Management Science and Financial Engineering
Vol 19, No 1, May 2013, pp.29-36
ISSN 2287-2043 | EISSN 2287-2361 |
http://dx.doi.org/10.7737/MSFE.2013.19.1.029

Common Due-Date Assignment and Scheduling on Parallel Machines with Sequence-Dependent Setup Times

Jun-Gyu Kim
Department of Industrial Engineering, Hanyang University
Jae-Min Yu
Department of Industrial Engineering, Hanyang University
Dong-Ho Lee*
Department of Industrial Engineering and Graduate School of Technology and Innovation Management, Hanyang University

(Received: December 18, 2012 / Revised: January 15, 2013 / Accepted: January 18, 2013)

ABSTRACT

This paper considers common due-date assignment and scheduling on parallel machines. The main decisions are: (a) deter-mining the common due-date; (b) allocating jobs to machines; and (c) sequencing the jobs assigned to each machine. The objective is to minimize the sum of the penalties associated with common due-date assignment, earliness and tardiness. As an extension of the existing studies on the problem, we consider sequence-dependent setup times that depend on the type of job just completed and on the job to be processed. The sequence-dependent setups, commonly found in various manufacturing systems, make the problem much more complicated. To represent the problem more clearly, a mixed integer programming model is suggested, and due to the complexity of the problem, two heuristics, one with individual sequence-dependent setup times and the other with aggregated sequence-dependent setup times, are suggested after analyzing the characteristics of the problem. Computational experiments were done on a number of test instances and the results are reported.

Keywords: Parallel Machines, Due-Date Assignment, Scheduling, Sequence-Dependent Setup Times, Heuristics

* Corresponding Author, E-mail: leman@hanyang.ac.kr

1. INTRODUCTION

Due-date assignment and scheduling have been received considerable attention in the last few decades due to the introduction of the just-in-time (JIT) concept. The JIT systems work in such a way that jobs are to be completed neither too early nor too late with respect to their due-dates. Also, they consider the due-date as a controllable variable, i.e., due-dates can be set by negotiation.

Assigning due-dates has a certain practical implication when a company offers due-dates to its customers during sale negotiations or offers a price reduction when the due-date is far away from the expected one. In fact, we can see many situations where due-dates are negotiated rather than simply set by customers. Here, the earlier the due-dates are set, the higher the probability of the loss of customer goodwill since the products may not be completed or delivered on time. On the other hand, the later the due-dates are set, the higher the probability of having high inventory due to the early completions of products. Therefore, due-date assignment is one of important practical decisions when considering customer orders.

Among various due-date assignment and scheduling problems, this paper focuses on the problem on parallel machines. The problem is to determine the common due-date as well as the allocation of jobs to parallel machines and the sequence of the jobs assigned to each machine. In general, prescribing a common due-date might represent a situation in which several items constitute a single customer order or reflect an assembly environment in which components should all be ready at the same time in order to avoid staging delays (Baker and Scudder, 1990).

There are a number of previous research articles on common due-date assignment and sequencing on a single machine. As a pioneering research, Panwalkar *et al.* (1982) consider the problem that minimizes the sum of the penalties associated with assigning common due-date, earliness and tardiness and suggest an optimal algorithm in which the common due-date is set in advance using the preliminary analysis and then each job is sequenced based on the weight value corresponding to each position. From this original work, there have been a number of research articles on the basic single machine problem (Baker and Scudder, 1989; Cheng, 1986) and its extensions (Biskup and Jahnke, 2001; Chen, 1996; Cheng and Kovalyov, 1996; Cheng *et al.*, 2002; Cheng, 1990; Dvir and George, 2006; Hall, 1986; Kim and Lee, 2009; Kim *et al.*, 2012; Ng *et al.*, 2003; Quaddus, 1987; Xia *et al.*, 2008). See Gordon *et al.* (2002) for a literature review on various common due-date assignment and scheduling problems.

