Management Science and Financial Engineering
Vol 20, No 1, May 2014, pp.1-9
ISSN 2287-2043 | EISSN 2287-2361 |
http://dx.doi.org/10.7737/MSFE.2014.20.1.001

Compromising Multiple Objectives in Production Scheduling: A Data Mining Approach

Wook-Yeon Hwang
Department of Mechanical and Industrial Engineering, Qatar University
Jong-Seok Lee*
Department of Systems Management Engineering, Sungkyunkwan University

(Received: April 4, 2014 / Revised: May 12, 2014 / Accepted: May 13, 2014)

ABSTRACT

In multi-objective scheduling problems, the objectives are usually in conflict. To obtain a satisfactory compromise and resolve the issue of NP-hardness, most existing works have suggested employing meta-heuristic methods, such as genetic algorithms. In this research, we propose a novel data-driven approach for generating a single solution that compromises multiple rules pursuing different objectives. The proposed method uses a data mining technique, namely, random forests, in order to extract the logics of several historic schedules and aggregate those. Since it involves learning predictive models, future schedules with the same previous objectives can be easily and quickly obtained by applying new production data into the models. The proposed approach is illustrated with a simulation study, where it appears to successfully produce a new solution showing balanced scheduling performances.

Keywords: Single-Machine Scheduling, Multiple Objectives, Data Mining, Random Forests

* Corresponding Author, E-mail: jongseok@skku.edu

1. INTRODUCTION

In production scheduling, there are several objectives among which a scheduler may consider only one or some at a time. Instances of such scheduling objectives to be optimized include makespan ($C_{\max}$), total weighted completion time ($\sum w_j C_j$), maximum lateness ($L_{\max}$), total tardiness ($\sum T_j$), and others. A single objective can be achieved in a reasonable time by applying an appropriate dispatching rule that prioritizes all jobs waiting for processing on a machine (Pinedo, 2005). The prioritization scheme may take into account the attributes of jobs, those of machines as well as the current time. For example, the earliest due date first (EDD) rule, which selects the job having the earliest due date to be processed next whenever a machine is available, tends to minimize the due date related objectives such as $L_{\max}$ and $\sum T_j$. On the contrary, the longest processing time first (LPT) rule is apt to balance the workload over the machines when they are in parallel, and to minimize $C_{\max}$ in a single machine environment. In practice, scheduling problems, however, often involve more than one objective and thus require a multiple criteria analysis. A difficulty with the multi-objective scheduling problems arises from that the objectives are often in conflict. For example, among the four objectives mentioned above, the two former focus on improving machine utilization and productivity that correspond to a production manager's interests, whereas the latter objectives are mainly regarded as measures of conformity with due dates that are related to customer satisfaction (Naderi *et al.*, 2010). Therefore, no single solution would simultaneously accomplish the multiple objectives and the compromise one should find as a trade-off in the conflicting circumstances.

Most machine scheduling problems with even a single objective are known to be a NP-hard problem (Pinedo, 2005). Therefore, the multi-objective scheduling problems would be in strong NP-hardness. For this reason, tradi-

tional mathematical programming methods are often infeasible to solve real field problems, and a majority of existing research works has thus focused on employing meta-heuristic techniques (Jones *et al*., 2002). Although they use different meta-heuristic algorithms, they are mainly based on an identical framework where the multiple objective functions are combined into a single one, such as in the weighted sum or utility functions; then, a meta-heuristic optimization is applied to find a near-optimal solution. Examples of using genetic algorithms (GA) include the research done by Taboada and Coit (2008), where the scheduling of a bottleneck operation from a real manufacturing line was addressed as well as the research by Arroyo and Armentano (2005), which applied a local search GA to a flowshop scheduling problem. Ahead of this, they also developed a constructive heuristic in order to generate better initial solutions for the GA (Arroyo and Armentano, 2004). The tabu search (TS) algorithm was used for a single machine problem with consideration of sequence-dependent setup times (Choobinech *et al*., 2006). Another TS combined with a variable neighborhood search was proposed by Gagne *et al*. (2005). Beginning from the work by Ulungu *et al*. (1998), the multi-objective scheduling based on simulated annealing (SA) has also made progress to date. Loukil *et al*. (2005) designed a general method, which approximates the set of all efficient schedules using the SA algorithm. They illustrated the method with several different systems ranging from single machine to flowshop problems. Based on this framework, they also conducted a real case study, which was a flexible job-shop problem with particular constraints such as batch production (Loukil *et al.*, 2007). The richness of research works in the intersection of multi-objective scheduling and meta-heuristics enables us to find more than the abovementioned.

