2 \dots k-1$, $n_{i-1}(t) > 0$, $n_i(t) < C_i$, and $x_i(t) < Z_i$;
- for $i = 1$, $n_1(t) < C_1$ and $x_1(t) < Z_1$;
- for $i = k$, $n_{k-1}(t) > 0$ and $x_k(t) < Z_k$.

Note that no additional capacity constraint for the finished goods inventory needs to be specified since Z_k limits the level of Buffer B_k. Furthermore, if there is a j such that $Z_j - Z_{j+1} > C_j$, then the system is equivalent to a CPP with the same parameters where Z_j is replaced by $C_i + Z_{j+1}$. Therefore, we can restrict our study to the control point policies whose parameters are such that $Z_i - Z_{i+1}$ is less than or equal to C_i for $i = 1 \dots k-1$.

We use L_k to denote a k-machine line controlled by the CPP, where the production times, the times to fail, and the times to repair of the machines are exponentially distributed. These systems are referred to as *CPP lines* here.

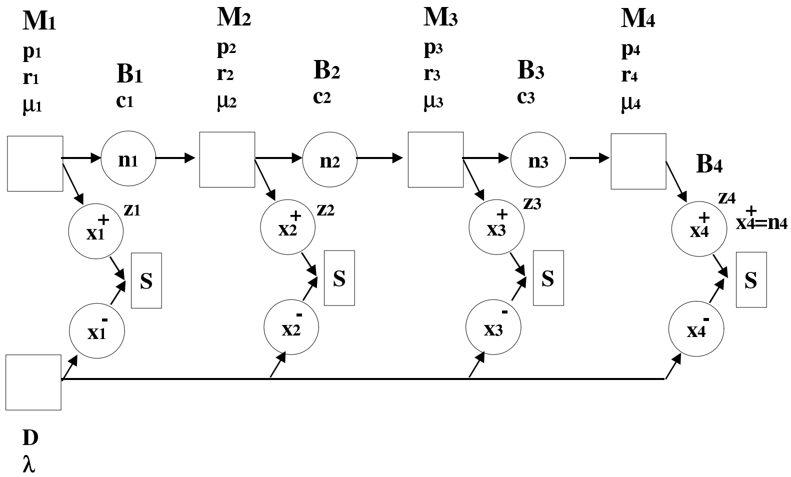

Figure 1: A Make-To-Stock Line Controlled by a Control Point Policy

2.2.2 Token-Based Definition

In the token-based representation of the CPP, represented in Figure 1, the demand process is modeled by Machine D, a perfectly reliable machine with a production rate equal to λ. Each of the surpluses and backlogs are represented by buffers, and when Machine D performs an operation, a token appears simultaneously in each backlog buffer. At such times, $x_i^-(t)$ increases by 1, for all $i \leq k$.

When M_i performs an operation, it removes a part from B_{i-1}, adds a part to B_{i-1}, and adds a token to the local surplus buffer. That is, $n_{i-1}(t)$ decreases by 1, $n_i(t)$ increases by 1, and $x_i^+(t)$ increases by 1.

If there are one or more tokens in the local surplus buffer *and* in the local backlog buffer, ie, if $x_i^+(t) \geq 1$ and $x_i^-(t) \geq 1$, the synchronization machine S (which is infinitely fast and perfectly reliable) instantaneously removes one token from each of them. Consequently, at least one of these buffers is empty at each time t.

The backlog buffers have no capacity constraint. The capacity of the surplus buffer is equal to Z_i. When this buffer is full, that is when $x_i = Z_i$, Machine M_i cannot operate. The behavior of this system is equivalent to the behavior of the system described in Section 2.2.1.

2.2.3 Performance Measures

We assume in the following that the parameters of the CPP line are such that the system is stable. That is, the system is able to satisfy the demand; and the throughput P_i of machine M_i is equal to λ in steady state. This is the statement of conservation of flow, or

$$P_1 = \ldots = P_k = \lambda. \tag{2}$$

The performance measure we are interested in is the service level β provided by the system, which is defined as

$$\beta = \text{prob}\{n_k(t) > 0 \text{ for every } t \text{ when a demand arrives}\}. \tag{3}$$

3 Decomposition

3.1 Overview

In this section, we propose a decomposition method to evaluate the service level of a transfer line controlled by the control point policy. Decomposition methods approximate the original system with a set of building blocks whose performance measures are easier to compute. Each buffer of a building block corresponds to a buffer in the original line. (In earlier formulations, the building block is a two-machine, one-buffer line, so there is only one buffer. In the analysis below, there are more.) The upstream machine approximates the flow of material (or tokens) coming from the upstream portion of the line into the buffer, while the downstream machine approximates the flow of material (or tokens) leaving the buffer and entering the downstream portion of the line. A set of equations is derived to compute the parameters of these machines. The purpose of these equations is to describe machines that are such that the behavior of flow of material in the buffers of each building block is close to the behavior of the flow of material in the corresponding buffers of the original line.

If we consider the token-based definition the system described in Section 2.2.2, this directly leads to the evaluation of loops. More precisely, machine M_i, M_{i+1}, the demand machine D and the two synchronous stations constitute a loop. In every closed loop, there is an invariant. It is determined by traversing the loop and adding the number of parts in buffers in which flow is in the same direction as the traversal, and subtracting the number in buffers where the flow is in the opposite direction. In this case, this becomes

$$x_{i+1}^{+} - x_{i+1}^{-} + x_i^{-} - x_i^{+} + n_i = \text{constant},$$

which, from the definition of x_i, becomes

$$x_{i+1} - x_i + n_i = \text{ constant}.$$

Equation (1) implies that the constant is 0.

The existing decomposition techniques for loops are only defined for single-loop systems, and appear to be less accurate than for acyclic systems despite considerable effort to improve their performance (Bonvik, 1995; Frein et al., 1996; Tolio and Gershwin, 1998). One possible reason for this difficulty comes from the fact that the equations which link the parameters of the two-machine lines do not directly represent the loop invariant.

To avoid this difficulty, we include the loops in the analysis of the subsystems. That is, we do not decompose the system into two-machine lines, but rather into $k-1$ CPP lines $L_2(i)$, $i = 1, \ldots, k-1$, as illustrated in Figure 2. Each subsystem $L_2(i)$ consists of a Poisson demand process, an upstream machine $M_u(i)$ with its surplus $x_u(i)$, a downstream machine $M_d(i)$ with its surplus $x_d(i)$, and an intermediate buffer $B(i)$ holding $n(i)$ parts. The arrival rate of the demand is the same as the arrival rate in the original line, λ. The capacity of $B(i)$ is C_i, the same as corresponding buffer B_i in the original line. The hedging points of the upstream and downstream machines are the same as the hedging points of Machine M_i and M_{i+1}, Z_i and Z_{i+1}. Machine $M_u(i)$ is characterized by a production rate $\mu_u(i)$, a failure rate $p_u(i)$ and a repair rate $r_u(i)$. Similarly, $\mu_d(i)$, $p_d(i)$ and $r_d(i)$ are the production rate, failure rate and repair rate of Machine $M_d(i)$. Machines $M_u(i)$ and $M_d(i)$ are sometimes called *virtual machines* or *pseudo-machines.*