Compared with those on a single machine, not much research has been done for the problem on parallel machines. Cheng (Cheng, 1989) considers the problem in which each machine must begin processing at time zero, i.e., no idle time is allowed, and suggests a list scheduling heuristic that minimizes the sum of the penalties associated with common due-date assignment, earliness and tardiness after generalizing the common due-date setting method of Panwalkar *et al.* (1982). Here, the list scheduling heuristic is similar to the earlier algorithms of Sundararaghavan and Ahmed (1984), Hall (1986) and Emmons (Emmons, 1987). Also, De *et al.* (1991) consider the generalized problem without the zero start time constraint and characterize the necessary conditions for the optimal schedules, and later, De *et al.* (1994) suggest a pseudo-polynomial time algorithm whose complexity depends on the number of parallel machines after proving that the generalized problem is NP-hard. Diamond and Cheng (Diamond and Cheng, 2000) develop another heuristic for the generalized problem without the zero start time constraint and show that it is asymptotically optimal as the number of jobs approaches to the infinity. Xiao and Li (Xiao and Li, 2002) develop a heuristic for the generalized problem without the zero start time constraint and prove its absolute worst case error bound, and Min and Cheng (Min and Cheng, 2006) suggest a genetic algorithm for the generalized common due-date assignment and scheduling problem. Recently, Kim and Lee (2012) improved the algorithms of Xiao and Li (2002) and Min and Cheng (2006), respectively.

As an extension of the previous articles, we consider the parallel machine problem with sequence-dependent setups that depend on the type of job just completed and on the job to be processed. The objective is to minimize the sum of the penalties associated with assigning the common due-date, earliness and tardiness. The sequence-dependent setup times, which make the scheduling problems much more complicated, are commonly found in various manufacturing systems. For example, in the injection molding process, only the mold change time is required if the same material is used between two consecutive jobs, while the screw cleaning times are required when there is a change of material (Kim *et al.*, 2007). To the best of the authors' knowledge, there is no previous research on common due-date assignment and scheduling on parallel machines with sequence-dependent setup times.

As stated earlier, the problem considered here has three decision variables: (a) determining the common-due-date; (b) allocating jobs to parallel machines; and (c) sequencing the jobs assigned to each machine. To represent the problem mathematically, a mixed integer programming model is suggested in this study. Then, due to the problem complexity, two heuristics, one with individual sequence-dependent setup times and the other with aggregated sequence-dependent setup times, are suggested after analyzing the characteristics of the problem. Computational experiments were done on randomly generated test instances, and the results are reported.

This paper is organized as follows. In the next section, the problem is described in more detail with a mixed integer programming model. Section 3 presents the heuristic algorithms and computational results are reported in Section 4. Finally, Section 5 gives concluding remarks and discussion of future research.

2. PROBLEM DESCRIPTION

Before describing the problem in more details, we present a general structure of parallel machine systems in Figure 1, where j_i and k_i imply the i-th job and the machine in which the i-th job is processed, respectively. Note that each job has a single operation to be performed on one of parallel machines. In general, the ordinary parallel machine scheduling problem has two decision variables: (a) allocating jobs to parallel machines (denoted by $j_i \,||\, k_i$ in Figure 1); and (b) sequencing the jobs assigned to each machine. After the job allocations are done, the resulting problem can be decomposed into single machine scheduling problems.

The problem considered in this paper can be briefly described as follows: *for a given set of jobs, the problem is to determine the common due-date, the allocation of jobs to machines and the sequence of the jobs assigned* *to each machine while considering the sequence-dependent setup times for the objective of minimizing the penalties associated with assigning the common due-date, earliness and tardiness.* It is assumed that the sequence-dependent setup time, depending on the type of job just completed and on the job to be processed, is deterministic and given for each pair of jobs. Also, the earliness (tardiness) penalty in the objective function is assumed to be directly proportional to the amount of earliness (tardiness).

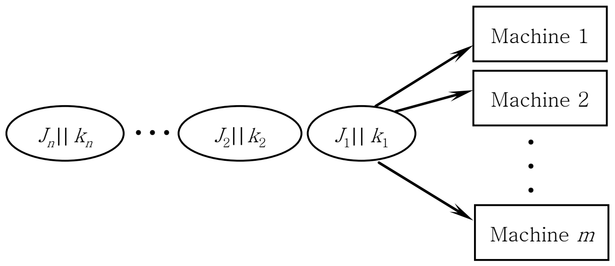


Figure 1. Structure of the Parallel Machine Process

We consider a static and deterministic version of the problem. That is, all jobs are ready for processing at time zero, i.e., zero ready times. Also, it is assumed that the job descriptors, such as processing times, setup times, and penalties, etc., are deterministic and given in advance. Other assumptions made for the problem are summarized as follows: (a) each machine can process only one job at a time; (b) once a job is determined to be processed on a machine, it will stay on the machine until its completion, i.e., no job preemption; and (c) idle times caused by machine breakdowns are not considered.

