Distributed Fuzzy Control of Flexible Manufacturing Systems

Anthony Angsana and Kevin M. Passino, *Member, IEEE*

Abstract— **A flexible manufacturing system (FMS) consists of a set of machines that are connected via tracks over which parts may be transported from one machine to another for processing. As parts arrive at a machine via the tracks, they are put in a buffer (queue) where they are held before they are processed. There is a local controller (scheduling policy) at each machine which uses the machine's buffer levels to decide which part type to process next; hence the overall controller for the FMS is physically distributed across the entire FMS with local schedulers at each machine. In this paper we show how to design a fuzzy controller for a single machine and show via simulation that its performance is comparable to conventional schedulers. In addition, we introduce an adaptive fuzzy controller which can automatically synthesize itself (or tune itself if there are machine parameter variations) to achieve good throughput rates for the single machine as compared with conventional schedulers. Next we show via simulations that by using such adaptive fuzzy controllers in a distributed fashion, we obtain a distributed fuzzy controller (DFC) which can automatically synthesize itself and lower the maximum buffer level more effectively than conventional schedulers. Finally, we illustrate the ability of the DFC and conventional schedulers to automatically tune themselves in case there are unpredictable machine parameter changes in an FMS. These final results show that while sometimes the DFC performs in a superior fashion, better scheduling policies are needed to guarantee high performance FMS operation in case there are unpredictable machine parameter changes.**

I. Introduction

THE flexible manufacturing system (FMS) that we consider is a system composed of several machines such as the one shown in Fig. 1 and is the same as the ones studied in [1], [2]. The system processes several different part-types (indicated by Pi, i=1,2,3 in Fig. 1). Each part-type enters the system at a prespecified rate and is routed in the system through a sequence of machines (indicated by $Mi, i = 1, 2 \cdots, 6$ in Fig. 1) over the transportation tracks (the arrows in Fig. 1). A part-type may enter the same machine more than once for processing (i.e., the FMS is "nonacyclic"). The length of processing time for each part-type at each machine is also prespecified. The same part-type may have different processing times for the same machine at different visits, i.e., a machine may process a part-type longer at its first visit than at its second. Each part that arrives at a machine is stored in a buffer until the machine is ready to process the part. There

Manuscript received July 1, 1993; revised April 18, 1994. Recommended by Associate Editor, X.-R. Chen. This work was supported in part by an Engineering Foundation Research Initiation Grant and by National Science Foundation Grant IRI-9210332.

The authors are with the Department of Electrical Engineering, Ohio State University, Columbus, OH 43210 USA.

IEEE Log Number 9406339.

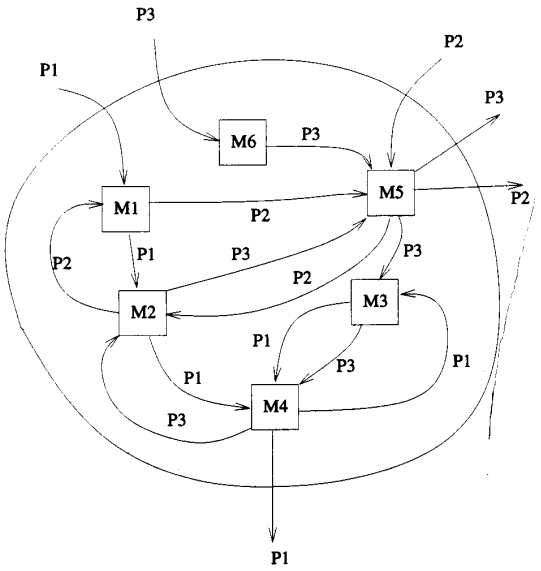


Fig. 1. Typical nonacyclic flexible manufacturing system.

are prespecified "set-up times" when the machine switches from processing one part-type to another. Each scheduler on each machine tries to minimize the size of the "backlog" of parts by appropriately scheduling the sequence of parts to be processed. The goal is to specify local scheduling policies that maximize the throughput of each part-type and hence minimize the backlog and the overall delay incurred in processing parts through the FMS.

While there has been a significant amount of work performed on the development and analysis of conventional schedulers for FMS [1]–[5], there have also been investigations into the use of artificial intelligence (AI) based techniques. For instance, the authors in [6] examine the use of negotiation as a cooperative problem-solving mechanism for matching the changing needs of FMS components to available systems resources. In [7], the authors develop an intelligent flexible manufacturing system scheduling framework which uses a hybrid architecture that integrates artificial neural networks and knowledge-based systems to generate real-time solutions for real-time scheduling of the FMS. In [8], the authors introduce "requirement-driven scheduling," an architecture for real-time distributed scheduling which attempts to satisfy several objectives on a real-time basis by integrating AI and operation research techniques to perform scheduling tasks,

including capacity planning, finite capacity scheduling, and sequencing for each machine. A detailed description of general flexible manufacturing systems and the use of knowledge-based systems to optimize the various manufacturing design and management problems is given in [9]. There exist relationships between the work in this paper and the research in "distributed intelligent systems" [10]–[17] since we can view the machines with fuzzy schedulers (fixed or adaptive) as "intelligent subsystems" and the tracks on which the parts flow as "communication links" (where an inherent message is represented by how many parts travel over the track where the amount dictates how much servicing time must be dedicated to a part). Each of the intelligent subsystems seeks to efficiently schedule part processing (a local goal) so that the overall goal of achieving high throughput rates for the whole FMS are achieved. More details on the relationships between the results reported in this paper and the research in distributed intelligent systems are provided in [18], [19]. While the approach in this paper is, perhaps, more similar in philosophy to some of the AI-based approaches, we were primarily motivated by the work in [1], [2] and our overall goal was to determine if fuzzy control (one type of "intelligent control") had anything to offer to the scheduling problem for FMS.

We begin by showing how to design a fuzzy controller for a single machine. We use simulations to illustrate that its performance is comparable to conventional scheduling policies such as the clear a fraction, clear largest buffer, and the clearing policy of Perkins and Kumar in Section IV of [1]. Next, we introduce an adaptive fuzzy controller for the single machine and show via simulation that it can automatically synthesize a scheduling policy that is as good as conventional schedulers. Then we show via simulation that if there are machine parameter changes, the adaptive fuzzy controller can tune itself so that it performs better than conventional fixed scheduling policies. Next, we utilize the adaptive fuzzy controllers as local scheduling policies on each machine in an FMS and augment each of these with a universally stabilizing supervisory mechanism from [2] to ensure stability. Our simulation results indicate that the resulting distributed fuzzy controller (DFC) can automatically synthesize itself to provide lower maximum buffer levels than conventional scheduling policies (sometimes at the expense of slightly increasing the average buffer levels). Finally we briefly study the performance of the DFC and conventional scheduling policies in reacting to machine parameter changes and show that: i) while the DFC appears promising there is no clear advantage to using either the DFC or conventional schedulers, and ii) there is a significant need for better scheduling policies for FMS which are subject to machine parameter variations.

Note that we consider our evaluation of the performance of the various intelligent control techniques to be somewhat preliminary as we only provide simulation results; no mathematical proof is given that provides general guarantees that there will be improved performance levels over conventional schedulers. We do, however, emphasize that the results in this paper are novel in that they: i) show via simulation how several intelligent controllers (the fuzzy controller, an adaptive fuzzy controller, and the DFC) can improve performance of FMS as compared to conventional schedulers and ii) provide a new approach to (distributed) adaptive fuzzy control which provides for automatic intelligent controller synthesis and tuning for performance enhancement in a single machine and a general FMS.

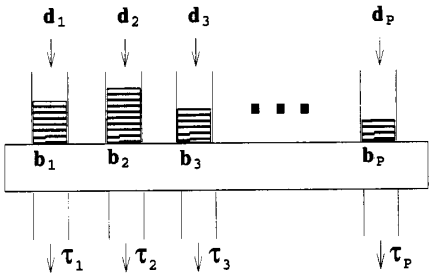


Fig. 2. Single machine with P part types.

In Section II we overview conventional scheduling policies, introduce a fuzzy controller (scheduler) for a single machine, and compare the fuzzy controller's performance to that of conventional policies. In Section III we introduce an adaptive fuzzy controller for a single machine and study its ability to i) automatically synthesize a fuzzy controller and ii) tune the fuzzy controller in case there are machine parameter variations. In Section IV we perform a study analogous to that in Section III but for FMS topologies rather than a single machine. Section V contains some concluding remarks and future directions for research.

