International Symposium on Scheduling 2004
May 24-26, 2004, Awaji-Yumebutai, Japan

COMPOSITION AND ACQUISITION OF RULES FOR FLEXIBLE SHOP SCHEDULING PROBLEMS

Kazutoshi Sakakibara
Graduate School of Science and Technology
Kobe University
Rokkodai, Nada-ku, Kobe 657-8501, JAPAN
sakaki@al.cs.kobe-u.ac.jp

Hisashi Tamaki and Shinzo Kitamura
Faculty of Engineering
Kobe University
Rokkodai, Nada-ku, Kobe 657-8501, JAPAN
{tamaki, kitamura}@al.cs.kobe-u.ac.jp

Hajime Murao
Faculty of Cross-Cultural Studies
Kobe University
Rokkodai, Nada-ku, Kobe 657-8501, JAPAN
murao@i.cla.kobe-u.ac.jp

Abstract

In this paper, we consider to solve an extended class of flexible shop scheduling problems under the condition that information on jobs to be processed is not given beforehand, i.e., under the framework of real-time scheduling. To find a solution, we apply such a method where jobs are to be dispatched by applying a set of rules (a rule-set), and propose an approach in which rule-sets are generated and improved by using the genetics-based machine learning technique. Through some computational experiments, the effectiveness and the potential of the proposed approach are investigated.

Keywords: Rule acquisition, genetics-based machine learning, flexible shop scheduling problem, simulation.

1 Introduction

Recently, scheduling has been recognized as one of the most important issues in the planning and the operation of manufacturing systems. Extensive researches on scheduling have been reported from the theoretical as well as the practical view points. Most of them deal with the static environment, i.e., problems on the assumption that all information with respect to jobs are given beforehand (Pinedo 2002, Blazewicz *et al.* 2001).

In actual manufacturing system, not a few unexpected troubles happen, and it is important to make feasible schedules promptly once any trouble happens. Furthermore, there is no guarantee that every parameters of the problems are given beforehand. These kinds of the problems are called real-time scheduling problems, to which rule-based dispatching approaches are widely applied (Shaw *et al.* 1992, Shafaei and Brunn 1999a, 1999b). In these approaches, the jobs are dispatched just when they arrive. However it is rather difficult to acquire effective rules for the scheduling from the practical view point.

In this paper, a class of flexible shop scheduling problems is considered in the framework of real-time scheduling. That is, each shop includes multiple parallel machines in the framework of job shop scheduling problems, and there are several auxiliary restrictions which originate from the necessity of set-up processes. In our past study,we have dealt with the problems as deterministic and static ones, and have proposed a method of modeling the problem based on a mathematical programming approach, in which an integer programming method and a genetic algorithm are combined to obtain good schedules fastly (Sakakibara *et al.* 2002).

In our approach, jobs are to be dispatched in real-time using some rules, and the dispatching rules are generated and improved, in an off-line manner, by using the genetics-based machine learning (GBML) frameworks (Smith 1980, Goldberg 1989). In implementing a GBML, we use the Pitt approach, where a rule-set is represented symbolically as an individual of genetic algorithms, and the fitness of

an individual, i.e., the (global) objective function value of the problem, is calculated according to the results of some simulations using the rule-set. By extending the ways of some applications adopting this framework presented so far (Sakakibara *et al.* 2003a, Sakakibara *et al.* 2003b), in the paper, a way of applying our Pitt approach to a class of flexible shop scheduling problems are proposed. Furthermore, some computational experiments are also shown, where the effectiveness and the potential of the proposed approach are discussed.

2 Description of the Problem

There are N^{M} machines M_i $(i = 1, \ldots, N^{\mathrm{M}})$ and N^{J} jobs J_j $(j = 1, \ldots, N^{\mathrm{J}})$. Each job J_j is processed in a machine environment with N^{S} stages(i.e., shop floors); At stage F_r $(r = 1, \ldots, N^{\mathrm{S}})$, there are n_{r} identical machines M_i $(i = N_{r-1} + 1, \cdots, N_r$ where $N_r = \sum_{t=1}^{r} n_t$ $(r \geq 1), N_0 = 0)$ in parallel. Each job J_j includes a series of n_j operations O_k $(k = N_{j-1} + 1, \ldots, N_j;\ N_j = \sum_{t=1}^{j} n_t;\ N_0 = 0)$, and these n_j operations are to be processed in this order. To each operation O_k $(k = 1, \ldots, N;\ N = N_{N^{\mathrm{J}}} = \sum_{j=1}^{N^{\mathrm{J}}} n_j)$, the set of available machines $\mathcal{M}_k$ and the type δ_k are associated. The type of operation represents the kind of production, and the combination of the machine and the type determines production speed.