To describe the problem more clearly, a mixed integer programming model is suggested. Note that the model extends that of Balakrishnan *et al.* (1999) by additionally representing the common due-date assignment. The following notations are used in the formulation.

Parameters

t_i	the processing time of job i, $i = 1, 2, \cdots, n$
s_{ij}	the setup time required between two consecutive jobs i and j
P_1	the penalty associated with assigning the common due-date
P_2	the penalty associated with earliness
P_3	the penalty associated with tardiness
L	a large number

Decision variables

d	the common due-date
x_{ik}	= 1 if job i is assigned to machine k, and 0 otherwise
y_{ij}	= 1 if job i directly precedes job j on the same machine, and 0 otherwise
C_i	the completion time of job i
E_i	the earliness of job i, i.e., $\max\{0, d\text{-}C_i\}$
T_i	the tardiness of job i, i.e., $\max\{0, C_i\text{-}d\}$

Now, the mixed integer programming model is given below.

[P] Minimize $P_1 \cdot d \cdot n + \sum_{i=1}^{n}\{P_2 \cdot E_i + P_3 \cdot T_i\}$

$$C_i - d = T_i - E_i \qquad \text{for all } i \qquad (1)$$

$$C_i - C_j + L \cdot (2 + y_{ij} - x_{ik} - x_{jk}) \ge t_i + s_{ji} \qquad \text{for all } i, j \text{ and } k \qquad (2)$$

$$C_j - C_i + L \cdot (3 - y_{ij} - x_{ik} - x_{jk}) \ge t_j + s_{ij} \qquad \text{for all } i, j \text{ and } k \qquad (3)$$

$$x_{ik} + \sum_{k^* \ne k} x_{jk^*} + y_{ij} \le 2 \qquad \text{for all } i, j \text{ and } k \qquad (4)$$

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \text{for all } i \qquad (5)$$

$$C_i \ge t_i + s_{0i} \qquad \text{for all } i \qquad (6)$$

$$x_{ik} \in \{0, 1\} \qquad \text{for all } i \text{ and } k \qquad (7)$$

$$y_{ij} \in \{0, 1\} \qquad \text{for all } i \text{ and } j \qquad (8)$$

$$d \ge 0 \qquad (9)$$

$$E_i \ge 0 \qquad \text{for all } i \qquad (10)$$

$$T_i \ge 0 \qquad \text{for all } i \qquad (11)$$

The objective function denotes the sum of the penalties that depend on the common due-date and the completion time of each job. Constraint (1) specifies the amounts of earliness and tardiness while ensuring that T_i and E_i cannot be positive at the same time. Constraints (2) and (3) establish the relationship between the completion times of jobs i and j assigned to the same machine. Using the binary variables y_{ij} and the large number L, these constraints enforce that there is sufficient time between the completions of jobs i and j, based on the order of jobs. Constraint (4) ensures that the job precedence between jobs i and j is relevant only if both jobs are assigned to the same machine. In other words, y_{ij} is equal to zero (implying job j before job i) or one (implying job i before job j) if both jobs i and j are assigned to the same machine, while it must equal to zero if these jobs are assigned to different machines. Constraint (5) implies that each job must be assigned to one machine and constraint (6) specifies the minimum completion time of each job, i.e., completion time of an arbitrary job should be larger than or equal to the sum of its processing time and the initial setup time. Finally, constraints (7), (8), (9), (10) and (11) represent the conditions of the decision variables.

The optimal solutions can be obtained by solving the model (P) directly using a commercial integer programming software package. However, it is not practical because of excessive computation time. We can easily see that the problem (P) is NP-hard since the problem (P) is the generalization of the parallel machine problem without the sequence-dependent setup times. Note that the special case is already known to be NP-hard (De *et al.*, 1994).

3. SOLUTION ALGORITHMS

This section explains the two heuristics suggested in this study, one with individual and the other with aggregated sequence-dependent setup times. Before presenting the algorithms, we first explain the method to find the optimal common due-date.

