Distributed Feedback Mechanism for Just-In-Time Scheduling Problem

Wei Weng, Shigeru Fujimura
Graduate School of Information, Production, and Systems, Waseda University
808-0135 2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, Japan
wengwei@toki.waseda.jp, fujimura@waseda.jp

***Abstract*—The problem considered in this research is the just-in-time scheduling of a manufacturing environment that is able to produce several different products. New jobs come randomly into the system, expected to become one of the products. Each job must go through multiple stages before it can be finished as a product. There are multiple machines at each stage, and the processing time of each product on each machine is different. There exits delivery time between stages. Previous researches did much on the static scheduling of such problem by using mixed integer linear programming. However, little efforts have been made on realtime scheduling, which means the release time of jobs is unknown. But such scheduling is becoming more and more important under the increasingly competitive manufacturing market. In this paper, two distributed feedback mechanisms are proposed to solve the realtime scheduling problem of minimizing earliness and tardiness penalties of all jobs. The simulation shows that the proposed distributed feedback mechanisms deliver quite competitive performance for the targeted problem.**

I. Introduction

In the manufacturing industry, some factories are divided into multiple stages so that processing and management become more flexible and easier. The jobs released into the system go through all the stages in sequence in order to become a product. There is at least one machine at each stage to carry out the processing of jobs. And there is delivery system between stages, such as assembly line, to carry semi-products of the previous stage to the next one. In such a manufacturing environment, as there are many different flow lines available for each job to go through the system, the scheduling problem is called Flexible Flow Shop (FFS), or Flexible Flow Line (FFL) problem. Many researches have been conducted on such problem as we briefly review some literatures.

An agent-based learning approach was once proposed by W. Brauer and G. Weiß [1] to solve the above mentioned problem targeting at minimizing makespan. In that paper, each machine makes choice for its finished job which machine of the next stage to go to, after which its predecessor machine will learn its minimum remaining time. Other early researches focused on relatively simple environments such as identical machines or few stages, which is summarized in the survey [2]. In recent years, different manufacturing environments with specific characters are designed and researched, for example, FFS problem with sequence-dependent setup times [3], with common cycle multi-product lot sizing [4], with a bottleneck stage [5], or with processor blocking [6].

All these researches have one thing in common, that is, they all make the scheduling plan in a static way. In other words, the release time of all jobs into the system is known in advance. The optimization of such static planning has been achieved throughout the history by using Mixed Integer Linear Programming (MILP), which is seen in the paper of T. Sawik [7] in 2000, and is still widely seen today, for example, Y. Liu [8] in 2008. The long historical period reveals that MILP is an important approach of solving the static FFS problem. By establishing different constraints of the model of the same problem, MILP may deliver different performance in the efficiency of achieving optimal solution. Therefore, the efficiency of convergency to the optimal is the goal of many current researches.

But the increasingly competitive and unpredictable market has raised the demand of realtime scheduling that is able to handle coming orders immediately without impairing a good profit. In realtime scheduling, the release time of each job will no longer be known a priori, which resembles the actual situation that a customer may come by the shop window and order a product at any time. Under such circumstance, the static optimization methods will no longer be of any service. Unfortunately, few valuable researches have been found that can address such problems. In order to fill this void, realtime scheduling is targeted in this paper. The characteristics of such realtime scheduling is that scheduling decisions are made in real time as jobs entering the system, no planning in advance, and no knowledge about coming orders. The system is working by itself to adjust to the coming orders or jobs. Such self improving system is very much dependant on effective feedbacks and cooperation between machines. This paper presents three distributed feedback mechanisms that are believed to be able to address the problem. A comparison between these feedback mechanisms is conducted to see what kind of information is the most useful one, and what kind of cooperation among machines are most helpful. A manufacturing environment similar with that is presented in many other researches, for example, [8], is employed as the simulation environment. The proposed feedback mechanisms are also compared with a dispatching rule developed in previous research [9] to evaluate the overall quality of the proposed mechanisms.