In a scheduling or dispatching process, the following restrictions should be taken into account.

(a) Restriction of machines: each operation can be processed on one of the fitted machines. (This is due to the mechanical structure, size, weight, etc. of the machines and the fitness of operations to machines.)

(b) Setup time: If the types of two operations which are processed successively on any machine differs, a setup time is needed between their processing. The setup time consists of two parts: the time required before processing and the time required after processing.

To describe the scheduling problem, we use the following notations and parameters. First, the basic parameters are

N^{S} : number of stages,
N^{M} : number of machines,
N^{J} : number of jobs, and
N^{O} : number of operations $(= N_{N^{\mathrm{J}}})$.

Then, the following parameters are associated with machines, jobs and operations. With each stage F_r $(r = 1, \ldots, N^{\mathrm{S}})$, the following parameters are associated:

n_r : number of machines.

With each machine M_i $(i = 1, \ldots, N^{\mathrm{M}})$, the following parameter is associated:

$\mu_{i\delta}$: time per unit product on M_i when an operation of type T_δ is processed.

With each job J_j $(j = 1, \ldots, N^{\mathrm{J}})$, the following parameters are associated:

d_j : due date,
r_j^{J} : required quantity, and

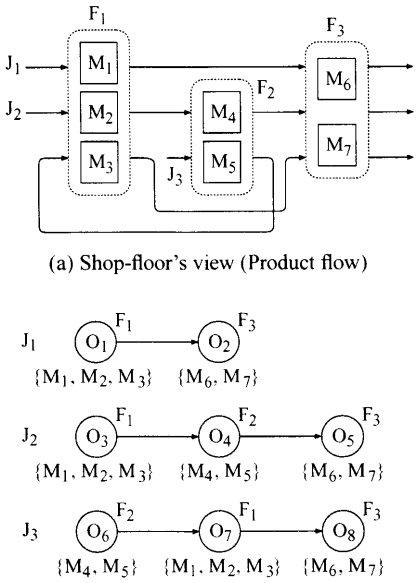

Fig. 1 An example of flexible shop problems.

n_j : number of operations.

With each operation O_k $(k = 1, \ldots, N^{\mathrm{O}})$, the following parameters are associated:

δ_k : type of O_k,
r_k^{O} : required quantity (the value of r_k^{O} is determined according to that of r_j^{J}),
s_k^1 : setup time required before processing,
s_k^2 : setup time required after processing,
$\mathcal{M}_k$: set of machines which can process O_k.

As for evaluation of schedules, various kinds of criteria may be considered. It is impossible, however, to take all of them into consideration. So, we consider here the maximum completion time $C_{\max}$ and the total tardiness T_{sum}:

$$z_1 = C_{\max} = \max_j t_{N_j}^{\mathrm{F}} = \max_k t_k^{\mathrm{F}}, \tag{1}$$

$$z_2 = T_{\mathrm{sum}} = \sum_j \left\{ \max\left(0, t_{N_j}^{\mathrm{F}} - d_j\right) \right\}, \tag{2}$$

where t_k^{F} represents the completion time of the operation O_k. Then, we use the weighted sum:

$$z = a_1 z_1 + a_2 z_2 = a_1 C_{\max} + a_2 T_{\mathrm{sum}} \tag{3}$$

as a scalar objective function, where the weights a_1 and a_2 are nonnegative.

The scheduling problem as defined above is categorized as flexible shop problems (or flexible job shop problems, generalized shop problems), which little have been studied (Blazewicz *et al.* 2001, Brucker 2001). In Fig. 1, an example of the flexible shop problems with $N^{\mathrm{M}} = 6$, $N^{\mathrm{J}} = 3$ and $N^{\mathrm{O}} = 8$ is shown.

