Design and Operation of Manufacturing Systems — The Control-Point Policy*

Stanley B. Gershwin
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307
gershwin@mit.edu
http://web.mit.edu/manuf-sys/www

December 19, 1999

Abstract

This paper synthesizes recent work by the author and his colleagues on the performance analysis of manufacturing systems and on real-time scheduling methods. It describes a decentralized scheduling/control policy and proposes a method for analyzing the performance of systems that are operated according to the policy. It presents three views of the production system: surplus-based, time-based, and token-based; and shows how they are equivalent in special cases. While there are no assurances of optimality, experience suggests that the policy has desirable characteristics. The paper also suggests how the physical resources and the control policy can be designed together.

1 Introduction

1.1 Goal of Paper

THE design and the operation of manufacturing systems are of great economic importance. In spite of the considerable literature on these topics, and the long history of manufacturing, there is no widespread agreement on how they can best be done. There are many academic approaches (queuing theory, dynamic programming, combinatorial optimization, etc.) and many approaches that are advocated among practitioners and consultants, but factory performance remains unpredictable.

*to appear, *IIE Transactions on Design and Manufacturing*, Special Issue on Decentralized Control of Manufacturing Systems.

Two of the major issues treated in the manufacturing systems research literature are performance analysis and decision and control policies. In this paper, we combine earlier work by the author and his colleagues on the decomposition method for factory performance prediction and design and the hedging point method for scheduling. We propose an approach that comprises design and operation in a unified manner. The method will eventually include algorithms for selecting buffer (in-process inventory storage) sizes and operational parameters.

The approach combines the following elements:

- a decentralized control policy for making scheduling decisions (which part to load on which machine, when) in real time, in response to random events such as machine failures, part or operator absence, etc.;
- a method for predicting the performance of the factory when operated according to that policy;
- a method for optimizing the parameters of the policy, including the sizes of the in-process inventory buffers;
- a method for factory design, ie, resource selection.

The approach is far from complete, and much work remains before it can be fully implemented. However, the first item has been defined and successfully implemented (with rough estimates for the parameters); the second is a subject of on-going research; preliminary work on the third is very encouraging; and work on the fourth has not begun in this context. This paper focuses on the first item and on the relationship between the first and second.

1.2 Background

1.2.1 Control Policies

There is a substantial literature on control policies for manufacturing systems. This literature includes research on simulation studies of specific policies on specific systems; queueing theoretical analysis of performance, stability, and optimal controls; fluid approximations of discrete systems; and many other relevant topics. Time and space limitations prevent an extensive discussion here. See Wein (1988), Dai and Meyn (1995), Altiok (1996), Bonvik (1996), Bonvik, Couch, and Gershwin (1997), and Meyn (1997) for surveys.

Policy classification We may classify many policies as *token-based*, *time-based*, or *surplus-based*. In surplus-based systems, decisions are made on the basis of how far cumulative production is ahead of, or behind, cumulative demand. The policy described here is first formulated as such a policy. Hedging point, two-boundary, and (one might argue) base stock policies are based on surplus and backlog. Much of the surplus-based literature is summarized in Section 2.2.

Time-based systems, including earliest due date (EDD), least slack (LS), and material requirements planning (MRP) operate on the basis of time. For example, EDD makes choices on the basis of the due dates of the parts available to be worked on. MRP systems attempt to determine the times when operations should take place, and it is up to the factory to make those events happen at those times.

Token-based systems, of which kanban is the most wide-spread and either Production Authorization Card — PAC — (Buzacott and Shanthikumar, 1993) or Extended Kanban Control System (Dallery and Liberopoulos, 1997, 2000; Liberopoulos and Dallery, 1997) may be the most general, involve the movement of tokens in the manufacturing system to trigger events. When an operation is performed or a demand arrives, a token is either created or taken from a specific location and then either destroyed or moved to another specific location. Only when a token exists at the first location (and only if space for tokens exist at the second) can the operation take place.

Wein observations In an extensive series of experiments on simulations of semiconductor fabrication facilities, Wein (1988) found that the most important factor in determining the performance of a factory was the process by which material is released into the system (discretionary input control). The decisions that are made once the material is inside (lot sequencing) are less important (assuming that they are reasonable).

The control point policy described here may be viewed as a consequence of Wein's observation: since the release policy is the most important factor in factory performance, the best policy is one that views all or many points as release points to the rest of the factory.

1.2.2 Decomposition

Decomposition is an analytical approximation technique for evaluating performance measures (especially production rate, average buffer levels, and probabilities of blockage and starvation) of queuing networks (especially manufacturing systems) for which exact analytic methods do not exist, particularly systems with finite buffers. It was developed for tandem systems in Gershwin (1987) and improved in Dallery, David and Xie (1988). It was extended to assembly/disassembly systems in Di Mascolo, David, and Dallery (1991) and Gershwin (1991). It is reviewed in Dallery and Gershwin (1992) and discussed in detail in Gershwin (1994) and Burman (1995). It is rapid and accurate for these systems (although no proof of convergence or bounds on accuracy is known). Recent work has extended these methods to systems with yield losses and rework (Helber, 1997, 1999; Dallery, 1999).

1.3 Control Point Policy Overview

The control point policy is a generalization of the hedging point or two-boundary policies. The policy limits the flow of material into a system, or further downstream if it is already in the system. There are two limitations: the first is a limit on surplus (as in the hedging point policy) or, equivalently, earliness or slack in a time-based policy. Parts are not allowed beyond a control point if the excess of production at that point over demand is too great. The second is an inventory limitation. Parts are not allowed in, or further downstream, if there are already too many of that part type in the system.

Parts are rank-ordered. If more than one part survives these two filters, the highest ranking part is chosen. This mechanism can be used to limit the dollar value of the inventory in the system, or the amount of space taken up. Alternatively, it can be used to favor some customers or products.

2 Design and Operation of Manufacturing Systems

2.1 Dynamic Programming Model and Analysis

2.1.1 Formulation

Consider a k-machine production system that produces l part types. A four-machine, two-part-type example is illustrated in Figure 1. Part type q ($q = 1, ..., l$) follows a sequence of $K(q)$ steps. It visits Machines $M_{S(1,q)}$, $M_{S(2,q)}$, ..., $M_{S(K(q),q)}$ in sequence, where $S(s,q) \in \{1, ..., k\}$. After receiving its sth operation at Machine $M_{S(s,q)}$, the part goes to Buffer B_{sq}, and then to Machine $M_{S(s+1,q)}$ for its $s+1$st operation. This notation allows us to model reentrant flow, which is an essential feature in semiconductor and some other kinds of manufacturing. It distinguishes the same part type (q) at the same machine ($M_i, i = S(s,q)$) at different stages (s).

For example, in Figure 1, two part types enter the system at Machine M_1. Type 1 parts then go to Buffer B_{11}, M_2, B_{21}, M_3, and B_{41}. They go back to M_1, for another operation, and then to B_{51}, M_2, B_{61}, M_3, B_{71}, M_4. After the second visit to M_4, they leave the system as finished parts. After M_1, type 2 parts go to Buffer B_{12}, M_2, B_{22}, M_3, B_{32}, M_4, and then leave the system.

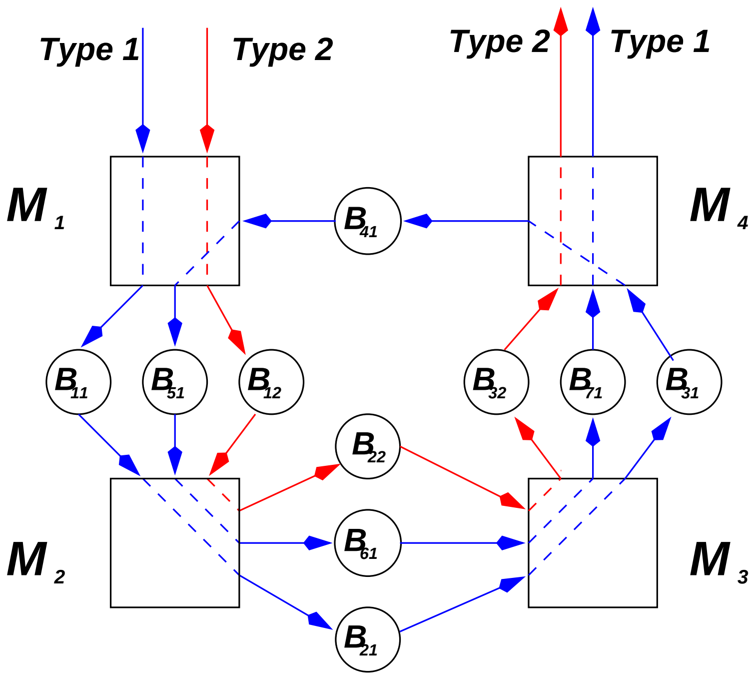

Figure 1: Production system with multiple parts and re-entrant flow

Material is modeled as continuous, ie, as a fluid. In-process inventories, surpluses (defined below), and production rates are represented by continuous (ie, real) variables that can take on any values within specified intervals.

The machines are unreliable. The fraction of time that M_i is operational is e_i and μ_{sq} is the maximum rate at which Machine M_i can do operations on type q parts ($i = S(s,q)$). Assume that the demand for type q parts is constant at rate $d_q > 0$, and that

$$\sum_{\{s,q|S(s,q)=i\}} \frac{d_q}{\mu_{sq}} < e_i \text{ for all } i \tag{1}$$

in which the sum is taken over all parts at all production stages that visit Machine M_i. This is a statement of *feasibility.* See the further discussion of feasibility in Section 2.1.4.

Let $u_{sq}(t)$ be the *instantaneous production rate* of type q parts at stage s (which is performed at machine M_i, $i = S(s,q)$) at time t. The production rates are the control variables in this formulation. The total amount of type sq material that has been processed by M_i during $(0,t)$ is

$$P_{sq}(t) = \int_0^t u_{sq}(\tau)d\tau. \tag{2}$$

Let Machine M_i's repair state at time t be given by $\alpha_i(t)$. When M_i is operational, $\alpha_i(t) = 1$. When it is under repair, $\alpha_i(t) = 0$. Then

$$\begin{array}{ll} \text{if } \alpha_i(t) = 0, & u_{iq}(t) = 0; \text{ for } q = 1,...,l; s = 1,...,K(q); S(s,q) = i \\ \text{if } \alpha_i(t) = 1, & \displaystyle\sum_{\{s,q|S(s,q)=i\}} \frac{u_{sq}(t)}{\mu_{sq}} \leq 1; u_{sq}(t) \geq 0. \end{array} \tag{3}$$

We define $u_{sq}(t)/\mu_{sq}$ to be the *fraction of the capacity of Machine M_i that is devoted to part type q at stage s at time t.*

The buffers are *homogeneous* — each buffer holds parts that are indistinguishable, except possibly according to arrival time, due date, etc. That means that only a single part type at a *single production stage* appears in each buffer.