II. Design of Fuzzy Controllers for a Single Machine

A. Conventional Scheduling Policies for the Single Machine

In this section, we briefly overview the operation of a single machine of an FMS of the type described in [1]. Fig. 2 illustrates a single machine that operates on P different part-types. The value of d_p represents the arrival rate of part-type p, and τ_p represents the amount of time it takes to process a part of type p. Parts of type p that are not yet processed by the machine are stored in buffer b_p. The single machine can process only one part at a time. When the machine switches processing from one part-type p to another part-type p', it will consume a set-up time $\delta_{p,p'}$. Here, as in [1], for convenience we will assume that all the set-up times are equal to a single fixed value δ (the conventional schedulers and the intelligent controllers studied here perform in a similar fashion for non-equal set-up times).

If a scheduling policy does not appropriately choose which part to process next, the buffer levels of the parts that are not processed often enough may rise indefinitely high, which can result in buffer overflow. To avoid that problem, the machine must have a proper scheduler. In addition to keeping the buffer levels finite, the scheduler must also increase the throughput of each part-type and decrease the buffer levels (i.e., decrease the backlog).

A block diagram of a single machine with its controller (scheduler) is shown in Fig. 3. The inputs to the controller are the buffer levels x_p of each part-type. The output from the

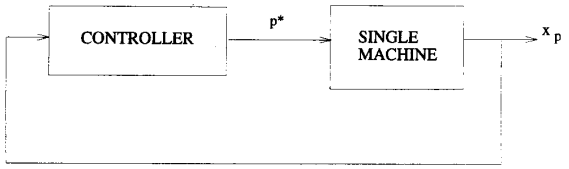


Fig. 3. Single machine with its controller (scheduler).

controller is $p*$, which represents the next part-type to process. To minimize the idle time due to setups, the machine will clear a buffer before it starts to process parts from another buffer. There are three clearing policies proposed in [1]: clear largest buffer (CLB), clear a fraction (CAF), and an unnamed policy in Section IV of [1] which we will refer as "CPK," to name it after the authors, Perkins and Kumar.

Let $x_p(T_n)$ represent the buffer level of b_p at T_n, the time at which the scheduler selects the next buffer of part-type $p*$ to clear. Let γ_p be any positive weighting factors (throughout this paper, we set the γ_p to one so that the AWBL, to be defined below, is "average work"). Each of the three clearing policies is briefly described as follows:

1) *CLB:* Select $p*$ such that $x_{p*}(T_n) \geq x_p(T_n)$ for all p.
2) *CAF:* Select $p*$ such that

$$x_{p*}(T_n) \geq \epsilon \sum_{p=1}^{P} x_p(T_n)$$

where ϵ is a small number, often set to 1/P.

3) *CPK:* Select $p*$ such that $p* = \text{argmax}_p\{\frac{x_p(T_n)+\delta d_p}{d_p\sqrt{\gamma_p\rho_p^{-1}(1-\rho_p)}}\}$, where $\rho_p = d_p\tau_p$.

In addition to these clearing policies, there exist many other policies that are used in FMS such as: first come, first serve (FCFS), first buffer, first serve (FBFS), last buffer, first serve (LBFS), least slack policy (LS), and earliest due date policy (EDD) [3].

A single machine is "stable" if the buffer level for each part-type is bounded, i.e., if there exists $m_p > 0$, $p = 1, 2, \cdots, P$, such that

$$\sup_t x_p(t) \leq m_p < +\infty \quad for\ p = 1, 2, \cdots, P.$$

A necessary condition for stability is that the machine load $\rho = \sum_{p=1}^{P} \rho_p < 1$ where $\rho_p = d_p\tau_p$. For the single machine case, the authors in [1] prove that all three policies described above cause the machine to be stable.

There are various ways to measure the performance of a scheduling policy. We can measure the average delays incurred when a part is processed in the machine. We can also measure the maximum value of each buffer level. The performance criterion proposed in [1] is a quantity called the average weighted buffer level (AWBL) defined as

$$AWBL = \liminf_{t\to\infty} \frac{1}{t}\int_0^t \left[\sum_p \gamma_p\tau_p x_p(s)\right] ds.$$

For any stable scheduling policy, the average weighted buffer level has a lower bound (LB) determined in [1] of

$$LB = \frac{\delta\left[\sum_p \sqrt{\gamma_p\rho_p\ (1-\rho_p)}\right]^2}{2(1-\rho)}.$$

Let $\eta = \frac{AWBL}{LB}$ be a measure of how close a scheduling policy is to optimal. An optimal scheduling policy has η equal to one. Any scheduling policy has $\eta \geq 1$. To compute the value of AWBL, we will of course have to choose some finite value of t to terminate our simulations. In [1], the authors compare the performance of CLB, CAF, and CPK for various machines and find that quite often the use of CPK results in values of η that are closest to one. It is for this reason that we will pay particular attention to comparing the performance of our intelligent controllers to CPK. Finally, note that if "idling" is allowed, the authors in [4] have shown that the bound LB can be improved. While we do not consider idling here, it does have the potential to improve performance.

B. Universally Stabilizing Supervisory Mechanism

The universally stabilizing supervisory mechanism (USSM), introduced in [2], is a mechanism that is used to govern any scheduling policy. There are two sets of parameters employed by the mechanism for the single machine, namely $\gamma$and z_p where it must be the case that

$$\gamma > \frac{\sum_b \max_{b'} \delta_{b',b}}{1-\rho}$$

and $z_p$can be chosen arbitrarily. The single machine will process parts of type p for exactly $\gamma d_p\tau_p$ units of time unless it is cleared first (if a part is currently being processed when this amount of time is up, the processing on this part is finished). Once the machine takes $\gamma d_p\tau_p$ units of time to process parts of type p or the parts of type p are cleared before $\gamma d_p\tau_p$ elapses, the machine will schedule another part to be processed next. In addition, the USSM has a first-in–first-out queue Q. When a buffer level x_p exceeds z_p and the buffer is not being processed or setup, that buffer will be placed into Q. When there is some buffer in the queue overruling the scheduling policy, the next buffer scheduled to be processed is the first buffer in the queue. Once that first buffer is processed, it leaves the queue, then any remaining buffers on the queue are processed. Hence, the USSM stabilizes any scheduling policy by truncating long production runs and by giving priority to buffers that become excessively high. Note that x_p is not exactly bounded by z_p since x_p can still increase while it is listed in the queue. On the other hand, x_p is affected by z_p. The larger z_p is, the larger the maximum of x_p tends to be. Also note that if the system is already stable (i.e., without the USSM) and the values of γ and z_p are large enough, the mechanism will not be invoked [2]. We will have occasion to use the USSM to ensure that our intelligent controllers are stable for both the single and multiple machine cases.

C. A Fuzzy Controller for a Single Machine

In this section, we propose a new scheduling policy which is designed to emulate a human scheduler. In particular,

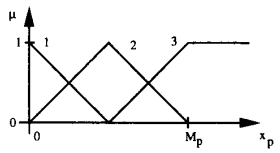


Fig. 4. Fuzzy sets for x_p.

we will show how to perform scheduling via a class of intelligent controllers called "fuzzy controllers" (FC); for an introduction to fuzzy control see [20], [21]. The FC is designed to be a clearing policy just as are CLB, CAF, and CPK. As there is no guarantee of stability when operating by itself, the FC is always augmented with the USSM in this paper.

Similar to the conventional scheduling policies CLB, CAF, and CPK, the inputs to the FC policy are the buffer levels x_p. The output of the FC is simply an index $p*$ indicating which one of the buffers will be processed next. The universe of discourse for each x_p is $[0, \infty)$. The universe of discourse of each x_p has several fuzzy sets. The membership functions for each fuzzy set are triangular except at the extreme right as shown in Fig. 4. Fig. 4 shows the membership functions μ for the case where the universe of discourse for x_p has three fuzzy sets. These fuzzy sets, indexed as 1, 2, and 3, indicate how "small," "medium," and "large," respectively, the value of x_p is. If the buffer level x_p exceeds M_p, the value of x_p is assumed to be M_p by the FC, where M_p must be predetermined. We will call this parameter M_p the saturation value of the FC for x_p and will spend a significant amount of time discussing the choice of M_p in the remaining sections of this paper.

