DYNAMIC INSTABILITIES AND STABILIZATION METHODS IN DISTRIBUTED REAL-TIME SCHEDULING OF MANUFACTURING SYSTEMS*

P. R. Kumar[†] and T. I. Seidman [‡]

Abstract

We consider manufacturing systems consisting of many machines and producing many types of parts. Each part-type requires processing for a specified length of time at each machine in a prescribed sequence of machines. Machines may require a set-up time when changing between part-types, and parts may incur a variable transportation delay when moving between machines. The goal is to dynamically schedule all the machines so that all the part-types are produced at the desired rates while maintaining bounded buffer sizes at all machines.

In this paper we study the interaction of two types of feedbacks, one caused by "cycles" of material flow in non-acyclic manufacturing systems, and the other introduced by the employment of closed-loop scheduling algorithms. We examine the consequences of this interaction for the stability properties of the manufacturing system in terms of maintaining bounded buffer levels.

First, we resolve a previously open problem by exhibiting the instability of all "clearing policies" for some non-acyclic manufacturing

*The research of the first author has been supported in part by the National Science Foundation under Grant No. ECS-88-02576, and in part by the Manufacturing Research Center of the University of Illinois, and the research of the second author has been supported in part by the Air Force Office of Scientific Research under Contract No. AFOSR-87-0190.

[†]Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign. **Mailing Address:** University of Illinois, C.S.L., 1101 W. Springfield Ave., Urbana, IL 61801. **Tel.** No. 217-333-7476. Please address all correspondence to the first author.

[‡]Department of Mathematics, University of Maryland Baltimore County, 5401 Wilkens Avenue, Baltimore, MD 21228.

systems. Surprisingly, such instabilities can occur even when all set-up times are identically zero, and they are induced purely by starvation of machines. Simultaneously however, there can exist certain exact sets of initial conditions for which a delicate stability does hold; however, it may not be robust. Second, we exhibit sufficient conditions on the system topology and processing and demand rates which ensure the stability of distributed clear-a-fraction policies. These conditions are easy to verify. Third, we study general supervisory mechanisms which will stabilize any policy. One such universal stabilization mechanism requires only a supervisory level which truncates all excessively long production runs of any part-type, and maintains a separate priority queue for all part-types with large buffer levels. It is easily implementable in a distributed fashion at the machines, and the level of supervisory intervention can also be adjusted as desired.

1 INTRODUCTION

Consider a manufacturing system with the following features.

(i) There are P types of parts labeled 1, 2, ,..., P and M machines labeled 1, 2, ,..., M.

(ii) Parts of type p enter the manufacturing system, at a rate of d_p parts/time-unit, at a machine $\mu_{p,1}$. Thereafter they visit machines $\mu_{p,2}, \mu_{p,3}, \ldots, \mu_{p,n_p}$, in order, where each $\mu_{p,i} \in \{1, 2, , \ldots, M\}$, and exit the system from machine μ_{p,n_p} as a finished product. We allow for the possibility that parts may visit a given machine more than once; this happens if $\mu_{p,i} = \mu_{p,j}$ for $i \neq j$.

(iii) At the i-th machine $\mu_{p,i}$ visited by parts of type p, they require a processing time $\tau_{p,i}$. (We shall refer to $\tau_{p,i}^{-1}$ as the *processing rate*). While awaiting processing there, they are stored in a buffer labeled $b_{p,i}$.

(iv) The buffers $B_m := \{b_{p,i} : \mu_{p,i} = m\}$ serve machine m. When machine m switches from processing parts in buffer $b \in B_m$ to processing parts in buffer $b' \in B_m$, it incurs a set-up time $\delta_{b,b'}$.

Our goal is to schedule the entire manufacturing system so that the levels of all the buffers at all the machines are bounded over all time, and hopefully small. If this is the case, we shall say that the manufacturing system is stable under the given scheduling policy.

In this paper we study the dynamic behavior of non-acyclic manufacturing systems. Due to the presence of "cycles" of material flows, there is the possibility of a destabilizing "positive feedback" effect. Simultaneously, there is also the effect of feedback caused by the deployment of "closed-loop" scheduling policies. Our broad objective in this paper is to study the interactions of these two feedbacks on the dynamics and stability of manufacturing systems, and the implications of this interaction for dynamic scheduling of manufacturing systems.

Our approach to scheduling is in contrast to "classical" scheduling which adopts a static, open-loop approach. Excellent references are the books by Baker [1], Conway, Maxwell and Miller [2], French [3], Coffman [4]; the surveys by Panwalker and Iskander [5], Graves [6], Lawler, Lenstra and Rinooy Kan [7], Lenstra and Rinooy Kan [8]; and a good recent reference is Adams, Balas and Zawack [9] which provides an efficient algorithm. In the classical approach, given a finite number of parts each requiring a specified processing time at various machines, the issue of scheduling is usually formulated as an optimization problem of sequencing the parts at the machines so as to minimize criteria such as makespan or weighted flowtime. Thus, one obtains a mixed integer/linear programming problem. A significant limitation of this approach is that large systems are typically intractable since the computational demands grow exponentially with the number of parts or machines. In fact a test problem consisting of ten parts and ten machines posed in 1963 in Muth and Thompson [10] has become notorious for its intractability, and only recently has its optimal solution been obtained; see [9].

The spirit of our approach is more in keeping with the pioneering work of Kimemia and Gershwin [11] who study dynamic "closed-loop" scheduling for systems with random machine failures. This line of work has been continued by Akella and Kumar [12], Bielecki and Kumar [13], Sharifnia [14] and Fleming, Sethi and Soner [15]. Our work in this paper can be regarded as occupying an even lower rung in the time-scale hierarchy of scheduling problems delineated by Gershwin [16], since we neglect the effect of machine failures but do incorporate the effect of set-up times [17].

The manufacturing systems considered in this paper have been introduced in Perkins and Kumar [18]. Purely for notational convenience we have not introduced here the operations of assembly and disassembly allowed there, as well as the possibility of bounded but variable transportation delays as parts move from machine to machine. However, all the results of this paper are valid for the most general model of [18].

We note also that neither in [18] nor here do we consider the case where there are many parallel machines with a part being free to choose which of the parallel machines to be processed at; rather we restrict our attention to situations where routes are specified a-priori.

While in Perkins and Kumar [18] the brunt of the analysis has been for "acyclic" systems (defined in Section 2), in this paper we concentrate on non-acyclic systems. Our main results here are the following. First we show that clear-a-fraction policies (see Section 2) do *not* stabilize all non-acyclic systems, thus resolving the significant open question posed in [18]. This shows the existence of a destabilizing "positive feedback" effect in such systems. We do this by presenting two examples of such systems where *all* clearing policies will result in linearly growing, and thus unbounded, peak buffer levels. In one of the examples, the cyclicity results from a single part-type returning to the same machines more than once, while in the other example no part-type ever revisits a machine. Moreover, in the latter example, we show that there are sets of initial conditions for which CAF policies are stable. Thus stability and instability are not properties independent of initial system states, and stable and unstable regimes can coexist simultaneously, as in nonlinear systems.

Our next result, also via an example, is that such instability of clearing policies can hold even when all set-up times are identically zero. In fact, we exhibit dynamic instabilities which are purely "starvation" induced. Till now it had been suspected that such instabilities could only manifest themselves in the presence of strictly positive set-up times.

Motivated by the existence of unstable modes of behavior, we obtain sufficient conditions on the system topology, the demand rates of part-types, and the processing rates of the machines, which guarantee that clear-a-fraction policies are stable for *all* initial conditions. These conditions generalize earlier results which were valid only for acyclic systems, and are tight for such systems. They are easy to verify.

Finally, for general non-acyclic systems, we examine distributed "supervisory" control mechanisms which will render *any* policy stable for all initial system states. We show that by adding a supervisory layer which (i) truncates all long production runs, and (ii) maintains a separate priority queue for buffers with large levels, we obtain an "universally" stabilizing safety mechanism which can be appended to any policy to render it stable.

The rest of this paper is organized as follows. Section 2 details some preliminaries. Section 3 shows the existence of dynamic instabilities in simple non-acyclic systems. Section 4 provides sufficient conditions for the stability of clear-a-fraction policies. In Section 5 we describe a stabilizing supervisory mechanism which can render any policy stable. Finally Section 6 contains some concluding remarks.

2 CLEARING POLICIES, NON-ACYCLIC SYSTEMS AND STABILITY

Let us denote by $x_{p,i}(t)$ the number of parts in buffer $b_{p,i}$ at time t, by $u_{p,i}(t)$ the cumulative input of parts into buffer $b_{p,i}$ over the time interval $[0,t]$, and by $y_{p,i}(t)$ the cumulative output of parts from $b_{p,i}$ over $[0,t]$. Thus, $x_{p,i}(t) = x_{p,i}(0) + u_{p,i}(t) - y_{p,i}(t)$.

We shall say that the manufacturing system is *stable* if all the buffer levels are bounded for all time, i.e. if

$$\sup_{0 \le t < \infty} x_{p,i}(t) < +\infty \qquad \text{for } 1 \le p \le P \text{ and } 1 \le i \le n_p.$$

It is clear that the *capacity condition*,

$$\rho_m := \sum_{\{(p,i):\mu(p,i)=m\}} d_p \tau_{p,i} < 1 \qquad \text{for } 1 \le m \le M, \tag{1}$$

is necessary for stability. In Perkins and Kumar [18] it has been shown that (1) is also sufficient for the existence of a scheduling policy which stabilizes the manufacturing system for all initial system states.