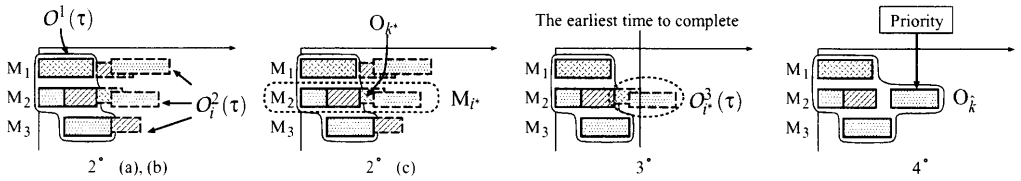

Fig. 2 Procedure for generating a schedule.

To represent a schedule, we have to determine the following items:

(i) Assignment of each operation to a certain machine,
(ii) Sequence of operations to be processed on each machine,
(iii) Start time of each operation, and
(iv) Completion time of each operation.

Here, the item (iv) is dependent on, i.e., is automatically determined by referring to, the items (i) to (iii). Moreover, the objective function of the problem considered here is a regular one, and it is known that the optimal schedule of the problems with regular objective function belongs to the set of active schedules (Pinedo 2002, Blazewicz *et al.* 2001). Under this situation, by fixing the items (i) and (ii), a feasible schedule can be uniquely determined. In the following, therefore, we consider a method of acquiring rules for determining the items (i) and (ii) efficiently, based on simulations in which the items (iii) and (iv) are calculated by using a prescribed procedure.

3 Priority Scheduling

In our approach to the scheduling problem, we use a set of rules (a rule-set) to make decisions in real-time throughout manufacturing. As mentioned in Section 2, the rule-set is used to determine the assignment of jobs to machines and the sequence of jobs on each machine. To acquire a good (possibly the best) rule-set beforehand, the GBML method is designed and applied (as described in Section 4), where several computer simulations are required for evaluating the rule-sets according to the schedules obtained by applying them. In the following, the method of computer simulations, i.e., the scheduling procedure, is described.

31 Scheduling procedure

The procedure for making a schedule of the problems is designed as follows (Fig. 2).

1° *Initialization*: Set $\tau = 0$ and $O^1(\tau) = \phi$, where τ and $O^1(\tau)$ represent the timing of dispatching and a set of operations that has been assigned, respectively.

2° *Selection of one machine for assignment*:

(a) For each M_i, first set $O_i^2(\tau) = \phi$, where $O_i^2(\tau)$ represents a set of possible operations to start processing on M_i. Then, for each operation $O_k \notin O^1(\tau)$ and for each machine which can process O_k, set $O_i^2(\tau) = O_i^2(\tau) \cup \{O_k\}$ if O_k is the first operation within a job (i.e., ${}^{\exists}J_j | k = N_j + 1$) or the operation directly preceding it has been completed (i.e., $O_{k-1} \in O^1(\tau)$).

(b) For each $O_k \in O_i^2(\tau)$, calculate the earliest time to start processing ($t_k^{S'}$) as follows:

$$t_k^{S'} = \begin{cases} \max\left(t_{k-1}^{F}, t_{\ell_i}^{F} + (s_{\ell_i}^2 + s_k^1)\Delta_{\ell_i k}\right), \\ \qquad \text{if } k \neq N_{j-1}+1, j = 1, \cdots, N^J, \\ \max\left(0, t_{\ell_i}^{F} + (s_{\ell_i}^2 + s_k^1)\Delta_{\ell_i k}\right), \\ \qquad \text{otherwise}, \end{cases} \tag{4}$$

where O_{ℓ_i} represents the currently last operation assigned on M_i, and $\Delta_{kk'}$ is a constant defined as

$$\Delta_{kk'} = \begin{cases} 0, & \text{if } \delta_k \neq \delta_{k'}, \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

(c) Among the set of $O_k \in \bigcup_i O_i^2(\tau)$, find the operation O_{k^*} and the machine M_{i^*} such that the completion time of O_{k^*} is the minimum.

3° *Determination of a set of candidate operations*: Find $O_{i^*}^3(\tau)$ ($\subseteq O_{i^*}^2(\tau)$) such that any operation O_k in $O_{i^*}^3(\tau)$ satisfies the condition $t_k^{S'} < t_{k^*}^{S'} + p_{i^*k^*}$.

4° *Selection of one operation for processing*: Find an operation $O_{\hat{k}}$ with the highest priority calculated by using a rule-set among the operations in $O_{i^*}^3(\tau)$. Then, set $t_{\hat{k}}^S = t_{\hat{k}}^{S'}$, $t_{\hat{k}}^F = t_{\hat{k}}^S + p_{i^*k^*}$, and $O^1(\tau) = O^1(\tau) \cap \{O_{\hat{k}}\}$.