3.1 Setting the Common Due-Date

The two propositions, adopted from Panwalkar *et al.* (1982) for the single machine problem, can be extended to the parallel machine problem after the jobs are sorted according to their completion times for a given job schedule on parallel machines. Also, they are valid for the single machine problem with sequence-dependent setup times (Kim and Lee, 2009).

Proposition 1 implies that the common due-date must coincide with the completion time of a job in a given job sequence. In the proposition, the job sequence S is specified by sorting the jobs in the non-decreasing order of their completion times. The proof is omitted here since they are straightforward even for the problem on parallel machines with sequence-dependent setup times.

Proposition 1: *For any specified job sequence S on parallel machines, there exists an optimal common due-date d that coincides with the completion time of one of the jobs in S.*

By differentiating the objective function with respect to d and setting it equal to zero, we can obtain the optimal common due-date. The detailed method is given in Proposition 2. In the proposition, $[k]$ denotes the index of the k-th job after sorting the jobs in the non-decreasing order of their completion times.

Proposition 2: *For any specified job sequence S, the optimal common due-date is equal to $C_{[k]}$, where k is the smallest integral value greater than or equal to $n \cdot (P_3 - P_1)/(P_2 + P_3)$, where P_1, P_2 and P_3 are the penalties associated with due-date assignment, earliness and tardiness, respectively.*

3.2 Heuristic Algorithms

Based on the method to set the common due-date, we suggest two heuristics that consist of two phases: obtaining an initial solution and improvement. Here, the two heuristics are different in the method to obtain the initial solution.

3.2.1 Individual Setup Heuristic

This heuristic, denoted by individual setup (IS) heuristic in this paper, obtains the initial solution by assigning jobs to the positions according to positional weights. Here, the positional weights are calculated using the idea of Diamond and Cheng (2000) that consider the parallel machine problem without sequence-dependent setup times. More formally, two weight values according to the characteristic of the h-th position on machine k are calculated as follows (See Diamond and Cheng (1994) for more details).

$$\lambda_{kh} = \begin{cases} \dfrac{nP_1}{m} + (h-1)\cdot P_2 & \text{if the position } (k, h) \text{ is early} \\ h \cdot P_3 & \text{otherwise,} \end{cases}$$

where the position (h, k) is called early if the completion time of the job assigned to the position is less than the common due-date.

After calculating the positional weights, the first job i_1^* is selected with the following condition

$$i_1^* = \arg\min_{i\in U}\left\{\arg\min_{j\in U, i\neq j}\{s_{ji}\} + t_i + \arg\min_{j\in U, i\neq j}\{s_{ij}\}\right\}$$

and then it is assigned to the position with the largest positional weight, where U denotes the set of unscheduled jobs. Note that the first job is the one that gives the minimum sum of the smallest setup time assignable before the job, its processing time, and the smallest setup time assignable after the job. Then, the second job i_2^*, which is assigned to the position with the second largest positional weight, is selected as follow. If the position with the second largest positional weight is located before the common due-date, the job i_2^* is selected as

$$i_2^* = \arg\min_{j\in U}\{BP_{j,i_1^*}\},$$

where $BP_{ji} = s_{ji} + t_j$. Otherwise, the job i_2^* is selected as

$$i_2^* = \arg\min_{j\in U}\{AP_{i_1^*,j}\},$$

where $AP_{ij} = s_{ij} + t_j$. In this way, the other jobs are selected and assigned to the remaining positions in the non-increasing order of the positional weight.

The improvement is done as follows. First, two machines with the largest and the smallest objective values are selected. Here, the objective value associated with machine k is calculated as

$$\sum_{i\in U_k}\{P_2 \cdot E_i + P_3 \cdot T_i\},$$

where U_k denotes the set of jobs assigned to machine k in the current schedule. Note that it is not needed to consider the penalty for the common due-date, i.e. $P_1 \cdot d \cdot n$, since it is a constant. Then, according to the numbers of jobs assigned to the two machines, the following two methods are repeatedly used until there is no further improvement. Let N_L and N_S denote the numbers of jobs assigned to the machines that have the largest and the smallest objective value, respectively.