This paper is organized as follows: Section II gives the problem description; Section III introduces a feedback mechanism presented in previous research; Section IV proposes two

978-0-7695-3641-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICIS.2009.12

more distributed feedback mechanisms; Section V shows the comparison and computational simulation result; Section VI concludes the paper with outlook on future work.

II. Problem Description

The following just-in-time scheduling problem is considered in this paper. The factory is able to manufacture P kinds of products (p = 1, 2, ..., P). The release time of each job Rt_j is unknown, in other words, a new order of a product (a job) may arrive at any time. Each job must be processed into one of the P kinds of products. There are S stages in the factory (s = 1, 2, ..., S), with M_s nonidentical machines at stage s. All jobs must go through all the S stages in sequence in order to be manufactured as a finished product. The processing time of each product at each machine of each stage is different. There exists delivery time $DT_{s,n}^{s-1,m}$ from machine m of stage $s-1$ to machine n of stage s. Each product has a due time Dt_p, which is the time period for product p to be made, and a due date D_p, which is the time at which product p should be finished and delivered to the customer. Let $Dt_{p(j)}$ and $D_{p(j)}$ denote the due time and the due date of job j, respectively, then

$$Rt_j + Dt_{p(j)} = D_{p(j)}. \tag{1}$$

If a job is finished before its due date, it has an earliness E_j. On the other hand, if a job is finished after its due date, it has a tardiness T_j. Let C_j denote the completion time of the job, then the earliness of a job j completed before its due date is given as:

$$E_j = max(0, d_p - C_j) \tag{2}$$

And the tardiness of a job completed after its due date is given as:

$$T_j = max(C_j - d_p, 0) \tag{3}$$

The earliness penalty e_p and tardiness penalty t_p for each product p are both given constants. The objective of this research is to minimize the total earliness and tardiness penalties of all jobs, or:

$$JIT = Min \sum_{j=1}^{J} (e_{p(j)} \times E_j + t_{p(j)} \times T_j). \tag{4}$$

where j is the index of job; $p(j)$, the product of job j; and J, the total number of jobs.

Besides, the following conditions are assumed in this paper:

(1) There is no obvious bottleneck stage in the system;
(2) One machine can only process one job at a time;
(3) The processing of each job is non-preemptive;
(4) Machines do not break down;
(5) There allows a queue before each machine, with jobs inside the queue processed in First-Come-First-Served sequence;
(6) Earliness penalties are always smaller than tardiness penalties.

The problem is characterized by online just-in-time scheduling since jobs are not released into the system before the planning begins, nor released at a known time, but released one after another randomly into the system during the production. Under such circumstance, static scheduling or planning methods cannot deliver any solution. Therefore, online planning methods using information of already processed jobs and working machines are quite desirable for such problem.

III. Previous Research Fruits

Previous research has proposed two dispatching rules, and an agent-based learning method for this problem [9], of which the agent-based learning method shows a comparatively good simulation result. That method involves some feedback and update process as the learning process continues.

In this paper, two more feedback mechanisms will be proposed to compare against the previously presented agent-learning feedback mechanism. As these methods have something in common, the comparison will be easier understood if the previous agent-learning approach is briefly introduced and illustrated in this paper. Therefore, a brief introduction of the previously presented feedback mechanism is given as follows.

Let PT_{psn} denote the processing time of product p by machine n at stage s; Q_{sn}, the queue time that a newly arrived job must wait before it can be started processing on machine n of stage s, then the time from a job ready to be delivered from machine m of stage $s-1$ to the time that the job comes out of (finished by) machine n of stage s can be presented as:

$$T_{out,psn} = max(Q_{sn}, DT_{s,n}^{s-1,m}) + PT_{psn} \tag{5}$$

Every time machine m of stage $s-1$ finishes a job, it calculates a predicted completion time of the job on each machine of the coming stage:

