A NEURAL NETWORK BASED CELL CONTROLLER FOR AUTOMATED MANUFACTURING CELLS

Albert Jones
National Institute of Standards and Technology
Gaithersburg, Maryland

Yuehwern Yih and Yu-Liang Sun
School of Industrial Engineering
Purdue University
West Lafayette, Indiana

Abstract

In this paper, we describe an adaptive controller for automated manufacturing cells. The controller inputs desired performance targets and job characteristics and outputs task sequences for each piece of equipment in the cell. It contains an adjustment module to rank performance measures and a collection of dispatchers to generate the task sequences. The rankings are done dynamically based on the current cell status. The task sequences are generated using neural networks. After we describe the controller, we discuss the results of some simulation studies which show how well it performs.

1. Introduction

The control of an automated manufacturing system is extremely difficult because of the complex interactions among machines, material transportation devices, and storage buffers (O'Grady and Lee, 1988). Frequently, a hierarchical control system is used to simplify implementation, to reduce functional responsibility, and to increase local authority. Several architectures for such a hierarchical control system have been proposed during the last 15 years (Jones, 90) and (Jones and Whitt, 86). While they differ in the number of levels, and the amount of responsibility and authority given to each level, most architectures have the notion of a cell level controller which performs detailed production scheduling . This cell controller determines task sequences for the equipment level controllers to carry out based on the production and performance requirements it receives from the shop level controller.

The literature on cell controllers can be divided into two classes based on the number of performance measures. A number of approaches allow only one scheduling performance measure such as Minimize Makespan. Others use a utility function approach to combine multiple objectives for evaluation. In this paper, we show how a neural network can be used to handle both cases. In addition, we demonstrate that this approach can adapt to changes in the state of the various pieces of equipment in the cell and to changes in the performance measures passed down from the shop controller.

This paper utilizes the neural network based scheduling approach defined in (Yih et al 1994) to define a collection of dispatchers. These dispacthers form the foundation of our cell controller. The paper is organized as follows. In Section 2, we review some pertinent literature related to efforts to design and build cell controllers and neural network based schedulers. In Section 3, we describe the manufacturing cell used for the simulation experiments. In Section 4, we provide a description of the internal structure of the controller and the neural network-based equipment dispatchers. In Section 5, we use the results of

several experiments originally described in (Yih et al 1994) to demonstrate the potential capabilities of the cell controller. Finally, some summary comments and conclusions are given in Section 6.

2. Literature Review

It has been know for a long time, that the ability of dispatching rules to optimize particular performance measures is highly dependent on the state of the system. Since the mid 1980s, many researchers have attempted to use a knowledge-based approach to capture these relationships. Under this approach, a central database contains the dispatching rules for creating and modifying schedules based on the current performance measure(s) and the current system state. Typically, these relationships are generated in one of two ways: simulation studies or expert schedulers. Whenever a decision is required, the cell controller will scan the database to find the condition which matches the current performance measure(s) and the current system state, choose the associated rule, and pass the results to the equipment controllers. The major drawback to this approach is the difficulty involved in creating and maintaining a knowledge base which contains enough rules to cover every possible state. O'Grady and Lee (1988) attempted to overcome this problem in PLATO-Z, which combined a rule-based expert system with a multi-blackboard/actor model. This control system was later implemented using an object-oriented programming technique (O'Grady and Seshadri, 1992).

Wu and Wysk (1988) found a way around this problem in their multi-pass expert system. Under this system, a fixed set of candidate rules are stored in the knowledge base. Each time a decision is needed, the performance of each candidate rule is evaluated by a simulation initiated to the current system state. The rule giving the best predicted performance is selected to generate the schedules. This eliminates the need to have a direct relationship between rules and system states. Cho and Wysk (1993) removed the fixed set requirement by using a neural network to select the candidate rules to be evaluated by the simulation. However, many of the simulations require a large computational effort. This limits the applicability of this approach to real-time scheduling.

Several other researachers have tried multi-layer neural networks to deal with scheduling and candidate rule selection. Chryssolouris et al. (1990) suggested that a multi-layer network could be used as the inverse function of simulation. This, in turn, could be used to estimate the system parameters needed to compute performance measures. They showed that neural networks could be trained to learn the inverse function of simulations used in the design of small manufacturing systems. Potvin et al. (1992) used this approach to build a dispatcher for automated vehicles. Rabelo et al. (1993) used neural networks to do candidate rule selection followed by a genetic algorithm to determine the final sequences.