When a manufacturing system is stable, it is clear that for each part-type p the cumulative output from the system over the time interval $[0,t]$ differs from the cumulative input into the system, td_p, by no more than a constant. Thus if the input rate d_p of parts of type p is the desired rate of production of such parts, as we intend to be the case, then the goal of producing parts at the given rate has been met.

For convenience, in the rest of this paper we assume that the part flows are continuous as opposed to discrete, though all results of this paper apply equally well to discrete flows also.

Let us define the following types of scheduling policies.

Definition 1 *Clearing Policies*

A scheduling policy will be called a "clearing policy" if it has the property that whenever a machine m is processing parts in a buffer $b \in B_m$, then it continues to remain set-up for and to process parts from b until the first time *thereafter that simultaneously the buffer b is empty and there is some other non-empty buffer $b' \in B_m$. At such a time, the machine then commences a set-up for one of the (possible many) non-empty buffers in B_m.*

A special sub-class of clearing policies is the class of "clear-a-fraction" (CAF) policies defined below.

Definition 2 *Clear-A-Fraction (CAF) policies*

A clearing policy will be called a "clear-a-fraction" (CAF) policy, if for each machine m there exists a constant $\epsilon_m > 0$, and a constant K_m such that if machine m commences a set-up to buffer $b_{p,i} \epsilon B_m$ at time t, then

$$x_{p,i}(t) \geq \epsilon_m \sum_{\{(q,j):\mu_{q,j}=m\}} x_{q,j}(t) \quad - K_m \tag{2}$$

It should be noted that one can easily construct CAF policies which also possess the following two properties.

i) they can be implemented in a *distributed* fashion at the machines, without requiring any sharing of information or coordination of action between machines; and

ii) they can be easily implemented in *real-time* without requiring any computation to determine when to change the set-up, and to which buffer.

Let us associate with the manufacturing system a directed graph $G = (V, A)$ where the vertex set V consists of the machines, and there is an arc $(m, m') \in A$ if and only if there is some (p, i) with $\mu_{p,i} = m$ and $\mu_{p,i+1} = m'$, i.e. some part-type visits m' immediately after m. We shall say that the manufacturing system is *acyclic* if the digraph G contains no directed cycles. In [18], Perkins and Kumar have shown that *all* CAF policies are stabilizing for *all* acyclic manufacturing systems for *all* initial states of the systems. Moreover, for systems consisting of just one machine, there are CAF policies whose performance, measured by the resulting average buffer levels, is very close to a theoretical lower bound; see [18]. (There is a slight difference in the formulation in this paper compared to that in [18], in that [18] implicitly assumes that even an idle time requires a set-up).

In this paper we focus on manufacturing systems that are not acyclic; we shall label them as *non-acyclic*.

3 DYNAMIC INSTABILITIES IN NON-ACYCLIC SYSTEMS

Can dynamically unstable behavior occur in non-acyclic systems? We will show below that even in apparently very simple systems, the mere existence of directed cycles of part flows can lead to a destabilizing "positive feedback" effect. Specifically, we provide two examples of systems where *all* clearing policies are unstable.

The first example illustrates two properties. First, instability is caused by a single part-type revisiting machines. Second, the unstable behavior persists even when all set-up times are *zero*, showing that it is caused purely by loss of machine capacities due to *starvation* by upstream machines. Previously it had been suspected that only positive set-up times could cause instability.

The second example also illustrates two phenomena. First it shows that even when no part-type ever revisits any machine, i.e. when the route followed by every part-type is acyclic, the mere fact that two different part-types together give rise to a directed cycle of material flow can also lead to instability. Secondly, it shows that stable *periodic* modes of behavior can coexist with unstable modes, as in nonlinear systems, and that whether one has stable or unstable behavior can depend on the *initial state* of the system.

These two examples thus resolve the open question posed in [18] whether clear-a-fraction policies stabilize all non-acyclic systems, and exhibit precisely the mechanisms underlying dynamic instability.

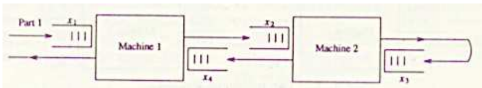

Figure 1: The System of Example 1

Example 1 Consider the system shown in Figure 1. There is a single part-type which first visits machine 1, then machine 2, then machine 2 again, and finally, machine 1 again. The successive buffers visited will be denoted by 1, 2, 3 and 4, respectively. The input rate d into buffer 1 is 1 part/time-unit, while the processing times required at the buffers are τ_1, τ_2, τ_3 and τ_4, respectively. Lastly, the times for setting-up to buffers 1 and 4 at machine 1 will be denoted by δ_1 and δ_4, while the times for setting-up to buffers 2 and 3 at machine 2 are δ_2 and δ_3. We shall assume that a clearing policy is used for scheduling; there is only *one* such policy.

We will assume that the processing times satisfy the following critical condition,

$$\tau_2 + \tau_4 > 1. \tag{3}$$

Of course, we will assume that the capacity condition (1) is satisfied, i.e.

$$\tau_1 + \tau_4 < 1 \tag{4}$$

and

$$\tau_2 + \tau_3 < 1. \tag{5}$$

Note that (3-5) necessarily imply that

$$\tau_1 < \tau_2 < 1 \quad \text{and} \quad \tau_3 < \tau_4 < 1. \tag{6}$$

We will analyze the evolution of the clearing policy.

Case 1: Positive set-up times

We first consider the usual situation where the set-up times are all positive, i.e. $\delta_1, \delta_2, \delta_3, \delta_4 > 0$.

Let us consider an initial time instant $t_1 = 0$ at which time the initial buffer levels are

$$x(t_1) := (x_1(t_1), x_2(t_1), x_3(t_1), x_4(t_1)) = (\xi, 0, 0, 0)$$

where $\xi > 0$ is large enough. We will also suppose that at time t_1, machine 1 is currently set-up for buffer 4, while machine 2 is set-up for buffer 3.

As in [18], throughout this paper, for simplicity of presentation, we will treat part flows as continuous rather than discrete; all the results remain valid for discrete flows with the only difference being that one will have to incorporate the effects of integer round-offs into the calculations.

We will analyze the future evolution of the system in stages, as in event-driven simulation. A graph of the behavior of the system is shown in Figure 2.

Stage 1. Machine 1 begins a set-up for buffer 1 at t_1 and completes the set-up at a time $t_2 := t_1 + \delta_1$. One has $x_1(t_2) = \xi + \delta_1$ due to the inflow of δ_1 units into buffer 1. However, one still has $x_2(t_2) = x_3(t_2) = x_4(t_2) = 0$.

Stage 2. Machine 2 begins a set-up for buffer 2 at t_2 and completes the set-up at time $t_3 := t_2 + \delta_2$. One has $x_1(t_3) = x_1(t_2) + \delta_2 - \frac{\delta_2}{\tau_1}$, since an amount δ_2 has entered buffer 1 while the amount $\frac{\delta_2}{\tau_1}$ has been processed and sent to

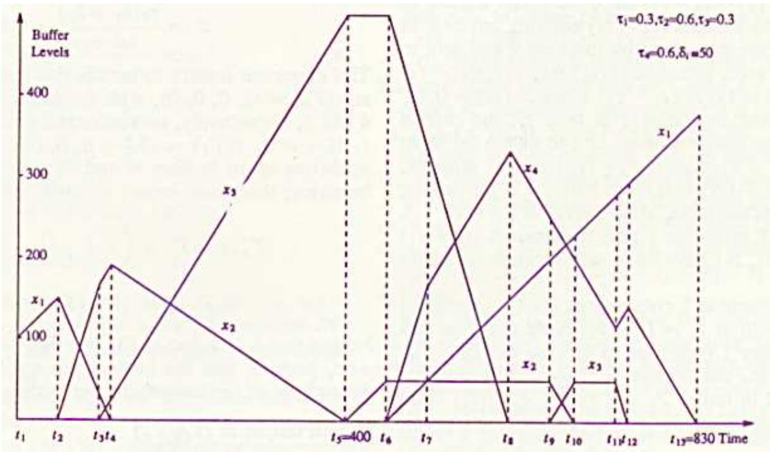

Figure 2: The Behavior of the System of Example 1 for the Case of Positive Set-Up Times

buffer 2. Thus $x_2(t_3) = \frac{\delta_2}{\tau_1}$, while $x_3(t_3) = x_4(t_3) = 0$. If ξ is large enough, we are guaranteed that buffer 1 does not empty in the interval $[t_2, t_3]$.
Stage 3. Machines 1 and 2 work on buffers 1 and 2, respectively, in the time-interval $[t_3, t_4]$ where $t_4 := t_2 + \frac{x_1(t_2)}{\tau_1^{-1}-1}$ is the time at which buffer 1 empties, i.e. $x_1(t_4) = 0$. The condition $\tau_1 < \tau_2$ of (6) ensures that the level of buffer 2 stays positive throughout $[t_3, t_4]$. Note that $x_2(t_4) = \xi + (t_4 - t_1) - \frac{1}{\tau_2}(t_4 - t_3)$ since the initial amount in buffer 1 at time t_1 plus the amount $(t_4 - t_1)$ entering buffer 1 in the time interval $[t_1, t_4]$ are processed by machine 1 and sent to buffer 2, while it has, in turn, processed and sent $\frac{1}{\tau_2}(t_4 - t_3)$ to buffer 3. Thus $x_3(t_4) = \frac{1}{\tau_2}(t_4 - t_3)$ while $x_4(t_4) = 0$.
Stage 4. After emptying its buffer 1 at time t_4, the *rate* at which machine 1 works on buffer 1 drops to 1 to match the input rate exactly. At $t_5 := t_4 + \frac{x_2(t_4)}{\tau_2^{-1}-1}$, buffer 2 empties, i.e. $x_2(t_5) = 0$. One has $x_1(t_5) = 0$ since buffer 1 is processing at exactly the same rate as the input. Also $x_3(t_5) = \xi + (t_5 - t_1)$, since the initial amount ξ in buffer 1 plus the amount $(t_5 - t_1)$ entering the system in $[t_1, t_5]$ have been sent through to buffer 3. Still, $x_4(t_5) = 0$.
Stage 5. At time t_5, machine 2 commences a set-up to buffer 3 and completes the set-up at $t_6 := t_5 + \delta_3$. Note that one still has $x_1(t_6) = 0$ since buffer 1 continues to process at exactly its input rate, $x_2(t_6) = \delta_3$ since the amount δ_3 processed by machine 1 in $[t_5, t_6]$ is deposited in buffer 2, and $x_3(t_6) = x_3(t_5)$ since machine 2 has not processed any parts from buffer 2 in $[t_5, t_6]$. Still,

$x_4(t_6) = 0$.