5° *Termination*: If $|O^1(\tau)| = N^O$ terminate. Otherwise set $\tau = \tau^*$ where τ^* is the earliest completion time among all machines, and go to 2°.

32 Calculation of priority

In the step 4° of the scheduling procedure in Section 3, it is required to calculate the priority of operations according to the status of the production system as well as the characteristics of operations.

As for the state parameters, we introduce the four kinds of indexes (variables) to represent the whole state space $\mathcal{S}$:

s^R : ratio of the number of completed operations,
s^C : makespan of the current (partial) schedule,
s^D : average of slacks of remaining operations.

Then, $\mathcal{S}$ is divided into n^S portions $\mathcal{S}_u$ ($u = 1, \ldots, n^S$) and,

for each subset $\mathcal{S}_u$, the priority ϕ_{uk} of O_k is defined as

$$\phi_{uk} = \sum_{\ell=1}^{n^A} w_{u\ell}\, a_{k\ell} \tag{6}$$

where

n^A : number of attributes of an operation,

$w_{u\ell}$: weight of the attribute A_ℓ for $\mathcal{S}_u$, and

$a_{k\ell}$: value of A_ℓ of O_k.

As for the attribute A_ℓ of an operation O_k, we introduce the followings :

A_1 : processing time,

A_2 : number of machines which can process it,

A_3 : slack for its due-date, and

A_4 : number of succeeding operations to complete a job.

To determine the weights $w_{u\ell}$, we use a production rule of the following type :

R: if ⟨condition⟩ then ⟨action⟩.

The ⟨condition⟩ part represents the current status of the system, i.e., $\mathcal{S}_u$. Moreover, the condition of

s^F : index of the stage F_r to process an operation

is also handled in the ⟨condition⟩ part. This is for dividing the rule-set according to the stage to process an operation. While the ⟨action⟩ part possesses the (integer) weight vector $w_u = (w_{u1}, w_{u2}, w_{u3}, w_{u4})$, where the range of each weight value is $\{-n^w, \cdots, n^w\}$.

As for the acquisition of a good rule-set, i.e., the adjustment of the weight vector $\{w_u\}$, we adopt a machine learning approach which is described in Section 4.

4 Method of Rule Acquisition

As mentioned in Section 3, we adopt a scheduling approach using the priority rules. It is rather difficult, in general, to acquire the effective rule-set in heuristic manner. Here, a genetics-based machine learning (GBML) technique (Goldberg 1989, Smith 1980) is used in order to generate and improve the rule-set efficiently,

The GBML is classified into two approaches : the Michigan and the Pitt approaches. In the former approach, a rule is represented symbolically as an individual of genetic algorithms (GA), and the credit of each rule is adjusted according to the result obtained by applying a rule-set. The rule is generated and improved by applying GA at certain time intervals, where the fitness of each individual is calculated according to its credit. On the other hand, in the latter approach, a rule-set is represented symbolically as an individual of GA, and the fitness of an individual is calculated based on the result, e.g., the simulation result, by applying a rule-set. In the following, because of its easier implementation, the Pitt approach of the GBML is adopted.

In Fig. 3, we show the framework of our Pitt approach, where the outline of the learning procedure is as follows :

1° Create an initial population $\mathcal{P}(1)$ with randomly generated n_p individuals (rule-sets), where each rule-set includes n_r rules. Initialize the upper bound of generation n_g and set the current generation $t = 1$.

2° Calculate the fitness of each individual by applying an individual (rule-set).

3° If $t < n_g$, go to 4°. Otherwise go to 5°.

4° Apply genetic operators to the population $\mathcal{P}(t)$, and generate $\mathcal{P}(t+1)$. Set $t = t+1$, and go back to 2°.

5° Terminate. The best-so-far individual possesses the best rule-set.

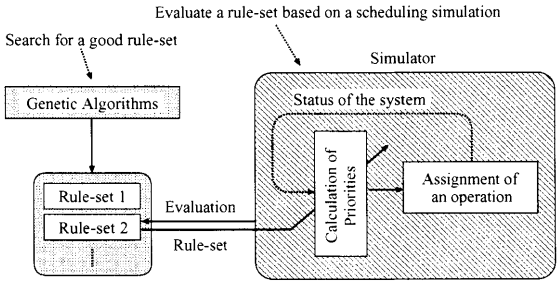

Fig. 3 Outline of the Pitt approach.