$$PCT_{jsn}^{M2M} = \underbrace{T_{j,past} + T_{out,psn}}_{\text{known}} + \underbrace{FT_{psn}}_{\text{to be learnt}} \tag{6}$$

where PCT_{jsn} stands for the predicted completion time of the job j if machine n is chosen at the coming stage s; $T_{j,past}$, the time that has already past in the manufacturing process of job j; FT_{psn}, the following time of machine n, or the remaining time after machine n of stage s for the product p. $T_{j,past}$ and $T_{out,psn}$ are known, but FT_{psn} is to be learnt by feedbacks from agents. After the machine agent chooses the machine with the least penalty calculated upon the difference between (6) and the job's due time $Dt_{p(j)}$, the selected agent will feedback and thus updating the following time information of its previous machine m by:

$$FT_{ps-1m} = \alpha \times FT_{ps-1m} + (1-\alpha) \times (T_{out,psn} + FT_{psn}) \tag{7}$$

As the feedback mechanism is carried out by machine n and machine m, this is named Machine to Machine (M2M) Feedback Mechanism in this paper. Fig.1 gives an illustration of the approach, the machine has just finished the yellow job, it chooses machine n of stage s for the job, then the selected machine feeds back to it the information marked red.

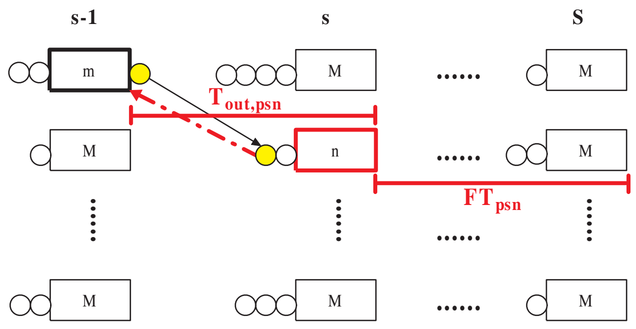

Fig. 1. M2M feedback mechanism.

IV. Proposed Feedback Mechanisms

The M2M feedback mechanism has been proved in previous research to be effective for the targeted problem, but still better solutions are believed to exist. Therefore, various feedback mechanisms have been tried, and two of them are to be presented in this paper. The reason why feedback mechanism has always been our first consideration is that the continuously changing factory situation and the randomly coming jobs into the system have caused great difficulty in pre-decision or pre-planning, thus the ideal way is to adjust the system to the changing demand and orders by online feedback and online improvement. In addition, the concurrent computing carried out by numerous local agents is able to save a lot of system resources and can be modified with considerable flexibility. With regards to these advantages, two more distributed feedback mechanisms are proposed in this paper for the objective of minimizing total job earliness and tardiness penalties.

Each machine represents an agent who only knows very limited information - the realtime situation of machines of the coming stage to which its finished jobs are to be delivered. After calculation, the machine will decide which machine of the coming stage the job should be delivered to, then the selected machine will feedback some useful information, as we discuss the approaches below.

A. SM2M Feedback Mechanism

The M2M feedback mechanism involves only information of the selected machine as feedback. But more inclusive information may help deliver a better performance, therefore, or Stage and Machine to Machine (SM2M) feedback mechanism is developed, in which not only the information of the machine but also the information of the stage is bound together when the feedback process is conducted. To be more precisely, not only the following time information of the selected machine n, but also the mean queue length of all machines of stage s are included in the feedback.

When a machine finishes a job, it will calculate a predicted completion time for job j on each machine of the next stage, PCT_{jsn}^{SM2M}, by using the same equation with PCT_{jsn}^{M2M}, i.e. equation (6). Then the machine predicts a penalty of the job on each machine of the coming stage by:

$$P_j = \begin{cases} eP_j, & PCT_{jsn}^{SM2M} < Dt_{p(j)}. \\ tP_j, & PCT_{jsn}^{SM2M} > Dt_{p(j)}. \end{cases} \tag{8}$$