Stage 7. At t_6 buffer 4 starts filling up, and so machine 1 commences a set-up for buffer 4, which is completed at $t_7 := t_6 + \delta_4$. One has $x_1(t_7) = \delta_4, x_2(t_7) = x_2(t_6), x_3(t_7) = x_3(t_6) - \frac{\delta_4}{\tau_3}$, and $x_4(t_7) = \frac{\delta_4}{\tau_3}$.

Stage 6. At $t_8 := t_7 + \tau_3 x_3(t_7)$, buffer 3 empties and so $x_3(t_8) = 0$. The condition $\tau_3 < \tau_4$ of (6) guarantees that machine 1 is still processing buffer 4, and so $x_4(t_8) = x_4(t_7) - (\frac{1}{\tau_4} - \frac{1}{\tau_3})(t_8 - t_7)$. The level of buffer 1 continues to increase due to the external input of rate 1, and so $x_1(t_8) = x_1(t_7) + (t_8 - t_7)$, and $x_2(t_8) = x_2(t_7)$.

Stage 8. Since buffer 3 empties at t_8, machine 2 commences a set-up for buffer 2 at t_8, and completes it at $t_9 := t_8 + \delta_2$. One has $x_1(t_9) = x_1(t_8) + (t_9 - t_8), x_2(t_9) = x_2(t_8), x_3(t_9) = 0$ and $x_4(t_9) = x_4(t_8) - \frac{1}{\tau_4}(t_9 - t_8)$. If ξ is large enough, buffer 4 will not empty in $[t_8, t_0]$.

Stage 9. At $t_{10} := t_9 + \tau_2 x_2(t_8)$, buffer 2 empties, i.e. $x_2(t_{10}) = 0$. One has $x_1(t_{10}) = x_1(t_9) + (t_{10} - t_9), x_3(t_{10}) = x_2(t_9)$, and $x_4(t_{10}) = x_4(t_9) - \frac{1}{\tau_4}(t_{10} - t_9)$. Again, if ξ is large enough, buffer 4 does not empty in $[t_9, t_{10}]$.

Stage 10. Machine 2 commences a set-up for buffer 3 at t_{10} which is completed at $t_{11} := t_{10} + \delta_3$. One has $x_1(t_{11}) = x_1(t_{10}) + (t_{11} - t_{10})$, $x_2(t_{11}) = 0$, $x_3(t_{11}) = x_3(t_{10})$, and $x_4(t_{11}) = x_4(t_{10}) - \frac{1}{\tau_4}(t_{11} - t_{10})$. The assumption that ξ is large enough guarantees that buffer 4 does not empty in $[t_{10}, t_{11}]$.

Stage 11. At t_{11}, machine 2 commences processing parts from buffer 3, emptying it at $t_{12} := t_{11} + \tau_3 x_3(t_{11})$. Thus $x_3(t_{12}) = 0$. Moreover, when ξ is large enough, machine 1 is still processing parts from buffer 4. Thus $x_1(t_{12}) = x_1(t_{11}) + (t_{12} - t_{11})$ and $x_2(t_{12}) = 0$. Finally, $x_4(t_{12}) = x_4(t_{11}) - \frac{1}{\tau_4}(t_{12} - t_{11}) + x_3(t_{11})$ since the contents of buffer 3 at t_{11} have been processed and sent to buffer 4.

Stage 12. At $t_{13} := t_{12} + \tau_4 x_4(t_{12})$, buffer 4 empties, i.e. $x_4(t_{13}) = 0$. One has $x_1(t_{13}) = x_1(t_{12}) + (t_{13} - t_{12}), x_2(t_{13}) = 0$, and $x_3(t_{13}) = 0$. Note that machine 1 is still set-up for buffer 4 at t_{13}, while machine 2 is still set-up for buffer 3.

A straightforward computation shows that $t_{13} = t_1 + (\lambda + \frac{\tau_2}{1-\tau_2})\xi + \alpha$, and $x_1(t_{13}) = \lambda\xi + \beta$, where

$$\lambda := \frac{\tau_4}{1 - \tau_2} > 1,$$

$$\alpha := \frac{(\tau_4 + 1)(\delta_1 + \delta_2)}{(1 - \tau_2)} + \delta_3(\tau_4 + 1) + \delta_4,$$

and

$$\beta := \frac{\tau_4(\delta_1+\delta_2)}{1-\tau_2} + \tau_4\delta_3 + \delta_4.$$

The important feature to note is that starting at a certain time T_0 at $x(T_0) = (\xi, 0, 0, 0)$, with machines 1 and 2 set-up to buffers 4 and 3, respectively, we return at a time $T_1 := T_0 + (\lambda + \frac{\tau_2}{1-\tau_2})\xi + \alpha$ to $x(T_1) = (\lambda\xi + \beta, 0, 0, 0)$ with machines 1 and 2 again set-up to buffers 4 and 3, respectively. This pattern of behavior therefore recurs infinitely often, and we obtain,

$$T_{n+1} = T_n + (\lambda + \frac{\tau_2}{1-\tau_2})x_1(T_n) + \alpha,$$

$$x(T_{n+1}) = (\lambda x_1(T_n) + \beta, 0, 0, 0).$$

Noting that $\lambda > 1$ due to (3), we thus obtain $\lim_{n\to\infty} x_1(T_n) = +\infty$, proving that the buffers are not bounded. Regarding the throughput of the manufacturing system we note that,

$$\frac{\text{System output in } [T_0, T_n]}{\text{System input in } [T_0, T_n]} = 1 - \frac{x_1(T_n) - x_1(T_0)}{T_n - T_0} \quad \to \quad \frac{1}{\tau_2 + \tau_4} < 1,$$

by virtue of (3). Hence the system does not produce parts at the desired (input) rate. ■

Case 2: No Set-Up Times

Now we turn to the situation where $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 0$, i.e. all the set-up times are *zero*. We will assume that the initial condition at $t_1 = 0$ is as before, i.e. $x(t_1) = (\xi, 0, 0, 0)$ where $\xi > 0$. Due to the absence of set-up times, the analysis is in fact a little easier than before.

Stage 1. At $t_2 := t_1 + \frac{x_1(t_1)}{\tau_1^{-1}-1}$, buffer 1 clears. During the interval $[t_1, t_2]$, machine 2 is processing buffer 2, and so $x(t_2) = (0, \xi - (\frac{1}{\tau_2} - 1)(t_2 - t_1), \frac{1}{\tau_2}(t_2 - t_1), 0)$. After t_2, since buffer 4 is empty, machine 1 continues to process buffer 1, but at the *reduced* rate of 1, due to *starvation.*

Stage 2. At time $t_3 = t_2 + \frac{x_2(t_2)}{\tau_2^{-1}-1}$, buffer 2 is cleared. Thus $x(t_3) = (0, 0, \xi + t_3 - t_1, 0)$. Since machine 2 starts processing buffer 3 at t_3, buffer 4 starts filling up, and so machine 1 starts processing buffer 4.

Stage 3. At time $t_4 = t_3 + \tau_3 x_3(t_3)$, buffer 3 clears. Hence $x(t_4) = (t_4 - t_3, 0, 0, (\frac{1}{\tau_3} - \frac{1}{\tau_4})(t_4 - t_3))$.

Stage 4. At time $t_5 = t_4 + \tau_4 x_4(t_4)$, buffer 4 clears. Thus $x(t_5) = (t_5 - t_3, 0, 0, 0)$.

A computation shows that $x_1(t_5) = \lambda\xi$ where $\lambda := \frac{\tau_4}{1-\tau_2} > 1$ as before. Thus at t_5, the state of the system is a magnified version of that at time t_1. Hence $\lim_{n\to\infty} x_1(t_{4n+1}) = +\infty$. ■

The main reason for the instability of this system is that during time intervals such as $[t_2, t_3]$, machine 1 is forced to work at the reduced rate of 1 on buffer 1, due to its buffers being starved of input. Thus, valuable processing capacity is lost. Also machine 2 is forced to be idle during intervals such as $[t_4, t_5]$ due to starved buffers. Thus the dynamic instability is purely starvation induced, and oscillatory. The oscillatory nature is caused solely by the non-acyclic nature of the manufacturing system.

The following example exhibits two further phenomena. First, it shows that dynamic instability can arise even if no part-type ever revisits a machine. It also shows that one can have stability or instability, depending on the system initial conditions.