Let $b_{sq}(t)$ be the *level* of (the amount of material in) Buffer B_{sq} at time t, $s = 1,...,K(q)-1$. Then

$$\frac{db_{sq}}{dt} = u_{sq} - u_{s+1,q} \tag{4}$$

Buffer levels must satisfy

$$0 \leq b_{sq} \leq N_{sq} \tag{5}$$

where N_{sq} is the *size* of Buffer B_{sq}.

Let $x_{K(q),q}$ satisfy

$$\frac{dx_{K(q),q}}{dt} = u_{K(q),q} - d_q, q = 1,...,l \tag{6}$$

This is the *system surplus*. It is the set of cumulative differences between production and demand. It is not bounded above or below. If it is negative, customers are not satisfied. If positive and large, there is a large finished goods inventory. Keeping $x_{K(q),q}(t)$ near zero is a major objective.

We generalize the concept of the system surplus by defining the *local surplus*

$$x_{sq} = \sum_{j=s}^{K(q)-1} b_{jq} + x_{K(q),q}, \tag{7}$$

which satisfies

$$x_{sq} - x_{s+1,q} = b_{sq}, \tag{8}$$

$$\frac{dx_{sq}}{dt} = u_{sq} - d_q \tag{9}$$

The local surplus is discussed in Section 2.1.3.

The system states are $(\alpha_1, ..., \alpha_k)$ and $(x_{11}, ..., x_{K(l)-1,l}, x_{K(l),l})$. A natural formulation of a dynamic programming problem is then

$$J = \min E \int_0^T g(b_{11}(t), b_{12}(t), ..., x_{K(l),l}(t))dt \tag{10}$$

subject to (3), (5), (6), (8), (9),
and the dynamics of α

in which g is a function that penalizes surplus, backlog, and inventory; and T is a long time. Typically, α is modeled with exponential distributions for time to fail and time to repair.

2.1.2 State Variable Inequality Constraints

Inequality (5) is a set of state-variable inequality constraints. These conditions can only be enforced by translating them into control constraints. They can be written

$$\text{If } b_{sq} = 0, \qquad \text{then } \frac{db_{sq}}{dt} \geq 0.$$

$$\text{If } 0 < b_{sq} < N_{sq}, \quad \text{then } \frac{db_{sq}}{dt} \text{ is unconstrained by (5).}$$

$$\text{If } b_{sq} = N_{sq}, \qquad \text{then } \frac{db_{sq}}{dt} \leq 0.$$

Using (4), these conditions become

$$\text{If } b_{sq} = 0, \quad \text{then} \quad u_{sq} - u_{s+1,q} \geq 0. \tag{11}$$

$$\text{If } 0 < b_{sq} < N_{sq}, \quad \text{then} \quad u_{sq} - u_{s+1,q} \text{ is unconstrained by (5).}$$

$$\text{If } b_{sq} = N_{sq}, \quad \text{then} \quad u_{sq} - u_{s+1,q} \leq 0. \tag{12}$$

2.1.3 Local Surplus

The local surplus $x_{sq}(t)$ is the difference between the total number of type q parts that have had their sth operation at time t and the cumulative demand for type q parts up to that time. The system surplus is the local surplus of the last production stage. Note that (5) and (8) imply that $x_{sq} \geq x_{s+1,q}$.

The local surplus summarizes downstream information in a way that helps decentralized decision-making. If the local surplus is negative, the downstream part of the system is starved; and if the local surplus is large and positive, it means that there is a lot of inventory somewhere downstream. Both of these events are undesirable, so we want the local surplus to be close to zero. In fact, if we view stage $s+1$ as the customer of stage s, the local surplus at that stage is like the system surplus for the upstream part of the system. See item 9 in Section 8.2.

2.1.4 Feasibility and Stability

Kumar and his colleagues have extensively investigated the sufficiency of (1) for feasibility. In general, they have found that it is possible to construct reasonable-seeming policies for which (1) is not sufficient, and that there are classes of policies for which (1) is sufficient.

Kumar and Seidman (1990) defined *clearing* policies as those in which, once a machine takes a part from a given buffer, it continues taking parts from that buffer until that buffer is empty. (Their definition is more precise.) They demonstrated, by exhibiting a counter-example, that (1) is not sufficient for feasibility (or *stability* as they termed it, because they study the levels of infinite buffers) for clearing policies. That is, they created a system that satisfied (1) and a clearing policy such that the demand could not be satisfied. Perkins and Kumar (1989) had earlier found a class of modified clearing policies that they showed to be stable if (1) is satisfied.

Lu and Kumar (1991) defined *buffer priority policies* as the set of policies in which parts are chosen to load into a machine according only to which buffer they come from. An example for the system in Figure 1 is a policy that insists that M_2 always chooses parts from B_{51} if that buffer is non-empty; and from B_{12} if B_{51} is empty and B_{12} is non-empty; and from B_{11} if it is non-empty and B_{12} and B_{51} are empty. They demonstrated, by using the same system as Kumar and Seidman (1990), that (1) is not sufficient for feasibility for buffer priority policies. They demonstrated sufficiency of (1) (with some other limitations on arrival and due date characteristics) for stability of the least slack and earliest due date policies.

Meyn (1995) and Dai (1995a, 1995b, 1996) established a relationship between the transience (instability) of a discrete network model and a continuous fluid approximation of it. Kumar and Meyn (1995) and Dai and Weiss (1996) establish stability results for reentrant systems by using Lyapunov methods. They show that if a certain matrix exists, or if the solution to an optimization problem has certain characteristics, then (1) implies stability. See also Down and Meyn (1997).

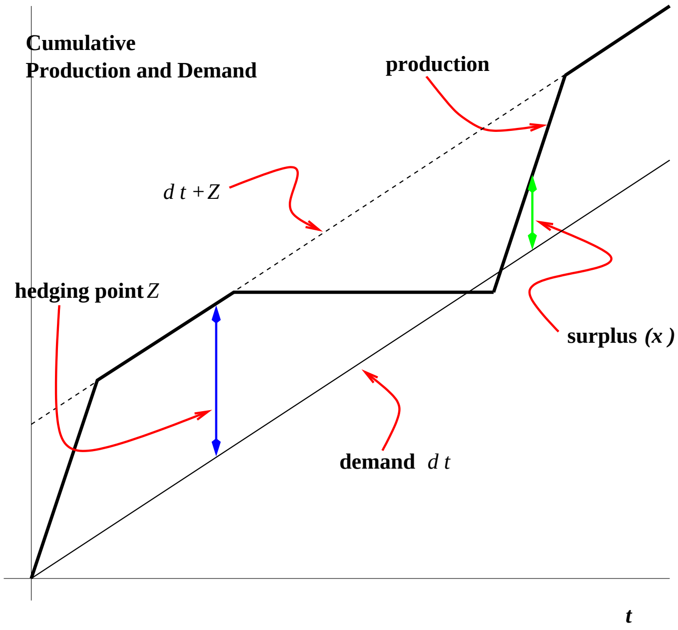

Figure 2: Hedging point policy dynamic behavior — single part type

2.2 Special Cases

2.2.1 One Machine, One Part Type ($k = 1, n = 1$)

For a single-stage, single-part-type system in which g was a generalized absolute value, Akella and Kumar (1986) (for a discounted version of this problem) and Bielecki and Kumar (1988) showed that the solution of this problem was in the form of a *hedging point policy*. That is, there is some number Z (the hedging point) such that the optimal control law is

$$\begin{array}{ll} \text{if } x > Z & u = 0 \\ \text{if } x = Z & u = d \\ \text{if } x < Z & u = \mu \end{array} \tag{13}$$

whenever the machine is up. When the machine is down, $u = 0$.

Under these rules, when $u = d$, x is constant (from (6) and (8)). According to the control law, $u = d$ when the machine is up and $x = Z$. As a consequence, once x reaches Z, x remains equal to Z until the machine fails.

Figure 2 illustrates the dynamic behavior of the hedging point policy. The horizontal axis represents time t and the vertical axis represents cumulative production (P) and demand. Since the demand rate is assumed constant, the cumulative demand during time interval $[0, t]$ is dt. The heavy broken line represents the cumulative production in $[0, t]$. Since the surplus $x(t)$ is the difference between cumulative production and cumulative demand, the cumulative production at time t is

$$P(t) = x(t) + dt.$$

When the machine is down, $u(t) = 0$ and the cumulative production is constant (from (2)). Whenever the machine is up and the surplus $x(t)$ is less than Z, $u(t)$ takes on its maximum value μ and the $P(t)$ graph is steepest. When the machine is up and $x(t)$ is equal to Z, $x(t)$ remains constant and

$$P(t) = Z + dt.$$

2.2.2 One Machine, Multiple Part Types ($k = 1, n \geq 2$)

Generalizations to more than one part type or more than one stage have proved to be difficult. Srivatsan (1993) studied a one-machine, two-part generalization of the Bielecki-Kumar (1988) problem in which the generalized absolute value function cost is $g = g_1 + g_2$, and

$$g_q = \begin{cases} -g_q^- x_q & \text{if } x_q < 0 \\ \\ g_q^+ x_q & \text{if } x_q \geq 0 \end{cases} \tag{14}$$

where g_q^- and g_q^+ are non-negative numbers. He was only able to obtain a solution under the special case in which the hedging point vector (Z_1, Z_2) was 0. Assuming that that policy can be extended to single-machine, n-part-type systems without set-up costs, the policy is a special case of that of Rishel (1975) and Kimemia and Gershwin (1983). The regions in x space are such that the policy is:

0. Before executing the policy, rank order the products.
1. Produce the highest ranking product, until its surplus reaches its hedging point. (The others fall behind.)
2. Keep the highest ranking product at its hedging point. Devote all remaining capacity to the second highest ranking product, until it reaches its hedging point. (The others fall further behind.)
3. Keep the two highest ranking products at their hedging points. Devote all remaining capacity to the third highest ranking product, until it reaches its hedging point. (The others fall still further behind.)
4. etc.