where

$$eP_j = [Dt_{p(j)} - PCT_{jsn}^{SM2M}]^+ \times e_{p(j)} \tag{9}$$

$$tP_j = [PCT_{jsn}^{SM2M} - Dt_{p(j)}]^+ \times t_{p(j)} \tag{10}$$

Based on these predicted penalties for the same job, the machine will determine the destination for the job by selecting the machine with the least predicted penalty. After that, the feedback process is conducted by the following equation:

$$\begin{aligned} FT_{ps-1m} &= \beta \times FT_{ps-1m} + (1-\beta) \\ &\times [max(\overline{Q_s}, \overline{DT_s^{s-1}}) + \overline{PT_{ps}} + FT_{psn}] \end{aligned} \tag{11}$$

where $\overline{Q_s}$ means the mean queue length of the machines of stage s; $\overline{PT_{ps}}$, the mean processing time of the product p at stage s; and $\overline{DT_s^{s-1}}$, the mean delivery time from stage $s-1$ to stage s.

Let M_s denote the number of machines at stage s, then:

$$\overline{Q_s} = \frac{\sum_{n=1}^{M_s} Q_{sn}}{M_s} \tag{12}$$

$$\overline{PT_{ps}} = \frac{\sum_{m=1}^{M_s} PT_{psm}}{M_s} \tag{13}$$

$$\overline{DT_s^{s-1}} = \frac{\sum_{m=1}^{M_{s-1}} \sum_{n=1}^{M_s} DT_{s,n}^{s-1,m}}{M_{s-1} \times M_s} \tag{14}$$

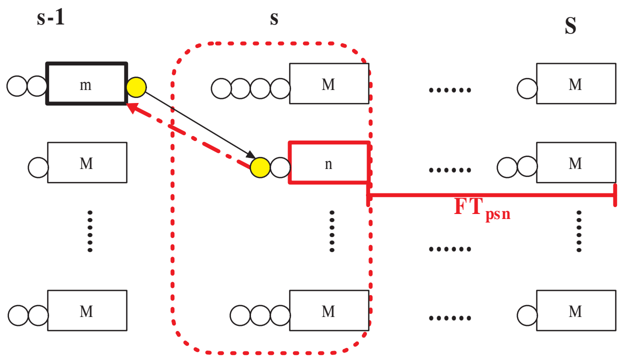

Fig. 2. SM2M feedback mechanism.

Since in this feedback, not only the information of the selected machine but also of the general (mean) queue and processing time situation of the coming stage is included, it can be called a Stage and Machine to Machine feedback mechanism, which means that stage information and machine information are both included in the feedback. Fig.2 illustrates this feedback mechanism. The marked job has just been finished by machine m of stage $s-1$, and is delivered to machine n of stage s. The information marked red is the feedback information, which includes the mean information of stage s.

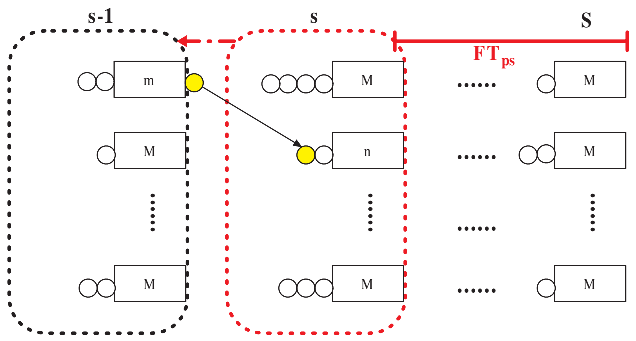

Fig. 3. S2S feedback mechanism.

B. S2S Feedback Mechanism

The multiple stage multiple machine environment provides a great opportunity to develop various feedback mechanisms whose feedback information might be restricted or inclusive. In the M2M feedback mechanism, the feedback information contains relatively limited local information of just one machine, while in the SM2M feedback mechanism, more information is included. However, further more inclusive feedback mechanism is still possible, between the previous stage and the coming stage. These different feedback mechanisms can help reveal the effectiveness of limited information feedback and inclusive information feedback for the targeted problem. In the following, a feedback mechanism called Stage to Stage (S2S) Feedback Mechanism is proposed and presented.