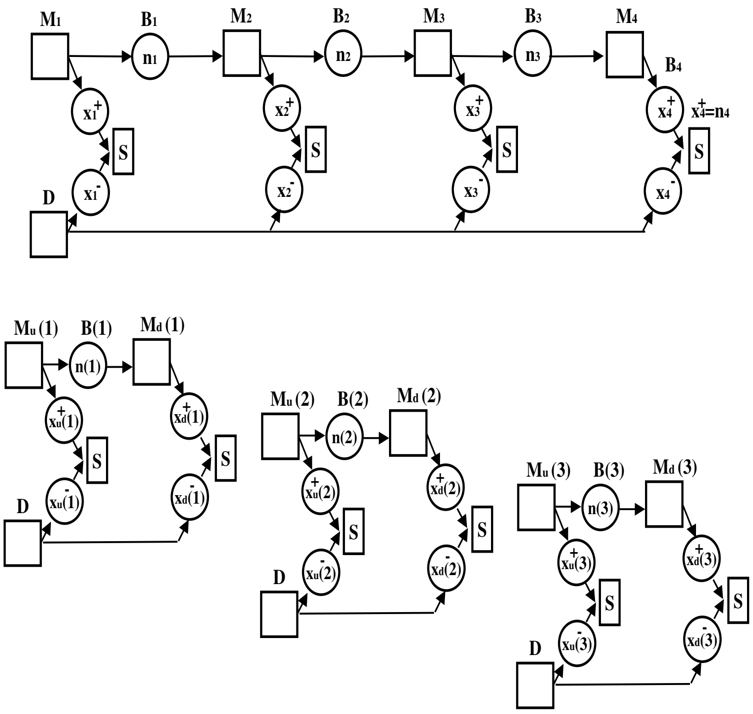

Figure 2: The Decomposition of a L_4 CPP line into three L_2 CPP lines

The values of the $6k$ unknowns $\mu_u(i)$, $p_u(i)$, $r_u(i)$, $\mu_d(i)$, $p_d(i)$ and $r_d(i)$, $i = 1 \ldots k$, are sought such that the behavior of the flow of material in $B(i)$ of CPP line $L_2(i)$ matches as closely as possible that of the flow in B_i of the original line. Furthermore, we want the behavior of $x_u(i)$ and $x_d(i)$ to approximate that of x_i and x_{i+1}. (Note that x_i can be approximated by *both* $x_u(i)$ and $x_d(i-1)$, and this is is considered in the formulation of the decomposition equations.) The analysis of $L_2(i)$

should maintain the loop invariant

$$x_u(i) - x_d(i) - n(i) = 0. \tag{4}$$

The last quantity we are interested in is the service level β. Since the buffer of $L_2(k)$ closely matches B_k, the service level of $L_2(k-1)$, denoted by $\beta(k-1)$ is used to approximate the service level of the original CPP Line, or

$$\beta \quad \approx \quad \beta(k-1) \tag{5}$$

where $\beta(k-1)$ is defined by (3) in the last line $L_2(k-1)$.

Having defined the the $6k$ unknowns, we now derive $6k$ equations that these unknowns must satisfy.

3.2 Parameters of the Two-Machine CPP Line

The failure and repair processes of the virtual machines $M_u(i)$ and $M_d(i)$ of $L_2(i)$ capture the interruption of upstream arrival processes, and the interruption of the departure process from Buffer B_i. We say that $M_u(i)$ is down if material is not flowing into B_i because of a failure upstream. More precisely (following the approach of Gershwin, 1994), we state that $M_u(i)$ is *down* (equivalently, $\alpha_u(i) = 0$) if

- M_i is down, or
- $n_{i-1} = 0$ and $M_u(i-1)$ is down.

and $M_u(i)$ is *up* ($\alpha_u(i) = 1$) otherwise.

Note that $M_u(i)$ can fail while $x_u(i) = Z_i$. (For example, consider what an observer in B_i sees when $x_i = Z_i$, $n_{i-1} = 0$ and M_{i-1} is up, and then M_{i-1} fails.) We can therefore not say that the failures of $M_u(i)$ are operation dependent, and taking this into account would make the decomposition equations messy. But $M_u(i)$ down with $x_u(i) = Z_i$ only happens when $x_{i-1} < Z_{i-1}$, $x_i = Z_i$, and $n_{i-1} = 0$, which does not occur frequently. Hence we assume in the following that $M_u(i)$ down implies that $x_u(i) = Z_i$. This considerably simplifies the method without significantly worsening its performance. (Note that when $Z_{i-1} = Z_i$, the assumption is exact.)

Similarly we say that $M_d(i)$ is *down* ($\alpha_d(i) = 0$) if

- M_{i+1} is down, or
- $n_{i+1} = C_{i+1}$ and $M_d(i+1)$ is down

and $M_d(i)$ is *up* ($\alpha_d(i) = 1$) otherwise. Again, $M_d(i)$ fail if $x_d(i) = Z_{i+1}$, but we ignore this event because its probability is small.

The parameters of the virtual machines are

$$
\begin{aligned}
r_u(i)dt &= P[\alpha_u(i;t+dt)=1|\alpha_u(i;t)=0, n_i(t)<C_i, x_i(t)<Z_i] &(6)\\
p_u(i)dt &= P[\alpha_u(i;t+dt)=0|\alpha_u(i;t)=1, n_i(t)<C_i, x_i(t)<Z_i] &(7)\\
\mu_u(i)dt &= P[n_i(t+dt)=n+1|\alpha_u(i;t)=1, n_i(t)=n<C_i, x_i(t)<Z_i] &(8)\\
r_d(i)dt &= P[\alpha_d(i;t+dt)=1|\alpha_d(i;t)=0, n_i(t)>0, x_{i+1}(t)<Z_{i+1}] &(9)\\
p_d(i)dt &= P[\alpha_u(i;t+dt)=0|\alpha_u(i;t)=1, n_i(t)>0, x_{i+1}(t)<Z_{i+1}] &(10)\\
\mu_d(i)dt &= P[n_i(t+dt)=n-1|\alpha_u(i;t)=1, n_i(t)=n>0, x_{i+1}(t)<Z_{i+1}] &(11)
\end{aligned}
$$

where $\alpha_u(i,t)=1$ indicates that $M_u(i)$ is up at time t and $\alpha_u(i,t)=0$ indicates that $M_u(i)$ is down up at time t; and $\alpha_d(i,t)$ is defined similarly.

We also denote by $e_u(i)$ and $e_d(i)$ the *efficiency* of M_u and M_d as

$$
\begin{aligned}
e_u(i) &= \frac{r_u(i)}{r_u(i)+p_u(i)}\\
e_d(i) &= \frac{r_d(i)}{r_d(i)+p_d(i)}.
\end{aligned}
$$

The arrival rate for $L_2(i)$ is the same as the arrival rate of the demand in the original system. Therefore,

$$\lambda dt = P[x_d(i,t+dt)=x-1|x_d(i,t)=x] \tag{12}$$

According to (4), x_d and x_u decrease at the same time when a demand arrives. λ could also be defined replacing x_d with x_u in (12).

3.3 Resumption and Interruption of Flow

We present two sets of equations linking the parameters of $L_2(i-1)$, $L_2(i)$ and $L_2(i+1)$. The resumption of flow equations capture the relationship between the repair probabilities, while interruption of flow equations describe the link between the failure probabilities. To end up with these equations we generalize in a straightforward way the derivation in Gershwin (1994) for the pure transfer line case, adding events corresponding to the control such as $\{X_i < Z_i\}$.