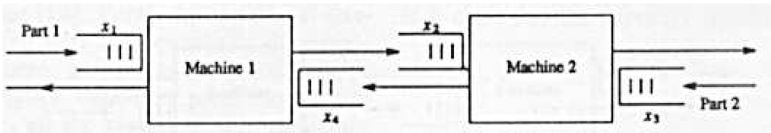

Figure 3: The System of Example 2

Example 2 Consider the system shown in Figure 3 where there are 2 part-types and 2 machines. Both part-types have an input rate of 1. Part-type 1 first visits buffer 1 at machine 1 and then buffer 2 at machine 2, while part-type 2 first visits buffer 3 at machine 2 and then buffer 4 at machine 1. The processing times of the parts in buffers 1, 2, 3 and 4 are τ_1, τ_2, τ_3 and τ_4, respectively, while the times for setting-up to these buffers are $\delta_1, \delta_2, \delta_3, \delta_4 > 0$.

We will assume that the parameters above satisfy,

$$\tau_2 + \tau_4 > 1, \tag{7}$$

$$\frac{\delta_1 + \delta_4}{1 - \tau_1 - \tau_4} = \frac{\delta_2 + \delta_3}{1 - \tau_2 - \tau_3} =: \eta, \tag{8}$$

$$\delta_4 < (1 - \tau_3 - \tau_4)\eta, \tag{9}$$

$$\delta_2 < (1 - \tau_1 - \tau_2)\eta, \tag{10}$$

and that the capacity condition (1) is met, i.e.

$$\tau_1 + \tau_4 < 1 \quad \text{and} \quad \tau_2 + \tau_3 < 1.$$

(A set of feasible parameters is $(\tau_1, \tau_2, \tau_3, \tau_4, \delta_1, \delta_2, \delta_3, \delta_4) = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{2}{3}, \frac{1}{30}, \frac{1}{5}, \frac{1}{20}, \frac{1}{20})$, for example). We will analyze this system when a scheduling policy of clearing type is employed; there is only one such clearing policy.

Case 1: A Periodic Regime

First we will show that there exists an initial condition for which all buffers are bounded; for this demonstration we do not need the assumption (7).

Consider a time $t = 0$ at which buffers 1 and 3 have just been cleared, and so a set-up commences for buffers 2 and 4. We will assume that $x_2(0) = x_4(0) = \eta > 0$, while $x_1(0) = x_3(0) = 0$, as stated.

Note that machine 1 completes the set-up for buffer 4 at time $t = \delta_4$, then clears buffer 4 at time $t = \delta_4 + \eta\tau_4$. and completes the set-up for buffer 1 at $t = \delta_4 + \eta\tau_4 + \delta_1$.

Meanwhile, machine 2 completes the set-up for buffer 2 at $t = \delta_2$, then clears buffer 2 at $t = \delta_2 + \eta\tau_2$, and completes the set-up for buffer 3 at $t = \delta_2 + \eta\tau_2 + \delta_3$.

Since $\delta_4 + \eta\tau_4 < \delta_2 + \eta\tau_2 + \delta_3$, by virtue of (9) and the second equality in (8), buffer 4 is cleared before machine 2 starts processing buffer 3, and so one does not need to consider the possibility that buffer 4's clearing is delayed. Similarly, due to $\delta_2 + \eta\tau_2 < \delta_4 + \eta\tau_4 + \delta_1$ by virtue of (10,8), buffer 2's clearing time is indeed $\delta_2 + \eta\tau_2$.

At $t = \delta_4 + \eta\tau_4 + \delta_1$, buffer 1 has accumulated $\delta_4 + \eta\tau_4 + \delta_1$ parts, which are cleared at time $t = (\delta_4 + \eta\tau_4 + \delta_1) + \frac{(\delta_4 + \eta\tau_4 + \delta_1)}{\tau_1^{-1} - 1} = \eta$. Similarly buffer 3 is cleared at time exactly equal to η.

Thus at $t = \eta$, we return to the same system state as that at $t = 0$, and so we have a *periodic* steady state where all buffers are bounded for all time. ■

It is worth noting that this synchronization of events depends on condition (8) holding *exactly*. It is a rather fragile stability, as we shall see in the next case.

Case 2: An Unstable Regime

Now we will consider a different initial condition for the system. At time t_1, we start with $x(t_1) = (\xi, 0, 0, 0)$ with $\xi > 0$ large enough, and machine 1 set-up for buffer 4, while machine 2 is set-up for buffer 3. We shall abbreviate the details of the arguments. (Conditions (8-10) are unnecessary for this analysis).

Stage 1. Machine 1 completes the set-up for buffer 1 at $t_2 = t_1 + \delta_1$, and $x(t_2) = (x_1(t_1) + \delta_1, 0, 0, \frac{1}{\tau_3}(t_2 - t_1))$. Note that machine 2 is processing at the reduced rate of 1 at buffer 3 in $[t_1, t_2]$.

Stage 2. Machine 2 completes the set-up for buffer 2 at $t_3 = t_2 + \delta_2$. $x(t_3) = (x_1(t_2) - (\frac{1}{\tau_1} - 1)(t_3 - t_2), \frac{1}{\tau_1}(t_3 - t_2), (t_3 - t_2), x_4(t_2))$.

Stage 3. Machine 1 clears buffer 1 at $t_4 = t_3 + \frac{x_1(t_3)}{\tau_1^{-1} - 1}$. $x(t_4) = (0, x_2(t_3) + (\frac{1}{\tau_1} - \frac{1}{\tau_2})(t_4 - t_3), x_3(t_3) + (t_4 - t_3), x_4(t_3))$.

Stage 4. Machine 1 completes the set-up for buffer 4 at $t_5 = t_4 + \delta_4$. $x(t_3) = ((t_5 - t_4), x_2(t_4) - \frac{1}{\tau_2}(t_5 - t_4), x_3(t_4) + (t_5 - t_4), x_4(t_4))$.

Stage 5. Machine 1 clears buffer 4 at $t_6 = t_5 + \tau_4 x_4(t_4)$. $x(t_6) = (x_1(t_5) + (t_6 - t_5), x_2(t_5) - \frac{1}{\tau_2}(t_6 - t_5), x_3(t_5) + (t_6 - t_5), 0)$.

Stage 6. Machine 1 completes the set-up for buffer 1 at $t_7 = t_6 + \delta_1$. $x(t_7) = (x_1(t_6) + (t_7 - t_6), x_2(t_6) - \frac{1}{\tau_2}(t_7 - t_6), x_3(t_6) + (t_7 - t_6), 0)$.

Stage 7. Machine 1 clears buffer 1 at $t_8 = t_7 + \frac{x_1(t_7)}{\tau_1^{-1} - 1}$. $x(t_8) = (0, x_2(t_7) + (\frac{1}{\tau_1} - \frac{1}{\tau_2})(t_8 - t_7), x_3(t_7) + (t_8 - t_7), 0)$. Since buffer 4 is empty, machine 1 continues to work on buffer 1 at the reduced rate of 1 after t_8.

Stage 8. Machine 2 clears buffer 2 at $t_9 = t_8 + \frac{x_2(t_8)}{\tau_2^{-1} - 1}$. $x(t_9) = (0, 0, x_3(t_8) + (t_9 - t_8), 0)$.

Any easy way to compute t_9 is as follows. During the period $[t_1, t_9]$ machine 2 has processed the amount ξ in buffer 1 at t_1 plus the amount $(t_9 - t_1)$ that came in during $[t_1, t_9]$. Thus $\frac{1}{\tau_2}(t_9 - t_1 - \delta_1 - \delta_2) = \xi + (t_9 - t_1)$, giving $t_9 = t_1 + \frac{\xi + \tau_2^{-1}(\delta_1 + \delta_2)}{\tau_2^{-1} - 1}$. Since buffer 3 was not processed during $[t_2, t_9], x_3(t_9) = t_9 - t_2 = t_9 - t_1 - \delta_1 = \frac{\xi + \tau_2^{-1}(\delta_1 + \delta_2)}{\tau_2^{-1} - 1} - \delta_1$.

Note that at t_9, the state of the system is the mirror-image of the state at t_1. Hence, by repeating the mirror image of the analysis of Stages 1-8, we arrive at a time t_{17} when $x(t_{17}) = (\lambda\xi + \alpha, 0, 0, 0)$ where $\lambda := \frac{\tau_2 \tau_4}{(1-\tau_2)(1-\tau_4)}$ and $\alpha := \left[\frac{\delta_1 + \delta_2}{1 - \tau_2} - \delta_1 + \frac{1}{\tau_4}(\delta_3 + \delta_4)\right]\left(\frac{1}{\tau_4} - 1\right)^{-1}$. Also, as at t_1, machine 1 is set-up for buffer 4 and machine 2 for buffer 3.

Hence the situation is the same as at t_1, except that the level of buffer 1, x_1, has been magnified. Thus $\limsup_t x_1(t) = +\infty$. ■

Thus we see that the same system can possess both stable and unstable modes of behavior. Moreover, for instability, it is not necessary for a part to revisit a machine.

4 A SUFFICIENT CONDITION FOR STABILITY OF CAF POLICIES

Once the existence of dynamic instability has been established, it is natural to ask what the conditions are under which CAF policies are stable for *all* initial conditions.

The weakest easily verifiable sufficient condition for such stability, which we have been able to determine, involves a consideration of the system topology as well as demand and processing rates.

The intuitive idea is as follows. Suppose that a machine m receives inflow of a part-type p from a machine m'. There are two cases to consider. If there is *any* flow of any type of part from m back to m', then m and m' are involved in a cycle of material flows, and we have already seen that in such cycles there is the potential for instability. Thus, to rule out such instability, we will require that machine m possess enough capacity to handle the "peak" inflow rate of parts of type p, i.e. the rate at which machine m' processes such parts. On the other hand, if there is *no* flow of parts of any type from m back to m', then m and m' are not involved in a cycle of material flows and so there is no danger of instability; consequently we only require that machine m have enough capacity to handle the *average* rate of inflow of parts of type p, which is d_p. The result below shows that when every machine has enough capacity to handle peak demand for parts coming to it from machines with which it is involved in a cycle, but only average demand when otherwise, then the resulting overall system is stable under CAF policies. This resulting capacity condition is more stringent than (1) because the rate at which a predecessor machine processes parts is always larger than the average inflow rate. In what follows we make this idea precise.