This statement can be simplified to: *Produce the highest ranking part whose surplus is below its hedging point.*

The ranking that follows from the generalized absolute value function cost is according to decreasing values of $g_q^- \mu_q$, the "$c\mu$ rule." The quantity g_q^- is the penalty that the business suffers for each unit of type q for each time unit in which it is backlogged. Unfortunately, Srivatsan and Dallery showed that this simple priority scheme is not always optimal when the hedging point is not zero (Srivatsan and Dallery, 1998).

2.2.3 Multiple Machines, One Part Type ($k \geq 2, n = 1$)

Van Ryzin (1987) and colleagues (Van Ryzin, Lou, and Gershwin, 1993) obtained numerical solutions for two-stage, single-part systems with discounted cost. In Gershwin (1997) the optimal production control policy for this class of systems is approximated by a policy of the form:

$$\text{For } i = 1, 2, \quad \begin{array}{ll} \text{if } x_i > Z_i & u_i = 0; \\ \text{if } x_i = Z_i & u_i = d; \\ \text{if } x_i < Z_i & \text{choose } u_i \text{ as large as possible.} \end{array} \tag{15}$$

In this expression, "As large as possible" means

- if $0 < x_1 - x_2 < N$, then $u_1 = \mu_1$ and $u_2 = \mu_2$;
- if $x_1 - x_2 = N$, then $u_1 = \min(\mu_1, u_2)$ (due to (12)) and $u_2 = \mu_2$; and
- if $x_1 - x_2 = 0$, then $u_1 = \mu_1$ and $u_2 = \min(u_1, \mu_2)$ (due to (11)).

Note that if $x_1 - x_2 = N$, and $u_2 < \mu_1$, then $x_1 - x_2$ remains constant (until the machine fails, or $x_1 = Z_1$ or $x_2 = Z_2$). A similar statement is true when $x_1 - x_2 = 0$.

Lou and his colleagues have expanded on Van Ryzin's results. Lou and Van Ryzin (1989) extend it to tandem three-machine systems and conjecture near-optimality for the two-boundary policy. They get exact numerical solutions using numerical dynamic programming methods. Lou and Kager (1989) and Yan, Lou, Sethi, Gardel, and Deosthali (1996) apply it to models of semiconductor fabrication facility and get good performance. A rigorous analysis of the two-machine solution was provided in Lou, Sethi, and Zhang (1994).

Following this literature, we can generalize this policy statement to lines of arbitrary length k. The buffer level is $b_i = x_i - x_{i+1}$, so we must have $0 \leq Z_i - Z_{i+1} \leq N_i$. The policy is the same as (15), but "as large as possible" means $u_i = \mu_i$ except when $x_i - x_{i+1} = N_i$, in which case $u_i = \min(\mu_i, u_{i+1})$; and when $x_{i-1} - x_i = 0$, in which case $u_i = \min(u_{i-1}, \mu_i)$.

Decentralization This policy is *decentralized.* Each decision is made on the basis of only local dynamic information: x_{i-1}, x_i, x_{i+1}, b_{i-1}, and b_i.

2.2.4 General Networks

Lou and Kager (1989) and Yan, Lou, S., Sethi, Gardel, and Deosthali (1996) develop the two-boundary methods for networks that involve reentrant flow. Gershwin (1997) states the policy in general terms. We restate it in the next section.

3 Two Statements of the Control Point Policy

In this section, we describe a generalization of the policies of Sections 2.2.1, 2.2.2, and 2.2.3 for the general formulation of Section 2.1.1. While this surplus-based version is a natural extension of those policies, it is awkward to state, impossible to implement, and it is not clear how to analyze it. Therefore, we present two equivalent policies (the time- and token-based versions) which, together, remedy these defects.

The reentrant nature of the material flow makes it essential to specify both the part type and the production stage that the part has reached. We therefore define *work item type sq* to be part type q at stage s.

3.1 Surplus-Based Policy Statement

We conjecture that the following policy provides good performance. This is a proposed suboptimal closed loop solution of (10) that specifies u_{sq}, the control variables, as a function of the state variables x_{sq} and α_i. Following this, we restate the solution in a form which may be implemented (Section 3.2.2).

Like most control policies, the solution has two elements: the preparatory phase (Step 0, when parameters are selected) and the real-time phase (Steps 1-4, when decisions are made in response to events).

0. Before executing the policy, rank order the work items at each machine. Parts at different production stages at the same machine may have to be compared to create this ranking. One possible ranking is described in Section 2.2.2. Also calculate the hedging points (Z_{sq}) and buffer sizes.

The following four steps form a real-time capacity allocation algorithm. In principle, it must be executed at every time instant t. (We describe a way of executing it only when necessary in Section 3.2.2.) For each machine $M_i, i = 1, ..., k$, if $\alpha_i(t) = 0$, then $u_{sq} = 0$ for all sq such that $S(s, q) = i$. If $\alpha_i(t) = 1$, divide the set of part types q at stage s (work item types sq) such that $S(s, q) = i$ into four sets. They are defined as:

$\mathcal{Q}_1$: the set of all work item types for which $x_{sq}(t) > Z_{sq}$; We have made too much of them, and we will not produce any now.

$\mathcal{Q}_2$: the set of work item types for which $x_{sq}(t) = Z_{sq}$ and for which there are no higher ranking work item types such that $x_{sq}(t) < Z_{sq}$; We will produce these parts at a rate no greater than demand.

$\mathcal{Q}_3$: the highest ranking parts such that $x_{sq}(t) < Z_{sq}$; We will produce these parts at the maximum possible rate subject to capacity constraints (3) and buffer constraints ((11) and (12)).

$\mathcal{Q}_4$: the lowest ranking parts such that $x_{sq}(t) < Z_{sq}$; We will *not* produce these parts because all capacity has been exhausted on $\mathcal{Q}_2$ and $\mathcal{Q}_3$. That is, (3) is satisfied with equality when summed over the part types in $\mathcal{Q}_2$ and $\mathcal{Q}_3$.

We construct these sets as follows.

- Initially, they are:

$\mathcal{Q}_1$: the set of all work item types for which $x_{sq}(t) > Z_{sq}$;

$\mathcal{Q}_2$: the set of work item types for which $x_{sq}(t) = Z_{sq}$ and for which there are no higher ranking work item types such that $x_{sq}(t) < Z_{sq}$;

$\mathcal{Q}_3$: the highest ranking work item type for which $x_{sq}(t) < Z_{sq}$. If there is no sq for which this is true, $\mathcal{Q}_3$ is initially empty.

$\mathcal{Q}_4$: all other work item types.

- Then we assign production rates and modify the sets according to the following *real-time capacity allocation algorithm*:

1. For each work item type sq in $\mathcal{Q}_1$, $u_{sq}(t)$=0.
2. For each work item type sq in $\mathcal{Q}_2$, $u_{sq}(t) = d_q$.
3. If $\mathcal{Q}_3$ is not empty, allocate as much of the remaining capacity as possible to the work item type in $\mathcal{Q}_3$, subject to the limitations discussed just below (15). That is, if the work item type in $\mathcal{Q}_3$ is called s^*q^*, define

$$\mathcal{U}_{s^*q^*} = \mu_{s^*q^*}\left(1 - \sum_{sq\in\mathcal{Q}_2}\frac{d_q}{\mu_{sq}}\right) \tag{16}$$

in which the sum is over all elements of $\mathcal{Q}_2$. Then

$$u_{s^*q^*}(t) = \begin{cases} \min(\mathcal{U}_{s^*q^*}, u_{s^*-1,q^*}(t)) & \text{if } b_{s^*-1,q^*} = 0; \\ \min(\mathcal{U}_{s^*q^*}, u_{s^*+1,q^*}(t)) & \text{if } b_{s^*q^*} = N_{s^*q^*}; \\ \mathcal{U}_{s^*q^*} & \text{otherwise.} \end{cases} \tag{17}$$

4.
   - If $\displaystyle\sum_{sq\in\mathcal{Q}_2\cup\mathcal{Q}_3}\frac{u_{sq}(t)}{\mu_{sq}} = 1$, then $u_{sq}(t) = 0\,\forall\, sq \in \mathcal{Q}_4$, and *stop*.
   - If $\mathcal{Q}_4$ is empty, *stop*.
   - Otherwise,
     (a) Move the work item type from $\mathcal{Q}_3$ into $\mathcal{Q}_2$. Do not change its production rate $u_{sq}(t)$.
     (b) Move the highest ranking work item type from $\mathcal{Q}_4$ into $\mathcal{Q}_3$. If there are more than one work item type with the same, highest, ranking, select one whose surplus is furthest behind its hedging point.
     (c) Go to Step 3.

By *stop*, we mean that all the capacity of Machine M_i has been allocated, and all $u_{sq}(t)$ have been specified.

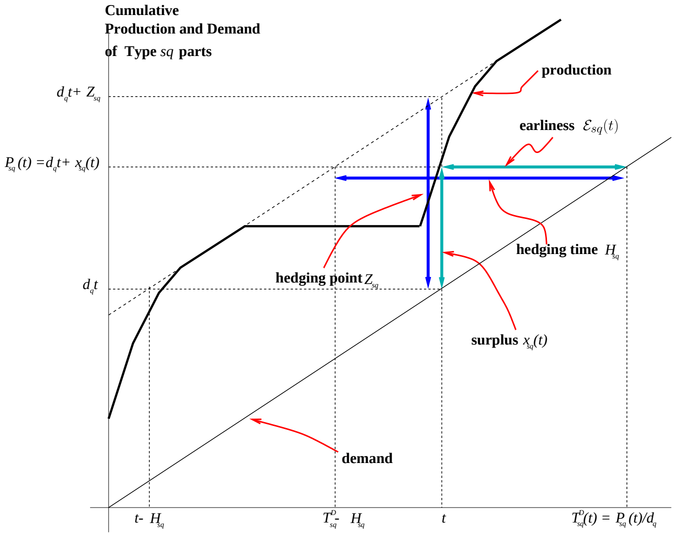

Figure 3: Earliness and lateness in multiple part type hedging point control

Explanation In Step 0, we determine the ranking, just as in Step 0 of Section 2.2.2. In Step 1, we assign production rate 0 to all work items whose surplus is greater than the hedging point. In Step 2, we maintain all the highest ranking work items at their hedging points. Step 3 and Step 4 assign the remaining capacity to one or more work item types subject to the "as large as possible" considerations of Section 2.2.3. In (16), $\mathcal{U}_{s^*q^*}$ is the maximum possible production rate of s^*q^* considering only short term capacity (3). The actual production rate is reduced in (17) if an adjacent buffer is empty or full. Capacity is allocated in rank order until either there is no capacity left or there are no work items left.