3.3.1 Interruption of Flow

The upstream Interruption of Flow equation is the corrected form of Equation (4.31) of Gershwin (1994). (For the correction, see `web.mit.edu/manuf-sys/www/gershwin.errata.html`.)

$$p_u(i) = p_i\frac{E_u(i)-P^{011}(i-1)}{E_u(i)} + \frac{r_u(i-1)P^{001}(i-1)}{E_u(i)}. \tag{13}$$

Similarly, the downstream Interruption of Flow equation is the corrected form of (4.32):

$$p_d(i) = p_{i+1}\frac{E_d(i) - P^{C11}(i+1)}{E_d(i)} + \frac{r_d(i+1)P^{C10}(i+1)}{E_d(i)}. \tag{14}$$

where

$$\begin{aligned}
P^{011}(i) &= P[n(i)=0, \alpha_u(i)=1, \alpha_d(i)=1, x_d(i) < Z_{i+1}] && (15)\\
P^{001}(i) &= P[n(i)=0, \alpha_u(i)=0, \alpha_d(i)=1, x_d(i) < Z_{i+1}] && (16)\\
P^{C11}(i) &= P[n(i)=C_i, \alpha_u(i)=1, \alpha_d(i)=1, x_u(i) < Z_i] && (17)\\
P^{C10}(i) &= P[n(i)=C_i, \alpha_u(i)=1, \alpha_d(i)=0, x_u(i) < Z_i] && (18)\\
E_u(i) &= P[n(i) < C_i, \alpha_u(i)=1, x_u(i) < Z_i] && (19)\\
E_d(i) &= P[n(i) > 0, \alpha_d(i)=1, x_d(i) < Z_{i+1}] && (20)
\end{aligned}$$

The production rate of Machine $M_u(i)$ is equal to $\mu_u(i)E_u(i)$, which also approximates the production rate of Machine M_i. But from (2), the production rate of Machine M_i is also equal to λ. Following the same approach for $E_d(i)$, we obtain the following equalities which can be used in (13) and (14):

$$\begin{aligned}
E_u(i) &= \frac{\lambda}{\mu_u(i)} && (21)\\
E_d(i) &= \frac{\lambda}{\mu_d(i)} && (22)
\end{aligned}$$

3.3.2 Resumption of Flow

These equations are exactly the same as those of Gershwin (1994):

$$\begin{aligned}
r_u(i) &= r_u(i-1)X(i) + r_i(1 - X(i)),\ i = 2\ldots k-1 && (23)\\
r_d(i) &= r_d(i+1)Y(i) + r_{i+1}(1 - Y(i)),\ i = 1\ldots k-2. && (24)
\end{aligned}$$

$$\begin{aligned}
X(i) &= \frac{P^{001}(i-1)r_u(i)}{p_u(i)E_u(i)}\\
Y(i) &= \frac{P^{C10}(i+1)r_d(i)}{p_d(i)E_d(i)}.
\end{aligned}$$

$E_u(i)$ and $E_d(i)$ can be evaluated from (21) and (22).

3.4 The Flow Rate-Idle Time Relationship

The production rate of Machine i in the line is equal to its production rate in isolation times the probability that the machine is working in the line. Due to Equation (2), it is also equal to the arrival rate and we have

$$\begin{aligned} \lambda &= e_i \mu_i P[n_{i-1} > 0 \text{ and } n_i < C_i \text{ and } x_i < Z_i] \\ &= e_i \mu_i (1 - P[n_{i-1} = 0 \text{ or } n_i = C_i \text{ or } x_i = Z_i]) \end{aligned} \tag{25}$$

The same equations hold for the upstream and downstream machines of $L_2(i)$

$$\lambda = e_u(i)\mu_u(i)(1 - P[n(i) = C_i \text{ or } x_u(i) = Z_i]) \tag{26}$$

$$\lambda = e_d(i)\mu_d(i)(1 - P[n(i) = 0 \text{ or } x_d(i) = Z_{i+1}]) \tag{27}$$

We then make the assumption (as in Gershwin, 1994) that the probability of B_{i-1} being empty and B_i being full is very small. That is, we approximate

$$P[n_{i-1} = 0 \text{ and } n_i = C_i] = 0. \tag{28}$$

We can write Equation (25) in two different ways:

$$\lambda = e_i \mu_i (1 - P[n_{i-1} = 0 \text{ or } x_i = Z_i] - P[n_i = C_i, x_i < Z_i]) \tag{29}$$

$$\lambda = e_i \mu_i (1 - P[n_i = C_i \text{ or } x_i = Z_i] - P[n_{i-1} = 0, x_i < Z_i]) \tag{30}$$

This is because, for three events A, B and C,

$$P[A \cup B \cup C] = P[\{A \cup C\} \cup \{B \cap \overline{C}\}] \tag{31}$$

$$= P[A \cup C] + P[B \cap \overline{C}] - P[A \cap B \cap \overline{C}]. \tag{32}$$

If $A = \{n_{i-1} = 0\}$, $B = \{n_i = C_i\}$, and $C = \{x_i = Z_i\}$, this becomes Equation (29). If for $A = \{n_i = C_i\}$, $B = \{n_{i-1} = 0\}$, and $C = \{x_i = Z_i\}$, it is Equation (30). In both cases, (28) implies that $P[A \cap B] = 0$.

The behavior of the surplus x_i of machine i is imitated by both the surplus $x_u(i)$ and by the surplus $x_d(i-1)$. Furthermore, buffer $B(i)$ in $L_2(i)$ behaves similarly to the corresponding buffer B_i in the line. We therefore obtain, from (29) and (30),

$$\lambda = e_i \mu_i (1 - P[n(i-1) = 0 \text{ or } x_d(i-1) = Z_i] - P[n(i) = C_i, x_u(i) < Z_i]) \tag{33}$$

$$\lambda = e_i \mu_i (1 - P[n(i) = C_i \text{ or } x_u(i) = Z_i] - P[n_{i-1} = 0, x_d(i) < Z_i]) \tag{34}$$

From (26) and (27), we obtain

$$P[n(i-1)=0 \text{ or } x_d(i-1)=Z_i] \quad = \quad 1-\frac{\lambda}{e_d(i-1)\mu_d(i-1)} \tag{35}$$

$$P[n(i)=C_i \text{ or } x_u(i)=Z_i] \quad = \quad 1-\frac{\lambda}{e_u(i)\mu_u(i)} \tag{36}$$

We use (35) in (33) (with i replaced by $i+1$), and (36) in (34), to obtain

$$\mu_d(i) \quad = \quad \frac{1}{e_d(i)} \frac{1}{\frac{1}{\mu_{i+1}e_{i+1}}+\frac{P_b(i+1)}{\lambda}},\ i=1\ldots k-2 \tag{37}$$

$$\mu_u(i) \quad = \quad \frac{1}{e_u(i)} \frac{1}{\frac{1}{\mu_i e_i}+\frac{P_s(i-1)}{\lambda}},\ i=2\ldots k-1. \tag{38}$$

where

$$P_b(i) \quad = \quad P[n(i)=C_i, x_u(i)<Z_i] \tag{39}$$

$$P_s(i) \quad = \quad P[n(i)=0, x_d(i)<Z_{i+1}]. \tag{40}$$

$P_s(i)$ is the probability that Machine $M_d(i)$ is starved when the control policy requires it to produce. Similarly $P_b(i)$ is the probability that Machine $M_u(i)$ is blocked when it should produce. These probabilities, as well as the others that appear in the Interruption of Flow and Resumption of Flow equations, are given by the solution of the two-machine CPP line $L_2(i)$, which is described in Section 4.

3.5 Boundary Conditions

The Resumption of Flow, Interruption of Flow, and the Flow Rate-Idle Time equations are $6(k-1)$ equations. The boundary conditions provide the remaining 6 equations:

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

4 Numerical Solution of the Two-Machine CPP Line

The equations of the decomposition method involve the probabilities of $L_2(i)$ being in particular states, namely $P^{011}(i)$, $P^{001}(i)$, $P^{C11}(i)$, $P^{C10}(i)$, $P_b(i)$ and $P_s(i)$. To evaluate these quantities, we analyze the $L_2(i)$ subsystem. For this purpose, it is most convenient to use the surplus-based formulation of $L_2(i)$. In this section, we describe its state space, evaluate its steady state probability distribution π numerically, and formulate the probabilities we need as functions of π.