Yih and Jones (1992) proposed to use multi-layer networks to select candidate rules. A multi-layer network will take the current system status and the desired performance measures to determine a matching score for each dispatching rule. This matching score provides a convenient mechanism for replacing natural language rules with numerical values. The rule with the largest matching score will be selected to dispatch the jobs. This concept has been adapted for the work presented in this paper.

3. The Experimental Manufacturing Cell

3.1 Cell Description

The automated manufacturing cell used to conduct the experiments in this research is shown in Figure 1. This cell is typical of many automated cells found in factories around the world. It contains five machines and a robot which serves as the material handling device. Each machine has its own finite capacity input

and output buffers. In addition, there is an infinite capacity an input/output buffer for the cell and a temporary buffer to prevent system deadlock.

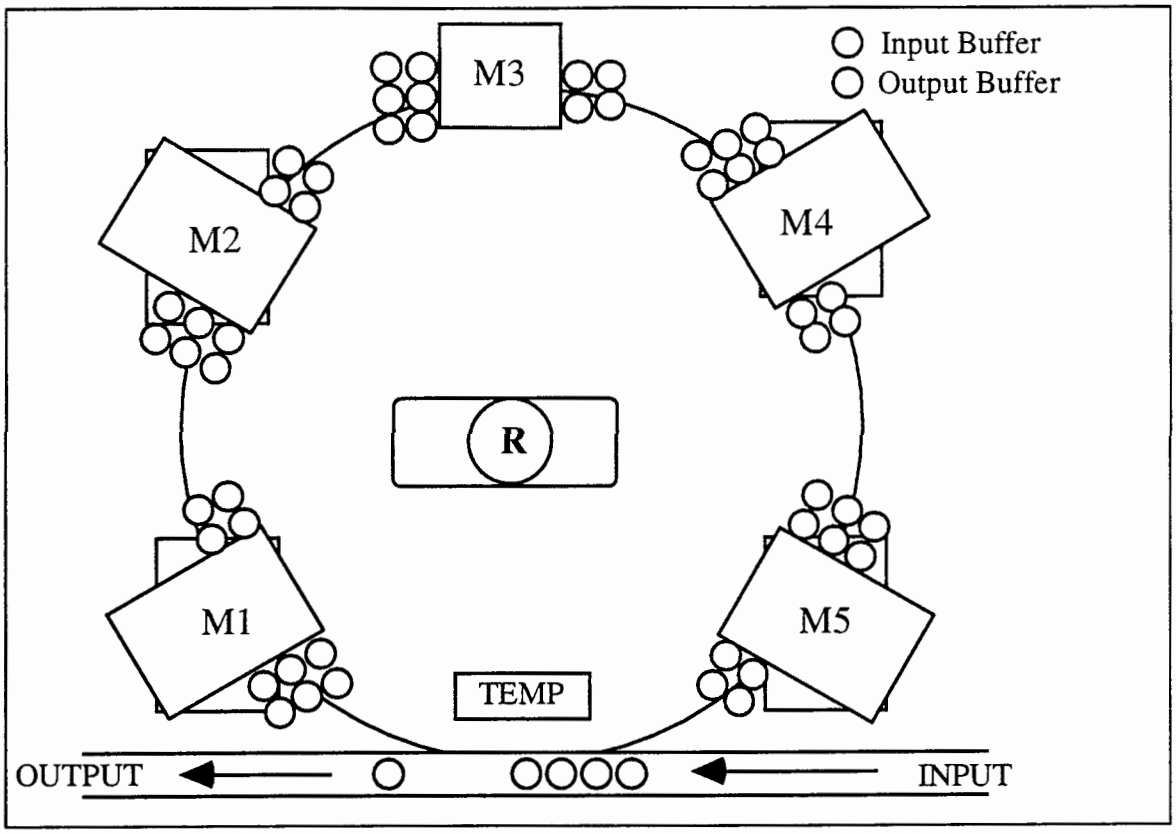

Figure 1. Experimental Manufacturing Cell

When a part arrives to this manufacturing cell, it waits in the system input buffer until the robot moves it to the input buffer at the first machine in its routing. In this study, we allow fifteen part routings through the cell. After visiting all of the machines in its routing, a part will be placed in the output buffer for the cell. Once there, it will eventually leave the cell. Each machine will process the parts waiting in its input buffer using the active dispatching rule. Once a machine finishes with a part, the completed part is automatically placed into the machine output buffer to await removal by the robot. The total time to complete processing at each machine is the sum of machining time plus setup time. In this study, the machining time for each part is provided with the routing information and does not depend on the order in which it is selected. The setup time, however, is sequence-dependent.