This policy is impossible to implement as written because it must be executed at every time instant t and because material is discrete. (In factories where material actually is fluid, the kind of flexibility assumed here is impossible.) The first issue can be overcome because the production rates that it calculates are piecewise constant, and we can determine the switching times. The second is more serious, and requires the revision and reinterpretation described below.

Decentralization This policy, and the variations described below, share the decentralization property of Section 2.2.3. The only dynamic information that is needed at each decision point comes from the local buffers and surpluses.

Dynamic behavior The dynamic behavior of this policy is illustrated in Figure 3. The vertical axis represents cumulative demand of type q $(d_q t)$ and the cumulative production of type q parts at stage s (the heavy broken line).

This figure is similar to Figure 2. Here, however, the production line does not follow a single slope as it makes its way back to the hedging point line $(d_q t + Z_{sq})$. It changes slope each time a capacity is shared with a new work item type, each time an upstream buffer changes from empty to non-empty (or vice versa), and each time a downstream buffer changes from full to non-full (or vice versa).

In addition, because stage s is not the final stage of the process for part type q, the local hedging point Z_{sq} must be greater than if s were the final stage. This is because of equation (7), in which the buffer levels b_{jq} are non-negative. In general, $Z_{sq} \geq Z_{s+1,q}$.

An alternative interpretation of this figure, which is based on time rather than surplus, is described in the following section.

3.2 Time-Based Policy Statement

3.2.1 Due dates and hedging times

Because the surplus-based policy statement is complex and probably difficult to implement, we present a time-based version of the policy. It is equivalent to the surplus-based version. In addition to being easier to implement, it establishes a connection with time-based policies and suggests new directions for development of the policy.

At time t, the amount of type q material that has been demanded since time 0 is $d_q t$. The amount that was actually produced at stage s at Machine i is

$$P_{sq}(t) = \int_0^t u_{sq}(r)dr = d_q t + x_{sq}(t). \tag{18}$$

The maximum that could have been produced according to this policy, if machine up time was not unfavorable, is $d_q t + Z_{sq}$. This is indicated in Figure 3.

To say it in another way, if the up times of all the machines in the system were not unfavorable, the required material, $d_q t$, could have been produced early. The earliest it could have been produced under this policy was the time t^Z_{sq} such that $d_q t^Z_{sq} + Z_{sq} = d_q t$, or

$$t^Z_{sq} = t - H_{sq}$$

in which we define the *hedging time*,

$$H_{sq} = Z_{sq}/d_q.$$

This suggests the following alternate statement of the generalized hedging point policy. Define

$$T^D_{sq}(t) = P_{sq}(t)/d_q.$$

This is the time at which we *should* have produced a total of $P_{sq}(t)$ units (if we were not concerned with the rest of the production process). In other words, $T^D_{sq}(t)$ is the *due date* of the type sq work item which is now being processed at Machine $M_{S(s,q)}$ at time t.

Also, define

$$\mathcal{E}_{sq}(t) = \frac{P_{sq}(t)}{d_q} - t = T^D_{sq}(t) - t \leq H_{sq}. \tag{19}$$

$\mathcal{E}_{sq}(t)$ represents the *earliness* of work item sq at time t. The inequality says that the earliness should be less than or equal to the hedging time. We can therefore interpret the hedging time H_{sq} as a conservative estimate of the time required between stage s and the end of the process for part type q: it represents how much earlier than its due date we would like to be doing the sth operation on a type q part.

The new statement is the same that of Section 3.1 except that we replace $x_{sq} > Z_{sq}$ with $\mathcal{E}_{sq}(t) > H_{sq}$; $x_{sq} = Z_{sq}$ with $\mathcal{E}_{sq}(t) = H_{sq}$; and $x_{sq} < Z_{sq}$ with $\mathcal{E}_{sq}(t) < H_{sq}$.

Another way of stating this would be to replace $x_{sq} > Z_{sq}$ with

$$t + H_{sq} < T^D_{sq}(t) \tag{20}$$

and so forth. Then the real-time capacity allocation algorithm becomes:

1. Do not produce any work item type sq for which (20) is true. That is, do not produce any work item which is too early (ie, which is a member of $\mathcal{Q}_1$).

2. Maintain

$$H_{sq} + t = T^D_{sq}(t) \tag{21}$$

   for any work item type sq for which (21) is true. In other words, do the sth operation on type q parts exactly H_{sq} time units earlier than their due dates, if possible. The set of work items for which (20) (and for which there are no higher ranking work items such that $H_{sq} + t > T^D_{sq}(t)$) is $\mathcal{Q}_2$.

3. Produce all others at the maximum rate possible subject to ranking and available capacity. Initially $\mathcal{Q}_3$ consists of the highest ranking work item type for which $H_{sq} + t > T^D_{sq}(t)$ and $\mathcal{Q}_4$ contains all types not in $\mathcal{Q}_1$, $\mathcal{Q}_2$, or $\mathcal{Q}_3$. $\mathcal{Q}_3$ and $\mathcal{Q}_4$ are then modified as in Step 4.

This policy is still not implementable because it deals in rates. To implement it, the control actions should describe how to select discrete parts to load and when to load them. This is described in the following section.

3.2.2 Adaptation for Discrete Parts

When material is discrete, at most only one work item type can be selected at a time. This is because we are deciding on a discrete action rather than a set of production rates. In addition, there are never — for all practical purposes — any work item types in $\mathcal{Q}_2$.

We assume that each work item has an individual due date. We define the due date of work item type sq to be the earliest due date of any work item in Buffer $B_{S(s-1,q)}$.

The set $\mathcal{Q}_3$ initially consists of the highest ranking work item type for which $H_{sq} + t > T^D_{sq}(t)$. However, if buffer $B_{S(s-1,q)}$ is empty or if $B_{S(s,q)}$ is full, work item type sq cannot be processed. Therefore, we replace that work item with the highest ranking work item type for which $H_{sq} + t > T^D_{sq}(t)$ *and* $B_{S(s-1,q)}$ is not empty *and* $B_{S(s,q)}$ is not full. These conditions are required because of (16) and (17).

The work item types for which $B_{S(s-1,q)}$ is not empty and $B_{S(s,q)}$ is not full are called *available*. The set of available work item types for which $H_{sq} + t > T^D_{sq}(t)$ are called *ready*.

Now, we need not consider the system at every time instant t — even in principle. Decisions can only be implemented when machines complete operations or when the repair of failed machines are complete. Therefore (since the real time computation is trivial) the policy need only be invoked at such times. Consequently, the policy is now:

0. Before executing the policy, calculate buffer sizes and hedging points; decide on the rank ordering for all the work item types at each machine.

1. When Machine M_i becomes idle, decide whether to operate it or wait:

   (a) Consider on the work items sq that are available, ie those for which the upstream buffer $B_{s-1,q}$ is not empty and the downstream buffer B_{sq} is not full.

   (b) Divide the available work item types into those that are ready not ready.

   (c) Ignore work item types that are not ready. Among those that are ready, select the highest ranking work item type sq. Operate on the part in $B_{S(s-1,q)}$ which has the earliest due date.

   (d) If there are no work items that satisfy (b), wait until the first work item does, and then go to (b).

3.2.3 The Control Point Policy

The policy, as stated so far, is executed at every machine M_i. We learned, very early in the experimental implementation (Section 7.3), that this is not necessary and may be expensive. Instead, it appears to be sufficient to implement it at a limited set of *control points*. At all other factory resources, a simple, sensible policy like FIFO can be used.

3.3 Comparisons with Other Policies

3.3.1 Least Slack

At Machine M_i, the expected amount of time before work item type sq ($S(s,q) = i$) leaves the system is ζ_{sq}. In the *least slack* policy (Lu and Kumar, 1991), the *slack* for a type q work item at Machine i is defined as the difference between its due date and ζ_{sq}. At each stage, the work item with the least slack is selected next. Lu and Kumar show that this policy works well.

Suppose the type sq work items have serial numbers starting at 0. If demand is constant, the due date of the work item with serial number P_{sq} is P_{sq}/d_q. According to the policy of Section 3.2.2, we want that work item to obtain its sth operation (at Machine M_i, where $i = S(s,q)$) at time $T^D_{sq} - H_{sq}$. Consequently, the hedging point and least slack policies are similar if $H_{sq} = \zeta_{sq}$. We can interpret $T^D_{sq} - H_{sq}$ as the slack time of part type q at stage s (Bonvik, 1994).

However there are differences. First, least slack does not distinguish between work items that are earlier than the slack time and those that are not. Instead, it ranks the work items in the order of slack time, and processes the work item with the earliest slack time. (Here, that would be the smallest value of $T^D_{sq} - H_{sq}$.) Second, least slack considers only slack time in selecting the work items, and no other criterion. This difference is not likely to be very important when there is enough excess capacity, but it can lead to delays for valuable parts, or parts for valued customers when capacity is tight.

3.3.2 MRP

Material Requirements Planning (Orlicky, 1975) MRP and some extensions are widely used to plan factory operations. The difference between MRP and least slack is that MRP systems require parts to be produced at *exactly, or as close as possible to*, an intermediate due date (ie, a slack time). If a part is early, it is made to wait; if it is late, the workers and managers view themselves as in trouble and do everything they can — including overtime, subcontracting, expediting, and short-term manual rescheduling — to catch up.

4 Token-Based Control Policies

4.1 Definitions

An important class of real-time manufacturing scheduling systems are token-based. That is, operations are performed if both parts and tokens (or kanbans, or authorization cards, etc.) are present. Many control policies are naturally token-based, such as kanban, base stock, CONWIP (Bonvik, 1996; Bonvik, Couch, and Gershwin, 1997; Dallery and Liberopoulos, 1997, 2000; Frein, Di Mascolo, and Dallery, 1994) and Buzacott and Shanthikumar's PAC (Production Authorization Card) system (Buzacott and Shanthikumar, 1992, 1993).