Table I shows a "complete" rule-base of a FC for a single machine that has three part-types using the fuzzy sets shown in Fig. 4. In each rule I_{x_p} represents the index of the fuzzy set, and J represents the part-type that is selected by the rule. Then, for instance, rule number 2 takes on the form:

$$\textbf{If } x_1 \text{ is small } \textbf{and } x_2 \text{ is small } \textbf{and } x_3 \text{ is medium}$$
$$\textbf{Then } p* = 3.$$

In other words, if the buffer levels of b_1 and b_2 are small and the buffer level of b_3 is medium, then process part-type 3. The part of type J that is selected in each rule has buffer level x_J that falls into a fuzzy set that has index I_{x_J} the largest compared to the other indexes. In some rules, there are indexes of fuzzy sets of several part-types that have equal largest value. In these cases, one of these part-types is selected arbitrarily in our rule-base. For example, the first rule in Table I is fixed to select part-type 1 even though the fuzzy set indexes of all part-types in the rule are equal to one. Therefore, this rule is biased toward part-type 1. Throughout this paper, if we use more fuzzy sets on the universe of discourse we will utilize a similar structure for the rule-base (i.e., uniformly distributed and symmetric membership functions). The output universe of discourse (the positive integers) has P membership functions denoted by μ_p where for each $p \in \{1, 2, \cdots, P\}$, $\mu_p(i) = 1$ for $i = p$, and $\mu_p(i) = 0$ for $i \neq p$. We use singleton fuzzification, max-defuzzification [20], [21] and the standard

TABLE I
COMPLETE RULE BASE OF A FC WITH THREE INPUTS AND THREE FUZZY SETS ON EACH UNIVERSE OF DISCOURSE

Rule No.	I_{x_1}	I_{x_2}	I_{x_3}	J
1	1	1	1	1
2	1	1	2	3
3	1	1	3	3
4	1	2	1	2
5	1	2	2	2
6	1	2	3	3
7	1	3	1	2
8	1	3	2	2
9	1	3	3	2
10	2	1	1	1
11	2	1	2	1
12	2	1	3	3
13	2	2	1	1
14	2	2	2	1
15	2	2	3	3
16	2	3	1	2
17	2	3	2	2
18	2	3	3	3
19	3	1	1	1
20	3	1	2	1
21	3	1	3	3
22	3	2	1	1
23	3	2	2	1
24	3	2	3	1
25	3	3	1	1
26	3	3	2	1
27	3	3	3	3

Zadeh's compositional rule of inference [20], [21] to pick $p*$, given the rule-base and particular values of x_p.

For P buffers and m fuzzy sets, the size of memory needed to store the rules is on the order of P^m; hence, the CLB, CPK, and CAF policies are simpler than the FC. We will show, however, that with the use of this slightly more complex scheduler we can get enhanced performance. It is possible to expand the FC to use the information about arrival rates, processing times, and the set-up times also. There may be significant improvements in performance if this information is represented with the control rules, however, the memory size can significantly increase too. In the interest of ensuring that the FC would be implementable in real-time we did not pursue this line of research. We note that our FC essentially "fuzzifies" the operation of the CLB policy in [1]; however due to the interpolation inherent in the implementation of the FC it will behave quite differently than the conventional CLB (as the simulation results below indicate).

Next, we simulate a single machine that uses CLB, CAF, CPK, and the FC so that we can compare their performance. Figs. 5 and 6 show the plots of the buffer levels of a single machine with three part-types for the first 10 production runs (a production run is defined as setting up for and processing all the parts in a buffer) and the last 30 production runs. The machine parameters are: $d_1 = 7$, $d_2 = 9$, $d_3 = 3$, $\tau_1 = 1/100$, $\tau_2 = 1/51$, $\tau_3 = 1/27$, and $\delta = 1$. The parameters M_1=35, M_2=35, and M_3=12 are selected based upon the maximum value x_p obtains when the CPK policy is used. Note that the first 10 production runs of the FC are very different from CPK. For large values of t, however, they are quite similar but not exactly the same, as indicated by

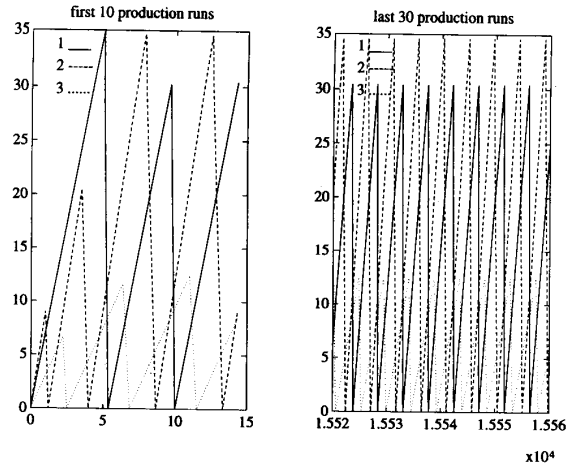


Fig. 5. Buffer levels using the CPK scheduling policy.

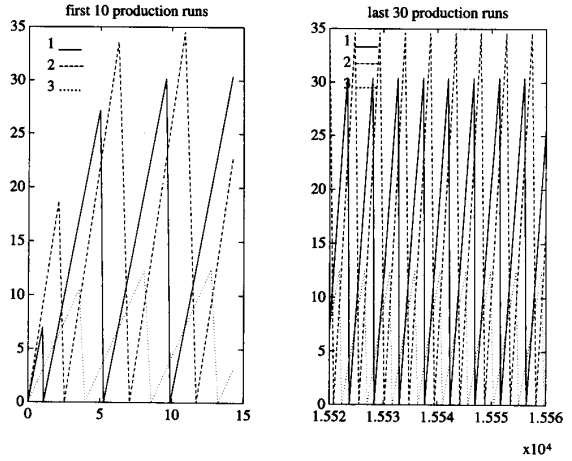


Fig. 6. Buffer levels using the FC scheduling policy.

the last 30 production runs when CPK and the FC are used. Even though the buffer levels are maintained at nearly the same heights, the periodic sequence of scheduling the part-types by CPK is $1, 3, 2, 1, 3, 2, \cdots$, whereas the sequence by FC is $1, 2, 3, 1, 2, 3, \cdots$.

Among the three schedulers, namely CLB, CAF, and CPK, CPK often yields the best performance, i.e., its η is closest to one [1]. The performance of the FC is compared to CLB, CAF, and CPK for several single machines below. The number of fuzzy sets is set to 3, 5, and 7 for each universe of discourse x_p so as to observe how the number of fuzzy sets can affect the performance of the FC. The first two machines are chosen from Section IV of [1].

Machine 1: d_1=7, d_2=9, d_3=3, τ_1=1/100, τ_2=1/51, τ_3=1/27, ρ=0.35758.

- CLB: η = 1.0863484
- CAF: η = 1.2711257
- CPK: η = 1.0262847
- FC: M_1=35, M_2=35, M_3=12; γ=34.0, z_1=30, z_2=30, z_3=30
  For three fuzzy subsets, η = 1.0263256
  For five fuzzy subsets, η = 1.0262928
  For seven fuzzy subsets, η = 1.0262928

These simulations show that FC can perform nearly as well as CPK can. Note also that we cannot significantly improve η by simply increasing the number of fuzzy subsets for the same M_p (for this machine).

Machine 2: d_1=18, d_2=3, d_3=1, τ_1=1/35, τ_2=1/7, τ_3=1/20, ρ=0.99286.

- CLB: η = 1.1738507
- CAF: η = 1.179065
- CPK: η = 1.0017406
- FC: M_1=3375, M_2=626, M_3=665; γ=1000.0, z_1=5000, z_2=5000, z_3=5000. For three fuzzy subsets, η = 1.0027945
  For five fuzzy subsets, η = 1.0027945
  For seven fuzzy subsets, η = 1.0013173

These simulations show that with the machine load closer to one, the FC can even work better than CPK provided that there are enough fuzzy sets on the input space. Next, we create a new machine that has a lower machine load, and compare the performance of the scheduling policies.

Machine 3: d_1=3.5, d_2=4.5, d_3=1.5, τ_1=1/100, τ_2=1/51, τ_3=1/27, ρ=0.17879.

- CLB: η = 1.0841100
- CAF: η = 1.3456014
- CPK: η = 1.0306833
- FC: M_1=23.6, M_2=25.1, M_3=5.6; γ=100.0, z_1=5000, z_2=5000, z_3=5000.
  For three fuzzy subsets, η = 1.0307992
  For five fuzzy subsets, η = 1.0319630
  For seven fuzzy subsets, η = 1.0306972
- Scheduler: FC with 3 fuzzy subsets; M_1=50, M_2=50, M_3=20; γ=100.0, z_1=5000, z_2=5000, z_3=5000: η = 1.2273009

These simulations show that the FC cannot perform any better than CPK when the machine load is small for this machine. Also note that if the parameters M_p are not set properly, the performance of the FC can degrade. This provides a motivation for studying how to tune M_p of the FC to optimize performance (we examine this in the next section). Next, we create another new machine that works on more part-types.