In the meanwhile, the operations research (OR) community has expended effort on using data mining methods in order to solve traditional OR-related problems. For example, Chen *et al*. (2013) recently proposed a data mining-based system in order to solve the local pickup and delivery problem. They transformed the problem into classification and regression tasks and then employed a tree induction method to obtain quality solutions within an acceptable computational time. The production scheduling problem has also been one of the applications of data mining and machine learning algorithms. Due date assignment problems on a job-shop environment could be addressed by using C4.5 algorithm (Sha and Liu, 2005). Another category of research considering the dispatching rule selection problem via predictive modeling has been conducted over the years. Its main idea consists of three phases−generating candidate dispatching rules, evaluating the efficiency of those through simulation, and selecting the best performing rule for scheduling using machine learning algorithms such as neural networks. The research by Kutanoglu and Sabuncuoglu (2001) and that of Metan *et al*. (2010) are examples belonging to this type of study. Koonce and Tsai (2000) used a data mining technique, namely, attributed-oriented induction method, in order to find similar patterns from the survived genetic solutions for a job-shop schedule so that the resulting patterns could be used to create a final schedule. A fairly distinctive research for the generation of new dispatching rules using data mining was studied by Li and Olafsson (2005). The production data for scheduling is first engineered to a flat structure so that it can be suitable for a classification task. They showed that decision tree learning from this data can be used as a new dispatching system and it could furthermore obtain previously unknown knowledge, which is useful to improve scheduling performances. However, their methodology was restricted to single objective problems.

From our review of relevant literatures, it appears that the research of applying data mining directly to production data in order to generate a new solution has been quite limited. To the best of our knowledge, using a data mining technique for multi-objective scheduling problems has not been considered before. To this end, we propose a novel data-driven approach for generating a single solution that compromises multiple rules pursuing different objectives. The proposed method uses a data mining technique, namely, random forests, in order to extract the logics of several historic schedules and aggregate those logics. Since it involves learning predictive models, future schedules having the same previous objectives can be easily and quickly obtained by applying new production data into the models. With regard to learning predictive models from the production data, our research may be thought of as an extension of Li and Olafsson (2005)'s work toward the multi-objective problems.

The rest of this paper has the following structure. Since our method is in the common area of Li and Olafsson (2005)'s research, we will review their approach in detail by focusing on how to directly apply a data mining technique to the production data in Section 2. Section 3 describes the details of the proposed approach with an illustrative example. In Section 4, an intensive simulation study is conducted to show how well our method performs as a tool for combining multiple objectives. This section also discusses future scheduling using the trained predictive models. Section 5 provides concluding remarks and future research directions.

2. LEARNING DISPATCHING RULES

Production scheduling can be characterized by several factors−attributes and number of jobs and machines, current time, status of machines, and others. Based on such information, an algorithmic logic determines a list of scheduling elements that includes when, on which machine, which job should be processed, and so on. In view of data mining, the scheduling procedure mentioned above contains the attribute (or predictor) and the

target (or response) concept. Therefore, it may be possible to reformulate a scheduling problem into a data mining problem by appropriately engineering the production data.

We now consider a single machine case for simplicity. Denote by $\mathbf{x}_j = (x_{j1}, x_{j2}, \cdots, x_{jp})$ a set of p attributes of job j, which may include its release time (r_j), due date (d_j), weight (w_j), processing time (p_j), and so on. For a pair of two jobs i and j, we additionally define a variable y_{ij} as below, since we need to determine which job should be dispatched first.

$$y_{ij} = \begin{cases} 1, & \text{if job } i \text{ is dispatched first} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

With a proper predictor generating function $g(\mathbf{x}_i, \mathbf{x}_j)$, a production dataset consisting of n jobs is now transformed into a flat structure $\mathbf{D} = \left(g(\mathbf{x}_i, \mathbf{x}_j), y_{ij}\right),\ i < j,$ to which we can directly apply a classification tool. In Li and Olafsson (2005)'s research, a simple function concatenating two sets of job attributes was used for $g(\mathbf{x}_i, \mathbf{x}_j)$, as described below.

$$g(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i, \mathbf{x}_j) = (x_{i1}, x_{i2}, \cdots, x_{ip}, x_{j1}, x_{j2}, \cdots, x_{jp}) \quad (2)$$

However, any other sophisticated $g(\mathbf{x}_i, \mathbf{x}_j)$, which can better discriminate between the two classes (1 and 0) and extract knowledge from production data, is more recommended. An interpretable classification method such as a decision tree is then applied to **D** in order to learn what scheduling scheme derived the job sequences. Let

$$f : \left\{g(\mathbf{x}_i, \mathbf{x}_j)\right\} \rightarrow y \in \{0, 1\} \quad (3)$$

denote the trained predictive model that determines which job is dispatched first between the two jobs i and j. Model f implies that "*Existing scheduling practices are generalized into explicit scheduling rules. These rules can then be applied both to situations that have occurred before and to new scenarios,*" (Li and Olafsson, 2005).