Consider Figure 4. This machine is different from those of Figure 1 because it is an *assembly/disassembly* machine. When it does an operation, it takes, simultaneously, one part from each of the upstream buffers and places, simultaneously, one part into each of the downstream buffers. The buffers are homogeneous. It can only do an operation if there is at least one part in each upstream buffer and one space in each downstream buffer. Imagine that some of the buffers hold tokens rather than parts. In that case, the operation is only allowed if there is a token in each upstream token buffer and a space in each finite downstream token buffer. These buffers then act as *authorizations* for operations.

The physical structure of the process is constrained by the nature of the production, and we assume that it is fixed, for the purpose of this paper. However, the flow of tokens can be anything we want, if it produces good factory behavior. That is, we may connect any machine with any other machine through a token buffer. The second machine is allowed to do an operation only if the first machine has put enough tokens in that buffer.

Consequently, a real-time scheduling policy is defined by the topology of the token network; the sizes of the token buffers; and the sizes of the part buffers.

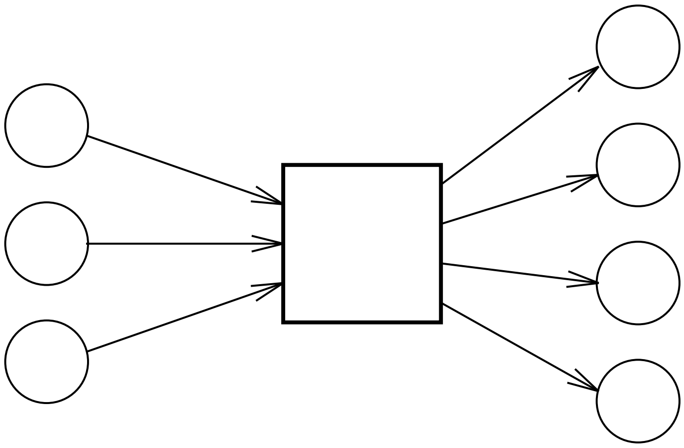

Figure 4: Assembly/Disassembly Machine

4.2 Equivalence of Hedging Point Policy for One-Machine, One-Part-Type Systems and Three-Machine Assembly

Figure 5 represents a three-machine assembly system in which Machine S takes parts from Buffers FG and B, assembles them, and sends them on to customers. Buffers FG and B get their parts from Machines M and D. Let us assume further that Machine S is infinitely fast and perfectly reliable. In that case, material cannot be in both buffers simultaneously; if there were something in both buffers, enough of it would be instantly removed from both so that at least one of the buffers would become empty. In addition, we assume that Buffer B is infinite, the size of Buffer FG is Z, which is finite, and Machine D is perfectly reliable and operates at speed $\mu_D = d$.

In that case, as shown in Bonvik (1996), the behavior of this system is *identical* to the system of Section 2.2.1, and that it can be treated analytically for a given Z. As in Bielecki and Kumar (1988), the optimal Z can be determined.

This system can be interpreted as follows: Machine D represents the *demand* process. When the sales department confirms an order, a token is generated by D and is put into B. If FG (the *finished goods* buffer) is not empty, Machine S (*synchronization* of demand and production) instantaneously assembles the order and a part and sends it out of the production area to the customer, to the shipping area, or elsewhere. If FG is empty, the order waits in B (the *backlog* buffer) until Machine M produces a part. On the other hand, if M produces a part when B is empty, the part waits in FG for an order.

4.3 Construction of Larger Equivalent Systems: One Part Type

Consider the three-machine production line in Figure 6. Suppose we control it using the methods of Section 2.2.3. Then we claim that its behavior is exactly the same as the assembly/disassembly system of Figure 7. In this figure, the control structure of Figure 5 is replicated for each machine M_i (and is drawn in lighter color than the rest of the figure). When M_i performs an operation, it removes a part from Buffer B_{i-1}, it adds a part to Buffer B_i and it creates a token and adds it to

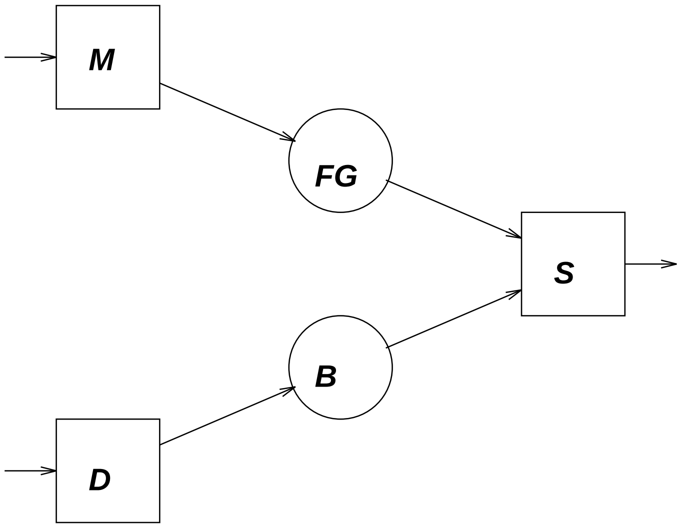

Figure 5: Three-Machine Assembly System

the upper buffer of the new structure. If Buffer B_{i-1} is empty, or if Buffer B_i or the new buffer is full, an operation cannot take place. Synchronization Machines S_1 and S_2 require demands in order to prevent the new upper buffers from filling up. Synchronization Machine S_3 regulates the delivery of completed items to customers.

The upper buffer in each control structure, which has size Z_i, performs the function of the hedging point; the lower buffer represents a local backlog. Since demand is considered to be given by a constant rate d, each machine D_i can separately represent the demand process dt.

A more general representation of the control policy appears in Figure 8. This is more appropriate if the demand is not constant. (See Section 6.) In this representation, whenever a demand arrives, a token appears simultaneously in all three backlog buffers. This structure allows each buffer to be notified when actual demands occur. However, as indicated in Section 5, there is a price to be paid for this increased realism.

4.4 Construction of Larger Equivalent Systems: Multiple Part Types

Figure 9 demonstrates how to transform a multiple-part-type machine that is controlled according to the policy of Section 2.2.2 into a token-based system. Since there are three part types, we have three demand generators, three synchronization machines, three surplus buffers and three backlog buffers. There is a strict precedence relationship among the part types at the original machine: it works on part type 1s unless the part type 1 surplus buffer is full. In that case, it works on type

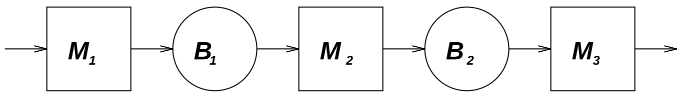

Figure 6: Three-Machine Production Line

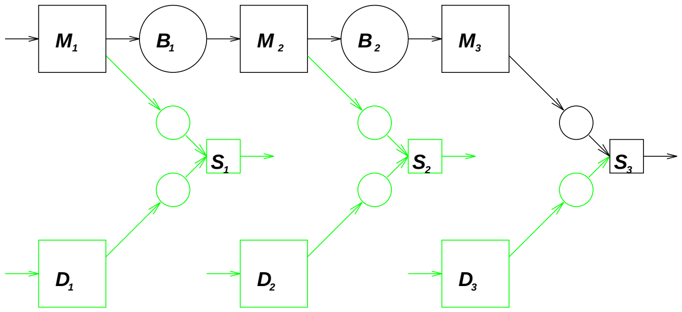

Figure 7: Three-Machine Production Line as a Tree-Structured Assembly System

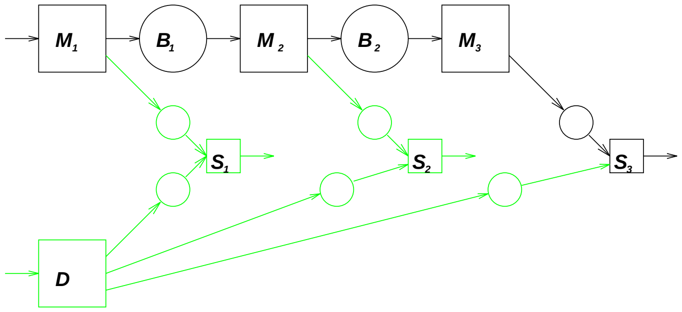

Figure 8: Three-Machine Production Line as an Assembly System with Loops

2 unless the type 2 surplus buffer is full. If it is, it works on type 3 unless that buffer is full. If it is, the machine is idle. For larger systems, we apply the hedging point policy to all machines as in Section 4.3.

4.5 Comparisons with Other Policies

The token systems described in this section do not deal explicitly with multiple-part-type systems. That is, they can be thought of as dealing with single part types, or with with parts that follow the same route through the factory, and that are identified with individual orders. In the latter case, it is not clear how to choose among parts if more than one is available to be processed at the same time.

4.5.1 Production Authorization Card

Buzacott and Shanthikumar (1993) describe a *Production Authorization Card* (PAC) system for coordinating material flow. It is an elaborate system of different kinds of tokens: order tags, requisition tags, and PA cards. Buzacott and Shanthikumar show that the PAC framework is a generalization that includes base stock, MRP, kanban, CONWIP and other policies as special cases. They derive performance bounds and exact evaluation for some special cases.

One feature in this framework that is not included in our token-based system is the possibility of delay: a parameter can be specified which is used to delay the movement of tokens. This allows

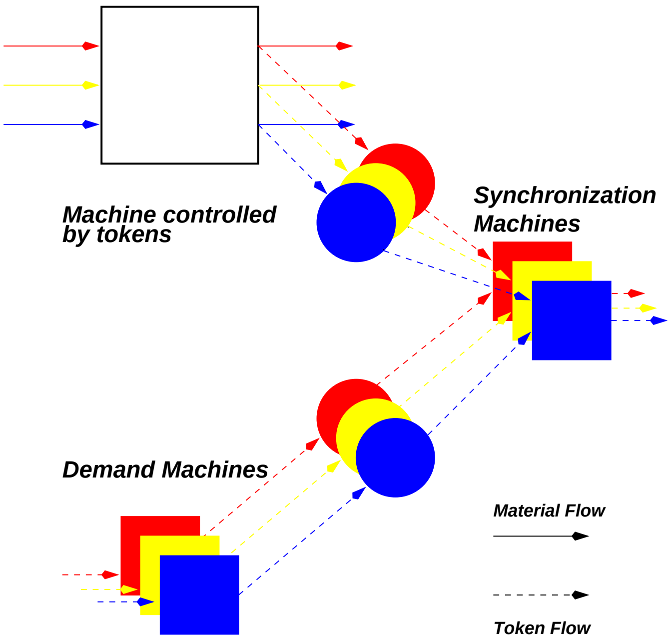

Figure 9: Three part type system

orders to be taken far in advance of their due dates without requiring material to be moved, or purchased, until it is needed. This feature limits inventory. Karaesmen, Buzacott, and Dallery, (1999) obtain analytical results for a special case.