The parameters which determine the behavior of line $L_2(i)$ are the two hedging points Z_i, Z_{i+1} and the buffer size C_i. The state of $L_2(i)$ is described by the number of parts $n(i)$ in $B(i)$, and the surpluses $x_d(i)$ and $x_u(i)$ of Machine $M_u(i)$ and Machine $M_d(i)$. Machine $M_u(i)$ produces if $n(i) < C_i$ and $x_u(i) < Z_i$, and $M_d(i)$ produces if $n(i) > 0$ and $x_d(i) < Z_{i+1}$. As a result, we have $x_u(i) \le Z_i$, $x_d(i) \le Z_{i+1}$ and $n(i) \le C_i$. But the population constraint states that $x_u(i) = x_d(i) + n(i)$. The state space is therefore the set of all $(n, x, \alpha_u, \alpha_d)$, where

$$\begin{aligned} 0 \le n \le \min(Z_i - x, C_i), \\ -\infty < x \le \min(Z_{i+1}, Z_i + n), \\ \alpha_u \in \{0,1\}, \text{ and } \alpha_d \in \{0,1\}. \end{aligned}$$

The transition rates of the continuous time Markov chain are obtained from equations (6) to (12), where x_{i+1}, x_i, and n_i are replaced by $x_u(i)$,$x_d(i)$, and $n(i)$. Figure 3 depicts the state space and shows the transition rates when $\alpha_u = \alpha_d = 1$. (The states indicated in gray are those in which $n = 0$ or $x = 0$). When a demand arrives at the system, x decreases. This corresponds to the arrow pointing downward on the figure. When the upstream machine produces a part, n increases (corresponding to the arrow pointing to the right), and when the downstream machine produces a part, n decreases and x increases (corresponding to the upper left diagonal arrow). The steady-state probability distribution $\pi(n, x, \alpha_u, \alpha_d)$ of the system satisfies the following equations. (For the sake of clarity we use r_u, p_u, r_d and p_d for $r_u(i)$, $p_u(i)$, $r_d(i)$ and $p_d(i)$.

Internal equations: For $0 < n < \min(Z_i - x, C_i)$, $-\infty < x < \min(Z_{i+1}, Z_i - n)$, and $\alpha_u \in \{0,1\}$, $\alpha_d \in \{0,1\}$,

$$\pi(n, x, \alpha_u, \alpha_d)(\lambda + \mu_u + \mu_d + \alpha_u p_u + (1-\alpha_u) r_u + \alpha_d p_d + (1-\alpha_d) r_d) =$$

$$\begin{aligned} &\lambda \pi(n, x+1, \alpha_u, \alpha_d) + \alpha_u \mu_u \pi(n-1, x, \alpha_u, \alpha_d) + \alpha_d \mu_d \pi(n+1, x-1, \alpha_u, \alpha_d) \\ &+ [\alpha_u r_u + (1-\alpha_u) p_u] \pi(n, x, 1-\alpha_u, \alpha_d) + [\alpha_d r_d + (1-\alpha_d) p_d] \pi(n, x, \alpha_u, 1-\alpha_d) \end{aligned}$$

Boundary equations: For $n = 0$, $x < Z_{i+1}$,

$$\pi(n, x, \alpha_u, \alpha_d)(\lambda + \mu_u + \alpha_u p_u + (1-\alpha_u) r_u + (1-\alpha_d) r_d) =$$

$$\begin{aligned} &\lambda \pi(n, x+1, \alpha_u, \alpha_d) + \alpha_d \mu_d \pi(n+1, x-1, \alpha_u, \alpha_d) \\ &+ (\alpha_u r_u + (1-\alpha_u) p_u) \pi(n, x, 1-\alpha_u, \alpha_d) + \alpha_d r_d \pi(n, x, \alpha_u, 1-\alpha_d) \end{aligned}$$

For $n = 0$, $x = Z_{i+1}$,

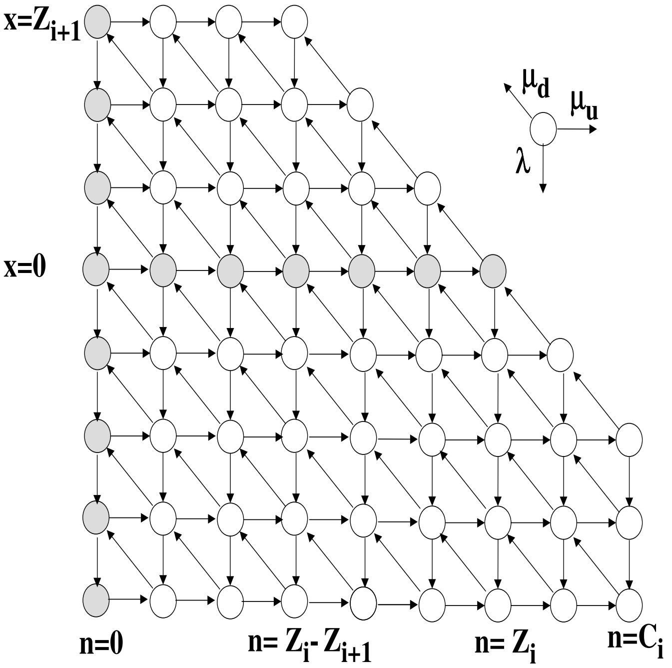

Figure 3: State Space of a Two-Machine CPP Line for $\alpha_u = \alpha_d = 1$

$$\pi(n,x,\alpha_u,\alpha_d)(\lambda + \mu_u + \alpha_u p_u + (1-\alpha_u)r_u + (1-\alpha_d)r_d)) =$$

$$\alpha_d\mu_d\pi(n+1,x-1,\alpha_u,\alpha_d) + [\alpha_u r_u + (1-\alpha_u)p_u]\pi(n,x,1-\alpha_u,\alpha_d)$$
$$+\alpha_d r_d\pi(n,x,\alpha_u,1-\alpha_d)$$

For $0 < n < Z_i - Z_{i+1}$, $x = Z_{i+1}$,

$$\pi(n,x,\alpha_u,\alpha_d)(\lambda + \mu_u + \alpha_u p_u + (1-\alpha_u)r_u + (1-\alpha_d)r_d) =$$

$$\alpha_u\mu_u\pi(n-1,x,\alpha_u,\alpha_d) + \alpha_d\mu_d\pi(n+1,x-1,\alpha_u,\alpha_d) + [\alpha_u r_u + (1-\alpha_u)p_u]\pi(n,x,1-\alpha_u,\alpha_d)$$
$$+\alpha_d r_d\pi(n,x,\alpha_u,1-\alpha_d)$$

For $n = Z_i - Z_{i+1}$, $x = Z_{i+1}$,

$$\pi(n,x,\alpha_u,\alpha_d)(\lambda + (1-\alpha_u)r_u + (1-\alpha_d)r_d) =$$

$$\alpha_u\mu_u\pi(n-1,x,\alpha_u,\alpha_d) + \alpha_d\mu_d\pi(n+1,x-1,\alpha_u,\alpha_d) + \alpha_u r_u\pi(n,x,1-\alpha_u,\alpha_d)$$
$$+\alpha_d r_d\pi(n,x,\alpha_u,1-\alpha_d)$$

For $n = Z_i - x,\ Z_i - C_i < x < Z_{i+1}$,

$$\pi(n,x,\alpha_u,\alpha_d)(\lambda + \mu_d + (1-\alpha_u)r_u + \alpha_d p_d + (1-\alpha_d)r_d) =$$

$$\alpha_u\mu_u\pi(n-1,x,\alpha_u,\alpha_d) + \alpha_d\mu_d\pi(n+1,x-1,\alpha_u,\alpha_d) + \alpha_u r_u\pi(n,x,1-\alpha_u,\alpha_d)$$
$$+[\alpha_d r_d + (1-\alpha_d)p_d]\pi(n,x,\alpha_u,1-\alpha_d)$$