Table 1. Dispatching List

JobID	J5	J1	J3	J4	J2
p_j	17	15	20	7	5
r_j	0	10	18	0	30

To help readers understand, we illustrated the above framework using a small example adopted from their work. The dispatching list shown in Table 1 was scheduled by the LPT rule; however, we assume that this is unknown. For every pair of two jobs i and j, we applied Equations (1) and (2) in order to construct **D**, as shown in Table 2. Note that the first two columns in Table 2 are to describe which pair of jobs is used for each row, and thus, they were not used for learning the model f.

Table 2. Engineered Dataset **D** for Data Mining

Job i	Job j	p_i	r_i	p_j	r_j	y_{ij}
J1	J2	15	10	5	30	1
J1	J3	15	10	20	18	1
J1	J4	15	10	7	0	1
J1	J5	15	10	17	0	0
J2	J3	5	30	20	18	0
J2	J4	5	30	7	0	0
J2	J5	5	30	17	0	0
J3	J4	20	18	7	0	1
J3	J5	20	18	17	0	0
J4	J5	7	0	17	0	0

By applying the well-known CART algorithm (Breiman *et al.*, 1999), we obtained the tree model, as illustrated in Figure 1 (Li and Olafsson (2005) used a different tree algorithm, C4.5, but a similar result was obtained). We can notice that the resulting tree successfully caught out the logic of the LPT rule and thus, it can be used for future scheduling according to the same logic. It states the followings: 'If $p_i < 11$, then dispatch job j first,' and 'Although $p_i \geq 11$, if $p_j \geq 12$, then dispatch job j first.' From this small and simple example, we have observed that learning from historical schedule data via data mining is a promising way to solve a production scheduling problem.

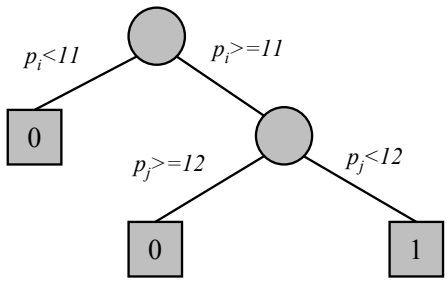


Figure 1. Decision tree learning from data in Table 2

3. DATA MINING APPROACH TO MULTI-OBJECTIVE SINGLE MACHINE SCHEDULING

As we noted in Section 2, the main benefit from applying data mining to an engineered production data is that the learned model contains the logic of how the jobs in a dispatching list were ordered. Our research begins from this simple notion that the predictive model encloses the scheduling objective. Assume that we have several dispatching lists of an identical set of jobs, where each list has been conducted with a different objective from others. Further suppose that we train a separate model for each of the dispatching lists. Then, each model contains its own scheduling objective. The

main issue for combining the multiple objectives is now to find an appropriate way to aggregate the trained models into a single model. For this purpose, we decided to employ another classification technique, namely, random forests, which will be briefly reviewed in the following subsection. We will then describe how the random forests can be used for generating a single solution pursuing multiple objectives in Section 3.2, which is followed by the subsection that provides an illustrative example with a small size of jobs. We will also discuss the way of controlling the weights of objectives in Section 3.4.

3.1 Supervised Learners: Random Forests

Random forest (Breiman, 2001), which is formed by a collection of multiple decision trees based on resampling technique, is one of the most often used meta-learning methods. Let $f_i(\mathbf{x}), i = 1, 2, \cdots, B,$ denote the ith decision tree in a random forest, where B is the total number of trees. Since each tree $f_i(\mathbf{x})$ in a forest equally casts one vote for predicting the final response, the random vectors of trees for a forest should be independently and identically distributed. Therefore, a bootstrap sample is drawn from the training data for each tree. In order to remove correlations among trees, the bootstrapped sample in fact has only m variables randomly chosen from p variables ($m << p$). Growing a tree can be conducted by a traditional tree algorithm, such as the CART algorithm. After we repeat the above procedure B times, the final prediction is determined by a majority voting scheme based on the predicted values from B trees, as denoted by Equation (4).

$$\hat{y} = majority\ vote \left\{ \hat{f}_i(\mathbf{x}) \right\}_{i=1}^{B} \tag{4}$$

Note that the above equation is equivalent to Equation (5) when 0.5 is used for the threshold value of classification, because only 1 or 0 is available for the output value (Hastie *et al.*, 2001). Hence, an object $\mathbf{x}$ will be classified into 1 if $\hat{\mathrm{H}} : \{\mathbf{x}\} > 0.5$.

$$\mathrm{H} : \{\mathbf{x}\} = \left\| \frac{1}{B} \sum_{i=1}^{B} f_i(\mathbf{x}) \right\| \tag{5}$$

3.2 Proposed Framework

Suppose that there is a set of jobs to be scheduled with multiple objectives. In this research, we further assume that each objective can be accomplished by the existing scheduling methods, and this assumption would be acceptable because numerous research have been conducted to date in order to find an optimal solution for a single objective case (Pinedo, 2005). Our proposed approach therefore begins with a set of jobs and its multiple schedules $I^{(k)}, k = 1, 2, \cdots, K,$ obtained by either the existing scheduling methods or the system experts' guidelines, where each schedule optimizes its individual objective.