Machine 4: d_1=1, d_2=1, d_3=1/0.9,d_4=1, d_5=1, τ_1=0.15, τ_2=0.2, τ_3=0.05, τ_4=0.1, τ_5=0.2, ρ=0.7055556.

- CLB: η = 1.0371536
- CAF: η = 1.2328696
- CPK: η = 1.0180188
- FC: M_1=14.16, M_2=13.81, M_3=28.06, M_4=14.98, M_5=13.63; γ=100.0, z_1=30, z_2=30, z_3=30, z_4=30, z_5=30.
  For three fuzzy subsets, η = 1.0157503
  For five fuzzy subsets, η = 1.0157503
  For seven fuzzy subsets, η = 1.0186198

These simulations show that FC can perform better than CPK. Note, however, that with seven fuzzy subsets, the FC performance degrades to be similar to CPK. This confirms the results of simulating machine 1 that we cannot be guaranteed that performance will improve by simply increasing the num-

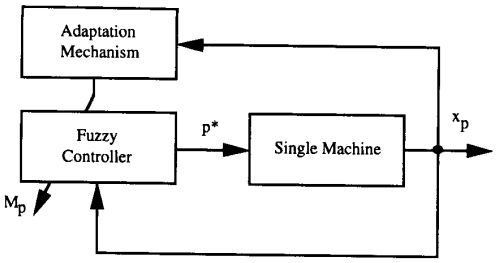


Fig. 7. Adaptive fuzzy controller (AFC) for a single machine.

ber of fuzzy subsets (this is not surprising as there are no performance guarantees for the heuristically constructed FC policy).

Our experience in simulation has shown that it is possible to tune the FC by choosing the values of M_p and the fuzzy sets to minimize η. We have used the following procedure to tune the FC to get smaller η: i) use i fuzzy sets and set the M_p all to unity, ii) run a simulation, iii) replace M_p with the maximum buffer levels obtained in x_p and re-run the simulation, and iv) repeat as necessary with $i+1$ fuzzy sets, $i+2$ fuzzy sets, etc. Using this tuning approach for Machines 1, 2, 3, and 4 of previous section we find that for three-buffer machines the results are as good as those of CPK and for the five-buffer machine the tuning method converges to a good result, even though the result is not quite as good as that of CPK. In the interest of space we omit the results of our detailed simulations but note that our experiences in tuning allowed us to develop the on-line tuning technique studied in the next section. It is interesting to note that other researchers have utilized iterative techniques for tuning FMS scheduling algorithms [3]. Here, our iterative tuning technique is for an entirely different class of scheduling policies than studied there.

III. An Adaptive Fuzzy Controller for a Single Machine

In this section we develop an adaptive fuzzy controller (AFC) for scheduling the single machine. In the AFC there is an adaptation mechanism which can automatically synthesize a FC policy, independent of the machine parameters. Moreover, if there are machine parameter changes during operation that still satisfy the necessary conditions for stability, the AFC will tune the parameters of the FC so that high performance operation is maintained. The USSM governs the AFC. Therefore, the complete scheduler consists of three layers. The bottom layer is simply the FC itself, the middle layer is the adaptation mechanism to be introduced here, and the top layer is the USSM which supervises the lower two layers to ensure stable operation.

If the parameters of the machine change, the USSM may not guarantee stability anymore since it assumes that the machine parameters stay constant. The parameter γ of the mechanism is dependent on the parameters of the machine whereas the parameters z_i are not. If the parameter γ is chosen large enough, the USSM may still provide stability over a large class of machine parameters. Since the USSM assumes constant machine parameters, however, stability is not guaranteed when the machine parameters change even if γ is large enough for the new machine parameters. It is for this reason that we split the adaptation problem into controller synthesis (i.e., determining the positioning of a fixed number of fuzzy sets by automatically picking M_p) and controller tuning (i.e., tuning the positioning of the fuzzy sets by changing M_p to react to machine parameter changes). In synthesis we are guaranteed stability while in tuning we have no proof that the policy is stable.

A. Automatic Controller Synthesis for a Single Machine

In this section, we introduce the adaptive fuzzy controller that has an adaptive mechanism which observes x_p, $p \in \{1, 2, 3, \cdots, P\}$ and automatically tunes the values of M_p. This adaptation mechanism shown in Fig. 7 adjusts the parameters M_p of the FC by using a moving window. The size of the window is not fixed, but is equal to the length of time for a fixed number of production runs. In this section we will use a window size of 10 production runs, while in the next section we will use a larger window size. Throughout this window the buffer levels x_p are recorded. The window slides forward at the end of each production run, and the values of M_p are updated to the maximum values of x_p over the last window frame. As M_p is updated, the fuzzy sets on the universe of discourse for x_p are shifted so that they remain symmetric and uniformly distributed [20], [21]. The fuzzification and defuzzification strategies, output fuzzy sets, and the rule-base remain constant so that the adaptation mechanism only adjusts the input fuzzy sets to improve machine performance. Basically, the AFC tunes the M_p values in search of a lower η. It does this by automatically adjusting the premise membership functions of the rules in Table I so that the FC policy appropriately fits the machine (i.e., so that the FC pays particular attention to part-types whose peak buffer levels have been rising).

Next, we show how the automatic tuning method can be used to synthesize the FC for the single machine. In particular, we will show how without any knowledge of the machine parameters our adaptation mechanism can synthesize a FC that can perform as well as the CPK policy. We shall first consider the same machines used in Section II-C. For each of the following machines, the number of fuzzy sets is set to five. The parameter M_p is initially set to one. The adaptive mechanism will adjust M_p at the end of each production run. The adjustment of M_p is based upon maximum value of the buffer levels x_p from the last 10 production runs.

Table II shows how each M_p is adjusted for the first 15 production runs in Machine 1. For later production runs, M_1 changes up and down slightly around 30.4, M_2 around 34.6, and M_3 around 12.45. After 10 000 production runs, we find that η=1.0263 which is the same as η=1.0263 produced by CPK after 10 000 production runs.

Table III shows that M_p for Machine 2 converges slowly compared to the previous machine. After 10 000 production runs, η=1.0993, which is worse than η=1.0017 produced by CPK after 10 000 production runs. When M_p is initially 10 000 instead of one and the adaptation mechanism updates the M_p every other 10 production runs, however, η from FC improves to 1.0018. This highlights an inherent problem

TABLE II
ADJUSTMENT OF M_p FOR MACHINE 1

Production Run No.	M_1	M_2	M_3
1	1.0000	1.0000	1.0000
2	7.0000	9.6774	3.2258
3	20.1929	18.6774	7.5599
4	27.1929	18.6774	10.5599
5	27.1929	24.5915	10.5599
6	27.1929	33.5915	12.2766
7	30.1793	33.5915	12.2766
8	30.1793	33.5915	12.2766
9	30.3823	33.5915	12.4396
10	30.3823	34.5243	12.4396
11	30.3823	34.5243	12.4396
12	30.3993	34.5243	12.4519
13	30.3993	34.6050	12.4519
14	30.3993	34.6050	12.4519
15	30.3993	34.6050	12.4519

TABLE III
ADJUSTMENT OF M_p FOR MACHINE 2

Production Run No.	M_1	M_2	M_3
1	1.000	1.000	1.000
10	284.8656	60.3010	25.9070
50	1266.295	253.6234	139.8938
100	2134.125	418.863	229.6020
200	3026.803	594.9258	328.7803
300	3401.508	667.5586	369.9785
400	3558.227	698.0391	386.8477
500	3626.016	710.8359	393.9297
600	3653.156	716.3438	396.9219
700	3664.406	718.5234	398.1875
800	3665.672	718.7578	398.3125
900	3665.672	718.7578	398.3125
1000	3665.672	718.7578	398.3125

TABLE IV
ADJUSTMENT OF M_p FOR MACHINE 3

Production Run No.	M_1	M_2	M_3
1	1.0000	1.0000	1.0000
2	3.5000	4.6632	1.5544
3	4.1897	9.1632	3.3499
4	8.3554	9.1632	4.8499
5	11.8554	10.4087	4.8499
6	11.8554	14.9087	4.8499
7	11.8554	14.9087	5.1652
8	12.3311	14.9087	5.1652
9	12.3311	14.9865	5.1652
10	12.3311	14.9865	5.1751
11	12.3383	14.9865	5.1751
12	12.3383	14.9865	5.1751
13	12.3383	14.9886	5.1753
14	12.3383	14.9886	5.1753

with the adaptation mechanism: the window size and M_p update strategy must be chosen in an ad hoc manner with no guarantees on performance levels (the results of this paper do, however, provide significant insights into the choice of the window size).