The robot is responsible for transporting parts from one machine to another and to/from each machine from/to the buffers for the cell. A part will wait in the cell input buffer or a machine output buffer until it is chosen as the next part to be moved. When available, the robot will select the next part to be moved based on the active dispatching rule. The selection is made from all the parts waiting in the cell. Once the part to be moved is selected, the robot will move to its current location, pick the part up, and then move it to its destination. The destination will be either the input buffer of the next machine in its process routing or the output buffer for the cell. No interruption is allowed during a transport cycle.

3.2 Performance Measures and Rules

In this study, we concentrated on four performance measures: cycle time, average waiting time, average tardiness, and tardy jobs ratio. The shop controller can input either a specified target (average waiting < 30 seconds) or a true objective function (minimize cycle time) - for our study we used targets. In computing these measures, two times are used: the time the part enters the cell input buffer and the time it enters the cell output buffer. That is, the part "arrives" when it enters the cell input buffer, and it "departs" when it reaches the cell output buffer. The time it may spend in the cell output buffer is not included in any computation.

Several of the most frequently used dispatching rules were used for our experiments. They are listed in Tables 1 and 2.

Table 1. Dispatching Rules for Machines

Rule	Description
FIFO	Operations are selected on first-come-first-serve basis.
EDD	The operation with the earliest process due date will be selected.
SPT	The operation with the shortest processing time will be selected.
STT	The operation with the shortest service time, which includes the processing time and required setup time, will be selected.
SST	The operation with the shortest setup time will be selected.
SRT	The operation with the shortest remaining processing time will be selected.
CR	The operation with the smallest CR ratio will be selected. The CR ratio is defined as: $CR = \frac{\text{Part Due Date} - \text{Current Time}}{\text{Remaining Processing Time}}$
SLACK	The operation with the least SLACK time will be selected. The SLACK time is defined by: SLACK = Process Due Date - Processing Time - Current Time

4. The Controller

The internal structure for the cell controller is shown in Figure 2. It contains an adjustment module and several dispatchers - one for the robot and for each machine in the cell. At each decision point - defined as the time when a machine needs to select the next part to process or the robot needs to select the next part to move - this controller picks the dispatching rule to be used to make that selection. This is how it works.

There are several potential advantages to this control scheme. First, since the the adjustment module recomputes performance ratios at every decision point, can effectively vary the relative priority of each performance measure as the system evolves over time. Second, changes in these relative priorities can have an immediate impact on the decisions because they are taken into account at every decision point. With this characteristic, the controller is able to quickly respond to the changes in the system state. Third, the shop contrller can also change the performance targets at any time to reflect changes to global system which may not be known to this particular cell controller.

Table 2. Dispatching Rules for the Robot

Rule	Description
SDIST / FIFO	The part with shortest transportation time including loading and unloading time will be selected. If tied, the first-come-first-serve basis will be applied.
FIFO / NEARQ	Parts will be selected upon the first-come-first-serve basis. The part in the nearest queue will be selected to break the ties.
LQL (FIFO) / NEARQ	The earliest arrival part of the longest queue will be selected. If tied in queue length, the part in the nearest queue will be chosen.
DOUT / SRT	The part which completes all the required operations will be selected. If more than one part is completed or none of them is finished, the part with shortest remaining process time will be chosen.
DIN / LQL	The part waiting in the system input buffer will be selected. If no newcomer waits, the LQL rule will be applied.
RSLACK / NEARQ	The part with least RSLACK time will be chosen while the NEARQ rule will applied to break the ties. The RSLACK is defined by: $$RSLACK = \frac{Due\ Date - \text{Re}maining\ \text{Pr}ocess\ Time - Current\ Time}{Number\ of\ \text{Re}maining\ Operations}$$

4.1 The Adjustment Module

At each decision point, a dispatcher will notify the adjustment module that it is time to move or machine a part. The adjustment module will compare the current values of the performance measures to the targets specified by the shop controller to determine the relative ranking, or priority, at this point in time. These rankings are determined by the ratio of the current performance level to the specified target. For our performance measures, the larger this ratio, the more critical the performance measure. This is so because smaller values are preferred for those performance measures. Whenever this ratio becomes larger than 1, this indicates that the current performance level has exceeded the specified target. This means that this performance measure should be given the highest priority by the equipment dispatchers at current decision point. Therefore, the effective range of these importance values is set between 0 and 1. A value of 0 means it has the lowest priority, while a value of 1 means it has the highest priority. For example, consider an average waiting time target of 30. If, at time t_1, the average waiting time is 10, then its ratio is 1/3. This may correspond to the lowest priortiy at time t_1. At some later time t_2, the avaerage waiting time maybe 35. Since a value of 35 exceeds the current target of 30, the adjustment module would give it a ratio of 1 and assign average waiting time the highest priority. Based on these priorities and its current status, the equipment dispatchers will choose a dispatching rule to select the next part to be processed or moved.