Similar with the above two feedback mechanisms, in this approach, each machine will calculate a predicted completion time for every job it finishes on each machine of the next stage:

$$PCT_{jsn}^{S2S} = \underbrace{T_{j,past} + T_{out,psn}}_{\text{known}} + \underbrace{FT_{ps}}_{\text{to be learnt}} \tag{15}$$

where FT_{ps} means the following time after stage s for product p. The machine will then predict a penalty for the job on each machine of the next stage, as described in (8), (9), and (10). After choosing the machine with the minimum penalty, the machine delivering the job from stage $s-1$ to stage s will update the following time of its own stage $s-1$ by:

$$\begin{aligned} FT_{ps-1} &= \gamma \times FT_{ps-1} + (1-\gamma) \\ &\times [max(\overline{Q_s}, \overline{DT_s^{s-1}}) + \overline{PT_{ps}} + FT_{ps}] \end{aligned} \tag{16}$$

This is a stage to stage approach, the following time of each machine no longer exits; instead, the following time of each stage is introduced. Upon such modification, the feedback and update process is performed more frequently and easily. The Fig.3 shows the S2S mechanism, in which the feedback is not between machines but between every two consecutive stages.

By now, the three distributed feedback mechanisms have been explained in detail. In the next section, these mechanisms will be compared, evaluated, and analyzed to see which one delivers the best performance, and thus revealing the best feedback approach for the targeted just-in-time scheduling problem.

V. Computational Simulation

In this section, the three distributed feedback mechanisms described are evaluated against each other in two manufacturing environments. Table I, II, and III form the first manufacturing environment. Table I describes the earliness penalty, tardiness penalty, and due time of each product that can be produced by the manufacturer. Table II gives the processing time of each product on each machine of each stage. The processing time is designed that fast machines are always faster than others in spite of the product kind. Table III offers the delivery time between stages. The delivery time to stage 1 means the delivery time from the system entry buffer to each machine of stage 1. These given data are fixed during the whole manufacturing process. The other manufacturing environment has 7 stages with 2 machines at each stage, the environment data is not presented in this paper due to limited space, but the result will be announced.

TABLE I
Product data.

	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
e_p	9	11	5	7	8
t_p	19	23	10	16	15
Dt_p	155	167	178	265	226

TABLE II
Processing time of each product.

		PT_{psm}				
s	m	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
1	1	2	3	5	8	1
	2	10	33	12	21	16
	3	22	42	23	33	17
	4	15	25	18	30	21
2	1	8	13	19	2	25
	2	21	25	35	15	37
	3	13	18	11	9	18
3	1	34	45	25	47	30
	2	24	19	15	33	17
	3	29	38	19	37	25
	4	15	12	6	22	11
4	1	36	20	34	28	17
	2	30	11	23	16	7
	3	15	6	12	9	2
5	1	17	24	38	27	7
	2	21	33	47	32	11
	3	36	40	49	39	15
	4	9	17	31	15	6
	5	28	35	40	30	8

The simulation is programmed in JAVA and run on a platform of Pentium IV 3.0G with 1G memory. In the simulation, α, β, γ are set to be 0.7, 0.5, and 0.6, respectively, as the result of testing. The initial FT_{psm} is set to be:

$$FT_{psm} = \sum_{k=s+1}^{S} (\overline{PT_{pk}} + \overline{DT_k^{k-1}}) \tag{17}$$

TABLE III
DELIVERY TIME BETWEEN STAGES.

s	m	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
in		8	2	10	16	-
1	1	6	4	5	-	-
	2	3	9	16	-	-
	3	10	15	8	-	-
	4	2	10	12	-	-
2	1	5	13	15	2	-
	2	11	3	9	13	-
	3	8	15	5	10	-
3	1	6	9	1	-	-
	2	9	12	5	-	-
	3	4	6	3	-	-
	4	3	18	12	-	-
4	1	25	3	1	11	17
	2	17	10	9	13	6
	3	10	8	6	15	4