For $n = C_i,\ x = Z_i - C_i$,

$$\pi(n,x,\alpha_u,\alpha_d)(\lambda + \mu_d + (1-\alpha_u)r_u + \alpha_d p_d + (1-\alpha_d)r_d) =$$

$$\alpha_u\mu_u\pi(n-1,x,\alpha_u,\alpha_d) + \alpha_u r_u\pi(n,x,1-\alpha_u,\alpha_d) + [\alpha_d r_d + (1-\alpha_d)p_d]\pi(n,x,\alpha_u,1-\alpha_d)$$

For $n = C_i,\ x < Z_i + C_i$,

$$\pi(n,x,\alpha_u,\alpha_d)(\lambda + \mu_d + (1-\alpha_u)r_u + \alpha_d p_d + (1-\alpha_d)r_d) =$$

$$\lambda\pi(n,x+1,\alpha_u,\alpha_d) + \alpha_u\mu_u\pi(n-1,x,\alpha_u,\alpha_d) + \alpha_u r_u\pi(n,x,1-\alpha_u,\alpha_d)$$
$$+[\alpha_d r_d + (1-\alpha_d)p_d]\pi(n,x,\alpha_u,1-\alpha_d)$$

From the steady state probabilities $\pi(n,x,\alpha_u,\alpha_d)$ of this system, we can compute the probabilities which appear in the decomposition equations. By definition (Equations (15) to (18) and (39) and (40),

$$P^{011}(i) = \sum_{x=-\infty}^{Z_{i+1}-1} \pi(0,x,1,1) \tag{41}$$

$$P^{001}(i) = \sum_{x=-\infty}^{Z_{i+1}-1} \pi(0,x,0,1) \tag{42}$$

$$P^{C11}(i) = \sum_{x=-\infty}^{Z_i-C_i-1} \pi(C_i,x,1,1) \tag{43}$$

$$P^{C10}(i) = \sum_{x=-\infty}^{Z_i-C_i-1} \pi(C_i,x,1,0) \tag{44}$$

$$P_b(i) = \sum_{x=-\infty}^{Z_i-C_i-1} \sum_{\alpha_u=0}^{1} \sum_{\alpha_d=0}^{1} \pi(C_i,x,\alpha_u,\alpha_d) \tag{45}$$

$$P_s(i) = \sum_{x=-\infty}^{Z_{i+1}-1} \sum_{\alpha_u=0}^{1} \sum_{\alpha_d=0}^{1} \pi(0,x,\alpha_u,\alpha_d) \tag{46}$$

$$\tag{47}$$

Note that the states $(0,x,\alpha_u,0)$ are transient since they can be reached at most from the states $(0,x,1-\alpha_u,0)$ and $(0,x+1,\alpha_u,0)$. For similar reasons, $(C_i,x,0,\alpha_d)$ are transient as well. Hence we can rewrite (46) and (45) as

$$P_s(i) = P^{011}(i) + P^{001}(i) \tag{48}$$
$$P_b(i) = P^{C11}(i) + P^{C10}(i) \tag{49}$$

The service level of the last machine is computed from the steady state probability distribution π of $L_2(k-1)$:

$$\beta(k-1) = \sum_{x=1}^{Z_{i+1}} \sum_{n=0}^{m(x)} \sum_{\alpha_u=0}^{1} \sum_{\alpha_d=0}^{1} \pi(n, x, \alpha_u, \alpha_d) \tag{50}$$
$$\text{where } m(x) = \min(Z_{i+1} - x, C_i).$$

This summation is over all states such that $x \geq 1$.

There are no existing analytical results for probability distribution π. We therefore compute the probability distribution numerically on a truncated state space. (For the sake of simplicity we use the same notation π for both the probability distribution defined on the entire state space and the probability distribution defined on the truncated space). We specify a maximum number of demands that can be waiting in the system x_{max}, and we limit the state space by requiring that $x \geq -x_{max}$. If there are x_{max} waiting demands in the system, any new demand is lost. At x_{max}, the internal equations are rewritten as follows:

$$\pi(n, x_{max}, \alpha_u, \alpha_d)(\mu_u + \mu_d + \alpha_u p_u + (1-\alpha_u) r_u + \alpha_d p_d + (1-\alpha_d) r_d) =$$

$$\lambda \pi(n, x_{max}+1, \alpha_u, \alpha_d) + \alpha_u \mu_u \pi(n-1, x, \alpha_u, \alpha_d) + [\alpha_u r_u + (1-\alpha_u) p_u] \pi(n, x_{max}, 1-\alpha_u, \alpha_d)$$
$$+ [\alpha_d r_d + (1-\alpha_d) p_d] \pi(n, x_{max}, \alpha_u, 1-\alpha_d)$$

and the boundary equations in x_{max} are modified in a similar way.

We compute π using a standard iterative algorithm (Stewart, 1994) that we denote by A_π. This algorithm consists in computing

$$\pi^{m+1} = [Q\delta t + I]\pi^m,$$

where Q is the generator matrix, δt a small time, and I the identity matrix. We chose $\delta t = 0.99/(\mu_u + \mu_d + \lambda + r_u + r_d + p_u + p_d)$. x_{max} is taken sufficiently large such that

$$\sum_{n,\alpha_u,\alpha_d} \pi(n, -x_{max}, \alpha_u, \alpha_d) < 10^{-4}.$$

(x_{max} is guessed. If this condition is not satisfied, x_{max} is increased and A_π is repeated.)

The iterative algorithm stops when the relative error of π between two consecutive iterations is less than 10^{-4}. As for all iterative algorithms, a key feature that determines the speed of the convergence of A_π is the initial guess π^0.

5 Solution of the Decomposition Equations

In this section, we propose an algorithm to compute the service level of a CPP Line with k machines. This algorithm solves the decomposition equations and then evaluates $\beta(k-1)$. The algorithm is similar to the fixed point iteration proposed by Dallery, David and Xie (1988), so we call it a DDX algorithm.

In this version of the DDX algorithm, one important step is to numerically solve $L_2(i)$ with Algorithm A_π. For the initial condition of A_π, we use the steady state probability of $L_2(i)$ computed in the previous iteration of DDX. Hence, the convergence of A_π and, as a result, the convergence of DDX are very fast.

The algorithm converges when the relative differences in all the parameters we need from the L_2 lines, between two consecutive iterations, are less than 10^{-4}. We then compute the service level of the CPP line.

Algorithm

- **Set** $r_u(i) = r_i$, $p_u(i) = p_i$, $\mu_u(i) = \mu_i$ for $i = 1, \ldots, k-1$

  and $r_d(i) = r_{i+1}$, $p_d(i) = p_{i+1}$, $\mu_d(i) = \mu_{i+1}$ for $i = 2, \ldots, k$.

  Also, choose an initial guess for all the probability distributions of $L_2(i), i = 1, \ldots, k-1$.

- **Repeat**

  1. for $i = 2$ to $k-1$
     (a) Find the steady state probability distribution $L_2(i)$ using A_π, with the the final distribution from the previous iteration as the initial guess.
     (b) Calculate $P^{011}(i)$, $P^{001}(i)$, and $P_s(i)$ using (41), (42) and (48).
     (c) Compute $E_u(i)$ from (21) and using $\mu_u(i)$ from the previous iteration.
     Compute $p_u(i)$ from (13).
     Compute $r_u(i)$ from (23) where $X(i)$ is obtained using $r_u(i)$ from the previous iteration.
     Compute $\mu_u(i)$ from (38).
  2. for $i = 1$ to $k-2$
     (a) Find the steady state probability distribution $L_2(i)$ using A_π, with the the final distribution from the previous iteration as the initial guess.
     (b) Calculate $P^{C11}(i)$, $P^{C10}(i)$, and $P_b(i)$ using (43), (44) and (49).
     (c) Compute $E_d(i)$ from (22) and using $\mu_d(i)$ from the previous iteration.
     Compute $p_d(i)$ from (14).
     Compute $r_d(i)$ from (24) where $Y(i)$ is obtained using $r_d(i)$ from the previous iteration.
     Compute $\mu_d(i)$ from (37).