4.5.2 Extended Kanban Control System

Liberopoulos and Dallery (1997) developed a framework for considering a wide class of systems. When the delay parameter is zero, the PAC and Karaesmen– Buzacott–Dallery policies are special cases. Base stock, hedging point, kanban, CONWIP and other policies are special cases of EKCS.

5 Decomposition and Optimization

Systems like the one in Figure 7 can be analyzed by decomposition. This is a conceptual extension of earlier decomposition work because none of the earlier authors contemplated tokens in the network. However, the methods work just as well for tokens as for parts.

However, decomposition methods are currently limited. There is considerable effort to extend these methods to systems with loops (Bonvik, 1996; Frein, Commault, and Dallery, 1996; Tolio and Gershwin, 1998), but at present, decomposition methods appear to be less accurate for loop systems than for acyclic systems; and there do not appear to be any published works on systems with more than one loop (as in Figure 8). In addition, existing decomposition methods work only for single part type systems. Recent work (Nemec, 1999; Syrowicz, 1999) has extended decompositions to small multiple-part-type systems.

Fast optimization methods have been based on decomposition (Schor, 1995; Schor and Gershwin, 1996; Gershwin and Schor, 1997). They will allow us to efficiently design real-time scheduling policies and manufacturing systems simultaneously. This may be more accurate than typical simulation because it predicts the performance of the manufacturing system when it is run with a well-designed policy that can be implemented easily. In typical factory simulations there may be no relationship between the control policy in the simulation and that which is implemented on the factory floor.

6 Extensions of the Demand Model

To make the policy practical in general, it must be extended to systems with non-constant demand. There are two cases to consider first.

Time-varying deterministic demand For this case, extending the implementation is easy, but extending the token-based analysis is not. The straight demand line $d_q t$ in Figure 3 is replaced by a non-decreasing function of time $D_q(t)$. For the surplus-based version of the policy, replace the straight hedging point line by $D_q(t) + Z_q$. For the time-based version, the due date $T_{sq}^D(t)$ is now the solution of

$$P_{sq}(t) = D_q(T_{sq}^D(t))$$

and (19) becomes, simply,

$$\mathcal{E}_{sq}(t) = T_{sq}^D(t) - t \leq H_{sq}.$$

However, evaluating and optimizing such a system by the methods proposed in Sections 4 and 5 might prove to be difficult. The decomposition method has not been extended to time-varying networks, and it is not clear how to do so, since it is based on approximating steady-state performance. For systems with small deterministic variations in demand, the best approach for evaluating the network might be to replace the demand by an average demand rate; for systems with large variations, it might be possible to replace the demand with a piecewise constant demand, and evaluate each demand rate separately.

Stationary stochastic demand Extending the implementation is less obvious in this case. For one thing, we must distinguish between build-to-order and build-to-stock systems. For the first, production of a work item cannot start until the order is received. This may be due to a policy, or more seriously, it may be due to the nature of the product. If each item is customized, or if the products are highly perishable, build-to-order is mandatory. For build-to-stock systems, the main consideration is the risk accounted for in (10) of excess or insufficient production.

There are then alternate possible approaches to implementation.

- For build-to-order, each item has a due date assigned before it enters the factory. That date is used to calculate earliness.

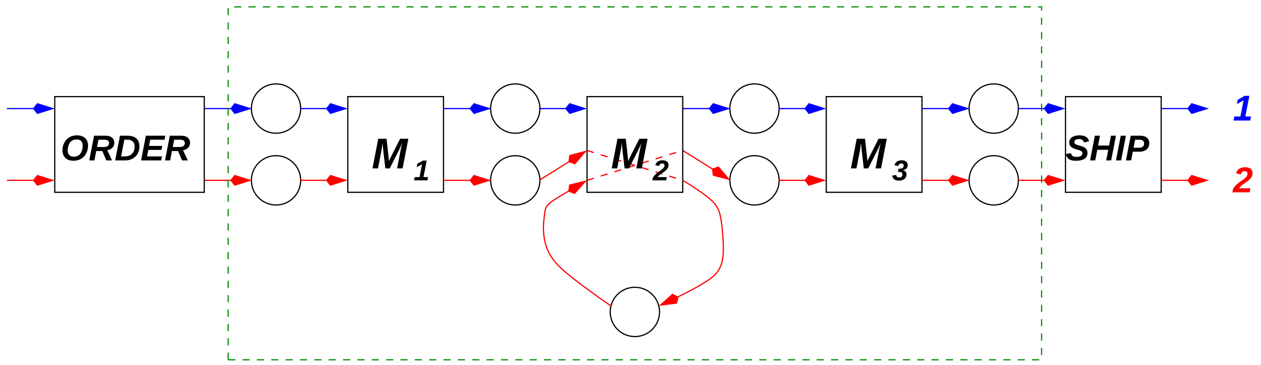

Figure 10: Small Reentrant System

- For pure build-to-stock, implement the policy as described, except that we interpret d_q as the expected demand rate (based on forecasts), and calculate due dates as in Section 3.2.

- Many systems (including long supply chains) are build-to-stock upstream and build-to-order downstream. That is, raw material is collected and production is begun on anticipation of orders, but production is not completed on an item until an order is received. The control policy is implemented accordingly.

Extending the token-based evaluation and optimization should not be difficult. We can generalize the demand machines in Figure 9 by making them unreliable with the same parameters (production rate, mean time to fail, and mean time to repair) as the real machines. This should add no additional difficulty to the decomposition method.

7 Simulation and Implementation

7.1 Small Reentrant System

7.1.1 Description

Figure 10 shows a small system that was simulated by Deshpande (1999). Here, we provide a summary of his most important observations. The purpose of the experiment was to study how the control point policy compared with others when parts may appear in a sequence that differs greatly from that of their due dates (ie, with *disorderly mingling of parts,* as he called it).

The system produces two part types. The high priority type travels straight through the system; the lower priority part visits Machine M_2 twice. The parts arrive at random times with randomly distributed due dates. Some parts are due later than others that arrive earlier. In addition, a poor strategy for selecting parts at M_2 can make the disorderly mingling worse. The machines are unreliable and have different exponentially distributed operation times.

Deshpande observed that the lower priority part received roughly the same performance among all the policies tested. For the higher priority part type, performance of the CONWIP, kanban,

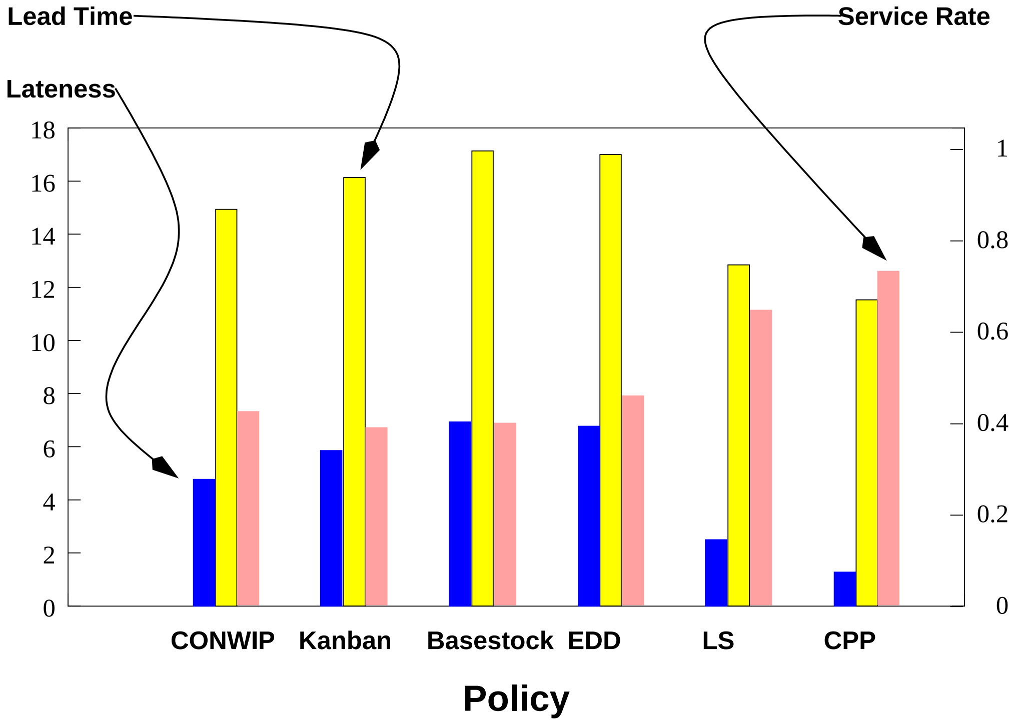

Figure 11: Simulation results

basestock, earliest due date (EDD), least slack (LS) and the control point policy (CPP) policy are shown in Figure 11. The lateness is shown in the dark bars and the lead time is shown in the light bars. Their magnitudes are indicated on the left scale, and smaller numbers are better. The service rate is shown in the medium gray bars, and is measured on the scale on the right. Larger numbers are better.

Among these policies, all except EDD have parameters that must be selected. CONWIP requires a population number; kanban requires kanban numbers, etc. For this to be a truly fair comparison, all policies would have to have been run with optimal values of their parameters, as was done in (Bonvik, Couch, and Gershwin, 1997). This was not feasible because of time constraints, so Deshpande performed a manual, informal parameter selection that he describes in Deshpande (1999). His goal was to maximize service rate, and he claims that his procedure produced reasonable values.