Let us first construct a directed graph $G = (V, A)$ which differs slightly from that in Section 2. Let V be the set of machines, and let the arc set A be defined as follows,

$$A := \{(m, m') | m \neq m', \text{ and there exists } (p, i) \text{ such that } \mu_{p,i} = m, \mu_{p,i+1} = m'\}.$$

The only difference between the graph defined here and that in Section 2 is that here we eliminate all "self-loops" consisting of arcs of the form (m, m) from A. (Thus the graph G defined here may have no directed cycles, and yet the manufacturing system can still be non-acyclic as defined in Section 2).

We shall say that m' is *reachable* from m if there is a directed path from m to m'; we shall write this as "$m \to m'$." If $m \to m'$ and $m' \to m$, we shall say that m and m' are *diconnected*, and write "$m \leftrightarrow m'$". By convention, we shall always say that $m \leftrightarrow m$. Note that "$\leftrightarrow$" is an *equivalence relation*; hence it partitions V into disjoint sets $G_1, \ldots, G_r$ with the properties that (i) $m \leftrightarrow m'$ if and only $m, m' \in G_i$ for some i, (ii) $G_i \cap G_j = \phi$ for $i \neq j$, and (iii) $\cup_i G_i = V$.

On $\{G_1, \ldots, G_r\}$ we have the following induced *partial ordering.* We will say that $G_i \to G_j$ if and only if there exist $m \in G_i, m' \in G_j$ with $m \to m'$.

Our sufficient condition for stability of CAF policies is given by the following theorem.

Theorem 1 *For each buffer $b_{p,i}$ define,*

$$\begin{aligned} \lambda_{p,i} &:= 0 \textit{ if } \mu_{p,j} = \mu_{p,i} \textit{ for all } j \leq i, \\ &:= j \textit{ if } \mu_{p,j} \neq \mu_{p,i} \textit{ but } \mu_{p,k} = \mu_{p,i} \textit{ for } j+1 \leq k \leq i, \end{aligned} \tag{11}$$

and then define the "prior machine" $\pi_{p,i}$ *visited by part-type p as follows:*

$$\begin{aligned} \pi_{p,i} &:= \phi \textit{ if } \lambda_{p,i} = 0, \\ &:= \mu_{p,\lambda_{p,i}} \textit{ otherwise.} \end{aligned} \tag{12}$$

For each buffer $b_{p,i}$ we next define the "modified input rate" $d'_{p,i}$ *as follows:*

$$\begin{aligned} d'_{p,i} &:= d_p \textit{ if } \text{either } \pi_{p,i} = \phi \text{ or } \pi_{p,i} \textit{ is not diconnected with } \mu_{p,i}, \\ &:= \frac{1}{\tau_{p,\lambda_{p,i}}} \textit{ if } \pi_{p,i} \leftrightarrow \mu_{p,i}. \end{aligned} \tag{13}$$

If the more stringent capacity condition,

$$\rho'_m := \sum_{\{(p,i):\mu_{p,i}=m\}} d'_{p,i}\tau_{p,i} < 1 \quad \textit{for all } m, \tag{14}$$

holds, then all *CAF policies are stable for* all *system initial conditions.*

Proof: Let us consider the equivalence classes of machines $G_1, \ldots, G_r$ induced by the relation "$\leftrightarrow$". As noted earlier, on $\{G_1, \ldots, G_r\}$ there is a natural partial ordering "$\to$", and let $\{G_1, \ldots, G_l\}$, say, be the set of *minimal* elements with respect to this partial ordering. (We say that G_i is "minimal" if there is no G_j such that $G_j \to G_i$).

First we will show that all the buffer levels at the machines in any of these minimal elements are bounded. For the sake of definiteness, consider a machine $m \in G_1$.

For each buffer $b_{p,i}$ with $\mu_{p,i} = m$ we define

$$\eta_{p,i} := \max\{j : \mu_{p,k} = m \text{ for } i \leq k \leq i + j - 1\}$$

and note that $(\eta_{p,i} - 1)$ is the number of times that part-type p subsequently "loops back" to m before visiting another machine or exiting from the system.

We shall consider the following "Lyapunov function,"

$$V_m(t) := \sum_{\{(p,i):\mu_{p,i}=m\}} x_{p,i}(t)[\tau_{p,i} + \tau_{p,i+1} + \ldots + \tau_{p,i+\eta_{p,i}-1}]. \tag{15}$$

Let $\{T_n\}$ be the sequence of times at which machine m commences a set-up, and let $(p_n^\star, i_n^\star)$ denote the buffer selected for the set-up of m commenced at time T_n. Note that by the property (2) of CAF policies

$$x_{p_n^\star, i_n^\star}(T_n) \geq \epsilon_m \sum_{\{(p,i):\mu_{p,i}=m\}} x_{p,i}(T_n) - K_m.$$

For any (p,i) with $\mu_{p,i} = m \in G_1$, if $\pi_{p,i} = \phi$, then $\mu_{p,i}$ is the *first* machine visited by part-type p. If $\pi_{p,i} \neq \phi$, then there is some prior machine $\pi_{p,i} \in \{1, 2, \ldots, M\}$, different from m. Note that since part-type p moves from $\pi_{p,i}$ to $\mu_{p,i}$, we have $\pi_{p,i} \to \mu_{p,i}$. If $\mu_{p,i} \not\to \pi_{p,i}$, then this would violate the minimality of G_1. Hence $\mu_{p,i} \leftrightarrow \pi_{p,i}$. Thus we see that the only two possibilities are $\pi_{p,i} = \phi$, or $\pi_{p,i} \leftrightarrow \mu_{p,i}$. Hence, by the definition of $d'_{p,i}$ in (13), *either* $d'_{p,i} = d_p$ in which case part-type p enters the system at machine m, *or* $d'_{p,i} = \frac{1}{\tau_{p,\lambda_{p,i}}}$. Since $\frac{1}{\tau_{p,\lambda_{p,i}}}$ is the *maximum* rate at which parts can move out of buffer $b_{p,\lambda_{p,i}}$ to buffer $b_{p,i}$ and into m, it follows that in either case, the maximum rate of entry of parts *into* m is $d'_{p,i}$. Thus,

$$\text{Inflow of work to } m \text{ in } [T_n, T_{n+1}] \leq \sum_{\{(p,i):\mu_{p,i}=m,\mu_{p,i-1}\neq m\}} d'_{p,i}[\tau_{p,i} + \tau_{p,i+1} + \ldots + \tau_{p,i+\eta_{p,i}-1}](T_{n+1} - T_n) \tag{16}$$

where *work* is measured in the time-units that machine m would take to process.

Now we consider two cases. If machine m has been *active* during the *entire* period $[T_n + \delta_{b_{p_{n-1}^\star, i_{n-1}^\star}, b_{p_n^\star, i_n^\star}}, T_{n+1}]$, then one has,

$$\begin{aligned} V_m(T_{n+1}) \leq V(T_n) \quad &+ \sum_{\{(p,i):\mu_{p,i}=m,\mu_{p,i-1}\neq m\}} d'_{p,i}[\tau_{p,i} + \tau_{p,i+1} + \ldots + \tau_{p,i+\eta_{p,i}-1}](T_{n+1} - T_n) \\ &- \quad (T_{n+1} - T_n - \delta_{b_{p_{n-1}^\star, i_{n-1}^\star}, b_{p_n^\star, i_n^\star}}) \end{aligned}$$

since by (16) the first term in the above summation represents an upper bound on the work that has arrived in the interval $[T_n, T_{n+1}]$, while the second term in the summation represents the work done in $[T_n+\delta_{b_{p^\star_{n-1},i^\star_{n-1}},b_{p^\star_n,i^\star_n}}, T_{n+1}]$, On the other hand, machine m could have been forced to process at *less* than the maximum rate due to *starvation*, which happens when $V_m(s) = 0$ for some $s \in (T_n, T_{n+1}]$. But then, since (14) implies $d'_{p,i} < \frac{1}{\tau_{p,i}}$, the rate at which work enters in the interval $[s, T_{n+1}]$ is less than the rate at which it is completed, and so it follows that $V_m(t) = 0$ for $s \le t \le T_{n+1}$, and in particular $V_m(T_{n+1}) = 0$. Hence, in either case, one has the inequality,

$$\begin{aligned} V_m(T_{n+1}) \le \max[0, V(T_n) \quad &+ \sum_{\{(p,i):\mu_{p,i}=m,\mu_{p,i-1}\neq m\}} d'_{p,i}[\tau_{p,i} + \tau_{p,i+1} + \ldots + \tau_{p,i+\eta_{p,i}-1}](T_{n+1} - T_n) \\ &- \quad (T_{n+1} - T_n - \delta_{b_{p^\star_{n-1},i^\star_{n-1}},b_{p^\star_n,i^\star_n}})] \end{aligned} \tag{17}$$

It is easy to see that since $\mu_{p,i} = \mu_{p,i-1} \Rightarrow d'_{p,i} = d'_{p,i-1}$ by virtue of (11,12,13), one has

$$\sum_{\{(p,i):\mu_{p,i}=m,\mu_{p,i-1}\neq m\}} d'_{p,i}[\tau_{p,i}+\tau_{p,i+1}+\ldots+\tau_{p,i+\eta_{p,i}-1}] = \sum_{\{(p,i):\mu_{p,i}=m\}} d'_{p,i}\tau_{p,i} = \rho'_m,$$

where the last equality follows from the definition in (14). Hence, (17) simplifies to,

$$V_m(T_{n+1}) \le \max[0, V_m(T_n) - (1-\rho'_m)(T_{n+1} - T_n) + \bar{\delta}], \tag{18}$$

where $\bar{\delta} := \max_{b,b' \in B_m} \delta_{b,b'} > 0$.