Table IV shows how each M_p is adjusted for the first 14 production runs of Machine 3. For later production runs, M_1 remains around 12.338, M_2 around 14.9886, and M_3 around 5.1753. After 10 000 production runs, we find that η=1.031 which is the same as the η produced by CPK after 10 000 production runs.

B. Automatic Controller Tuning for a Single Machine

In this section we investigate whether the AFC and CPK can adjust themselves to disturbances/failures that may occur during the operation of a single machine. The disturbance/failure may be in the form of changes in arrival rates, processing times, and/or set-up times. To observe how FC and CPK adjust to machine parameter changes, first we use the same machine parameters and part-types and switch the part-types to arrive at different buffers. Following this, we will investigate the tuning capabilities of the adaptation mechanism by examining the effects of changing the machine load. In the simulations, the machine parameters stay constant for the first 10 000 production runs, and then the machine parameters are changed and remain constant at different values for the next 10 000 production runs. When the parameters are changed, the parameters M_p of the FC are continued from the last production run. For the last 10 000 production runs, the CPK schedules based upon the former machine parameters while the AFC adjusts itself to improve performance.

1) Switching Buffers: Case 1 Old Machine: d_1=7, d_2=9, d_3=3, τ_1=1/100, τ_2=1/51, τ_3=1/27. *New Machine:* d_2=7, d_3=9, d_1=3, τ_2=1/100, τ_3=1/51, τ_1=1/27.

The AFC maintains the same η at 1.026 whereas η of CPK becomes worse from 1.027 to 1.237.

Case 2 Old Machine: d_1=18, d_2=3, d_3=1, τ_1=1/35, τ_2=1/7, τ_3=1/20. *New Machine:* d_2=18, d_3=3, d_1=1, τ_1=1/35, τ_2=1/7, τ_3=1/20.

The value of η of the AFC improves from 1.0993 to 1.0018 whereas η of CPK becomes worse from 1.0017 to 1.1965. The AFC is expected to perform similarly since the parameters of the machines are similar.[1]

2) Machine Load Variations: Case 3 Old Machine (ρ = 0.99286)*:* d_1=18, d_2=3, d_3=1, τ_1=1/35, τ_2=1/7, τ_3=1/20. *New Machine* (ρ = 0.35758)*:* d_1=7, d_2=9, d_3=3, τ_1=1/100, τ_2=1/51, τ_3=1/27.

This is a transition from a high to a low machine load. η of the AFC changes from 1.0993 to 1.0263 as expected. On the other hand, η of CPK changes from 1.0017 to 1.0477 instead of 1.0263. Note that CPK can still perform reasonably well as the machine parameters change from highly loaded to a lightly loaded machine.

Case 4 Old Machine (ρ = 0.35758)*:* d_1=7, d_2=9, d_3=3, τ_1=1/100, τ_2=1/51, τ_3=1/27. *New Machine* (ρ = 0.99286)*:* d_1=18, d_2=3, d_3=1, τ_1=1/35, τ_2=1/7, τ_3=1/20.

This is a transition from a low to a high machine load. η of the AFC changes from 1.0263 to 1.0993 as expected. On the other hand, η of CPK changes from 1.0263 to 1.106 instead of 1.0017, i.e., its performance degrades.

The results show that the AFC we have developed has the capability to maintain good performance even if there were significant changes in the underlying machine parameters (representing, e.g., machine failures). CPK is dependent on the exact specification of the machine parameters and hence its performance can degrade if the parameters change. We found similar improvements in performance as compared to CLB and CAF but do not report these here in the interest of space. While these results are encouraging we have only considered

[1] Recall, however, that in Section II-C we describe that there are some rules in the rule-base of the FC that are biased toward some part-types. Therefore, when we switch the order of indexing the part-types, the performance of the FC can be different.

the single machine case up till now (and only for a limited number of types of machines in simulation). In the next section we will show how adaptive fuzzy controllers placed at each machine in an FMS can adapt to changes in many machine parameters and achieve high performance operation.

IV. Distributed Fuzzy Control for FMS

In this section, we will extend the use of the adaptive fuzzy controller to general FMS consisting of several distributed interconnected machines by applying an adaptive fuzzy controller to each machine to obtain a distributed (adaptive) fuzzy controller. We will investigate whether each adaptive fuzzy controller can achieve its local goal which is to minimize the backlog of each machine and hence achieve the overall goal of minimizing the backlog for the entire FMS.

A. Nonacyclic FMS

A nonacyclic manufacturing system, which consists of several interconnected machines, is described as follows [1], [2]: There are P part-types, with each type labeled 1, 2, $\cdots, P$. There are M machines, with each machine labeled 1, 2,$\cdots$, M. A part of type p enters the system at machine $\mu_{p,1}$. Thereafter, it visits $\mu_{p,2}$, $\mu_{p,3}$, $\cdots$ and exits the system from machine μ_{p,n_p} where n_p is the total number of times part p visits the machines in the system and $\mu_{p,i} \in \{1, 2, \cdots, M\}$. A part p can visit a machine more than once, i.e., it can be that $\mu_{p,i} = \mu_{p,j}$ for $i \neq j$. Part-type p arrives at the system at the rate d_p parts/time-unit. Parts that arrive at $\mu_{p,i}$ are not necessarily processed immediately; they may be stored at buffer $b_{p,i}$ for some length of time. Once they are scheduled for processing, each part-type p requires a processing time $\tau_{p,i}$ at the ith machine in its processing path $\mu_{p,i}$. 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 buffers $b' \in B_m$ such that $b' \neq b$, it requires a set-up time $\delta_{b,b'}$. There is a transportation delay when a part is moved from one machine to another. For convenience we assume that the delay is zero in this paper. Fig. 1 shows an example of a nonacyclic manufacturing system. There are three different part-types and six machines. For part-type 1, the sequence of machines it visits is $\mu_{1,1} = 1$, $\mu_{1,2} = 2$, $\mu_{1,3} = 4$, $\mu_{1,4} = 3$, and $\mu_{1,5} = 4$. Note that part-type 1 visits machine 4 twice.

A necessary condition for stability is that the machine load of each machine (ρ_m) for $m = 1, \cdots, M$

$$\rho_m = \sum_{\{(p,i):\mu_{p,i}=m\}} d_p \tau_{p,i} < 1.$$

The three clearing policies CLB, CAF, and CPK can cause a general nonacyclic FMS to be unstable. For general cases, a CAF policy can still be employed to make the system stable, but the policy must be modified. Such a modified CAF policy is introduced in [1] and is called distributed CAF policies with backoff (DCAF).

We will use the maximum buffer level denoted by $\Theta_{p,i}$ and average weighted buffer level denoted by θ_m for local performance measures to indicate how well each scheduler performs locally on each machine. The θ_m is defined as

$$\theta_m = \liminf_{t\to\infty} \frac{1}{t} \int_0^t \left[\sum_{(p,i):\mu_{(p,i)}=m} \tau_{p,i} x_{p,i}(s) \right] ds.$$

We will use $\text{avg}\{\theta_m\} = \frac{\sum_{m=1}^{M} \theta_m}{M}$, the average of the average weighted buffer levels and $\max_{p,i}\{\Theta_{p,i}\}$, the maximum of the maximum buffer levels as global performance measures for the entire FMS. Let $O_p(t)$ denote the total number of parts of type p leaving the FMS up to time t. Let $I_p(t)$ denote the total number of new parts of type p entering the FMS up to time t. Let $\beta_p(t)$ denote the backlog of part-type p at time t, then $\beta_p(t)$ is simply $\beta_p(t) = I_p(t) - O_p(t)$. Let $\psi_p(t)$ denote the average backlog of part-type p at time t, then $\psi_p(t)$ is defined as

$$\psi_p(t) = \frac{1}{t} \int_0^t \beta_p(\tau) d\tau.$$

Let $\text{avg}\{\psi_p t\}$ denote the average of average backlogs, i.e., $\text{avg}\{\psi_p(t)\} = \frac{\sum_{p=1}^{P} \psi_p(t)}{P}$. Let $\Psi_p(t) = \sup_t \beta_p(t)$ denote the maximum backlog of each part-type p up to time t and $\max_p\{\Psi_p\}$ denote the maximum of the maximum backlogs. We will use $\psi_p(t)$, $avg\{\psi_p(t)\}$, and $\Psi_p(t)$ as global performance measures for FMS.