The CPP, which delivered 73% of its parts on or before their due dates, had substantially better service rates than the others. The differences among the lateness of the policies are dramatic. The average lead time that was demanded is 10. While none of the policies achieves this average lead time, the CPP again performs substantially better than the others.

7.1.2 Interpretation

The control point policy does well because it is constantly resequencing parts. The least slack policy does next best, but it evidently schedules some operations too early (since that is one of the ways it differs from the control point policy), and this leads to greater inventory and mixing

Production	WIP	Lead time	Lateness
(363,269,263)	160	144434.05	1343.06
+(28,30,24)		14132.92	6836.66
(363,269,263)	160	144515.07	335.71
+(33,31,34)		12419.84	2111.86
(363,269,263)	160	144332.27	622.06
+(40,32,33)		12206.77	2875.14

Table 1: CONWIP simulation — Section 7.2.1

of later parts with earlier parts. The earliest due date policy does worse because it does a worse job of resequencing and it does not limit inventory. The other policies limit inventory, but do no resequencing at all. See Deshpande (1999) for details.

7.2 Semiconductor Fab

In this section, we describe simulations of a small semiconductor fabrication facility in which three part types are produced. There are 59 distinct machine types, and a total of 93 machines. Type 1 parts receive 171 process steps; type 2 parts get 224 process steps; and type 3 parts have 270 process steps. All reported times are in minutes, and all production quantities are in lots of 12 wafers. In the CPP implementation, there was one control point: the cleaning work center.

7.2.1 Demand less than, but close to capacity

Table 1 contains the results of a simulation in which a modified CONWIP policy was used. For each experiment in the table, the first line represents the total number of parts produced, the mean total WIP, the mean lead time, and the mean lateness. Lateness is measured by only counting the time a part is late; if it is early, its lateness is counted as zero, not a negative number. In all three experiments, the CONWIP population was set at 160. This number was used by the management of the fab.

In the second line, the first column indicates the number of parts of each type that were left in the finished goods buffer at the end of the run because they were made earlier than their due dates. The last two columns show the standard deviation of lead time and lateness.

Tables 1– 3 show the results of a set of runs in which the demands are (363, 269, 263). As we will see, these demands are feasible. Table 2 shows the corresponding results for the control point policy. Almost all measures are substantially better. Most significantly, the CONWIP policy produces far in excess of the requirements, while the control point policy is almost exactly on target.

The CONWIP population was reduced to the same level as in the control point run, and Table 3 was produced. Now CONWIP underproduces and its lateness is considerably worse.

Production	WIP	Lead time	Lateness
(362,268,263)	50.73	46956.72	863.05
+(0,0,1)		11138.75	1921.71
(363,266,262)	50.93	47190.22	1114.61
+(0,0,0)		11116.74	2549.96
(363,269,263)	50.49	46771.68	680.66
+(0,1,1)		11065.21	1875.78

Table 2: Control point simulation — Section 7.2.1

Production	WIP	Lead time	Lateness
(355,268,263)	51	46365.57	8671.39
+(0,0,0)		9377.21	10799.61
(356,264,263)	51	46893.75	3095.72
+(0,0,0)		10300.90	7575.73
(359,268,263)	51	46534.00	7558.66
+(0,0,2)		9529.48	9941.98

Table 3: CONWIP simulation — reduced population — Section 7.2.1

7.2.2 Demand much less than capacity

Here the demand is reduced to (218, 161, 158). Table 4 shows the results when the CONWIP policy is used with the original population of 160. Lateness is reduced to zero, but only at the expense of a very large finished goods inventory. Table 5 shows the results of a set of control point policy runs. The overrun has nearly vanished and the WIP is substantially lower. Lateness has increased, but it is still very small compared with the lead time.

Based on the results of Table 5, the CONWIP policy is run again with a population of 30, and the results are shown in Table 6. There is still overproduction. The decrease in lateness from Table 5 is not important.

Production	WIP	Lead time	Lateness
(218,161,158)	160	235672.62	0
+(57,43,45)		20891.13	0
(218,161,158)	160	235679.23	0
+(57,44,46)		20890.85	0
(218,161,158)	160	235184.34	0
+(57,45,46)		20880.34	0

Table 4: CONWIP simulation — Section 7.2.2

Production	WIP	Lead time	Lateness
(218,161,158)	29.93	46114.81	20.39
+(2,1,1)		11249.51	201.77
(218,161,158)	29.95	46139.23	44.81
+(2,2,1)		11261.26	364.36
(218,161,151)	29.98	46185.18	90.76
+(1,2,0)		11219.65	559.67

Table 5: Control point simulation — Section 7.2.2

Production	WIP	Lead time	Lateness
(218,161,158)	30	44982.80	0
+(5,4,6)		8927.03	0
(218,161,158)	30	44982.80	0
+(5,5,7)		8927.03	0
(218,161,158)	30	44982.80	51.01
+(4,4,6)		8925.22	488.49

Table 6: CONWIP simulation — reduced population — Section 7.2.2

7.2.3 Conclusion

These preliminary results suggest that the control point policy performs considerably better than the policy that had been in use in the system that was simulated. Of course, more experiments are needed to confirm this. If this continues to hold, it suggests that the control point policy is a reasonable one for practical factory control.

7.3 Boeing Implementation

The policy was implemented at the Boeing Flap Support Business Unit in Portland, Oregon. This factory makes roughly 200 different part types which are fabricated and assembled. Some of the parts follow reentrant processing paths. In the part of the factory that we worked with, there are roughly 40 different work stations including a batching machine.

Boeing personnel implemented the policy with little difficulty. The required real-time information was available from an existing MRP system. In the course of this experiment, it became clear that control need not be exerted at every resource in the factory. The blocking criterion was modified to limit the number of parts of each type between successive control points.

The policy had only a short time to work, and the implementation was interrupted by a reorganization and a reduction in demand. We therefore have no quantitative results. Subjective impressions, however, indicate that the factory seemed to run a bit more smoothly and predictably than before. See Deshpande (1999) for details.

8 Conclusion

8.1 Summary

This paper describes a connection between dynamic programming-based (hedging point or control point) and other scheduling policies for manufacturing systems. It shows that single-part type systems operated according to the hedging point policy can be treated as token-based systems. Thus, these and factories run with certain other policies can be analyzed and optimized by means of decomposition methods. In particular, the paper suggests a future method for choosing hedging points for multiple-stage systems.

8.2 Future Research

There are many opportunities for significant research in this area.

1. Consider policy variations. For example, when there only limited control points, perhaps we should consider a part type to be blocked at a control point when the total amount of that type of material between the current and the next control point reaches a limit.

2. Perform extensive simulation experiments to test how well the control point policy performs; perform factory experiments for the same purpose and to verify that it is not difficult to implement.

3. Develop decomposition methods for token-based control schemes. Extensions of existing methods include the analysis of continuous material (fluid) models with

   - multiple part types (see Nemec, 1999; Syrowicz, 1999)
   - non-acyclic network with multiple loops
   - reentrant systems
   - yield losses and alternate routing (see Helber, 1997, 1999; Dallery, 1999).

4. Develop optimization methods for these kinds of systems.

5. Carry out the demand model extensions proposed in Section 6.

6. Improve assembly performance by using a token to coordinate production. This may involve a term like $\int g(x_a - x_b)\ \mathrm{dt}$ in the cost function (10), where a and b are resources that process parts that will be assembled together downstream.

7. Develop real-time extensions for batching and setups, and modify the decomposition method accordingly.

8. Investigate the stability of the control point policy by means of the methodology of Kumar, Meyn, Dai, and their colleagues.

9. Develop a dynamic programming decomposition of (10). For each machine M_i, find a function g^i such that minimizing

$$E\int_0^T g^{sq}(b_{11}(t), b_{12}(t), ..., x_{i(\ell),\ell}(t))dt,$$

(where $S(i(\ell), \ell) = i$) is the same as, or approximately the same as, minimizing (10). If such a function exists, we have broken the factory into two parts: M_i and above, and the rest. (Of course, we have to do it in a way that makes sense. Perhaps M_i cannot be inside a reentrant loop.) This generalizes the view of the factory in (10), and may lead to improved computational methods.

Acknowledgments

This material 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.

I am grateful for the non-financial support provided by the Boeing Company Flap Support Business Unit of Portland, Oregon (especially by Gary Collison) for the experimental implementation. Thanks are due to Sawan Deshpande for his work with Boeing and with the simulation, and to Omar Gzouli for his simulation work.

The paper has been much improved as a result of the attention given to it by two anonymous reviewers, and by Michael Caramanis.

References

[1] R. Akella and P. R. Kumar (1986), "Optimal Control of Production Rate in a Failure Prone Manufacturing System," *IEEE Transactions on Automatic Control*, Vol. AC-31, No. 2, pp. 116-126, February, 1986.

[2] T. Altiok (1996), *Performance Analysis of Manufacturing Systems,* Springer Series in Operations Research, Springer, 1996.

[3] T. Bielecki and P. R. Kumar (1988), "Optimality of Zero-Inventory Policies for Unreliable Manufacturing Systems," *Operations Research,* Volume 36, No. 4, July-August, 1988, pp. 532-541.

[4] A. M. Bonvik, private communication, 1994.

[5] A. M. Bonvik (1996), Performance Analysis of Manufacturing Systems Under Hybrid Control Policies, Ph.D. Thesis, Massachusetts Institute of Technology, Operations Research Center, June, 1996.

[6] A. M. Bonvik, C. E. Couch, and S. B. Gershwin (1997), "A Comparison of Production-Line Control Mechanisms," *International Journal of Production Research*, Vol. 35, No. 3, pp. 789-804, 1997.

[7] M. H. Burman (1995), "New Results in Flow Line Analysis", Ph.D. Thesis, Massachusetts Institute of Technology, Operations Research Center, June, 1995.

[8] J. A. Buzacott and J. G. Shanthikumar (1992), "Design of Manufacturing Systems Using Queuing Models," *Queueing Systems Theory and Applications, Special Issue on Queueing Models of Manufacturing Systems*, Volume 12, No. 1-2, December, 1992, pp. 135-213.

[9] J. A. Buzacott and J. G. Shanthikumar (1993), *Stochastic Models of Manufacturing Systems*, Prentice Hall, 1993.

[10] J. G. Dai (1995a), "On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," *Annals of Applied Probability*, Volume 5, pp. 49-77 (1995).