As a consequence of (18) one has the following implication:

$$(T_{n+1} - T_n) \ge \frac{2\bar{\delta}}{1-\rho'_m} \Rightarrow V_m(T_{n+1}) \le \max[0, V_m(T_n) - \bar{\delta}]. \tag{19}$$

Moreover by the "clearing" property:

$$x_{p^\star_n,i^\star_n}(T_n) \ge \frac{2\bar{\delta}}{(1-\rho'_m)\tau_{p^\star_n,i^\star_n}} \Rightarrow T_{n+1} - T_n \ge \frac{2\bar{\delta}}{1-\rho'_m}. \tag{20}$$

Also, by the CAF property (2):

$$\sum_{\{(p,i):\mu_{p,i}=m\}} x_{p,i}(T_n) \ge \frac{2\bar{\delta}}{\epsilon_m(1-\rho'_m)\tau_{p^\star_n,i^\star_n}} + \frac{K_m}{\epsilon_m} \Rightarrow x_{p^\star_n,i^\star_n}(T_n) \ge \frac{2\bar{\delta}}{(1-\rho'_m)\tau_{p^\star_n,i^\star_n}}. \tag{21}$$

Finally, by (15):

$$V_m(T_n) \geq \frac{2\bar{\delta}\bar{\tau}}{\epsilon_m(1-\rho'_m)\underline{\tau}} + \frac{K_m\bar{\tau}}{\epsilon_m} \Rightarrow \sum_{\{(p,i):\mu_{p,i}=m\}} x_{p,i}(T_n) \geq \frac{2\bar{\delta}}{\epsilon_m(1-\rho'_m)\tau_{p_n^\star,i_n^\star}} + \frac{K_m}{\epsilon_m}, \tag{22}$$

where $\bar{\tau} := \max_{\{(p,i):\mu_{p,i}=m\}}[\tau_{p,i}+\tau_{p,i+1}+\ldots+\tau_{p,i+\eta_{p,i}-1}]$ and $\underline{\tau} := \min_{\{(p,i):\mu_{p,i}=m\}} \tau_{p,i}$.

From (22,21,20,19), it thus follows that by concatenating the implications:

$$V_m(T_n) \geq \Gamma_m \Rightarrow V_m(T_{n+1}) \leq V_m(T_n) - \bar{\delta},$$

where $\Gamma_m := \max\left(\frac{2\bar{\delta}\bar{\tau}}{\epsilon_m\underline{\tau}(1-\rho'_m)} + \frac{K_m\bar{\tau}}{\epsilon_m}, \bar{\delta}\right)$. This proves the ultimate boundness of $V_m(T_n)$, and thus also of $V_m(t)$ since $V_m(t) \leq V_m(T_n) + \bar{\delta}\rho'_m$ for all $t \in [T_n, T_{n+1}]$.

Thus we have shown that the levels of the buffers of all the machines in the minimal elements $G_1, \ldots, G_l$ are bounded by, say, Γ.

The proof now proceeds by induction. Since the remaining steps of the induction are all similar, for simplicity of notation we show the second step of the induction process only.

Let us now delete these minimal elements from $\{G_1, \ldots, G_r\}$, and consider the *reduced set* $\{G_{l+1}, \ldots, G_r\}$ with the *same* (i.e. a restriction of the) partial order "$\rightarrow$" as before. Let $G_{l+1}, \ldots, G_{l+q}$, say, be the *new* minimal elements of $\{G_{l+1}, \ldots, G_r\}$ with respect to the partial order "$\rightarrow$". Consider an arbitrary minimal element, say G_{l+1}, and consider an arbitrary machine m in it, i.e. $m \in G_{l+1}$.

Let us fix attention on a buffer $b_{p,i} \in B_m$. If $\pi_{p,i} = \phi$, then part-type p enters the system at m, and so the rate of entry of such parts is $d'_{p,i} = d_p$ by (13). If not, then there is a previous machine $\pi_{p,i} \in \{1, \ldots, M\}$ visited by the part-type p. Note that $\pi_{p,i} \notin \cup_{n=l+2}^{r} G_n$ since that would violate the minimality of G_{l+1}. Hence either $\pi_{p,i} \in G_{l+1}$ or $\pi_{p,i} \in \cup_{n=1}^{l} G_n$. If $\pi_{p,i} \in G_{l+1}$, then since $d'_{p,i} := \frac{1}{\tau_{p,i}}$ by (13), it follows that the maximum rate at which parts move from $b_{p,\lambda_{p,i}}$ to m is $d'_{p,i}$. Lastly, if $\pi_{p,i} \in \cup_{n=1}^{l} G_n$, then $\pi_{p,k} \in \cup_{n=1}^{l} G_n$ for all $1 \leq k \leq i-1$, since $G_1, \ldots, G_l$ were originally minimal, and so part-type p enters the system at a rate d_p at one of the machines in $\cup_{n=1}^{l} G_n$. However, since all the buffers of such machines are bounded by Γ,

$$\text{Inflow of parts in } [T_n, T_{n+1}] \leq d'_{p,i}[T_{n+1} - T_n] + \Gamma, \tag{23}$$

since $d'_{p,i} := d_p$ in this case.

Note therefore, that in *all* cases, one has an inequality such as (23) (where Γ can even be set to 0 in the first two cases), and so,

$$\begin{aligned}
\text{Inflow of } \textit{work} \text{ to } m \text{ in } [T_n, T_{n+1}] \quad &\le \quad \sum_{\{(p,i):\mu_{p,i}=m,\mu_{p,i-1}\neq m\}} \{d'_{p,i}[\tau_{p,i} + \ldots + \tau_{p,i+\eta_{p,i}-1}](T_{n+1} - T_n) \\
&+ \quad \Gamma(\tau_{p,i} + \ldots + \tau_{p,i+\eta_{p,i}-1})\} \\
&= \quad \rho'_m(T_{n+1} - T_n) + \bar{\Gamma},
\end{aligned}$$

where $\bar{\Gamma} := \sum_{\{(p,i):\mu_{p,i}=m,\mu_{p,i-1}\neq m\}} \Gamma[\tau_{p,i} + \ldots + \tau_{p,i+\eta_{p,i}-1}]$. This is a simple generalization of the bound in (16).

Now let us examine the work done by machine m in the time interval $[T_n + \delta_{b_{p^\star_{n-1},i^\star_{n-1}},b_{p^\star_n,i^\star_n}}, T_{n+1}]$, in which period machine m is active. If it did *not* process at the maximum rate of $\frac{1}{\tau_{p^\star_n,i^\star_n}}$ throughout this time interval, then there is a time instant $s \in (T_n, T_{n+1}]$ at which $V(s) = 0$, since machine m must have been starved at some such time instant. If $s = T_{n+1}$, then $V_m(T_{n+1}) = 0$. If not, let s be the last such time instant. In $[s, T_{n+1}]$, machine m works at the maximum rate, and so the work *completed* is $(T_{n+1} - s)$. However then the work entering in $[s, T_{n+1}]$ can be at most $\rho'_m(T_{n+1} - s) + \bar{\Gamma}$.

Thus, by covering all cases, we obtain the relation,

$$\begin{aligned}
V_m(T_{n+1}) \quad &\le \quad \max\{0, V_m(T_n) - (1 - \rho'_m)(T_{n+1} - T_n) + \delta_{b_{p^\star_n,i^\star_n},b_{p^\star_n,i^\star_n}} + \bar{\Gamma}, \max_{T_n \le s \le T_{n+1}} [\bar{\Gamma} - (1 - \rho'_m)(T_{n+1} \\
&\le \quad \max\{\bar{\Gamma}, V_m(T_n) - (1 - \rho'_m)(T_{n+1} - T_n) + \delta_{b_{p^\star_n,i^\star_n}} + \bar{\Gamma}\},
\end{aligned}$$

which is a simple generalization of the recurrence bound (17). Thus by repeating the earlier arguments, we obtain the ultimate boundness of $V_m(T_n)$. The boundedness of $V_m(t)$ follows by noting that $V_m(t) \le V_m(T_n) + \bar{\delta}\rho'_m + \bar{\Gamma}$ for all $t \in [T_n, T_{n+1}]$.

Continuing by induction, one can next remove $\{G_{l+1}, \ldots, G_{l+q}\}$, and by a repeated application of this technique, the proof is completed. ■

5 A UNIVERSALLY STABILIZING SUPERVISORY MECHANISM

The sufficient condition (14) for CAF policies to be stable for all system initial conditions is a more stringent requirement than the capacity condition (1) since $\rho'_m \ge \rho_m$. Hence rather than restrictively requiring (14) to hold, we are interested in obtaining policies which are stable whenever (1) holds. However, as shown in Section 3, dynamic instability is a very real possibility. Is there a simple "stabilization" mechanism by which a supervisor can modify *any* scheduling policy so that it becomes stable when only (1) holds?

We will now show there does exist such a "universal" safety mechanism, which can moreover be trivially implemented by a supervisor, in a distributed fashion, at the various machines. Essentially this mechanism consists simply of:

i) truncating all long productions runs, and

ii) maintaining a separate first-come first-serve (FCFS) priority queue Q_m for large buffers, at each machine m.