4.2 Equipment Dispatchers

After receiving the relative rankings of performance measures from the adjustment module, the equipment level dispatchers will select the proper dispatching rule based on these rankings and the current equipment status. In this paper, we use a neural network to perform this selection. The neural network will take the attributes describing the current system state and these performance rankings as its input. It will then provide a ranking for the available dispatching rules as its output. From this output, the rule with the highest ranking will be chosen to dispatch the next job. We describe how this works in the following sections.

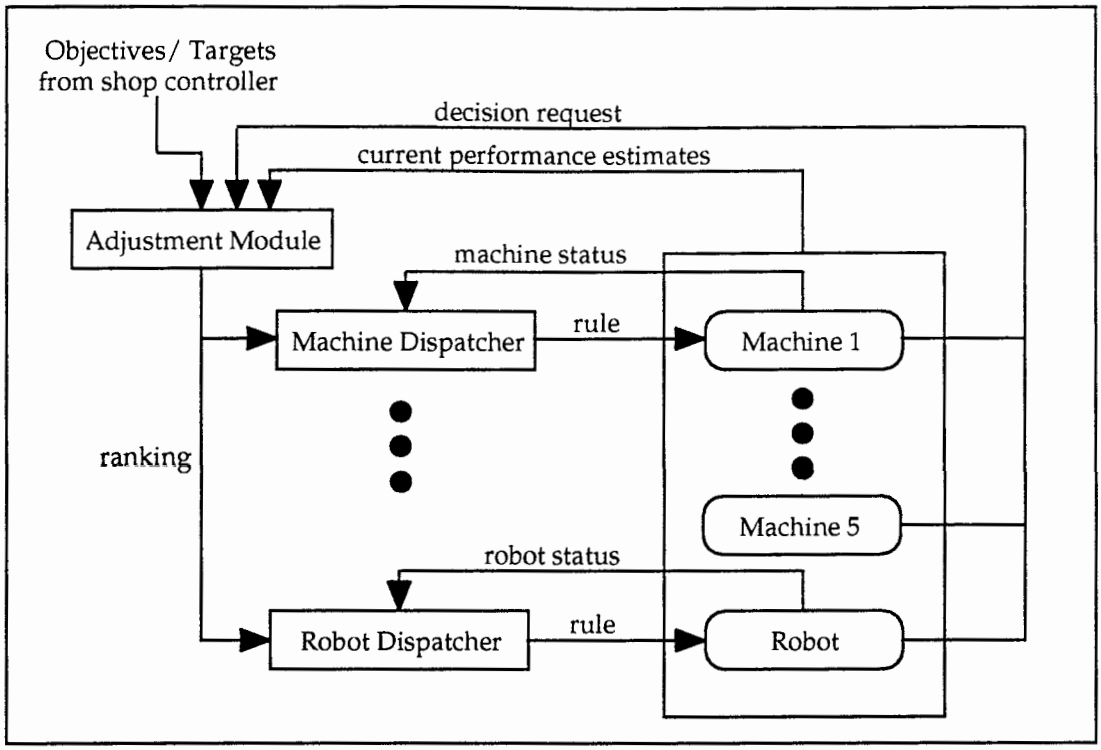

Figure 2. Internal Structure of Cell Controller

4.3 Training Data Sets

To obtain a well-trained network for each equipment dispatcher, we must generate training data sets. Each training data set consists of the two inputs listed above (system state and performance rankings) and one output (a prioritized list of the best dispatching rules).

In this study, state includes current queue length, mean and standard deviation of processing time, mean and standard deviation of slack time, mean and standard deviation of completion percentage, as well as the optimistic and pessimistic setup times. The optimistic setup time is set to the minimum of the sequence-dependent setup times for the machines and the shortest travel time between machines for the robot controller. The pessimistic setup time is chosen to be the longest of these times. The completion percentage of a part is defined as the ratio of the processing time already completed to the total required processing time. This can be used to compute how much work still remains for this part.