We now address the multi-objective scheduling problem of finding a single solution, which is denoted by I^{new}, by aggregating the $I^{(k)}$'s. Recall that a predictive model with a single scheduling objective can be constructed. It implies that we can build multiple models that generate different sequences of jobs for an identical set of jobs. Since a decision tree produces a single output for y_{ij}, it would be difficult to determine the final output based on a few values of y_{ij} from several decision trees if they do not agree on the same value. Since a random forest consists of many decision trees, we believe that using multiple random forests for building multiple dispatching systems is a good way to aggregate the trained models. Based on this feasible idea, we now describe the proposed approach as below.

0. Let $\mathbf{X}_{\text{raw}} = \{\mathbf{x}_j\}$ denote the production data consisting of n jobs (j = 1, 2, ⋯, n), where each job j has p attributes, i.e., $\mathbf{x}_j = (x_{j1}, x_{j2}, \cdots, x_{jp})$, and $I^{(k)}$ denotes the schedule of the n jobs with the kth objective, k = 1, 2, ⋯, K.
1. Convert $\mathbf{X}_{\text{raw}}$ into $\mathbf{X}_{\text{eng}} = \left(g(\mathbf{x}_i, \mathbf{x}_j)\right), i < j,$ by an attribute generating function g.
2. Repeat for each k, k = 1, 2, ⋯, K.
   2.1 Prepare $\mathbf{y}^{(k)} = \left(y_{ij}^{(k)}\right), i < j,$ from $I^{(k)}$.
   2.2 Let $\mathbf{D}^{(k)} = \left(\mathbf{X}_{\text{eng}}, \mathbf{y}^{(k)}\right)$ denote the data for training the kth random forest.
   2.3 Using $\mathbf{D}^{(k)}$, train a random forest $H^{(k)}$ consisting of B trees $\left\{f_b^{(k)}\right\}, b = 1, 2, \cdots, B.$
3. Compute $\hat{\mathbf{y}} = \left(\hat{y}_{ij}\right)$ by
$$\hat{y}_{ij} = majority\ vote \left\{ \left\{ f_b^{(k)}\left(g(\mathbf{x}_i, \mathbf{x}_j)\right) \right\}_{b=1}^{B} \right\}_{k=1}^{K}$$
$$= \left\| \frac{1}{KB} \sum_{k=1}^{K} \sum_{b=1}^{B} f_b^{(k)}\left(g(\mathbf{x}_i, \mathbf{x}_j)\right) \right\|, i < j.$$
4. Find a new schedule I^{new} from $\hat{\mathbf{y}}$.

As mentioned earlier, Step 0 implies that our proposed method is to combine multiple rules of n jobs into a single compromise schedule. If we do not have the schedules, $I^{(k)}$'s, we may want to apply the existing single objective scheduling methods to $\mathbf{X}_{\text{raw}}$ in order to prepare those schedules. For example, if our final goal is to find a single solution that simultaneously minimizes $\sum T_j$ and $C_{\max}$, in other words, a single solution that compromises between the objectives $\sum T_j$ and $C_{\max}$, we can apply the EDD and LPT rules to our production data before going through the remaining steps. As noted before, Step 1 is to convert the production data into a flat form that is suitable for a classification task. In Step 2, we train K random forests using an identical set of predictors and K different response variables. Although, in this research, Step 1 through Step 2.2 are described for the case of a single machine environment, properly modifying these steps will be able to extend the proposed framework into the other scheduling models such

as job shop problems. However, we do not give our attention to this as it is out of our research scope. As a step for determining the final value of y_{ij}, Step 3 decides which job should be dispatched first by the majority voting scheme. It is worth noting that we aggregate all $f_b^{(k)}$'s for the final $\hat{y}_{ij}$, unlike a usual random forest that does so at each forest level. Hence, job i would be dispatched earlier than job j, if more than half of KB trees return a value of $f_b^{(k)} = 1$. This means that although dispatching job j first helps to optimize a specific objective, we sacrifice this for improving other conflicting objectives. Once the sequences of all pairs of jobs are determined, we can obtain a new schedule I^{new} by decoding them in Step 4.

3.3 Numerical Example

A small scheduling problem consisting of 10 jobs will be considered in this subsection in order to illustrate the proposed framework. Production information of the 10 jobs in terms of the release time (r_j), due date (d_j), weight (w_j), and processing time (p_j) is given in Table 3.