For each machine m, let γ_m be a large number satisfying,

$$\gamma_m(1-\rho_m) > \sum_{b \in B_m} \max_{b' \in B_m} \delta_{b',b}. \tag{24}$$

Since $\rho_m < 1$ by the capacity condition (1), such a choice is clearly feasible. Second, we shall arbitrarily choose for each buffer $b_{p,i}$ a nonnegative number $z_{p,i}$. The precise operation of the supervisor is given by the following five rules to be implemented in a distributed fashion at each machine m.

The Truncation Rule No processing run of buffer $b_{p,i}$ at machine m is ever allowed to continue beyond $\gamma_m d_p \tau_{p,i}$ time units.

The Rule for Entering $\mathbf{Q_m}$ A buffer $b_{p,i}$ enters the tail of the priority queue Q_m if (i) $b_{p,i}$ is not being processed or set-up, *and* (ii) its buffer level $x_{p,i}$ exceeds $z_{p,i}$.

The Buffer Selection Rule If $Q_m \neq \phi$ when a processing run terminates, then the buffer at the head of the priority queue Q_m (i.e. according to a FCFS discipline) is chosen next for processing.

The Rule for Leaving $\mathbf{Q_m}$ A buffer leaves the priority queue Q_m when it is taken up for processing, i.e. a set-up is commenced.

The Rule for the Processing Time for a Buffer from $\mathbf{Q_m}$ If a buffer $b_{p,i}$ from Q_m is taken up for processing, then it is processed for *exactly* $\gamma_m d_p \tau_{p,i}$ time units, unless it clears before this amount of time has elapsed.

It should be noted that the above rules do not restrict in any way which buffer is chosen for processing if Q_m is empty. Thus, except for the truncation rule, the supervisory mechanism intervenes only when buffer levels become large, and is otherwise unobtrusive.

A second point to note is that after completing a processing run, a buffer $b_{p,i}$ may immediately enter Q_m; this happens if its buffer level at the end of the processing run is larger than $z_{p,i}$.

It is worth mentioning that there is considerable flexibility in choosing the frequency of supervisory intervention. For example, if $z_{p,i} := 0$, then the supervisor simply enforces a First-Come-First-Serve (FCFS) discipline among the non-empty buffers, truncating their processing runs after $\gamma_m d_p \tau_{p,i}$ time units. Thus the supervisor always intervenes. On the other hand, if a policy is already stable, and if both $z_{p,i}$ and γ_m are chosen large enough, then the supervisor never intervenes. Hence, by choosing $z_{p,i}$ and γ_m, the degree of supervisory intervention can be adjusted.

In what follows we prove that this supervisory mechanism guarantees stability.

The following preliminary lemma exhibits four key consequences of the supervisory mechanism.

Lemma 1 *Suppose buffer $b_{p,i}$ enters Q_m (where $m = \mu_{p,i}$) at a time instant t_{in} and has its subsequent processing run completed at a time t_{complete}.*

i) *Then,*

$$t_{\text{complete}} - t_{\text{in}} \leq \gamma_m - \zeta_m, \tag{25}$$

where

$$\zeta_m := \gamma_m(1 - \rho_m) - \sum_{b \in B_m} \max_{b' \in B_m} \delta_{b',b}. \tag{26}$$

Note that $\zeta_m > 0$ by (24).

ii) *Let $\beta_{p,i}(t) := \sum_{j=1}^{i} \tau_{p,i} x_{p,j}(t)$ be the backlog of work for machine m in the* prior *buffers visited by part-type p. Then $x_{p,i}(t_{\text{in}}) \geq \gamma_m d_p \Rightarrow \beta_{p,i}(t_{\text{complete}}) \leq \beta_{p,i}(t_{\text{in}}) - \zeta_m d_p \tau_{p,i}$.*

iii) *Suppose t' is such that $x_{p,i}(\sigma) \geq \max(\gamma_m d_p, z_{p,i})$ for all $\sigma \in [t_{\text{in}}, t']$. Then $t' - t_{\text{in}} \leq \left(1 + \frac{\beta_{p,i}(t_{\text{in}})}{\zeta_m d_p \tau_{p,i}}\right)(\gamma_m - \zeta_m)$.*

iv) *Suppose $[t, t']$ is* any *interval such that*

$$x_{p,i}(\sigma) \geq g_{p,i} \text{ for all } \sigma \in [t, t'],$$

where $g_{p,i} := \max(\gamma_m d_p, z_{p,i}) + \gamma_m d_p$, then

$$t' - t \leq a_{p,i} + c_{p,i}\beta_{p,i}(t),$$

where $a_{p,i} := \left(2 + \frac{(\gamma_m - \zeta_m)}{\zeta_m}\right)(\gamma_m - \zeta_m)$ *and* $c_{p,i} := \frac{(\gamma_m - \zeta_m)}{\zeta_m d_p \tau_{p,i}}$.

Proof: (i) When $b_{p,i}$ enters Q_m, at worst all other buffers $b \in B_m$ can be ahead of $b_{p,i}$ in Q_m. By the truncation rule, allowing for set-up times, buffer $b_{p,i}$ is guaranteed to have its processing run terminated by,

$$\begin{aligned} t_{\text{complete}} &\leq t_{\text{in}} + \sum_{\{(q,j):\mu_{q,j}=m\}} \gamma_m d_q \tau_{qj} + \sum_{b \in B_m} \max_{b' \in B_m} \delta_{b',b} \\ &= t_{\text{in}} + \gamma_m \rho_m + \sum_{b \in B_m} \max_{b' \in B_m} \delta_{b',b} \qquad \text{(by (1))}. \\ &= t_{\text{in}} + \gamma_m - \zeta_m \qquad \text{(by (26))}. \end{aligned}$$

which proves (i).

(ii) If $x_{p,i}(t_{\text{in}}) \geq \gamma_m d_p$ at a time t_{in} that $b_{p,i}$ enters Q_m, then since its buffer cannot be cleared in less than $\gamma_m d_p \tau_{p,i}$ time units of processing, it follows from the rule for the processing time for a buffer from Q_m that the subsequent processing time lasts exactly $\gamma_m d_p \tau_{p,i}$ time units. In the time interval $[t_{\text{in}}, t_{\text{complete}}]$, a quantity $d_p(t_{\text{complete}} - t_{\text{in}})$ of parts of type p have entered the system at machine $\mu(p,1)$. Thus,

$$\begin{aligned} \beta_{p,i}(t_{\text{complete}}) &= \beta_{p,i}(t_{\text{in}}) + d_p \tau_{p,i}(t_{\text{complete}} - t_{\text{in}}) - \gamma_m d_p \tau_{p,i} \\ &= \beta_{p,i}(t_{\text{in}}) + d_p \tau_{p,i}(t_{\text{complete}} - t_{\text{in}} - \gamma_m) \\ &\leq \beta_{p,i}(t_{\text{in}}) - d_p \tau_{p,i} \zeta_m \qquad \text{(by (i))}. \end{aligned}$$

(iii) The proof is by contradiction. Suppose not, i.e. $(t' - t) > (1 + \frac{\beta_{p,i}(t_{\text{in}})}{\zeta_m d_p \tau_{p,i}})(\gamma_m - \zeta_m)$. Let $t_{\text{complete}} > t_{\text{in}}$ be the next time its processing run is terminated. By (i) it follows that $t^{(1)}_{\text{complete}} - t^{(1)}_{\text{in}} \leq \gamma_m - \zeta_m$, and so $t^{(1)}_{\text{complete}} \leq t'$. Hence $x_{p,i}(t^{(1)}_{\text{complete}}) \geq \zeta_{p,i}$ again, and so $b_{p,i}$ again enters Q_m at $t^{(2)}_{\text{in}} := t^{(1)}_{\text{complete}}$, and correspondingly let $t^{(2)}_{\text{complete}}$ be the next time at which it again completes a processing run. Again, by (i), $(t^{(2)}_{\text{complete}} - t_{\text{in}}) \leq 2(\gamma_m - \zeta_m)$. If now, $2 \leq \left(1 + \frac{\beta_{p,i}(t_{\text{in}})}{\zeta_m d_p \tau_{p,i}}\right)$, then $b_{p,i}$ again enters Q_m at $t^{(3)}_{\text{in}} := t^{(2)}_{\text{complete}}$, and so on. If $n := \lfloor 1 + \frac{\beta_{p,i}(t_{\text{in}})}{\zeta_m d_p \tau_{p,i}} \rfloor$, then there are n such intervals $[t_{\text{in}}, t^{(1)}_{\text{complete}}], [t^{(2)}_{\text{in}}, t^{(2)}_{\text{complete}}], \ldots, [t^{(n)}_{\text{in}}, t^{(n)}_{\text{complete}}]$ with