Case	p_1	r_1	p_2	r_2	p_3	r_3	p_4	r_4	p_5	r_5	Z_5
1	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5	8
2	0.2	0.5	0.1	0.7	0.2	0.5	0.1	0.7	0.2	0.5	6

$\mu_1 = \mu_4 = \mu_3 = \mu_4 = \mu_5 = 1$ and $\lambda = 0.4$

Table 1: Parameters of the Homogeneous Cases

Until

the relative differences between the current and the previous iteration values of
$\mu_u(i)$, $r_u(i)$, $p_u(i)$ for $i = 1, \ldots, k-1$ and of
$\mu_d(i)$, $r_d(i)$, $p_d(i)$ for $i = 2, \ldots, k$,
are less than 10^{-4}.

- **Return** $\beta(k)$ using (50).

6 Numerical Results

In this section, we study the accuracy of the approximation technique. The results of the decomposition method were compared to simulation results for many examples. Each simulation consisted of 20 independent replications, where a replication ran for 10^5 time units, after a warm-up period of 10^3 time units. Typically, the confidence intervals in our examples are such that $\beta \in [\bar{\beta} - 0.003, \bar{\beta} + 0.003]$ with probability .95, where $\bar{\beta}$ is the sample average. In most of the cases we studied, the time to convergence of the decomposition method is less than one minute, while the simulation ran for 15-20 minutes.

First, we present some typical results obtained for the homogeneous case, when the line is balanced or unbalanced. Results for non-homogeneous lines are then described. Finally, we present comparisons of buffer level evaluations.

6.1 Balanced Homogeneous Line

In this example we consider a line of five identical machines. The production rate of each machine is equal to 1 and the demand arrival rate is equal to 0.4. The failure and repair rates of the machines are described in Table 1 (Case 1). The buffer capacities are all equal to C. We assume that $Z_i - Z_{i+1} = \Delta Z$ does not depend on i for $i \in \{1, \ldots, 4\}$. Hence, given Z_5, all the hedging points can be computed recursively. Here, we fix $Z_5 = 8$. Then, if $\Delta Z = 3$, we have

$$Z_5 = 8, Z_4 = 11, Z_3 = 14, Z_2 = 17, Z_1 = 20.$$

We study the effects of C and ΔZ on the accuracy of the decomposition method.

Figure 4 presents the service level β obtained by the simulation and the decomposition method, for increasing values of C when $\Delta Z = 3$. Recall that the capacity of the last buffer is constrained

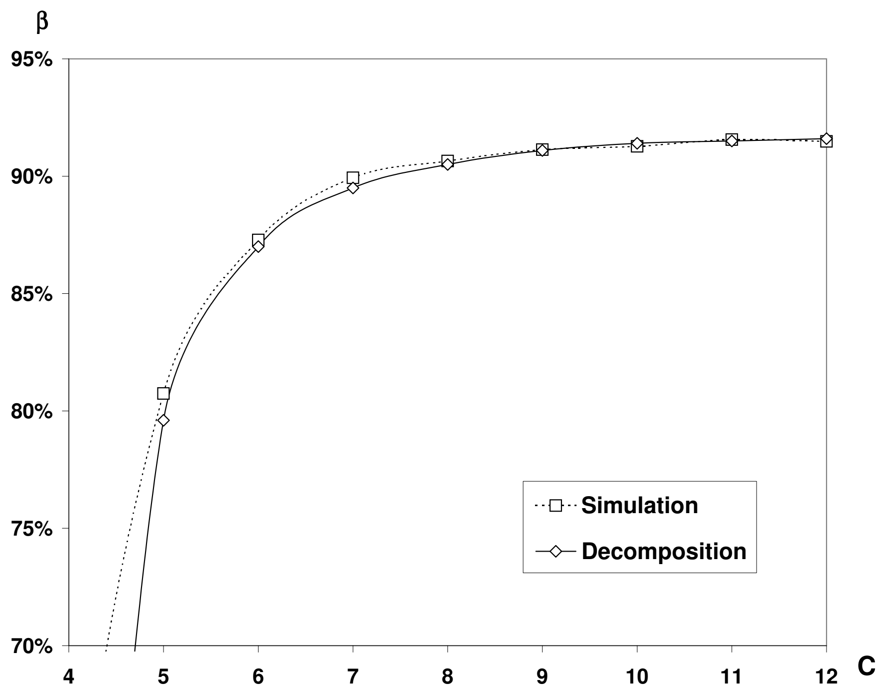

Figure 4: Service Level β as a Function of the Buffer Sizes C for Case 1

by Z_5, so that for all C, the maximum number of parts in the last buffer remains equal to 8. (See Section 2.2.1.) The shape of the simulation curve is well approximated by the shape of the decomposition curve. Both curves are increasing and reach a finite limit when C is large. This limit corresponds to the service level of a pure basestock policy where no buffer capacity constraint is specified. Furthermore, when $C \geq 5$, our method shows very good accuracy: the relative error between simulation and decomposition method is less than 1.5%. When $C < 4$, both the simulation and the decomposition method establish that the system is not stable, and is not able to satisfy arriving demands with the inventory of finish goods. When $C = 4$, the system is almost saturated. In this case the decomposition method converges slowly and does not perform very well. This is the only value of C for which the method is not accurate.

Figure 5 presents the service levels obtained by the simulation and the decomposition method, for increasing value of ΔZ, when $C = 8$. Note that for $\Delta Z \geq 8$, the service level is constant since the control policy is equivalent to the control policy where $Z_i = Z_{i+1} + C$, $i = 4 \ldots 1$. (See Section 2.2.1.) Again, the increasing shape of the decomposition curve accurately reflects the shape of the simulation curve. Furthermore, when $\Delta Z \geq 3$ the relative errors between simulation and decomposition method are less than 1%. When $\Delta Z = 2$, the relative error is equal to 1.9%.

When $\Delta Z = 0$, the control policy reduces to a single-stage basestock policy, but the decomposition method is not accurate in this case. This is because all surpluses x_i are sometimes at their hedging points Z_i at the same time. In this case, since $n_i = Z_{i+1} - Z_i = 0$ for $1 \leq i < 5$, all intermediate buffers are empty, and the last buffer contains $Z_5 = 8$ parts. As soon as a demand arrives at the system, the policy requires each machine to produce. But, machines M_2 to M_5 are