B. Automatic Synthesis/Tuning of Distributed Fuzzy Controllers

In this section, we develop a technique to automatically synthesize distributed fuzzy controllers for a nonacyclic FMS. The DFC consists of a set of AFC policies each assigned to control a machine in the FMS. The number of inputs to each AFC depends on the number of buffers on the machine at which the AFC is scheduling. Each AFC has the same rule-base structure and are all initialized with the same membership functions. The number of fuzzy sets is set to five (unless indicated otherwise) for each universe of discourse for $x_{p,i}$. The parameters of each FC will change as each AFC adjusts itself to control its machine. As in the single machine case, we utilize USSM's to ensure stability; therefore there will be as many AFC's and USSM's as the number of machines.

Let $M_{p,i}$ be the saturation value for $x_{p,i}$ as $M_p$was the saturation value for x_p in single machine case. The FC for each machine is automatically synthesized in a similar manner to how it was for the single machine described in Section III-A. The parameters $M_{p,i}$ of each FC are initially set to an arbitrary large number (we use 10 000). Each AFC also utilizes a moving window. The size of the window is the same as the length of time for 40 production runs (unless we indicate otherwise). The window slides forward at the end of each production run and the values of $M_{p,i}$ of each FC are updated to the maximum buffer level $x_{p,i}$ during the last window frame.

Again we note that we split "synthesis" from "tuning" to i) illustrate how the technique can be used to automatically generate controllers that can perform better than the scheduling policies in [1], and ii) to illustrate how the DFC can learn to accommodate for parameter changes/failures in the machines.

TABLE V
PERFORMANCE COMPARISON FOR FEEDFORWARD STRUCTURE

	Sum of All Maximum Backlog $\sum \Psi_p$	Maximum of All Maximum Backlog $\max_p\{\Psi_p\}$	Average of All Average Backlog $avg\{\psi_p\}$	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	1580	361	122.2	1957	326	6.7
CAF	2298	494	132.7	2910	431	7.2
CPK	1631	468	132.4	1933	425	6.8
DFC	1589	359	121.2	1868	323	6.6
DFC*	1496	338	134.7	1868	323	6.5
DFC11	1558	338	121.6	1879	326	6.5
DCAF	2559	676	192.3	3196	675	9.0

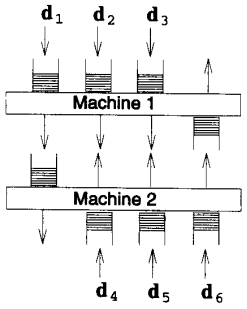


Fig. 8. Feedforward structure.

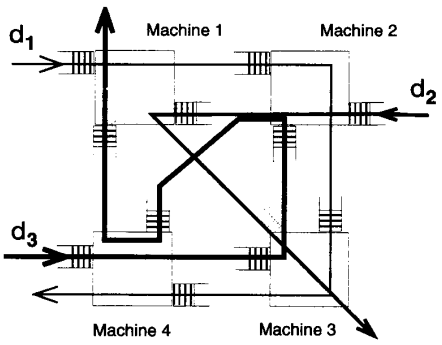


Fig. 9. Cellular structure I.

Moreover, we split the two cases since for case i) we are guaranteed stability while in case ii) we are not.

1) Automatic Synthesis of Distributed Fuzzy Controllers: In this section, we show the results of performing automatic DFC synthesis for several FMS structures. We compare the performance of the DFC with other scheduling policies such as CLB, CAF, CPK and distributed CAF with backoff. In each structure of FMS that we consider, the parameters of the USSM are set large enough so that the USSM never intervenes (we have found that if the parameters of the USSM are chosen too small the USSM can adversely affect performance).

a) Feedforward Structure: For the FMS shown in Fig. 8, the arrival rates are d_1=7, d_2=9, d_3=3, d_4=8, d_5=6, and d_6=5. The processing times for part-type 1 are $\tau_{1,1}$=1/100, and $\tau_{1,2}$=1/100. The processing time for part-type 2 is $\tau_{2,1}$=1/51. The processing time for part-type 3 is $\tau_{3,1}$=1/27. The processing time for part-type 4 is $\tau_{4,1}$=1/60. The processing time for part-type 5 is $\tau_{5,1}$=1/40. The processing times for part-type 6 are $\tau_{6,1}$=1/45, and $\tau_{6,2}$=1/100. The set-up times are all δ=5.0.

Each machine in this FMS is similar to a single machine with the exception that the arrival rate of parts in one of the buffers in each machine is not fixed since these parts are from another machine. The primary objective in studying this structure is to investigate how the scheduling in each machine is affected by these non-fixed arrival rate parts. Table V shows the comparison of the performance measures of each scheduling policy. For the DFC policy, we update the parameters $M_{p,i}$ based upon the buffer level $x_{p,i}$ over the last 100 production runs. The DFC* denotes a similar DFC except that $M_{p,i}$ is fixed with the values from the end of the simulation for the DFC. The DFC11 denotes a similar DFC as the first which has the $M_{p,i}$ arbitrarily initialized to 10 000 as in the first DFC, but the number of fuzzy sets of DFC11 is increased from five to 11. The table indicates that: i) these nonfixed arrival rate parts do affect the scheduling of each machine, ii) the CPK policy that performs the best in the single machine case is no longer the best scheduler, iii) the DFC dominates all conventional schedulers in almost all performance criteria, except that the sum of all maximum backlogs of DFC (1589) is slightly larger than that of CLB (1580), iv) the sum of all maximum backlogs of the DFC is reduced when the initial values of $M_{p,i}$ are properly set (i.e., in DFC*), but in this case the average of all average backlog increases from 121.2 to 134.7, and v) increasing the number of fuzzy sets can also improve performance. Overall, we see that DFC* dominates all the conventional schedulers for all performance criteria except for the average of the average backlogs.

b) Cellular Structure I: For this FMS shown in Fig. 9, the arrival rates are d_1=0.5, d_2=2, and d_3=1. The processing times for part-type 1 are $\tau_{1,1}$=0.4, $\tau_{1,2}$=0.2, $\tau_{1,3}$=0.4, and $\tau_{1,4}$ = 0.5. The processing times for part-type 2 are $\tau_{2,1}$=0.15, $\tau_{2,2}$=0.05, and $\tau_{2,3}$ = 0.05. The processing times for part-type 3 are $\tau_{3,1}$=0.1, $\tau_{3,2}$=0.05, $\tau_{3,3}$=0.2, $\tau_{3,4}$=0.1, and $\tau_{3,5}$ = 0.05. The set-up times are all δ=10.

Table VI shows the comparison of the scheduling policies CLB, CAF, CPK, DFC, and DCAF. DFC40 represents DFC schedulers that adapt their $M_{p,i}$ parameters based upon the buffer levels $x_{p,i}$ over the last 40 production runs. DFC10 represents the same, but only over 10 production runs. Note that $\sum \Psi_p$ of DFC40 is the smallest and $\sum \Psi_p$ of DCAF is

TABLE VI
PERFORMANCE COMPARISON FOR CELLULAR STRUCTURE I

	Sum of All Maximum Backlog $\sum \Psi_p$	Maximum of All Maximum Backlog $\max_p\{\Psi_p\}$	Average of All Average Backlog $avg\{\psi_p\}$	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	530	242	86.7	1268	174	10.0
CAF	585	245	92.2	1525	208	10.2
CPK	548	275	90.8	1243	198	9.7
DFC10	495	226	99.0	1309	201	10.8
DFC40	411	200	79.8	984	161	8.9
DCAF	831	346	233.6	1462	213	24.0

TABLE VII
PERFORMANCE COMPARISON FOR CELLULAR STRUCTURE II

	Sum of All Maximum Backlog $\sum \Psi_p$	Maximum of All Maximum Backlog $\max_p\{\Psi_p\}$	Average of All Average Backlog $avg\{\psi_p\}$	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	144	42	18.4	374	31	3.2
CAF	249	76	23.2	613	63	4.0
CPK	159	51	22.7	368	44	3.8
DFC	149	50	18.1	346	39	3.3
DFC*	125	36	18.8	315	29	3.3
DCAF	433	132	85.4	757	84	12.0

the largest. This table indicates that: i) for this FMS that is inherently stable, DCAF is not desired since the policy wastes too much idle time, ii) the size of the window that is used to adapt $M_{p,i}$ can affect the performance of the DFC (we have not found a method to select the optimal size for the window automatically, so far, we can only set the window size in an ad hoc manner), iii) the CPK again does not perform as well as expected from the single machine case, iv) among the conventional schedulers, the CLB is better than the CAF and CPK in almost all performance criteria, and v) the DFC40 totally dominates the conventional schedulers for this particular FMS.

c) Cellular Structure II: For this FMS shown in Fig. 10, the arrival rates are d_1=1.0, d_2=1.0, d_3=1.0, and d_4=1.0. The processing times for part-type 1 are $\tau_{1,1}$=0.2, $\tau_{1,2}$=0.1, $\tau_{1,3}$=0.1, and $\tau_{1,4}$=0.05. The processing times for part-type 2 are $\tau_{2,1}$=0.15, $\tau_{2,2}$=0.15, $\tau_{2,3}$=0.1, and $\tau_{2,4}$=0.1. The processing times for part-type 3 are $\tau_{3,1}$=0.25, $\tau_{3,2}$=0.2, and $\tau_{3,3}$=0.15. The processing times for part-type 4 are $\tau_{4,1}$=0.3, $\tau_{4,2}$=0.1, $\tau_{4,3}$=0.05, and $\tau_{4,4}$=0.05. The set-up times are all δ=1.0.