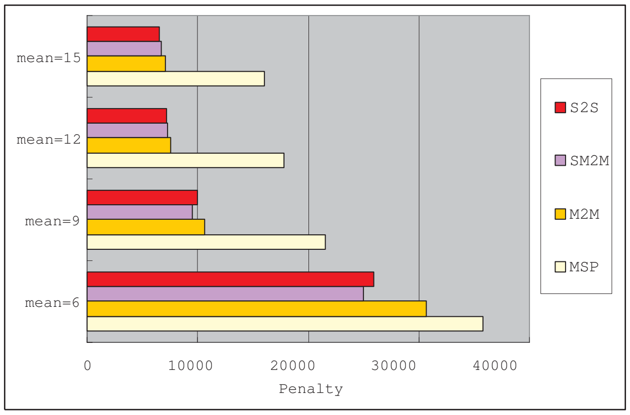

Fig. 4. Penalties of different approaches under various system loads for 100 jobs.

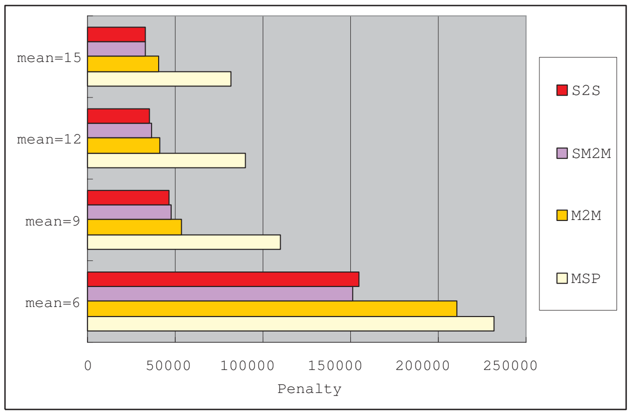

Fig. 5. Penalties of different approaches under various system loads for 500 jobs.

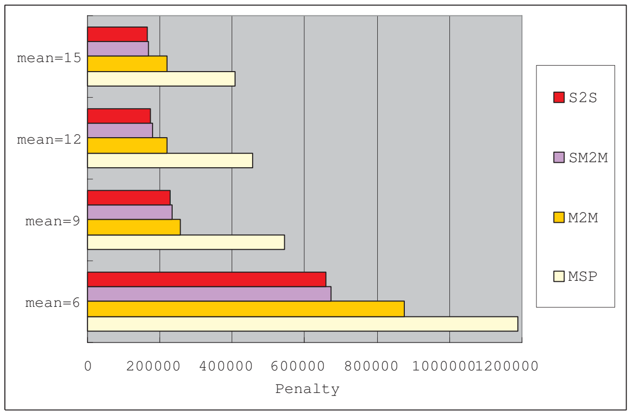

Fig. 6. Penalties of different approaches under various system loads for 2500 jobs.

The new job arrival ratio, or the time interval between every two consecutively arrived new jobs, complies with an exponential distribution whose mean is given. Each job must be processed into one of the five products given in Table I. The distribution mean is set around the system capacity 9, which means that at mean = 9, the mean queue length of all machines in the system does not increase continuously with the increase of jobs released into the system under the dispatching rule Minimum Stage Penalty [9]; in other words, the system can be considered as under a balanced status.

Therefore, the system is simulated under mean = 6, mean = 9, mean = 12, and mean = 15, distributing evenly around the system balanced capacity. Three scenarios have been tested with 100, 500, and 2500 jobs, respectively. The program has been run 200 times for each scenario to get the average result given in Fig.4, 5, and 6.

The comparative performance of each approach is the same in both the manufacturing environments, which means the result figures are also very similar in both cases. Therefore, only the result figures of the first manufacturing environment given in Table I, II, and III are presented in this paper.