Table 3. Data for Production Scheduling

JobID	Release Time (r_j)	Due Date (d_j)	Weight (w_j)	Processing Time (p_j)
J1	9	61	3	11
J2	0	39	4	19
J3	4	35	3	3
J4	2	80	1	9
J5	7	86	5	19
J6	5	79	1	18
J7	5	55	3	16
J8	1	46	5	3
J9	7	68	2	8
J10	6	67	1	3

From this example, we attempt to find a compromise sequence of the jobs that simultaneously minimizes total tardiness ($\sum T_j$), total weighted completion time ($\sum w_j C_j$), and makespan ($C_{\max}$), which are known to be in conflict for achievement at the same time. To obtain three $I^{(k)}$'s, we first apply the EDD, the WSPT, and the LPT rule to the given dataset. The resulting sequences of the jobs and the corresponding performances are shown in Table 4. As expected, the job sequence by the EDD rule shows best performance on the $\sum T_j$, and the WSPT and the LPT rule generated solutions that have the best performance on the $\sum w_j C_j$ and the $C_{\max}$, respectively. Due to the size of the problem, there is no big difference in the $C_{\max}$ performance among the three schedules. Using these job sequences and the data in Table 3, we attempted to obtain another solution by applying the proposed method. We therefore trained three random forests based on the transformed datasets $\mathbf{D}^{(k)}$, $k = 1, 2, 3$, where each dataset consists of 45 instances and 9 variables including the class variable. The number of trees (B) in each forest was set to 100, meaning that the value of y_{ij} is determined by aggregating 300 predictions in Step 3 of the proposed method. The classification result was then converted into the job sequence, which is shown on the last row of Table 4.

As can be seen from the table, the proposed approach appeared to successfully generate a solution showing balanced performances. Although the proposed method showed the worse total tardiness than the EDD rule as expected, it was evidently better than the WSPT and the LPT rule. Similarly, the total weighted completion time by our method is smaller than that of the EDD and of the LPT rule, whereas the WSPT rule showed the best performance. The proposed method also showed the best value in the makespan. Despite its small size and simplicity, this example clearly demonstrates that our data mining approach can be used to compromise among multiple objectives in production scheduling.

3.4 Controlling Weights of Objectives

The majority voting scheme in Step 3 of the proposed method implies that all predicted values from KB trees equally contribute to the final value of $\hat{y}_{ij}$. This also means that the K objectives are considered equally important. In practice, a system operator, however, may want to give different weights on the scheduling objectives. For example, in semiconductor manufacturing, the primary goal is to maximize the throughput of the facility, while responding promptly to the customer demands is considered a bit less important (Gupta and Sivakumar, 2005). To ensure that the proposed method can be adjusted to such problems, namely, multi-objective scheduling problems with different objective weights, we attempted to modify Step 3 using Equation (6) defined as below:

$$\hat{y}_{ij} = \left\| \sum_{k=1}^{K} w_k \cdot \frac{1}{B} \sum_{b=1}^{B} f_b^{(k)}\left(g\left(\mathbf{x}_i, \mathbf{x}_j\right)\right) \right\| \qquad (6)$$

where $\sum_{k=1}^{K} w_k = 1$. Since we train K random forests and aggregate the KB predictions at a time, to assign different weights to the objectives, we apply a specific weight w_k to the kth set of B predictions before the aggregation. Notice that Equation (6) is equivalent to the original one when $w_1 = w_2 = \cdots = w_K = \frac{1}{K}$.

To demonstrate how Equation (6) works, we conducted a simulation study using the example of 10 jobs in the previous subsection. The two objectives, the total tardiness and the total weighted completion time, which are evidently in conflict, were under our consideration. We denoted w_1 and w_2 by the weights of the former and the latter, respectively. Increasing the value of w_1 from 0

Table 4. Results of the Example

Dispatching system	Job sequence	Performance		
		ΣT_j	$\Sigma w_j C_j$	$C_{\max}$
EDD	J2-J3-J8-J7-J1-J10-J9-J6-J4-J5	35	1443	109
WSPT	J8-J3-J10-J1-J5-J9-J2-J7-J4-J6	99	1129	110
LPT	J2-J5-J6-J7-J1-J4-J9-J3-J8-J10	253	2027	109
Proposed	J2-J3-J8-J1-J9-J10-J5-J7-J6-J4	77	1295	109

Table 5. Job Sequences on Varying Weights

w_1	w_2	Job sequence	ΣT_j	$\Sigma w_j C_j$
0.00 ~ 0.30	1.00 ~ 0.70	J8-J3-J10-J1-J5-J9-J2-J7-J4-J6	99	1129
0.35	0.65	J8-J3-J10-J1-J9-J2-J5-J7-J6-J4	89	1159
0.40	0.60	J8-J3-J10-J1-J2-J5-J7-J9-J6-J4	88	1171
0.45	0.55	J3-J8-J10-J1-J2-J7-J9-J5-J6-J4	66	1286
0.50	0.50	J3-J8-J2-J10-J1-J7-J9-J5-J6-J4	62	1306
0.55	0.45	J8-J2-J3-J1-J7-J10-J9-J5-J6-J4	52	1270
0.60 ~ 0.65	0.40 ~ 0.35	J2-J3-J8-J7-J1-J10-J9-J6-J5-J4	45	1417
0.70 ~ 1.00	0.30 ~ 0.00	J2-J3-J8-J7-J1-J10-J9-J6-J4-J5	35	1443