For the DFC policy, we update the parameters $M_{p,i}$ based upon the buffer level $x_{p,i}$ over the last 60 production runs. Table VII shows the comparison of the performance measures of each policy for Cellular Structure II. There are some performance measures of the DFC that are better than those of CLB, but there are also some that are worse. The DFC* (DFC with $M_{p,i}$ initialized in the same manner as for the feedforward structure) almost totally dominates CLB with the exception that the average of all average backlogs and the average of all average weighted buffer levels of CLB are slightly better. This indicates that DFC* attempts to maintain smaller values of the maximum of each buffer level by allowing slightly higher average buffer levels.

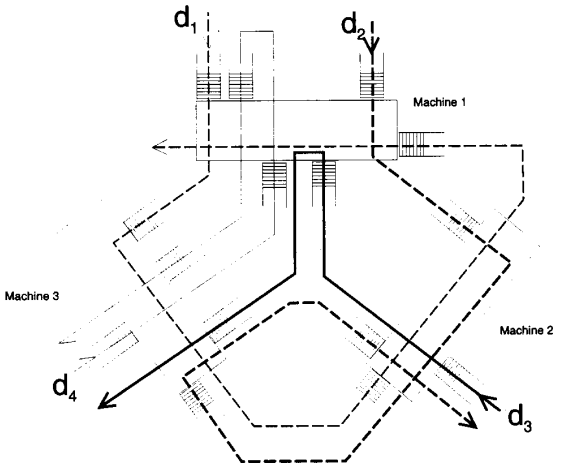


Fig. 10. Cellular structure II.

d) Reentrant Structure: For the FMS shown in Fig. 11, the arrival rate is d_1=4/3. The processing times are $\tau_{1,1}$=0.3, $\tau_{1,2}$=0.25, $\tau_{1,3}$=0.1, $\tau_{1,4}$=0.1, $\tau_{1,5}$=0.2, $\tau_{1,6}$=0.2, $\tau_{1,7}$=0.05, $\tau_{1,8}$=0.1, and $\tau_{1,9}$ = 0.05. The set-up times are all δ=50.0.

Table VIII shows the comparison of the performance measures of each policy for the Reentrant Structure. DFC40 represents DFC schedulers that adapt their $M_{p,i}$ parameters based upon the buffer levels $x_{p,i}$ over the last 40 production runs. DFC120 represents the same, but over 120 production runs. The table indicates several ideas that we have seen from the last two structures which are i) the window size for adapting the parameters of the DFC must be large enough, and ii) the DFC120 almost dominates all performance criteria.

TABLE VIII
PERFORMANCE COMPARISON FOR REENTRANT STRUCTURE

	Maximum Backlog Ψ_p	Average Backlog ψ_p	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	2107	1372.9	6009	1238	73.5
CAF	2089	1333.6	6126	1251	71.2
CPK	2247	1392.5	6511	1499	71.0
DFC40	2216	1412.1	6042	1338	73.7
DFC120	1929	1350.5	5328	921	72.3
DCAF	4273	3862.9	8424	1176	159.4

TABLE IX
PERFORMANCE COMPARISON FOR CELLULAR STRUCTURE III

	Sum of All Maximum Backlog $\sum \Psi_p$	Maximum of All Maximum Backlog $\max_p\{\Psi_p\}$	Average of All Average Backlog $avg\{\psi_p\}$	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	447	259	131.2	1394	160	10.5
CAF	445	265	134.0	1425	140	10.7
CPK	761	407	214.5	2475	275	16.1
DFC	403	244	135.1	1306	131	10.6
DCAF	721	406	335.3	1617	156	22.7

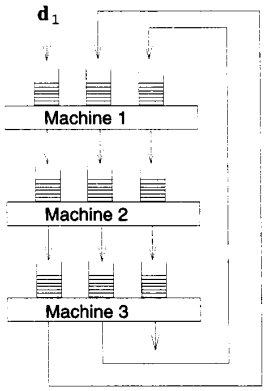


Fig. 11. Reentrant structure.

It has the smallest maximum backlog and maximum buffer levels, but the average backlog and the average of average weighted buffer levels are slightly worse than CAF and CPK respectively (again the DFC is lowering some performance measures at the expense of others).

e) Cellular Structure III: For this FMS shown in Fig. 12, the arrival rates are d_1=5/6 and d_2=4/3. The processing times for part-type 1 are $\tau_{1,1}$=0.4, $\tau_{1,2}$=0.4, $\tau_{1,3}$=0.2, $\tau_{1,4}$=0.2, $\tau_{1,5}$=0.1, $\tau_{1,6}$=0.4, $\tau_{1,7}$=0.3, $\tau_{1,8}$=0.15, and $\tau_{1,9}$=0.1. The processing times for part-type 2 are $\tau_{2,1}$=0.5, $\tau_{2,2}$=0.2, $\tau_{2,3}$=0.1, $\tau_{2,4}$=0.2, $\tau_{2,5}$=0.3, $\tau_{2,6}$=0.4, $\tau_{2,7}$=0.3, $\tau_{2,8}$=0.1, and $\tau_{2,9}$=0.1. The set-up times are all δ=2.0.

Table IX shows the comparison of the performance measures of each policy for cellular structure III. The buffer levels $x_{p,i}$ of the last 100 production runs are used to update the $M_{p,i}$ for the DFC. Again, the DFC dominates the maximum criteria, but the average criteria are slightly worse than those of CLB.

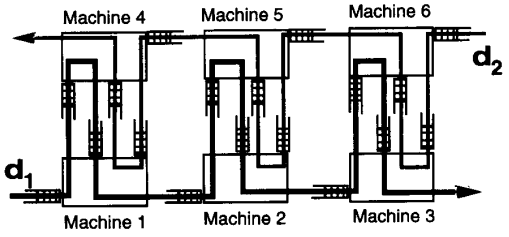


Fig. 12. Cellular structure III.

2) Automatic Tuning of Distributed Fuzzy Controllers: In this subsection, we show that the DFC can automatically retune if there is a change of machine parameters in the reentrant structure above (we have performed similar studies for the other FMS structures in the previous section but omit the details in the interest of space). We have two sets of machine parameters for the reentrant structure. To model the occurrence of machine parameter changes, in the simulation we initialize all the $M_{p,i}$ to 10 000 for one set of parameters. After 10 000 production runs from the first machine we change the machine parameters to another set and continue the simulation for another 10 000 production runs from the first machine. All the performance measures are initiated at the instant of the occurrence of the machine parameter changes. We repeat the same process for other scheduling policies such as CLB, CAF, and CPK. We do not include DCAF because the simulations above indicate that it performs relatively poorly even without any change of machine parameters.