In implementing GBML, we have to prefix a genetic representation and genetic operators.

(a) *Genetic representation* : In the Pitt approach, a rule-set is represented symbolically as an individual, and includes a set of n_r rules :

$$\{R_1, R_2, \ldots, R_{n_r}\}.$$

The fitness value of an individual, which corresponds to the objective function value of the problem, is calculated by using the equation (3) based on the obtained schedule by simulating the situation in which a schedule is generated by applying each rule-set.

(b) *Genetic operators* :
In calculating the fitness value, we have to evaluate the schedule. Much computation is to be required in evaluating a solution, e.g., in solving larger-size examples. Then, the steady-state genetic algorithm (Back *et al.* 2000) is implemented in order to reduce the number of evaluations.
In our implementation of the steady-state genetic algorithms, two individuals are paired randomly in a population and a offspring is created from them by adopting genetic operators. Then, the fitness of the offspring is compared to those of the parent individuals. If the offspring is better than the worse one of the parents, the worse parent is replaced by the offspring. As for the genetic operators, the uniform crossover (rate p_c) and the one-locus-exchange mutation (rate p_m) are applied.

5 Computational Examples

We have prepared 3 types of the flexible shop problem ($\mathcal{E}^A$, $\mathcal{E}^B$ and $\mathcal{E}^C$) by setting the parameters as shown in Table 1, and 20 instances (E_i^A, E_i^B and E_i^C; $i = 1, \cdots, 20$) have been composed randomly for each type. In Table 1, indexes

Table 1 Parameters of the Problems

	N^{M}	N^{J}	η	N	(a_1, a_2)
$\mathcal{E}^{\mathrm{A}}$	6	6	18	13.6	(0.5, 0.5)
$\mathcal{E}^{\mathrm{B}}$	12	9	45	36.2	(0.5, 0.5)
$\mathcal{E}^{\mathrm{C}}$	11	70	219	190.6	(0.5, 0.5)

Table 2 Setting of the GA Parameters

Population size n_{p}	100
Number of generations n_{g}	5000
Mutation rate p_{m}	0.1 / individual
Number of rules n_{r}	100

of the gap between the due-date and the completion time of a job defined as

$$\eta = \frac{q - d^{\mathrm{AVG}}}{p^{\mathrm{AVG}}}, \tag{7}$$

where q, d^{AVG} and p^{AVG} represent the total processing time, the average due-dates and the average processing time of all operations respectively, are also shown as characterizing the tightness of the due-dates.

These examples have been solved by using two proposed methods introduced in Section 3 and Section 4, where two types of rule structure (R_1 and R_2) are implemented as follows:

R_1 : The ⟨condition⟩ part is consists of only the current status of the system, i.e., s_u.

R_2 : The ⟨condition⟩ part is consists of s_u and the condition of s^{F}.

The parameters of a rule are set as $n^S = 10^3$ and $n^{\mathrm{w}} = 4$ ($\ell = 1, \cdots, 4$), and that of the GBML are set as shown in Table 2. In the experiments, 10 trials have been executed by changing initial conditions, and then the best schedule (the schedule generated by using the acquired rule-set) has been selected for each trial.

In Table 3, the minimum, the maximum, the average and the standard deviation of the ratios among 20 instances are summarized, where the ratio ρ_i is defined, for each instance ($\mathrm{E}_i^{\mathrm{A}}$, $\mathrm{E}_i^{\mathrm{B}}$ and $\mathrm{E}_i^{\mathrm{C}}$), as

$$\rho_i = \frac{f_i^{\mathrm{GBML}}}{f_i^{\mathrm{GA}}}. \tag{8}$$

In Equation (8), f_i^{GBML} and f_i^{GA} represent the objective value of the best schedule acquired by the proposed approach, and the objective value obtained by using the GA directly to each instance $\mathrm{E}_i^{\mathrm{A}}$, $\mathrm{E}_i^{\mathrm{B}}$ and $\mathrm{E}_i^{\mathrm{C}}$, respectively.