Case 1: $N_L \neq N_S$

In this case, the insertion method is used from the first job (in sequence) assigned to the machine with the largest objective value. More specifically, to balance the workloads assigned to machines, a job assigned to the machine with the largest objective value is removed from the original place and then it is inserted to the best position that gives better objective value on the other machine. For a graphical description, see Figure 2(a) in which the dashed arrows denote the possible insertions. Finally, among all possible insertions, selected is the one that gives the best solution.

Case 2: $N_L = N_S$

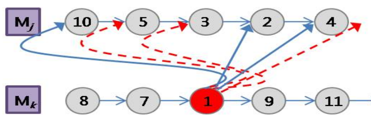
In this case, the interchange method is used by selecting one job for each of the two machines, i.e., those with the largest and the smallest objective values. First, among those assigned to machine k with the largest objective value, the job i^* is selected with the smallest earliness penalty, i.e.,

$$i^* = \arg\min_{i \in U_k} \{E_i\}\ .$$

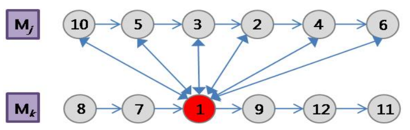
Note that U_k was defined as the set of jobs assigned to machine k in the current schedule. Then, the amount of improvement is checked after interchanging job i^* with those assigned to the other machine with the smallest objective value. For a graphical description, see Figure 2(b) in which the solid bi-directional arrows denote the possible interchanges. If there is no improvement, another job that has the smallest tardiness penalty is selected on the machine with the largest objective value, i.e.,

$$i^{**} = \arg\min_{i \in U_k} \{T_i\}\ .$$

Finally, this is repeatedly done until there is not further improvement.



(a) Insertion method



(b) Interchange method

Figure 2. Pictorial Descriptions of Insertion and Interchange Methods

3.2.2 Aggregated Setup Heuristic

As stated earlier, the second heuristic, denoted by the aggregated setup (AS) heuristic in this paper, is the same as the first one except for the method to obtain the initial solution (The initial solution is improved using the interchange method explained earlier).

In the AS heuristic, the initial solution is obtained by solving the relaxed problem in which sequence-dependent setup times are aggregated and added to the corresponding processing times. More formally, the processing time of job j is set as

$$t_j + \min_{\forall i (i \neq j)} \{s_{ij}\}\ .$$

To solve the relaxed problem without sequence-dependent setup times, we use the two-phase algorithm of Kim *et al.* (2012). More specifically, the two-phase algorithm can be briefly explained as follows.

In the first phase, the positional weight for the h-th position on machine k is calculated as

$$\lambda_{kh} = \begin{cases} \dfrac{nP_1}{m} + (h-1)\cdot P_2 & \text{if the position } (k,h) \text{ is early} \\ h \cdot P_3 & \text{otherwise,} \end{cases}$$

and the initial solution is obtained by matching the longest job to the position with the smallest positional weight, the second longest job to the position with the second smallest positional weight, and so on. Here, if the smallest positional weight occurs in the early form, the corresponding job is assigned to the first possible position. Otherwise, the job is assigned to the last possible position. Then, the optimal common due-date d^* is fixed using Proposition 2.

In the second phase, the initial solution is improved using the following two propositions. (See Kim and Lee (2006) for their proofs.) Proposition 3 specifies the condition that can improve a given solution when all tardy jobs at two machines are interchanged at the same time while maintaining the job sequence at each machine. In the proposition given below, n^T_k, s_k and $t_{i(l),k}$ denote the number of tardy jobs, the start time of the first tardy job, and the processing time of the l-th tardy job on machine k, respectively.

Proposition 3: *For arbitrary two machines k_1 and k_2 with $n^T_{k1} > n^T_{k2}$* and $s_{k1} > s_{k2}$, *if the following condition holds,*

$$\sum_{l=1}^{n^T_{k_1}} \max\left\{(l-1)\cdot(s_{k_1} - s_{k_1} - t_{i(l),k_1}),\ 0\right\} \times P2 < (n^T_{k_1} - n^T_{k_2})\cdot(s_{k_1} - s_{k_2})\cdot P_3$$

then interchanging all the tardy jobs on machines k_1 and k_2 improves the current solution.

Proposition 4 specifies the condition that can improve a given solution when the last tardy job at a machine is moved to the last position in sequence at another machine. In the proposition given below, t_{lk} and $C_k^{\max}$ denote the processing time of the last job and the maximum completion time on machine k, respectively.