Single machine simulations are then conducted to determine the impact of different dispatching rules on the four system performance measures discussed in Section 3.2: cycle time, average waiting time, average tardiness, and tardy jobs ratio. Since we are using single machine simulations, it is not possible to estimate global performance measures such as cycle time, average tardiness, and tardy jobs ratio directly. These can only be estimated when the part departs from the system. Instead, we use the following computations which are based on machine rather than system times:

(1) time at machine/number of machines in routing instead of cycle time

(2) delay measured from an expected machine completion instead of average tardiness

(3) tardy jobs ratio based on expected machine completion instead of tardy jobs ratio

It is easy to show that each of these single machine computations have a high positive correlation with the corresponding global system measure. That is, whenever these single machine measures are low (high) the corresponding global system measure will also be low (high). Thus, it is our belief, that in order to reduce a particular global measure (cycle time) it is sufficient to train the network to pick a dispatching rule which reduces the corresponding single machine measure (time at machine/number of machines in routing). The process of selecting the best network configuration from the training data we generated based on this philosophy is described in the next section.

4.4 The Best Network

Since a neural network can only deal with numerical data, the dispatching rules are represented by a "matching score". The concept of the matching score was introduced by Yih and Jones (1992). The higher the score, the better the dispatching rule. In this study, the matching score formula was

$$MS_{jk} = 0.9 - \frac{0.8}{n} \sum_{i=1}^{n} \left[\text{Max} \left\{ (R_{ij} - R_{ik}), 0 \right\} \right]^2$$

where

MS_{jk} : Matching Score for rule j compared with rule k

n : total number of performance measures

R_{ij} : the ratio of i^{th} performance of rule j

Because only the highest matching score in the network is of interest, an evaluation criterion (EC) was used to evaluate different network configurations during the training phase:

$$EC = \sqrt{\frac{1}{N} \sum_{i} (D - AH_i)^2}$$

where

EC : Evaluation Criterion

N : number of patterns

D : highest matching score in each pattern

AH_i : the highest output value in training pattern i

To prevent overfitting problems, the cross validation method (Hansen and Salamon 1990, Levin et al. 1990) was applied. Several network topologies were examined to select a well-trained network for each equipment level controller. The one with the lowest EC value was chosen for each equipment level controller.

4.5 Best Dispatching Rule Results

Table 3 provides a matrix of two digit codes that denote the rule applied to machines and the rule applied to the robot, respectively. From simulation results we were able to determine that code 41, which applies dispatching rule STT to machines and SDIST to the robot, achieved the best results for cycle time and average waiting time. Code 23, with EDD for machines and LQL for the robot, obtains the best result in average tardiness. Code 46, which combines dispatching rule STT for machines and SLACK for the robot, yielded the lowest tardy job ratio. The obvious conclusion is that there is no single rule combination which can dominate, or even do well, in all performance measures.

Table 3. Coding Matrix

Dispatching rule for the robot		Dispatching rule for machines							
		1	2	3	4	5	6	7	8
		FIFO	EDD	SPT	STT	SST	SRT	CR	SLACK
1	SDIST	11	21	31	41	51	61	71	81
2	FIFO	12	22	32	42	52	62	72	82
3	LQL	13	23	33	43	53	63	73	83
4	DOUT	14	24	34	44	54	64	74	84
5	DIN	15	25	35	45	55	65	75	85
6	RSLACK	16	26	36	46	56	66	76	86

5. Experiments

We conducted three different sets of simulation experiments in this study. The first was to determine how well a given dispatching code performed against all performance measures. The second was to determine how well the proposed cell controller performed with multiple performance measures. The third was to show how well the controller responded to changes in performance targets. All simulation trials were initialized to the same state run for 50,000 time units.

5.1 The Best Against the Rest

Since it is not possible to pick a single dispatching code which simultaneously optimizes all performance measures, we conducted an experiment to see how far from "optimal" things could get. To compare different performance measures for the same dispatching code, we used something called Performance Deviation. Performance Deviation (PD) is defined by the percentage of the average difference in each performance measure from the best dispatching rule, with regard to the range between the best and the worst ones in this study. A larger PD value indicates worse performance in the associated measurement. Also, the best single rule in each performance measure will result in a PD value of 0%, while the worst one will have a PD value of 100%. Consider, for example, codes 41 and 23 from table 4. Code 41, which optimizes both cycle time and average waiting time, has a PD value of 21.39% for average tardiness. On the other hand, code 23, which optimizes average tardiness, results in PD values of 45.98% and 42.54% in cycle time and average waiting time, respectively.