In applying the GA, an individual is represented by using a string of the priorities of operations and a schedule is generated by using the procedure introduced in Section 3. The parameters setting of the GA is the same as shown in Table 2. In Table 3 the statistic scores of the ratios ρ^* defined as

$$\rho_i^* = \frac{f_i^{\mathrm{GBML}}}{f_i^{\mathrm{CPLEX}}}, \tag{9}$$

Table 3 Results of Computational Experiments

(a) GBML using R_1

Problem	Ratio ρ (ρ^*)			
	Min.	Max.	Avg.	Std. Dev.
$\mathcal{E}^{\mathrm{A}}$	0.96 (1.00)	1.06 (1.14)	1.01 (1.05)	0.03 (0.04)
$\mathcal{E}^{\mathrm{B}}$	0.97	1.03	0.99	0.01
$\mathcal{E}^{\mathrm{C}}$	0.96	1.04	1.00	0.02

(b) GBML using R_2

Problem	Ratio ρ (ρ^*)			
	Min.	Max.	Avg.	Std. Dev.
$\mathcal{E}^{\mathrm{A}}$	0.95 (1.00)	1.02 (1.08)	1.00 (1.04)	0.02 (0.03)
$\mathcal{E}^{\mathrm{B}}$	0.97	1.02	0.99	0.01
$\mathcal{E}^{\mathrm{C}}$	0.96	1.04	1.00	0.02

Table 4 Number of Rules Applied in Scheduling

(a) GBML using R_1

Problem	# of Rules
$\mathcal{E}^{\mathrm{A}}$	9.0
$\mathcal{E}^{\mathrm{B}}$	7.2
$\mathcal{E}^{\mathrm{C}}$	7.9

(a) GBML using R_2

Problem	# of Rules
$\mathcal{E}^{\mathrm{A}}$	12.0
$\mathcal{E}^{\mathrm{B}}$	14.2
$\mathcal{E}^{\mathrm{C}}$	18.1

where f_i^{CPLEX} represents the optimal schedule acquired by using the CPLEX (a commercial package of mathematical programming solver) (CPLEX 2001), are also shown for the class $\mathcal{E}^{\mathrm{A}}$. To investigate the obtained rule-set, the number of rules applied in scheduling is shown in Table 4.

From Table 3, we can observe that:

(a) good schedules are obtained sufficiently fast by applying the best rule-set obtained by the proposed method specially in the condition of the rule structure R_1.

From Table 4, it is observed that several rules in the best rule-set acquired by the GBML using R_1 never be applied for the scheduling. In the case of the GBML using R_2, we can observe that more rules are applied for the scheduling than the that of the GBML using R_1.

Furthermore, the robustness of the acquired rule-sets have been examined. First, rule-sets have been acquired by using the proposed GBML method where the fitness value is calculated based on the average of the objective values to newly composed 20 examples $\mathrm{E}_i^{\mathrm{LA}}$, $\mathrm{E}_i^{\mathrm{LB}}$ and $\mathrm{E}_i^{\mathrm{LC}}$ for each type, i.e., $\mathcal{E}^{\mathrm{A}}$, $\mathcal{E}^{\mathrm{B}}$ and $\mathcal{E}^{\mathrm{C}}$. Then, these rule-sets have been applied to another 20 examples $\mathrm{E}_i^{\mathrm{EA}}$, $\mathrm{E}_i^{\mathrm{EB}}$ and $\mathrm{E}_i^{\mathrm{EC}}$ for each one. In Fig. 4, ratios to the best values of $\mathrm{E}_i^{\mathrm{EA}}$, $\mathrm{E}_i^{\mathrm{EB}}$ and $\mathrm{E}_i^{\mathrm{EC}}$ are summarized where the ratio $\rho_i^\star$ is defined, for each instances, as

$$\rho_i^\star = \frac{f_i^{\mathrm{R}}}{f_i^\star}. \tag{10}$$

In Equation (10), f_i^{R} and $f_i^\star$ represent the objective value of a schedule obtained by using the acquired rule-set and that

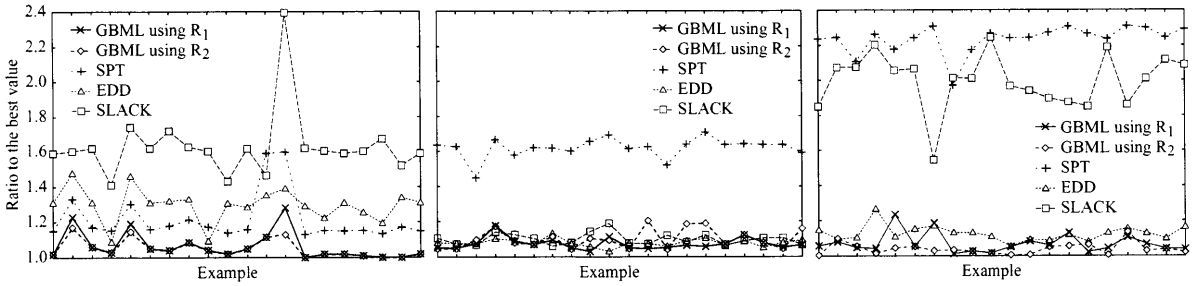

Fig. 4 Robustness of the obtained rule-set.