Proposition 4: *For arbitrary two machines k_1 and k_2, if*

$$C_{k_1}^{\max} - C_{k_2}^{\max} > t_{lk_1},$$

moving the last job on machine k_1 to machine k_2 improves the current solution.

After improving the initial solution (for the relaxed problem) using the above two propositions, the current solution is improved further by changing the given job sequence at each machine. First, the jobs assigned to each machine are sorted in the non-increasing order of their processing times. Second, according to the sorted list, two new job sequences for the current machine are generated as follows.

(a) If the total processing time of the early jobs (after the job is assigned to the early position) is less than or equal to the optimal common due-date d^*, the current job is assigned to the early possible position. Otherwise, the current job is assigned to the tardy possible position.
(b) If the total processing time of the tardy jobs (after the job is assigned to the tardy position) is less than or equal to ($C_k^{\max}$-d^*), the current job is assigned to the tardy possible position. Otherwise, the current job is assigned to the early possible position.

Third, for each job sequence at the current machine, the early jobs are sorted in a non-increasing order of their processing times while the tardy jobs are sorted in a non-decreasing order of their processing times, which results in the well-known V-shape job sequence. Then, of the two job sequences, the better one is selected for the current machine if it improves the current solution.

4. EXPERIMENTAL RESULTS

To test the performances of the two heuristics suggested in this study, computational experiments were done on a number of test instances and the results are reported in this section. The performance measures used are: (a) percentage deviations from the optimal solution values for small-sized test instances; (b) the relative performance ratios for medium to large-sized test instances; and (c) CPU seconds. Here, the optimal solutions for small-sized test instances were obtained by solving the mixed integer programming model (P) using CPLEX 10.1, commercial integer programming software. Also, the relative performance ratio of heuristic a for a problem is defined as

$$[(C_a - C_{best}) / C_{best}] \cdot 100(\%),$$

where C_a is the objective value obtained from heuristic a and C_{best} is the best objective value for that problem among those obtained from the two heuristics. The algorithms and the program to generate integer programs were coded in C and the tests were done on a personal computer with a Pentium processor operating at 3.0 GHz clock speed.

For the test on small-sized problems, 90 instances were generated, using the method of Kim and Lee (2009), for each combination of three levels of the number of machines (2, 3 and 4) and three levels of the number of jobs (8, 9 and 10). The processing times were generated from $DU(5, 100)$, where $DU(a, b)$ denotes the discrete uniform distribution with range $[a, b]$. Also, the sequence-dependent setup times for all pairs of jobs were generated from $DU(25, 75)$. Finally, the penalties associated with common due-date assignment (P_1), earliness (P_2), tardiness (P_3) were generated from $DU(1, 10)$, $DU(P_1, P_1+10)$ and $DU(P_1, P_1+11)$, respectively. Here, the penalties were generated in such a way that $0 < P_1 \leq P_3$ since the case with $P_1 > P_3$ has the optimal solution with $d = 0$ and hence can be solved in polynomial time (Diamond and Cheng, 2000). For the test on large-sized problems, 120 instances were generated for each combination of three levels of the number of machines (4, 8 and 12) and six levels of the number of jobs (20, 40, 60, 80, 100 and 120).

The results for small-sized test instances are summarized in Table 1 that shows the percentage gaps from the optimal solution values and CPU seconds. It can be seen from the table that the IS heuristic (considering individual sequence-dependent setup times) is better than the AS heuristic (aggregating sequence-dependent setup times) because of its larger search space. In fact, the gaps of the IS heuristic (AS heuristic) range from 5.2% (4.6%) to 9.1% (13.3%). The overall average gaps of the IS and the AS heuristics were 7.2% and 9.2%, respectively. Also, the average amounts of improvement from the initial solutions were 10.8% and 10.6% for the IS and the AS heuristics, which shows the effectiveness of the improvement method suggested in this paper. In the absolute sense, however, the gaps from the optimal solution values are relatively large, which implies that it may be needed to develop more efficient algorithms after analyzing the properties of the problem, especially those for the sequence-dependent setup times. Finally, the two heuristics required very short computation times. However, due to the complexity of the problem, CPLEX required much longer computation times although it gave optimal solutions.