to 1 (decreasing the value of w_2 from 1 to 0) obtained different job sequences, as presented in Table 5. Notice that the first and the last row of the table correspond to the second and first row of Table 4 individually. The objective values (performance measures) of each schedule are shown in the last two columns of the table and are also depicted in Figure 2. As the value of w_1 increases, the total tardiness tends to decrease, whereas the total weighted completion time increases. Specifically, by assigning a bigger weight to the total tardiness objective, we could obtain a solution showing better performance on it and vice versa. This example shows that the proposed approach has the capability of controlling the weights of the objectives. From the set of candidate solutions obtained by varying the weights, the most preferable one to the system can be selected.

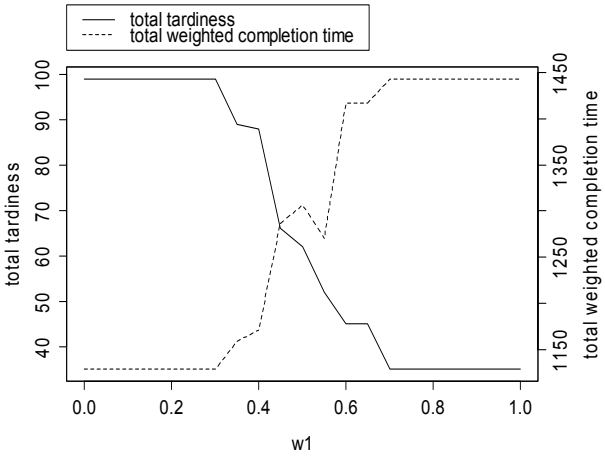


Figure 2. $\sum T_j$ and $\sum w_j C_j$ on varying w_1

4. SIMULATION STUDY

To generalize the results of our case study introduced in Section 3, we also examined our proposed method through an intensive simulation study consisting of two parts. The first set of experiments was aimed at generalizing the proposed approach to a larger size of scheduling problems and analyzing how much the resulting solutions are compromised among multiple objectives. We then considered the training and testing concept of the predictive models. The second part of the experiments was conducted in order to examine the ability of the trained random forests to schedule new jobs with the same previous objectives.

4.1 Scheduling with Two Objectives

As also considered in Section 3, the clearly conflicting two objectives, total tardiness and total weighted completion time, were chosen to be compromised in this section. We set the number of jobs n = 50, meaning 1225 instances in the classification task, and considered four attributes of each job j for scheduling, which are the release time (r_j), due date (d_j), weight (w_j), and processing time (p_j). They were randomly generated using the following probabilistic patterns.

Release time (r_j, j = 1, 2, ⋯, n)~*Uniform* [0, 10]
Processing time (p_j, j = 1, 2, ⋯, n)~*Uniform* [1, 20]

Due date (d_j, $j = 1, 2, \cdots, n$)~$r_j + \sum_{j=1}^{n} \frac{p_j}{2}$
$+ Uniform\left[-\sum_{j=1}^{n} \frac{p_j}{4}, \sum_{j=1}^{n} \frac{p_j}{4} \right]$

Weight (w_j, $j = 1, 2, \cdots, n$)~$Uniform$ [1, 5]

The EDD and the WSPT rule were accordingly selected as competitors of the proposed method in order to analyze how much compromise can be acquired by our method compared to those. Hence, their performances on the chosen objectives were used as baselines to observe how much improvement can be achieved by the proposed method on each objective. We repeated the scheduling experiment 10 times and the results are summarized in Table 6.