As can be seen from the result figures, S2S feedback mechanism generally delivers the best performance, followed by SM2M, further followed by M2M. But when mean=6, which means the system is working under a congested situation, the SM2M feedback mechanism outperforms S2S in the 100-job and 500-job cases, which implies that S2S feedback mechanism may not be very suitable for a congested system. Anyway, S2S and SM2M both outperform M2M, which reveals that inclusive feedback information can help achieve a better performance. In addition, since S2S feedback information is updated on a more frequent basis, the result may indicate that frequent updating is useful in such feedback mechanism. The M2M feedback mechanism is overshadowed by both SM2M and S2S feedback mechanism, but the feedback mechanisms generally deliver much better solutions than the dispatching rule Minimum Stage Penalty. Accordingly, it can be concluded that the distributed feedback mechanisms are able to offer competitive performance for the just-in-time FFS scheduling problem.

VI. Conclusion and Outlook

This paper works on the just-in-time FFS scheduling problem, aiming at minimizing total earliness and tardiness penalties of all jobs. The problem is featured by realtime job release, so that static planning cannot solve the problem. Efforts have been made on several distributed feedback mechanisms in which the system learns by itself during the production process to progress towards the system objective. Realtime feedback, learning, and improvement are believed to be of contribution to the targeted problem. And parallel computing provides flexibility that no other algorithm can offer. The proposed feedback mechanisms have been tested in a common multi-stage manufacturing environment, compared with a previous presented feedback mechanism, and against a dispatching rule. The simulation shows that inclusive feedback information is desirable for the problem, and the feedback mechanisms deliver much better overall performance than the dispatching rule, which means such learning and adjusting mechanism is fit for the problem.

Future work may introduce unexpected disturbances into the manufacturing environment such as machine breakdown or other emergencies, so as to test the proposed feedback mechanisms under more complicated environments, through which improvements may be made.

References

[1] W. Brauer, G. Weiß, *Multi-Machine Scheduling - A Multi-Agent Learning Approach*, 3rd International Conference on Multi-Agent Systems (ICMAS), pp. 42-48. Paris, France, 1998.

[2] R. Linn, W. Zhang, *Hybrid Flow Shop Scheduling: A survey*, Computer & Industrial Engineering, vol. 37, pp. 57-61, 1999.

[3] M. E. Kurz, R. G. Askin, *Scheduling Flexible Flow Lines with Sequence-Dependent Setup Times*, European Journal of Operational Research, vol. 159, pp. 66-82, 2004.

[4] B. Akrami, B. Karimi, S. M. M. Hosseini, *Two Metaheuristic Methods for the Common Cycle Economic Lot Sizing and Scheduling in Flexible Flow Shop with Limited Intermediate Buffers: The Finite Horizon Case*, Applied Mathematics and Computation, vol. 183, pp. 634-645, 2006

[5] C. L. Chen, C. L. Chen, *Bottleneck-based Heuristics to Minimize Total Tardiness for the Flexible Flow Line with Unrelated Parallel Machines*, Computers & Industrial Engineering, In press, 2008.

[6] R. T. Moghaddam, N. Safaei, F. Sassani, *A Memetic Algorithm for the Flexible Flow Line Scheduling Problem with Processor Blocking*, Computers & Industrial Engineering, In press, 2009.

[7] T. Sawik, *Mixed Integer Programming for Scheduling Flexible Flow Lines with Limited Intermediate Buffers*, Mathematical and Computer Modeling, vol. 31, pp. 39-52, 2000.

[8] Y. Liu, I. A. Karimi, *Scheduling Multistage Batch Plants with Parallel Units and No Intermediate Storage*, Computers and Chemical Engineering, vol. 32, pp. 671-693, 2008.

[9] W. Weng, S. Fujimura, *Online Scheduling of Flexible Flow Shop Manufacturing*, 2nd International Symposium on Applied Computing and Computational Sciences (ACCS), Sanya, Hainan Island, China, 2009. In press.