Table 4. Best Dispatching Rule for Each Performance Measure

	Cycle time	Average waiting time	Average tardiness	Tardy jobs ratio
Code 41	0%	0%	21.39%	1.62%
Code 23	45.98%	42.54%	0%	56.96%
Code 46	18.35%	13.26%	13.58%	0%

5.2 Multiple Performance Measures

To handle multiple performance measures, the adjustment module must provide, as we discussed above, provide the equipment level dispatchers with the relative ranking of all measures. Moreover, it must reevaluate these rankings at each decision point. Using this ranking together with the current machine status, the neural network will determine the matching score for each rule and the one with highest score

will be selected to dispatch the next part. To show how this might work, we ran several simulation experiments to look at pairs of performance criterion with equal rank given to each criterion. Then we conducted aonther analysis to look at all four simultaneously, to determine if a single code could be used. The experiments and their associated performance deviations are listed in Table 5. A value in a particular column indicates that the measure was included in the experiment, a "-" indicates it was not.

For the pairs-anaylsis, each experiment consisted of two parts. First we ran the entire simulation using only the best dispatching rule. Then, we ran the simulation again and allowed the controller to change dispatching rules at each decision point in the simulation. At the end of the run, we computed the various performance measures and their associated PD values. The results are shown in Table 5. A negative PD value indicates that the controller performed better than the best dispatching rule. A positive sign indicates that the dispatching rule performed better. In either case, the magnitude of the number indicates how much better. For example, in Ex 1 and Ex 5, the controller did better than the dispatching rules in both criteria concerned. In experiments Ex 3, the controller did better in one measure cycle time (PD value of -1.18%) but did worse for tardy job ratio (PD value of 5.83%). Similar results were obtained for Ex 4. In experiments Ex 2 and Ex 6, the controller actually did worse, to varying degrees, in both performance measures.

Table 5. Simulation Results - PD Values

	Performance Measures			
	Cycle time	Average waiting time	Average tardiness	Tardy jobs ratio
Ex 1	- 1.52%	- 0.04%	—	—
Ex 2	0.01%	—	15.87%	—
Ex 3	- 1.18%	—	—	5.83%
Ex 4	—	- 3.42%	17.30%	—
Ex 5	—	- 4.37%	—	- 1.45%
Ex 6	—	—	4.03%	25.23%
Ex 7	-8.00%	- 3.55%	19.27%	- 1.65%

We showed in Table 4, that it is not possible to optimize all of our global performance measures simultaneously by picking a single code in advance and using it for the entire simulation. In fact, choosing anything but the best code can lead to very poor performance. We ran another experiment (Ex 7) to determine if the cell controller could improve on these results. We can see from the last line in Table 5 that, while it did poorly for average tardiness, the controller surpassed the best dispatching rule for the other three performance measures.

5.3 Changing Performance Targets

In the previous experiments, we kept the targets for the performance measures constant during all simulation runs and allowed the controller to select dispatching rules based on the changing state of the system. We also conducted an experiment to determine how well the controller responded when the targets changed. We chose the combination of average tardiness and average waiting time because the controller did well for one but not the other (see Table 5). The change performance targets is shown below:

- At time 0, the initial target vector was [14, 160, 20, 0.5] for cycle time, average waiting time, average tardiness, and tardy jobs ratio, respectively.
- At simulation time 15,000, the objective vector changed to [14, 120, 60, 0.5].
- At simulation time 35,000, the objective vector became [14, 120, 20, 0.5].

The simulation results are illustrated in Figure 3 for the average waiting time and average tardiness. until about time 5,000 both performance measures are above their target values. At around time 5,000 the waiting time is below 160, but the tardiness is still above 20 indicating that the controller will continue to give it top priority. As the simulation approaches time 15,000 the waiting time is still below 160 and the tardiness is still above 20, but decreasing. If we follow trajectory 1, we can see that the tardiness will eventually reach the target. At time 15,000 the waiting time target is decreased to 120 and the tardiness target is increased to 60. Now, the controller will switch priorities because tardiness is now below target, but the waiting time is above target.

By looking at the graphs, we can see the impact of the controller's decisions. In the top graph, the trajectory changes from 1 to 2, indicating a gradual reduction in waiting time. In the bottom graph, a similar change takes place. However, in this case we see a gradual increase in the tardiness. As the simulation time approaches 35,000 we can see that both performance measures are decreasing. Waiting time is still not below target, but if we continue to follow trajectory 2, we can see that it will cross the target threshold at about time 37,000.