[11] J. G. Dai (1995b), "Stability of open multiclass queueing networks via fluid models," in the *Proceedings of the IMA workshop on stochastic networks*, Frank Kelly and Ruth Williams Editors. pp. 71-90 (1995), Springer-Verlag, New York.

[12] J. G. Dai (1996), "A fluid-limit model criterion for instability of multiclass queueing networks," *Annals of Applied Probability*, Vol 6, pp 751-757 (1996).

[13] J. G. Dai and S. Meyn (1995), "Stability and Convergence of Moments for Multiclass Queueing Networks via Fluid Limit Models," *IEEE Transactions on Automatic Control*, Vol. AC-40, pp. 1889-1904, 1995.

[14] J. G. Dai and G. Weiss, "Stability and Instability of Fluid Models for Re-Entrant Lines", *Mathematics of Operations Research*, Vol. 21, pp. 115–134 (1996).

[15] Y. Dallery (1999), "Extending the Scope of Analytical Methods for Performance Evaluation of Manufacturing Flow Systems," Second Aegean International Conference on "Analysis and Modeling of Manufacturing Systems," May 16-20, 1999, Tinos Island, Greece, 1999. (http://www.samos.aegean.gr/icsd/secaic/)

[16] Y. Dallery, R. David, and X.-L. Xie (1988), "An Efficient Algorithm for Analysis of Transfer Lines with Unreliable Machines and Finite Buffers," *IIE Transactions,* Vol. 20, No. 3, pp. 280-283, September, 1988.

[17] Y. Dallery and S. B. Gershwin, "Manufacturing Flow Line Systems: A Review of Models and Analytical Results," *Queueing Systems Theory and Applications, Special Issue on Queueing Models of Manufacturing Systems*, Volume 12, December, 1992, pp. 3-94.

[18] Y. Dallery and G. Liberopoulos (1997), "Extended Kanban Control system: A New Kanban-Type Pull Control Mechanism for Multi-Stage Manufacturing Systems," Laboratoire MASI, Universite Pierre et Marie Curie, Paris, December, 1997.

[19] Y. Dallery and G. Liberopoulos (2000), "Extended kanban control system: combining kanban and base stock," to appear in *IIE Transactions on Design and Manufacturing,* Vol. 32, 2000.

[20] S. Deshpande (1999), "A Scheduling Policy Experiment for Lean Implementation," MIT Technology and Policy Program MS Thesis, 1999.

[21] M. Di Mascolo, R. David, and Y. Dallery (1991), "Modeling and Analysis of Assembly Systems with Unreliable Machines and Finite Buffers," *IIE Transactions*, Vol. 23, No. 4, December 1991, pp. 315-331.

[22] D. Down and S. Meyn (1997), "Piecewise Linear Test Functions for Stability and Instability of Queueing Networks," *Queueing Systems: Theory and Applications*, Volume 27, pp. 205-226, 1997.

[23] Y. Frein, C. Commault and Y. Dallery (1996), "Modeling and analysis of closed-loop production lines with unreliable machines and finite buffers," *IIE Transactions*, 1996.

[24] Y. Frein, M. Di Mascolo and Y. Dallery (1994), "On the design of generalized kanban control systems," *International Journal of Operations and Production Management*, Vol. 15, No. 9, pp. 158-184.

[25] S. B. Gershwin (1987), "An Efficient Decomposition Method for the Approximate Evaluation of Tandem Queues with Finite Storage Space and Blocking," *Operations Research.*, pp. 291-305, March-April, 1987.

[26] S. B. Gershwin (1991), "Assembly/Disassembly Systems: An Efficient Decomposition Algorithm for Tree-Structured Networks," *IIE Transactions*, Vol. 23, No. 4, December 1991, pp. 302-314.

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

[28] S. B. Gershwin (1997), "Design and Operation of Manufacturing Systems — Control- and System-Theoretical Models and Issues," Proceedings of the 1997 American Control Conference, Albuquerque, New Mexico, June, 1997, pages 1909-1913.

[29] S. B. Gershwin and J. E Schor (1997), "Efficient Algorithms for Buffer Space Allocation," *Performance Evaluation and Optimization of Production Lines,* May 19-21, 1997, on Samos Island, in the Department of Mathematics, University of the Aegean, GR-832 00 Karlovassi, Samos Island, Greece, 1997. To appear in *Annals of Operations Research.*

[30] S. Helber (1997), "Approximate Analysis of Unreliable transfer Lines with Splits in the Flow of Material," *Performance Evaluation and Optimization of Production Lines,* May 19-21, 1997, on Samos Island, in the Department of Mathematics, University of the Aegean, GR-832 00 Karlovassi, Samos Island, Greece. To appear in *Annals of Operations Research.*

[31] S. Helber (1999) *Performance Analysis of Flow Lines with Non-Linear Flow of Material*, Springer Lecture Notes in Economic and Mathematical Systems, Volume 473, Springer, 1999.

[32] F. Karaesmen, J. A. Buzacott, and Y. Dallery, (1999), "Integrating Advance Information in Pull Type Control Mechanisms for Multi-Stage Production," Second Aegean International Conference on "Analysis and Modeling of Manufacturing Systems," May 16-20, 1999, Tinos Island, Greece, 1999. (`http://www.samos.aegean.gr/icsd/secaic/`)

[33] P. R. Kumar and T. I. Seidman (1990), "Dynamic Instabilities and Stabilization Methods in Distributed Real-Time Scheduling of Manufacturing Systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 3, pp. 289-298, March 1990.

[34] P. R. Kumar and T. I. Meyn (1995), "Stability of Queueing Networks and Scheduling Policies," *IEEE Transactions on Automatic Control*, pp. 251-260, vol. 40, no. 2, February 1995.

[35] J. G. Kimemia and S. B. Gershwin (1983), "An Algorithm for the Computer Control of Production in a Flexible Manufacturing Systems," *IIE Transactions*, Vol. 15, No. 4, pp. 353-362, December, 1983. Reprinted in *Modeling and Control of Automated Manufacturing Systems* by Alan A. Desrochers, IEEE Computer Society Press Tutorial, 1990.

[36] G. Liberopoulos and Y. Dallery (1997), "A Unified Framework for Pull Control Mechanisms in Multi Stage Manufacturing Systems," Laboratoire MASI, Universite Pierre et Marie Curie, Paris, October, 1997.

[37] Lou, S.X.C., Sethi, S.P. and Zhang, Q. (1994), "Optimal Feedback Production Planning in a Stochastic Two-Machine Flowshop," *European Journal of Operational Research*, 73, 1994, pp. 331-345.

[38] S. X. C. Lou, and P. Kager (1989), "A Robust Production Control Policy for VLSI Wafer Fabrication," *IEEE Transaction on Semiconductor Manufacturing*, Nov. 1989, Vol. 2, pp. 159-164.

[39] Lou, S.X.C., Van Ryzin, G. (1989), Optimal Control Rules for Scheduling Job Shops, *Annals of Operations Research*, Vol. 17, 1989, pp. 233-248.

[40] S. H. Lu and P. R. Kumar (1991), "Distributed Scheduling Based on Due Dates and Buffer Prioritization," *IEEE Transactions on Automatic Control*, Vol. 36, December, 1991.

[41] S. Meyn (1995), "Transience of Multiclass Queueing Networks via Fluid Limits," *Annals of applied probability*, 1995.

[42] S. Meyn (1997), "Stability and Optimization of Queueing Networks and Their Fluid Models," *Proceedings of the Summer Seminar on "The Mathematics of Stochastic Manufacturing Systems"* American Mathematical Society, 1997.

[43] J. E. Nemec (1999), "Diffusion and Decomposition Approximations of Stochastic Models of Multiclass Processing Networks," MIT Operations Research Center Ph. D. thesis, February, 1999.

[44] J. Orlicky (1975), *Material Requirements Planning*, McGraw-Hill, 1975.

[45] J. R. Perkins and P. R. Kumar, "Optimal Control of Pull Manufacturing Systems," *IEEE Transactions on Automatic Control*, vol. 40, number 12, pp. 2040-2051, Dec. 1995.

[46] R. Rishel (1975), "Dynamic Programming and Minimum Principles for Systems with Jump Markov Disturbances," *SIAM Journal on Control,* Vol.13, No.2, February, 1975.

[47] J. E. Schor (1995), "Efficient Algorithms for Buffer Allocation," MIT EECS MS Thesis, May, 1995.

[48] J. E. Schor and S. B. Gershwin (1996), "Efficient Algorithms for Buffer Allocation," 1996 Manufacturing and Service Operations Management Conference Proceedings, Dartmouth College, The Amos Tuck School, Hanover, New Hampshire 03755.

[49] D. Syrowicz (1999), "Decomposition Analysis of a Deterministic, Multiple-Part-Type, Multiple-Failure-Mode Production Line," MIT EECS Master's thesis, June, 1999.

[50] N. Srivatsan (1993), "Synthesis of Optimal Policies for Stochastic Manufacturing Systems," Ph.D. Thesis, Massachusetts Institute of Technology, Operations Research Center, September, 1993.

[51] N. Srivatsan and Y. Dallery (1998), "Partial characterization of optimal hedging point policies in unreliable two-part-type manufacturing systems," *Operations Research*, Vol. 46, pp. 36-45, 1998.

[52] T. Tolio and S. B. Gershwin (1998), "Throughput Estimation in Cyclic Queueing Networks with Blocking," *Annals of Operations Research*, Volume 79, pp. 207–229 March, 1998.

[53] G. J. Van Ryzin (1987), "Control of Manufacturing Systems with Delay," MIT EECS Master's thesis, June, 1987.

[54] G. J. Van Ryzin, X.-C. Lou, and S. B. Gershwin (1993), "Production Control for a Tandem Two-Machine System," *IIE Transactions*, Volume 25, No. 5, pp. 5-20, September, 1993.

[55] L. Wein (1988), "Scheduling Semiconductor Wafer Fabrication," *IEEE Transactions on semiconductor manufacturing*, Vol. 1, No. 3, August 1988.

[56] Yan, H., Lou, S.X.C., Sethi, S.P., Gardel, A., and Deosthali, P. (1996), "Testing the Robustness of Two-Boundary Control Policies in Semiconductor Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 9, N. 2, May 1996, pp. 285-288.