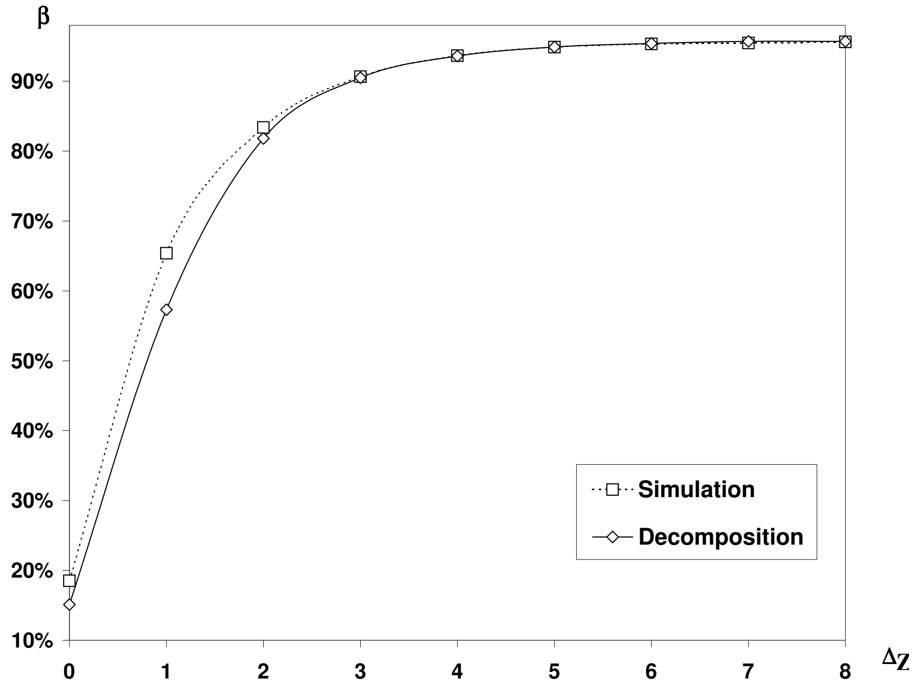

Figure 5: Service Level β as a Function of ΔZ for Case 1

starved. This situation occurs frequently in this case since the system tends to maintain the surpluses at their hedging points. Hence, the last four machines are often starved, even though no upstream machine has failed. As a result, approximating the virtual machines' times to produce with exponential random variables is not accurate and the decomposition method does not perform well when the control policy is very close to a single-stage basestock policy ($\Delta Z \leq 1$). More refined decomposition methods may improve the algorithm, for example methods like those of Dallery and Le Bihan (1999) or Tolio and Matta (1998).

6.2 Unbalanced Homogeneous Line

In this section, the machines are not identical. Compared to Case 1, the efficiencies of Machine M_2 and Machine M_4 are increased by reducing their failure rates and increasing their repair rates. The parameters of the machines are presented in the last row (Case 2) of Table 1. We still have $\lambda = 0.4$, and all production rates μ_i are equal to 1.

Figure 6 presents the service level β for increasing values of C, when $\Delta Z = 3$. The results of the decomposition method are very accurate for $C \geq 4$. The relative error is less than 1.5%. When $C < 3$, the simulation shows that the system is not stable. Since we have increased the efficiency of two machines, it is not surprising that the system is stable for lower values of C compared to Case 1. When $C = 3$, the system is almost saturated, and the decomposition method converges slowly and does not perform well.

The decomposition method gives optimistic results. This is because Machines 2 and 4 have

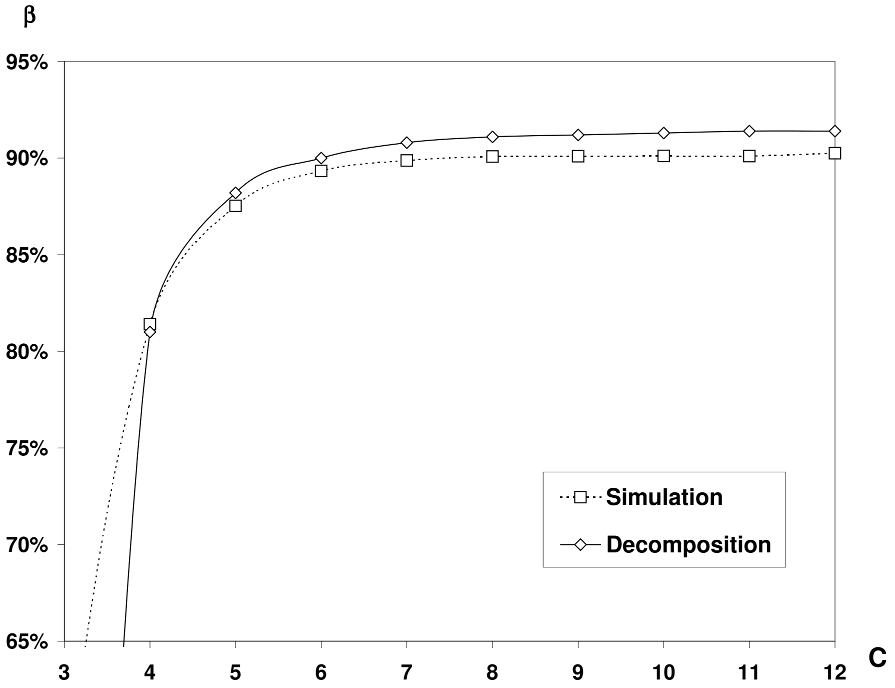

Figure 6: Service Level as a Function of the Buffer Sizes for Case 2

larger efficiencies than the other machines in the line. As a consequence, the probability that buffer B_1 is empty and buffer B_2 is full at the same time is not very small (and similarly for probability of B_3 being empty and B_4 being full simultaneously). The approximation we made in (28) is therefore not valid, but the relative error is still less than 1.5%. A similar phenomenon occurs in the decomposition technique to compute the throughput of a flow line with a finite buffer control policy, when one machine is much more efficient than its neighbors (Gershwin, 1994).

Figure 7 presents the service levels obtained by simulation and the decomposition method, for increasing values of ΔZ, when $C = 6$. Again, for $\Delta Z \geq 6$, the service level is constant. The decomposition method is still accurate: for $\Delta Z \geq 1$ the relative error is less than 2.3%. When $\Delta Z = 0$, however, the method is less accurate. In this case, the control policy is close to a single-stage basestock policy and the exponential parameters of the virtual machines may not be sufficient to represent the rest of the line in the subsystem. The discussion in Section 6.1 applies here as well.

6.3 Non-homogeneous Line

In this section, the machines have different production rates. We consider the production line of the previous section (Case 1 of Table 1), and we fix $\Delta Z = 3$ and $C = 8$. As before, μ_1, μ_3 and μ_5 are equal to 1. We vary μ_2 and μ_4 according to $\mu_2 = \mu_4 = \mu$.

Figure 8 depicts the service levels obtained by simulation and the decomposition method for increasing values of μ. The decomposition method gives good results, and the relative error is less than 2%. For $\mu = 0.6$ the system is not stable.