At time 35,000 the controller must react to a change in the average tardiness target. Both graphs begin a new trajectory along 3. As expected, waiting time begins to degrade slightly, but tardiness begins to improve slightly. These results indicate that the controller can adapt quickly to changes in performance targets. They also show that the controller can effect positive changes in individual performance targets. But, just as we saw in table 5, it can not always improve multiple performance measures simultaneously.

6. Summary and Conclusions

A neural network based cell controller, consisting of an adjustment module and one dispatcher for each piece of equipment, was proposed for scheduling and controlling a sample manufacturing cell. The adjustment module uses specified targets for several performance measures from the shop controller and the current performance estimates to determine the relative ranking of those performance measures. This ranking indicates the relative importance of these perfromance measures at each decision point. Based on these rankings and current equipment status, the equipment dispatchers will select a dispatching rule. This dispatching rule will be used to generate task sequences for machines and robots, which will be used until the next decision point. Each of these dispatchers was implemented using a neural network.

A simulation model of the sample manufacturing cell was constructed to evaluate the performance of the proposed controller. The initial simulation results were used to 1) determine the best of the available dispatching rules for each performance measure, and 2) train the neural network dispatchers. After this, two additional sets of experiments were run to determine how well the cell controller performed when given multiple performance targets, and when the targets changed during the simulation. The results show that the controller performs well under some combinations of performance measures and poorly under others. We also showed that the controller is able to respond quickly to changes in performance measure targets on an individual basis.