For the reentrant structure that we consider, we will first describe two sets of parameters and then illustrate the performance of each policy for each set of parameters separately. Finally, we show the performance of each policy when the FMS changes its parameters from the first set to the second

TABLE X
PERFORMANCE COMPARISON FOR REENTRANT STRUCTURE, SET II

	Maximum Backlog Ψ_p	Average Backlog ψ_p	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	2803	2185.0	7941	1250	102.8
CAF	3669	2407.5	12219	2066	111.5
CPK	3587	2481.1	11831	1978	110.4
DFC	2803	2176.8	8241	1250	102.5

TABLE XI
PERFORMANCE COMPARISON FOR REENTRANT STRUCTURE FROM SET I TO SET II

	Maximum Backlog Ψ_p	Average Backlog ψ_p	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	3151	2636.8	9273	1500	105.5
CAF	3691	2734.1	12194	2078	109.8
CPK	3173	2633.5	9528	1500	106.2
DFC	3151	2033.1	9144	1500	105.9

TABLE XII
PERFORMANCE COMPARISON FOR REENTRANT STRUCTURE FROM SET II TO SET I

	Maximum Backlog Ψ_p	Average Backlog ψ_p	Sum of All Maximum Buffer Level $\sum \Theta_{p,i}$	Maximum of All Maximum Buffer Level $\max_{p,i}\{\Theta_{p,i}\}$	Average of All Average Weighted Buffer Level $avg\{\theta_m\}$
CLB	2753	1358.0	7719	1500	72.1
CAF	3424	1439.8	8806	2323	71.4
CPK	2938	1530.0	8775	1636	77.1
DFC	2636	1306.3	7152	1478	73.6

set and also from the second set to the first set. For the first set of parameters (Set I), we use the same parameters as in Section IV-B-1) "Reentrant Structure." The second set of parameters (Set II) is also similar to the first set, except that the arrival rate d_1 is changed from 4/3 to 2, $\tau_{1,5}$ from 0.2 to 0.1, $\tau_{1,9}$ from 0.05 to 0.15, and the set-up times δ from 50.0 to 25.0. For Set I: $\rho_1 = 0.600$, $\rho_2 = 0.733$, and $\rho_3 = 0.467$. For Set II: $\rho_1 = \rho_2 = \rho_3 = 0.9$ (more heavily loaded machines).

The performance for the first set of parameters of the Reentrant Structure is shown in Table VIII and the second set in Table X. In this case, the DFC updates its $M_{p,i}$ over the last 120 production runs of $x_{p,i}$. Table X indicates that for this particular structure and parameters, the DFC is similar to CLB and both are better schedulers than CAF and CPK. The sum of all maximum buffer level of CLB is smaller, however, than that of the DFC.

a) Reentrant structure, changing from set I to set II: Table XI shows the performance measures after 10 000 production runs from the instant the machines change their parameters. Each machine load increases to 0.9 and the set-up times are reduced to half of the old ones. These tables indicate that the DFC adjusts to changes better than any of the other clearing policies. It adjusts better than CLB even though they perform similarly in Reentrant Structure, Set II. Both have the same maximum backlog, but the DFC has less average backlog than CLB and slightly more average of all average weighted buffer levels.

b) Reentrant structure, changing from set II to set I: Table XII shows the performance measures after 10 000 production runs from the instant the machines change their parameters. The table indicates that the DFC adjusts to machine parameter changes better than the others according to all performance measures except the last one.

While the above results seem to indicate that the DFC is the preferred scheduling policy, clearly they are not conclusive. In fact, in other simulations for other FMS structures and different machine parameter variations we found certain cases where some of the conventional policies out performed the DFC in several respects. Hence, the only conclusion that we can draw from the simulation results of this subsection is that while the DFC looks promising there is a need for a significant amount of work on the development of scheduling policies for the case where FMS are subject to machine parameter variations.

V. CONCLUDING REMARKS

We found that the fuzzy controller and adaptive fuzzy controller can show definite performance enhancements, and in most cases the DFC was the best scheduler among all scheduling strategies in the sense that it attempted to make the peak value of each buffer level and maximum backlog smaller by sometimes allowing their averages to be slightly larger. The DFC performed the best as long as the window size for updating the parameters $M_{p,i}$ was selected appropriately. We

did not develop a method to automatically select the window size for a given FMS, however, in general the performance of the DFC could be improved by enlarging the size of the window. In some cases, the DFC performed slightly worse than one of the conventional schedulers, but once we selected proper initial values for the parameters $M_{p,i}$ of the DFC, it outperformed the conventional schedulers. We emphasize that the distributed intelligent controllers for the FMS were designed so that they could automatically synthesize/tune themselves. We have illustrated that although this form of distributed adaptive system can offer improvements in the performance for FMS in certain cases, it is, in general, hard to choose certain parameters in the DFC and its performance is not always as good as conventional schedulers. Clearly there is the need for future work in i) developing scheduling policies for FMS subject to machine parameter variations and failures, ii) determining explicit mathematical conditions under which the FC, AFC, and DFC policies are known to perform better than conventional schedulers, and iii) investigating how information from multiple machines can be exploited in scheduling FMS via distributed intelligent control.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] J. R. Perkins and P. R. Kumar, "Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Trans. Automat. Contr.*, vol. 34, pp. 139–148, Feb. 1989.

[2] P. R. Kumar and T. I. Seidman, "Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 289–298, Mar. 1990.

[3] S. H. Lu and P. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. Automat. Contr.*, vol. 36 no. 12, pp. 1406–1416, 1991.

[4] C. Chase and P. Ramadge, "On real-time scheduling policies for flexible manufacturing systems," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1–9, 1991.

[5] K. L. Burgess and K. M. Passino, "Stable distributed scheduling policies for manufacturing systems," in *Proc. IEEE Conf. Decis. Contr.*, Orlando, FL, 1994.

[6] I. MacLeod and V. Lun, "An assessment of the negotiation metaphor for flexible manufacturing system control," *Eng. Appli. Artificial Intelligence*, vol. 37 no. 3, pp. 167–175, 1991.

[7] L. C. Rabelo and S. Alptekin, "A hybrid neural and symbolic processing approach to flexible manufacturing systems scheduling," in *Hybrid Architectures for Intelligent Systems*, A. Kandel and G. Langholz, Eds. Florida: CRC Press, 1992, ch. 18.

[8] K. Hadavi, W.-L. Hsu, T. Chen, and C.-N. Lee, "An architecture for real-time distributed scheduling," *AI Mag.*, vol. 13, no. 3, pp. 46–56, 1992.

[9] A. Kusiak, *Intelligent Manufacturing Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1990.

[10] R. G. Smith and R. Davis, "Frameworks for cooperation in distributed problem solving," *IEEE Trans. Syst. Man, and Cybernetics*, vol. SMC-11 no. 1, pp. 61–69, Jan. 1981.

[11] M. S. Fox, "An organizational view of distributed systems," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-11, no. 1, pp. 70–80, Jan. 1981.

[12] V. R. Lesser and D. D. Corkill, "Functionally accurate, cooperative distributed systems," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-11, no. 1, pp. 81–96, Jan. 1981.

[13] V. R. Lesser and D. D. Corkill, "The use of meta-level control for coordination in a distributed problem solving network," in *Proc. 8th Int. Joint Conf. Artificial Intelligence*, Karlsruhe, West Germany, 1983, pp. 748–756.

[14] S. C. Shapiro, "Distributed problem solving," in *Encyclopedia of Artificial Intelligence.* Canada: Wiley, 1987, pp. 245–251.

[15] E. H. Durfee, V. R. Lesser, and D. D. Corkill, "Cooperation through communication in a distributed problem solving network," *Distributed Artificial Intelligence*, M. N. Hunhs, Ed. London: Pitman, 1987, ch. 2.

[16] A. H. Levis, "Modeling and design of distributed intelligence systems," in *An Introduction to Intelligent and Autonomous Control*, P. J. Antsaklis and K. M. Passino, Eds. Boston, MA: Kluwer Academic, 1993.

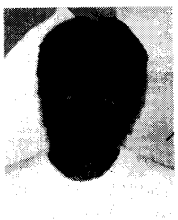
[17] E. H. Durfee, "The distributed artificial intelligence melting pot," *IEEE Trans. Syst., Man, Cybernetics*, vol. 21, pp. 1301–1306, Nov./Dec. 1991.

[18] A. Angsana and K. M. Passino, "Distributed intelligent control of flexible manufacturing systems," in *Proc. Amer. Contr. Conf.*, San Francisco, CA, 1993, pp. 1520–1524.

[19] A. Angsana, "Distributed intelligent control of flexible manufacturing systems," Master's thesis, Dept. Elec. Eng., Ohio State Univ., Columbus, 1992.

[20] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—part I," *IEEE Trans. Syst., Man, Cybernetics*, vol. 20, no. 2, pp. 404–418, Mar./Apr. 1990.

[21] ——, "Fuzzy logic in control systems: Fuzzy logic controller—part II," *IEEE Trans. Syst., Man, Cybernetics*, vol. 20, no. 2, pp. 419–435, Mar./Apr. 1990.



Anthony Angsana was born in Medan, Indonesia. He received the B.S. degree and the M.S. degree in electrical engineering in 1990 and 1992, respectively, from Ohio State University, Columbus, Ohio. His areas of interest include fuzzy control, intelligent control, and scheduling flexible manufacturing systems.

Kevin M. Passino for a photograph and biography, please see this issue, p. 405.