Similar results, which can be seen from Table 2, were obtained from the test on large-sized instances, i.e., the IS heuristic gives better solutions than the AS heuristic. Also, the average amounts of improvement from the initial solutions were 2.4% and 16.3% for the IS and the AS heuristics, which implies that the IS heuristic gives much better initial solutions than the AS heuristic. Also, as in the test on small-sized instances, the two heuristics required very short computation times. In fact, the heuristics gave solutions for the largest test instances with 12 machines and 120 jobs within 0.6 seconds.

Table 1. Test Results for the Heuristics on Small-Sized Test Instances

Number of machines	Number of jobs	IS		AS		CPLEX
		Gap[1]	CPU	Gap	CPU	CPU
2	8	7.1	< 0.01*	5.2	< 0.01	37
	9	5.2	< 0.01	4.6	< 0.01	232
	10	5.5	< 0.01	10.6	< 0.01	1818
3	8	7.4	< 0.01	8.4	< 0.01	6
	9	6.1	< 0.01	8.9	< 0.01	345
	10	8.2	< 0.01	12.5	< 0.01	796
4	8	7.7	< 0.01	8.9	< 0.01	70
	9	8.3	< 0.01	10.1	< 0.01	50
	10	9.1	< 0.01	13.3	< 0.01	1118

[1] average percentage deviations from optimal solution values out of 10 test instances.
* average CPU second less than 0.0005s.

Table 2. Test Results for the Heuristics on Large-Sized Test Instances

Number of machines	Number of jobs	IS		AS	
		RPR[1]	CPU	RPR	CPU
4	20	2.2	0.01	8.1	0.01
	40	2.2	0.03	7.2	0.01
	60	2.4	0.09	6.7	0.07
	80	2.0	0.28	5.4	0.23
8	40	0.3	0.02	8.0	0.01
	60	0.8	0.08	6.4	0.03
	80	0.5	0.18	6.8	0.08
	100	1.7	0.36	5.8	0.21
12	60	1.4	0.08	7.5	0.02
	80	0.1	0.16	8.6	0.05
	100	1.0	0.30	7.2	0.14
	120	1.8	0.56	8.9	0.29

[1] average relative performance ratio out of 10 test instances.

5. CONCLUSIONS

This paper considered the problem of determining the common due-date as well as the schedule on parallel machines for the objective of minimizing the sum of penalties associated with common due-date assignment, earliness and penalties. As an extension of the previous research, the sequence-dependent setup times were explicitly considered. Since the problem is known to be NP-hard, we suggested two heuristic algorithms in which an initial solution is obtained and then it is improved. Computational experiments were carried out on a number of test instances and the results showed that the heuristic considering individual sequence-dependent setup times is better than the heuristic with aggregating sequence-dependent setup times. In addition, the absolute performances of the two heuristics, i.e., gaps from the optimal solution values, were shown for small-sized test instances. Finally, the two heuristics were very fast and hence can be used for practical problems.

This research can be extended in several ways. First, the search heuristics, such as simulated annealing, genetic algorithm, and tabu search, can be used to increase the solution qualities although they may require more computation time. Second, it is needed to extend the problem by considering other due-date types, such as distinct due-dates and generalized due-dates.

ACKNOWLEDGEMENT

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2007-331-D00547).

REFERENCES

Baker, K. R. and G. D. Scudder, "On the assignment of optimal due-dates," *Journal of the Operational Research Society* 40 (1989), 93-95.

Baker, K. R. and G. D. Scudder, "Sequencing with earliness and tardiness penalties: a review," *Operations Research* 38 (1990), 22-36.

Balakrishnan, N., J. J. Kanet, and V. Sridharan, "Early/tardy scheduling with sequence dependent setups on uniform parallel machines," *Computers and Operations Research* 26 (1999), pp. 127-141.

Biskup, D. and H. Jahnke, "Common due-date assignment for scheduling on a single machine with jointly reducible processing times," *International Journal of Production Economics* 69 (2001), 317-322.

Chen, Z. L., "Scheduling and common due-date assignment with earliness-tardiness penalties and batch delivery costs," *European Journal of Operational Research* 93, (1996), 49-60.

Cheng, T. C. E. and M. Y. Kovalyov, "Bath scheduling and common due-date assignment on a single machine," *Discrete Applied Mathematics* 70 (1996), 231-245.