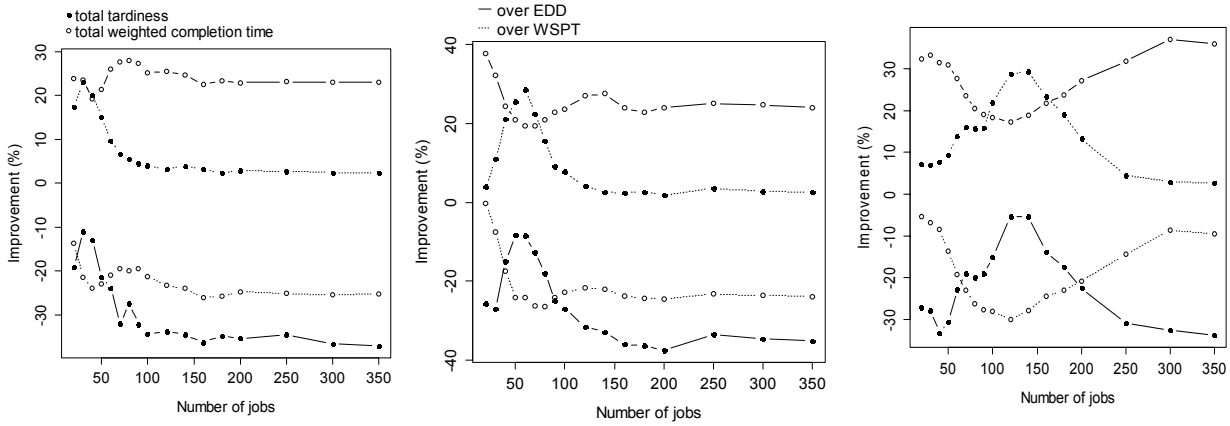
Similarly to the results in Section 3, the proposed approach appeared to successfully schedule the jobs for compromising between the two objectives. As expected, the EDD rule showed the best performance in total tardiness whereas the WSPT did in total weighted completion time. The performances of the schedules obtained by the proposed method appeared in the middle of those by the EDD and the WSPT on both objectives, which means that they are in a balanced manner between the two objectives. From the last four columns of the table, we can notice that the proposed method improved 26.96% on average over the WSPT rule in total tardiness and 18.92% on average over the EDD rule in total weighted completion time. As we can also expect, those improvements sacrificed the opposite objectives individually.

4.2 Learning Schedulers

We mentioned earlier that since the proposed approach involves training predictive models, which are multiple random forests $H^{(k)}$'s, future scheduling with the same previous objectives can be easily and quickly done by applying the new production data into the models. In

Table 6. Performance Comparison of the Proposed Method with EDD and WSPT

Replication	Performance Measurement						% improvement of proposed method			
	ΣT_j			$\Sigma w_j C_j$			in ΣT_j		in $\Sigma w_j C_j$	
	EDD	WSPT	Proposed	EDD	WSPT	Proposed	over EDD	over WSPT	over EDD	over WSPT
1	1338	2467	1734	41347	23284	30999	-22.84%	29.71%	25.03%	-24.89%
2	2076	3121	2373	43607	28161	34979	-12.52%	23.97%	19.79%	-19.49%
3	1804	2668	1874	43210	25741	34138	-3.74%	29.76%	21.00%	-24.60%
4	1011	2043	1568	39485	25876	32692	-35.52%	23.25%	17.20%	-20.85%
5	1373	2172	1721	32783	18382	26518	-20.22%	20.76%	19.11%	-30.68%
6	1034	2160	1463	37599	24877	32304	-29.32%	32.27%	14.08%	-22.99%
7	2022	3346	2626	44009	26031	35268	-23.00%	21.52%	19.86%	-26.19%
8	1524	2523	1852	41010	26588	35507	-17.71%	26.60%	13.42%	-25.12%
9	806	1878	1271	29344	18045	22594	-36.59%	32.32%	23.00%	-20.13%
10	1526	2617	1847	39394	25232	32792	-17.38%	29.42%	16.76%	-23.05%
						Average	-21.88%	26.96%	18.92%	-23.80%
						Standard deviation	9.58%	4.11%	3.48%	3.14%



(a) Train Size: 30 Jobs (b) Train Size: 50 Jobs (c) Train Size: 100 Jobs

Figure 3. Results of Applying Test Datasets to Trained Models

this subsection, we attempted to examine this capability through a simulation study.

The experimental settings are the same as in Section 4.1, except for the addition of the testing concept. The production data for both training and testing was generated from the job attribute distributions described in Section 4.1. The two objectives, total tardiness and total weighted completion time, were also considered in this experiment. The predictive models, which are two random forests, were learned based on two sets of responses generated by the EDD and the WSPT rule, respectively. By applying a new dataset to the learned models, we obtained the job sequence of the new data, and then compared its performances with those of the solutions from the EDD and WSPT rule in terms of the percentage improvement in each of the objectives over other rules, as done in Section 4.1. To observe if the size of training and testing sets affects the scheduling performances, we considered three different sizes (30 jobs, 50 jobs and 100 jobs) for training and a wider range of testing size (from 20 jobs to 350 jobs). We repeated the experiment 100 times for each testing size. The whole results were summarized as graphs in Figure 3.

Notice that the solid and the hollow circles indicate performance measurements and the solid and the dotted lines correspond to the dispatching rules compared with the proposed method. The graphs can therefore be read by their combination. For example, the solid line with the hollow circles represents the percentage improvement of the proposed method in the total weighted completion time over the EDD rule. Since the experiment was repeated 100 times for each test size, we reported the average value in the figure.

The overall appearance in this simulation study is that the trained models could still generate the compromise solutions between the two objectives for the new datasets. Similar to the results in Section 4.1, about 20~30% performance improvement on each objective was obtained. Accordingly, its corresponding negative improvement also occurred in the process of compromising the objectives. It seems that the training size somehow affects the quality of the solution for the testing set. As the size difference between training and testing increases, the agreement between the two objectives tends to be biased to one of those, suggesting that it is better to apply a new dataset in a similar size with the training size. Nonetheless, some allowable tolerance seems to exist. For example, in Figure 3(c), the trained scheduler from 100 jobs generated balanced performance improvements for a testing size ranging from 50 to 250.