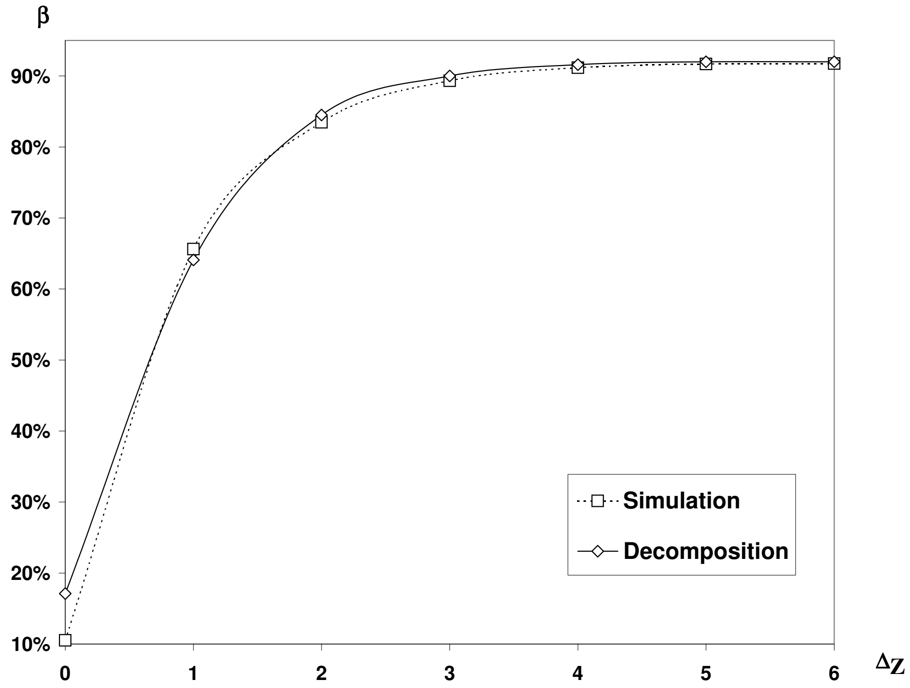

Figure 7: Service Level as a Function of ΔZ for Case 2

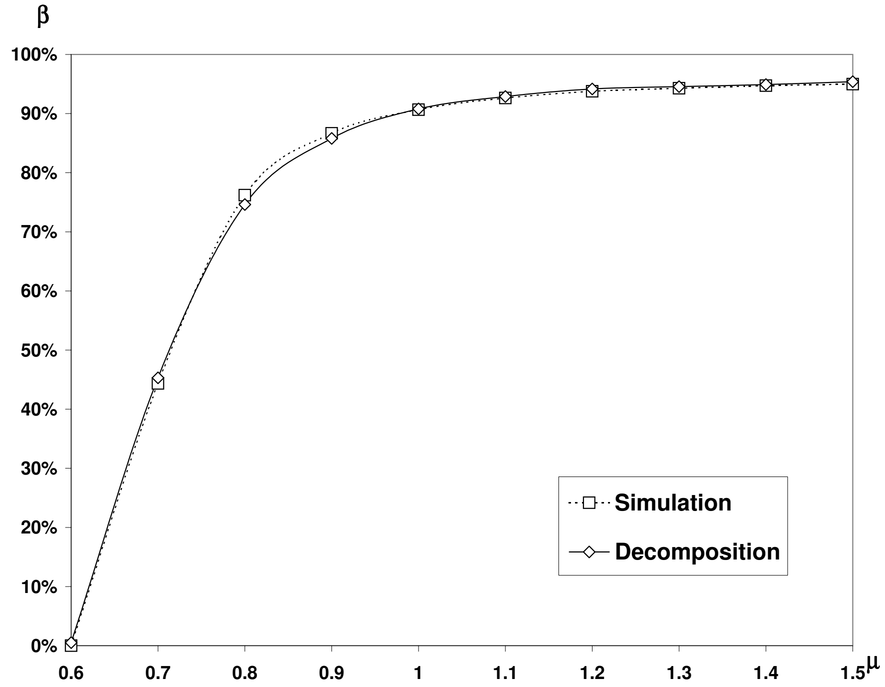

Figure 8: Service Level as a Function of $\mu = \mu_2 = \mu_4$ for a Non-homogeneous Case

Case 1	$\bar{n}_1$	$\bar{n}_2$	$\bar{n}_3$	$\bar{n}_4$	$\bar{x}_5^+$	β
Simulation	3.4	3.18	2.22	3.7	5.7	0.905
Decomposition	3.42	3.22	3.1	2.92	5.6	0.908
%error	-1%	-1%	-39%	21%	2%	-0.30%
Case 2	$\bar{n}_1$	$\bar{n}_2$	$\bar{n}_3$	$\bar{n}_4$	$\bar{x}_5^+$	β
Simulation	2.91	3.18	1.77	3.93	4.23	0.89
Decomposition	2.87	3.39	2.74	3.30	4.32	0.91
%error	1%	-7%	-55%	16%	-2%	-1.83%

Table 2: Average Buffer Levels

6.4 Average Buffer Levels

Here, we illustrate the accuracy of the decomposition method for calculating average buffer levels. Table 2 presents the approximation and simulation results corresponding to Case 1 and Case 2 of Table 1. We fix $\Delta Z = 3$, and $C = 8$ for Case 1, $C = 6$ for Case 2. The results are not as accurate as those for the service level. This effect has already been observed in the decomposition of a production line with a finite buffer control policy (Dallery and al., 1988; Frein et al., 1996; Bonvik, 1996; Tolio and Gershwin, 1998).

7 Conclusions and Future Research

In this paper, we have introduced a decomposition approximation for the performance evaluation of an exponential model of a make-to-stock line controlled by the control point policy. The accuracy of the method is similar to the decomposition method developed to compute the throughput of a finite-buffer production line (Gershwin, 1994). For typical cases, the iterative algorithm converges quickly and provides good results. This suggests that the proposed technique can be used as part of an algorithm to optimize the parameters of the CPP (as it as been done for the finite-buffer production line by Schor and Gershwin, 2000).

On the other hand, although we have illustrated our approach for the exponential model, it can be extended to the continuous flow model (Dallery et al., 1989), or to multi-failure mode line models (Tolio and Matta, 1998). This would improve the accuracy of our method, since the techniques developed for these cases already improve the analysis of simple transfer lines. Moreover, this could allow us to refine the virtual machines we have introduced here, so that our method can perform better when two consecutive hedging points are equal or almost equal.

The method can also be directly extended for demand which arrives according to more complex processes than the Poisson process. For instance, the decomposition equations do not change if we assume phase-type distributions for the time between two demand arrivals.

Finally, our method is the first approach using decomposition techniques to evaluate the performance of a control policy with two parameters per stage. This type of policy is difficult to analyze since it introduces multiple information loops that existing decomposition methods have not analyzed. Our approach resolves this problem for the CPP, and should provide a good starting point for evaluating other two-parameter control policies.

Acknowledgments

This paper is based upon work partially supported by the National Science Foundation Grant DMI-9713500, and the Semiconductor Research Corporation under contract No. 97-FJ-493. I am also grateful for the financial support that has been provided by the Lean Aircraft Initiative and the MIT Leaders for Manufacturing Program.

References

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

[2] A. M. Bonvik. *Performance Analysis of Manufacturing Systems Under Hybrid Control Policies.* PhD thesis, Massachusetts Institute of Technology, Operations Research Center, June 1996.

[3] M. H. Burman. *New Results in Flow Line Analysis.* PhD thesis, Massachusetts Institute of Technology, Operations Research Center, June 1995.

[4] J. A. Buzacott and J. G. Shantikumar. *Stochastic Models of Manufacturing Systems.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[5] Y. Dallery and H. Le Bihan. An improved decomposition method for the analysis of production lines with unreliable machines and finite buffers. *International Journal of Production Research*, 37(5):1093–1117, 1999.

[6] Y. Dallery and G. Liberopoulos. Extended kanban control system: combining kanban and base stock. *IIE Transactions on Design and Manufacturing*, 32, 2000.

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

[8] Yves Dallery, René David, and Xiao-Lan Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on automatic control*, 34(9):943–953, 1989.

[9] F. de Vericourt, F. Karaesmen, and Y. Dallery. Dynamic scheduling in a make-to-stock system: A partial characterization of optimal policies. *Operations Research*, to appear in Vol. 48(5), 2000.

[10] Maria Di Mascolo, Yannick Frein, and Yves Dallery. An analytical method for performance evaluation of kanban controlled production systems. *Operations Research*, 44(1):50–64, 1996.

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

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

[13] S. B. Gershwin. Design and operation of manufacturing systems - the control-point policy. *IIE Transactions*, 2000.

[14] Stanley B. Gershwin and James E. Schor. Efficient algorithms for buffer space allocation. *Annals of Operations Research*, 2000.

[15] O. Gzouli. Comparison of scheduling policies by simulation. Master's thesis, Massachusetts Institute of Technology, 2000.

[16] A. Ha. Optimal dynamic scheduling policy for a make-to-stock production system. *Operations Research*, 45, 1997.

[17] F. Karaesmen and Y. Dallery. A performance comparison of pull type control mechanisms for multi- stage production control. *International Journal of Production Economics*, to appear, 1999.

[18] George Liberopoulos and Yves Dallery. A unified framework for pull control mechanisms in multi-stage manufacturing systems. Unpublished manuscript, 1997.

[19] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton, 1994.

[20] T. Tolio and S. B. Gershwin. Throughput estimation in cyclic queueing networks with blocking. *Annals of Operations Research*, 79, March 1998.

[21] T. Tolio and A. Matta. A method for performance evaluation of automated flow lines. *Annals of the CIRP*, 47, 1998.

[22] M. H. Veatch and L. M. Wein. Optimal control of a two-station tandem production/inventory system. *Operations Research*, 42(2):337–350, 1994.