Cheng, T. C. E., "A heuristic for common due-date assignment and job scheduling on parallel machines," *Journal of the Operational Research Society* 40 (1989), 1129-1135.

Cheng, T. C. E., "A note on the common due-date assignment problem," *Journal of the Operational Re-*

search Society 37 (1986), 1089-1091.

Cheng, T. C. E., "Common due-date assignment and scheduling for a single processor to minimize the number of tardy jobs," *Engineering Optimization* 16 (1990), 129-136.

Cheng, T. C. E., Z. L. Chen, and N. V. Shakhlevich, "Common due-date assignment and scheduling with ready times," *Computers and Operations Research* 29 (2002), 1957-1967.

De, P., J. B. Ghosh, and C. E. Wells, "Due-date assignment and early/tardy scheduling on identical parallel machines," *Naval Research Logistics* 41 (1994), 17-32.

De, P., J. B. Ghosh, and C. E. Wells, "On the multiple-machine extension to a common due date assignment and scheduling problem," *Journal of the Operational Research Society* 42 (1991), 419-422.

Diamond, J. E. and T. C. E. Cheng, "Error bound for common due-date assignment and job scheduling on parallel machines," *IIE Tranactions* 32 (2000), 445-448.

Dvir, S. and S. George, "Two due-date assignment problems in scheduling a single machine," *Operations Research Letters* 34 (2006), 683-691.

Emmons, H., "Scheduling to a common due date on parallel uniform processors," *Naval Research Logistics* 34 (1987), 803-810.

Gordon, V., J. M. Proth, and C. Chu, "A survey of the state-of-the-art of the common due-date assignment and scheduling research," *European Journal of Operational Research* 139 (2002), 1-25.

Hall, N. G., "Scheduling problems with generalized due-dates," *IIE Transactions* 18 (1986), 220-222.

Hall, N. G., "Single and multiple processor models for minimizing completion variances," *Naval Research Logistics* 33 (1986), 49-54.

Kim, J.-G. and D.-H. Lee, "Algorithms for common due-date assignment and sequencing in a single machine with sequence-dependent setup times," *Journal of the Operational Research Society* 60 (2009), 1264-1272.

Kim, J.-G., J.-S. Kim, and D.-H. Lee, "Common due-date assignment and sequencing with sequence-dependent setup times: a case study on a paper remanufacturing system," *Management Science and Financial Engineering* 18 (2012), 1-12.

Kim, J-.G., J.-S. Kim, and D-.H. Lee, "Fast and meta-heuristics for common due-date assignment and scheduling on parallel machines," *International Journal of Production Research* 60 (2012), 6040-6057.

Kim, S.-I., H-.S. Choi, and D-.H. Lee, "Scheduling algorithms for parallel machines with sequence-dependent setup and distinct ready times: minimizing total tardiness," *Proceedings of the Institution of Mechanical Engineers*, *Part B: Journal of Engineering Manufacture* 221 (2007), 1087-1096.

Min, L. and W. Cheng, "Genetic algorithms for the optimal common due-date assignment and the optimal scheduling policy in parallel machine earliness/ tardiness scheduling problems," *Robotics and Computer-Integrated Manufacturing* 22 (2006), 279-287.

Ng, C. T. D., T. C. E. Cheng, M. Y. Kovalyov, and S. S. Lam, "Single machine scheduling with a variable common due-date and resource-dependent processing times," *Computers and Operations Research* 30 (2003), 1173-1185.

Panwalkar, S. S., M. L. Smith, and A. Seidmann, "Common due-date assignment to minimize total penalty for the one machine scheduling problem," *Operations. Research* 30 (1982), 391-399.

Quaddus, M. A., "A generalized model of optimal due-date assignment by linear programming," *Journal of the Operational Research Society* 38 (1987), 353-359.

Sundararaghavan, P. S. and M. U. Ahmed, "Minimizing the sum of absolute lateness in single-machine and multi-machine scheduling," *Naval Research Logistics* 31 (1984), 325-333.

Xia, Y., B. Chen and J. Yue, "Job sequencing and due-date assignment in a single machine shop with uncertain processing times," *European Journal of the Operational Research* 184 (2008), 63-75.

Xiao, W. Q. and C. L. Li, "Approximation algorithms for common due-date assignment and job scheduling on parallel machines," *IIE Transactions* 34 (2002), 467-477.