$t_{\text{in}}^{(k)} = t_{\text{complete}}^{(k-1)}$ for $2 \leq k \leq n$, at the commencement of each of which, $x_{p,i}(t_{\text{in}}^{(k)}) \geq \zeta_{p,i}$. Hence, by (ii),

$$\begin{aligned}
\beta_{p,i}(t_{\text{complete}}^{(n)}) &\leq \beta_{p,i}(t_{\text{in}}) - n\zeta_m d_p \tau_{p,i} \\
&< \beta_{p,i}(t_{\text{in}}) - \left(\frac{\beta_{p,i}(t_{\text{in}})}{\zeta_m d_p \tau_{p,i}}\right)\zeta_m d_p \tau_{p,i} \qquad (\text{since } n > \frac{\beta_{p,i}(t_{\text{in}})}{\zeta_m d_p \tau_{p,i}}) \\
&< 0,
\end{aligned}$$

which is a contradiction, since the backlog $\beta_{p,i}(t)$ is always nonnegative.
(iv) At time t since $x_{p,i}(t) \geq z_{p,i}$, if $b_{p,i}$ is not in Q_m, then it is either being processed or set-up, by the rule for entering Q_m. In any event, let s denote the time of the end of the resulting production run. If $b_{p,i}$ is being processed or set-up at t, then $s - t \leq \max_{b,b' \in B_m} \delta_{bb'} + \gamma_m d_p \tau_{p,i}$, where the first term on the right-hand side allows for a set-up time, and the last term represents the maximum length of a production run, due to the truncation rule. In case $b_{p,i} \in Q_m$ at t, then $s - t \leq \gamma_m - \zeta_m$ by (i). Since $\gamma_m - \zeta_m > \max_{b,b' \in B_m} \delta_{bb'} + \gamma_m d_p \tau_{p,i}$ by (26), it follows that in all cases,

$$s - t \leq \gamma_m - \zeta_m \tag{27}$$

Moreover, by the truncation rule, the number of parts processed in a run is less than or equal to $\gamma_m d_p$. Hence $x_{p,i}(s) \geq x_{p,i}(t) - \gamma_m d_p \geq \max(\gamma_m d_p, z_{p,i})$; and so $b_{p,i}$ enters Q_m at time s. By (iii) it follows that

$$t' - s \leq \left(1 + \frac{\beta_{p,i}(s)}{\zeta_m d_p \tau_{p,i}}\right)(\gamma_m - \zeta_m),$$

and from (27) we thus obtain,

$$t' - t \leq \left(2 + \frac{\beta_{p,i}(s)}{\zeta_m d_p \tau_{p,i}}\right)(\gamma_m - \zeta_m). \tag{28}$$

Now note that the maximum rate at which $\beta_{p,i}$ can grow is $d_p \tau_{p,i}$, and so by (27), we have

$$\beta_{p,i}(s) \leq \beta_{p,i}(t) + d_p \tau_{p,i}(\gamma_m - \zeta_m).$$

Substituting in (28) yields the result. ■

Our main theorem concerning the stabilizing property of this supervisory mechanism is the following.

Theorem 2 *If the above supervisory mechanism is applied to all the machines, then all buffer levels are bounded over all time for all initial system states.*

Proof: Consider any part-type p. We will prove by induction on i that the buffer levels $x_{p,i}(t)$ are bounded for $0 \leq t < +\infty$.

Consider the first machine $m = \mu_{p,1}$ visited by part-type p. Note that, as a consequence, $\beta_{p,1}(t) = \tau_{p,1} x_{p,1}(t)$.

Let

$$t^{(1)} := \inf\{t \geq 0 | x_{p,1}(t) \geq g_{p,1}\},$$

and inductively define for $k \geq 1$,

$$\bar{t}^{(k)} := \inf\{t > t^{(k)} | x_{p,1}(t) < g_{p,1}\},$$

$$t^{(k+1)} := \inf\{t > \bar{t}^{(k)} | x_{p,1}(t) = g_{p,1}\},$$

with the convention that $\inf \phi := \infty$. Note that if $t^{(k)} < +\infty$, then $\bar{t}^{k} < +\infty$ by Lemma 1(iv). Moreover, if any $t^{(k)} = +\infty$, then $x_{p,1}(t)$ is bounded (since it is Lipschitz and has no "finite escape time"). So let us suppose $t^{(k)} < +\infty$ for all k. Note also that again by the Lipschitz property, if we can prove that $\{\bar{t}^{(k)} - t^{(k)}\}$ is a bounded sequence, then $x_{p,1}(t)$ is bounded.

Let us consider $k \geq 2$, and note that $\beta_{p,1}(t^{(k)}) = \tau_{p,1} x_{p,1}(t^{(k)}) = \tau_{p,1} g_{p,1}$, and so by Lemma 1(iv),

$$\bar{t}^{(k)} - t^{(k)} \leq a_{p,1} + c_{p,1} g_{p,1} \tau_{p,1} \quad \text{for all } k \geq 2.$$

Since $x_{p,1}(t)$ is Lipschitz, it follows that it is bounded.

Now suppose for induction that,

$$x_{p,j}(t) \leq \Gamma_j \quad \text{for all } t \geq 0, \quad 1 \leq j \leq i-1.$$

Consider $b_{p,i}$. Define, as earlier,

$$t^{(1)} := \inf\{t \geq 0 | x_{p,i}(t) \geq g_{p,i}\},$$

and inductively define for $k \geq 1$

$$\bar{t}^{(k)} := \inf\{t > t^{(k)} | x_{p,i}(t) < g_{p,i}\}$$

$$t^{(k+1)} := \inf\{t > \bar{t}^{(k)} | x_{p,i}(t) = g_{p,i}\}.$$

Again, if $t^{(k)} = +\infty$ for any k, then $x_{p,i}(t)$ is bounded. So let us suppose that $t^{(k)} < +\infty$ for all k. Now note for $k \geq 2$,

$$\begin{aligned}
\beta_{p,i}(t^{(k)}) &= \sum_{j=1}^{i} \tau_{p,i} x_{p,j}(t^{(k)}) \\
&\leq \sum_{j=1}^{i-1} \tau_{p,i} \Gamma_j + \tau_{p,i} x_{p,i}(t^{(k)}) \\
&= \sum_{j=1}^{i-1} \tau_{p,i} \Gamma_j + \tau_{p,i} g_{p,i}
\end{aligned}$$

Hence, by Lemma 1(iv),

$$\bar{t}^{(k)} - t^{(k)} \leq a_{p,i} + c_{p,i} \tau_{p,i} \left[g_{p,i} + \sum_{j=1}^{i-1} \Gamma_j \right].$$

Since $x_{p,i}(t)$ is Lipschitz, it follows that it is bounded. ∎

The mechanism described above can therefore be used as a supervisory "safety net" which can be used with any policy to prevent it from dynamic instability. Moreover, it does not require a "centralized" supervisor; a distributed implementation is feasible since each machine can act as its own supervisor.

6 CONCLUDING REMARKS

The present paper provides an analysis of the dynamics of non-acyclic manufacturing systems. We now have an understanding of the interaction of the feedback effect caused by flows along directed cycles, with the feedback introduced by employing closed-loop scheduling policies.

An important open issue is the obtaining of tight bounds on buffer levels for systems with many machines. For the single-machine case, such bounds are available in [18], and policies, whose performance measured in terms of average weighted buffer levels is very close to a theoretical lower bound, are also available. However, for general systems with many machines, a good understanding is not available, even for the simpler class of acyclic systems. Given the importance of good performance, this issue needs to be properly addressed.

It is also useful to both model and analyze the problem of dynamically scheduling the transportation subsystem. While the machine scheduling algorithms of this paper can be used with any "stable" transportation scheduling policy which guarantees bounded transportation delays, the precise analysis of performance is not available.

Finally, the treatment of random uncertainties such as yields, machine failures, rework, demand changes, etc., is very much of an open, and important problem.

References

[1] K. Baker, *Introduction to Sequencing and Scheduling.* New York: John Wiley, 1974.

[2] R. W. Conway, W. L. Maxwell and L. W. Miller, *Theory of Scheduling.* Reading, MA: Addison-Wesley, 1967.

[3] S. French, *Sequencing and Scheduling.* New York: John Wiley, 1982.

[4] E. G. Coffman, *Computer and Job-Shop Scheduling Theory.* New York: John Wiley, 1976.

[5] S. S. Panwalker and W. Iskander, "A survey of scheduling rules," *Operations Research*, vol. 25, pp. 45–61, January–February 1977.

[6] S. Graves, "A review of production scheduling," *Operations Research*, vol. 29, pp. 646–675, July-August 1981.

[7] E. L. Lawler, J. K. Lenstra and A. H. G. Rinooy Kan, "Recent developments in deterministic sequencing and scheduling," *Deterministic and Stochastic Scheduling*, 1982. Reidel, Dordrecht.

[8] J. Lenstra and A. H. G. Rinooy Kan, "Sequencing and scheduling," Preprint.

[9] J. Adams, E. Balas and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, pp. 391–401, March 1988.

[10] J. F. Muth and G. L. Thompson, *Industrial Scheduling.* Englewood Cliffs, NJ: Prentice Hall, 1963.

[11] J. Kimemia and S. B. Gershwin, "An algorithm for the computer control of production in flexible manufacturing systems," *IIE Transactions*, vol. 15, pp. 353–362, December 1983.

[12] R. Akella and P. R. Kumar, "Optimal control of production rate in a failure prone manufacturing system," *IEEE Transactions on Automatic Control*, vol. AC-31, pp. 116–126, February 1986.

[13] T. Bielecki and P. R. Kumar, "Optimality of zero-inventory policies for unreliable manufacturing systems," *Operations Research*, vol. 26, pp. 532–546, July–August 1988.

[14] A. Sharifnia, "Production control of a manufacturing system with multiple machine states," *IEEE Transactions on Automatic Control*, vol. 33, pp. 620–626, July 1988.

[15] problem with randomly fluctuating demand," *SIAM Journal on Control and Optimization*, vol. 25, pp. 1494–1502, November 1987.

[16] S. B. Gershwin, "A hierarchical framework for discrete event scheduling in manufacturing systems," in *IIASA Workshop on Discrete Event Systems: Models and Applications*, August 3-7 1987. Sopron, Hungary.

[17] S. Gershwin, "Stochastic scheduling and set-ups in manufacturing systems," in *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, (K. E. Stecke, ed.), pp. 431–442, 1986. Elsevier, Amsterdam.

[18] J. Perkins and P. R. Kumar, "Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Transactions on Automatic Control*, vol. AC-34, pp. 139–148, February 1989.