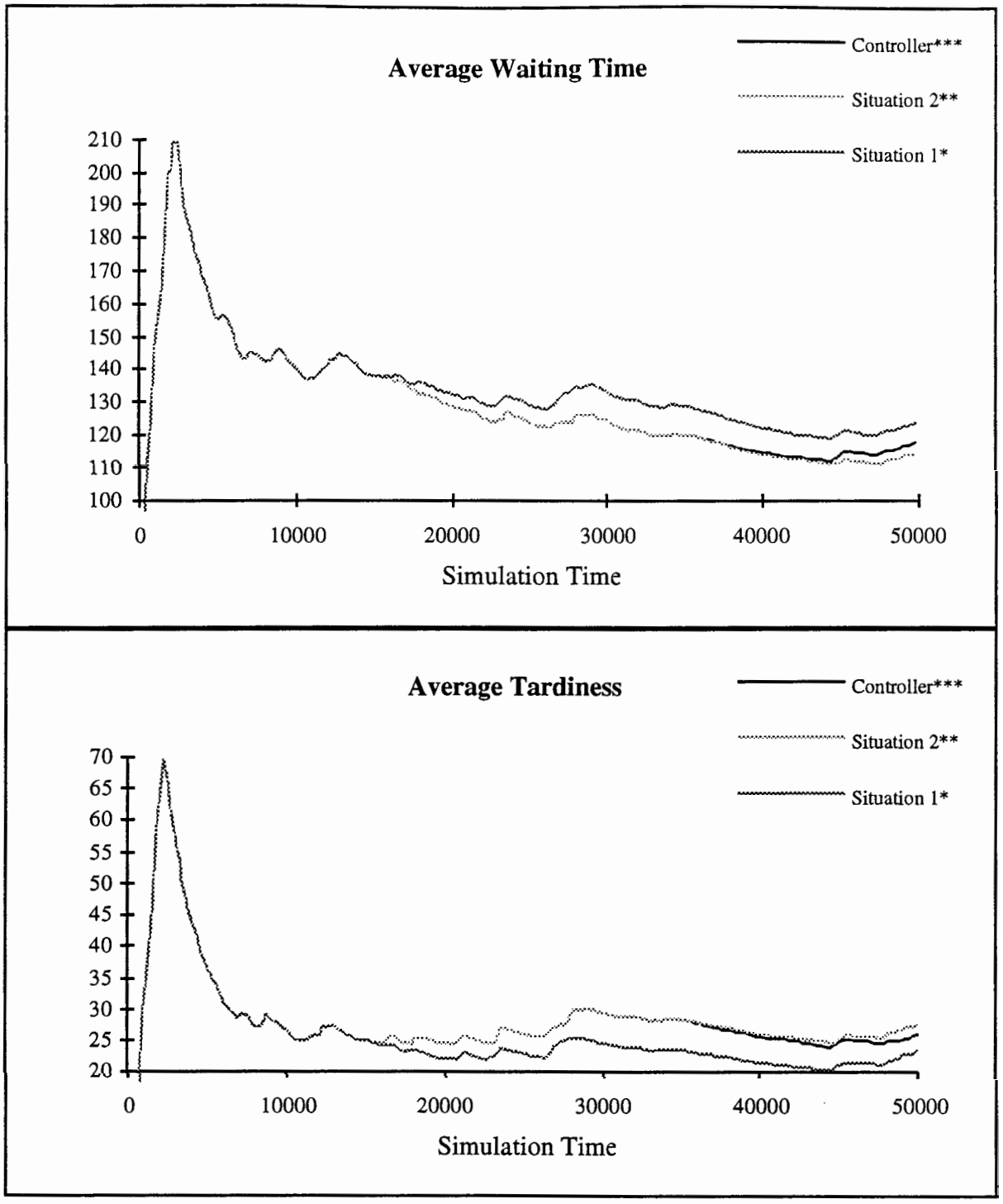

Figure 3. Simulation Results for Changing Performance Targets

References

Arizono, I., Yamamoto, A., and Ohta, H., 1992, Scheduling for minimizing total actual flow time by neural networks. *International Journal of Production Research*, **30** (3), 503-511.

Cho, H. and Wysk, R. A., 1993, A robust adaptive scheduler for an intelligent workstation controller. *International Journal of Production Research*, **31** (4), 771-789.

Chryssolouris, G., 1986, A decision making framework for manufacturing systems. *National Bureau of Standards, Special Publication 724*, 79-86.

Chryssolouris, G., Lee, M., Pierce, J., and Domroese, M., 1990, Use of neural networks for the design of manufacturing systems. *Manufacturing Review*, **3** (3), 187-194.

Foo, Y. S. and Takefuji, Y., 1988a, Stochastic neural networks for solving job-shop scheduling. part 1: problem representation. *IEEE International Joint Conference on Neural Networks*, **2**, 275-282.

Foo, Y. S. and Takefuji, Y., 1988b, Stochastic neural networks for solving job-shop scheduling. part 2: architecture and simulation. *IEEE International Joint Conference on Neural Networks*, **2**, 283-290.

Jones, A., (ed), 1990, Proceedings of CIMCON'90, *NIST Special Publication 785*, NIST, Gaithersburg, MD 20899, May 1990.

Jones, A., and Whitt, N., (eds), 1986, Proceedings of the Factory Standards Model Conference, National Bureau of Standards, Gaithersburg, Md 20899, November 1985.

Hansen, L. K. and Salamon, P., 1990, Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12** (10), 993-1001.

Levin, E., Tishby, N., and Solla, S. A., 1990, A statistical approach to learning and generalization in layered neural networks. *Proceedings of the IEEE*, **78** (10), 1568-1574.

Lippmann, R. P., 1987, An introduction to computing with neural nets. *IEEE ASSP Magazine*, **4** (2), 4-22.
O'Grady, P. and Lee, K. H., 1988, An intelligent cell control system for automated manufacturing. *International Journal of Production Research*, **26** (5), 845-861.

O'Grady, P. and Seshadri, R., 1992, Operation of X-Cell — an intelligent cell control system. *Computer - Integrated Manufacturing Systems*, **5** (1), 21-30.

Potvin, J., Shen, Y., and Rousseau, J., 1992, Neural networks for automated vehicle dispatching. *Computers and Operations Research*, **19** (3/4), 267-276.

Rabelo, L., Yih, Y., Jones, A., and Tsai, J., 1993, Intelligent scheduling for flexible manufacturing systems. *Proceedings of IEEE International Conference on Robotics and Automation*, **3**, 810-815.

Wu, S. D. and Wysk, R. A., 1988, Multi-pass expert control system - a control scheduling structure for flexible manufacturing cells. *Journal of Manufacturing Systems*, **7** (2), 107-120.

Yih, Y. and Jones, A. T., 1992, Candidate rule selection to develop intelligent scheduling aids for flexible manufacturing systems (FMS). *Proceedings of a Joint German/US Conference, Hagen, Germany, June 25-26*, 201-217.

Sun, Y., Yih, Y., Jones, A., and Rabelo, L., 1994, A Neural Network Based Scheduler For Flexible Manufacturing Systems, *Proceedings of the 1994 International Mechanical Engineering Congress and Exposition, Chicago, IL., November 6-11*, Vol 2, 61-69.

Zhang, C., Yan, P., and Chang, T., 1991, Solving job-shop scheduling problem with priority using neural network. *IEEE International Joint Conference on Neural Networks*, 1361-1366.

Zhou, D. N., Cherkassky, V., Baldwin, T. R., and Olson, D. E., 1991, A neural network approach to job-shop scheduling, *IEEE Transactions on Neural Networks*, **2** (1), 175-179.