of the best among schedules obtained by using the GBML, the GA method and the CPLEX methods. In Fig. 4, the results obtained by using 3 kinds of typical heuristic rules, i.e., SPT (shortest processing time), EDD (earliest due date) and SLACK.

From Fig. 4, it is observed that good schedules are obtained by using SPT to the problem with small η value, and by using EDD to the problem with large η value. On the other hand, any single heuristic rule does not show good performance to 3 kinds of problems considered here. As for the best rule-sets acquired by using the GBML method,

(b) the schedules obtained by applying the best rule are at most 5% worse than the best schedule for every problem.

Thus, the results by using proposed methods indicate good robustness.

As a result of the above observations, the proposed approach is effective for finding good rule-set for real-time scheduling.

6 Conclusion

In this paper, we deal with an extended class of flexible shop scheduling problems, and consider a solution under the conditions of real-time scheduling. To find a solution, we adopt such a method in which jobs are to be dispatched by a rule-set, and effective rule-sets are to be acquired by using a GBML technique. From computational examples, we have confirmed the usefulness of our proposed method.

The following issues are still left for further studies; (i) to investigate the definition and the division form of status of the production system in order to reduce the rules never match the input status, and (ii) to extend the proposed approach to the on-line learning.

References

Bäck, T., Fogel, D. B. and Michalewicz, Z. (2000), *Evolutionary Computation 1*, Institute of Physics Publishing, Bristol.

Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G. and Weglarz, J. (2001), *Scheduling Computer and Manufacturing Processes* (2nd Ed.), Springer-Verlag, New York.

Brucker, P., *Scheduling Algorithms* (3rd Ed.), Springer-Verlag, New York.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston.

ILOG Inc. (2001), *CPLEX 8.0*, http://www.ilog.co.jp/.

Pinedo, M. (2002), *Scheduling: Theory, Algorithms, and Systems* (2nd Ed), Prentice-Hall, New Jersey.

Sakakibara, K., Tamaki, H., Murao, H. and Kitamura, S. (2002), Mathematical Modeling and Hybrid Solution for a Class of Flexible Shop Scheduling Problems, *Proc. of Int. Symp. on Scheduling 2002*, pp. 93-96.

Sakakibara, K., Tamaki, H., Murao, H., Kitamura, S., Iwatani, T. and Matsuda, K. (2003a), A Genetics-Based Machine Learning Approach for Real-Time Scheduling, *IEEJ Trans. on Electronics, Information and Systems*, Vol. 123, No. 4, pp. 823-831 (in Japanese).

Sakakibara, K., Tamaki, H., Murao, H. and Kitamura, S. (2003b), Metaheuristics Approach for Rule Acquisition in Flexible Shop Scheduling Problems, *The 5th Metaheuristics Int. Conference*, CD-ROM paper.

Shafaei, R. and Brunn, P. (1999a), Workshop Scheduling Using Practical (Inaccurate) Data Part 1 : The Performance of Heuristic Scheduling Rules in a Dynamic Job Shop Environment Using a Rolling Time Horizon Approach, *Int. J. of Production Research*, Vol. 37, No. 17, pp. 3913-3925.

Shafaei, R. and Brunn, P. (1999b), Workshop Scheduling Using Practical (Inaccurate) Data Part 2 : An Investigation of the Robustness of Scheduling Rules in a Dynamic and Stochastic Environment, *Int. J. of Production Research*, Vol. 37, No. 18, pp. 4105-4117.

Shaw, M. J., Park, S. and Raman, N. (1992), Intelligent Scheduling with Machine Learning Capabilities – The Induction of Scheduling Knowledge, *IEE Transactions*, Vol. 24, No. 2, pp. 156-168.

Smith, S. F., *A Learning System Based on Genetic Algorithms*, PhD Thesis, Univ. of Pittsburgh.