5. CONCLUSIONS

In this research, we proposed a novel data mining-based approach to the multi-objective single-machine scheduling problems, which employs the random forests. From the notion that a trained model based on a well-engineered production data contains the scheduling objective, we proposed to employ a meta-learning algorithm in order to extend the concept of learning a historic schedule to the multi-objective problems. By training multiple random forests and aggregating their predictions at a time, the proposed method finds the job schedule, which is in agreement between multiple objectives. We also showed that the proposed approach has the capability to control the weights of objectives. Using the training and testing concepts, the trained predictive models, in other words, the learned artificial schedulers can easily perform the scheduling task for the new data. Although our method is described for a single machine environment in this research, we believe that the main idea of the proposed framework, which is to train multiple models and aggregate multiple predictions, will be valid for other scheduling problems by modifying the steps for engineering production data. We consider this as our future research direction in the realm of data mining and scheduling. Since in practice a scheduler is more likely to consider multiple objectives rather than a single one, we believe that the proposed method could be a useful tool in production scheduling.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A1012153).

REFERENCES

Arroyo, J. E. C. and V. A. Armentano, "A partial enumeration heuristic for multi-objective flowshop scheduling problems," *Journal of the Operational Research Society* 55, 9 (2004), 1000-1007.

Arroyo, J. E. C. and V. A. Armentano, "Genetic local search for multi-objective flowshop scheduling problems," *European Journal of Operational Research* 167, 3 (2005), 717-738.

Breiman, L., "Random forests," *Machine Learning* 45, 1 (2001), 5-32.

Breiman, L., J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, CRC Press, New York, 1999.

Chen, W., J. Song, L. Shi, L. Pi, and P. Sun, "Data mining-based dispatching system for solving the local pickup and delivery problem," *Annals of Operations Research* 203, 1 (2013), 351-370.

Choobinech, F. F., E. Mohebbi, and H. Khoo, "A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times,"

European Journal of Operational Research 175, 1 (2006), 318-337.

Gagne, C., M. Gravel, and W. L. Price, "Using metaheuristic compromise programming for the solution of multi-objective scheduling problem," *Journal of the Operational Research Society* 56, 6 (2005), 687-698.

Gupta, A. K. and A. I. Sivakumar, "Single machine scheduling with multiple objectives in semiconductor manufacturing," *International Journal of Advanced Manufacturing Technology* 26, 9/10 (2005), 950-958.

Hastie, T., R. Tibsharani, and J. Friedman, *The Elements of Statistical Learning*, Springer, New York, 2001.

Jones, D. F., S. K. Mirrazavi, and M. Tamiz, "Multi-objective meta-heuristic: An overview of the current state-of-the-art," *European Journal of Operational Research* 137, 1 (2002), 1-9.

Koonce, D. A. and S. C. Tsai, "Using data mining to find patterns in genetic algorithm solutions to a job shop schedule," *Computers and Industrial Engineering* 38, 3 (2000), 361-374.

Kutanoglu, E. and I. Sabuncuoglu, "Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop," *Journal of Manufacturing Systems* 20, 4 (2001), 264-279.

Li, X. and S. Olafsson, "Discovering dispatching rules using data mining," *Journal of Scheduling* 8, 6 (2005), 515-527.

Loukil, T., J. Teghem, and P. Fortemps, "A multi-objective production scheduling case study solved by simulated annealing," *European Journal of Operational Research* 179, 3 (2007), 709-722.

Loukil, T., J. Teghem, and D. Tuyttens, "Solving multi-objective production scheduling problems using metaheuristics," *European Journal of Operational Research* 161, 1 (2005), 42-61.

Metan, G., I. Sabuncuoglu, and H. Pierreval, "Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining," *International Journal of Production Research* 48, 23 (2010), 6909-6938.

Naderi, B., R. Tavakkoli-Moghaddam, and M. Khalili, "Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan," *Knowledge-Based Systems* 23, 2 (2010), 77-85.

Pinedo, M., *Planning and Scheduling in Manufacturing and Services*, Springer, New York, 2005.

Sha, D. Y. and C. H. Liu, "Using data mining for due date assignment in a dynamic job shop environment," *International Journal of Advanced Manufacturing Technology* 25, 11/12 (2005), 1164-1174.

Taboada, H. A. and D. W. Coit, "Multi-objective scheduling problems: Determination of pruned Pareto sets," *IIE Transactions* 40, 5 (2008), 552-564.

Ulungu, E. L., J. Teghem, and C. Ost, "Efficiency of interactive multi-objective simulated annealing through a case study," *Journal of the Operational Research Society* 49, 10 (1998), 1044-1050.