Management Science and Financial Engineering
Vol 18, No 1, May 2012, pp.13-20
ISSN 2287-2043 | EISSN 2287-2361 |

Scheduling Algorithms for the Maximal Total Revenue on a Single Processor with Starting Time Penalty

Un Gi Joo*
Department of Industrial and Management Engineering, Sun Moon University

(Received: December 8, 2011 / Accepted: March 10, 2012)

ABSTRACT

This paper considers a revenue maximization problem on a single processor. Each job is identified as its processing time, initial reward, reward decreasing rate, and preferred start time. If the processor starts a job at time zero, revenue of the job is its initial reward. However, the revenue decreases linearly with the reward decreasing rate according to its processing start time till its preferred start time and finally its revenue is zero if it is started the processing after the preferred time. Our objective is to find the optimal sequence which maximizes the total revenue. For the problem, we characterize the optimal solution properties and prove the NP-hardness. Based upon the characterization, we develop a branch-and-bound algorithm for the optimal sequence and suggest five heuristic algorithms for efficient solutions. The numerical tests show that the characterized properties are useful for effective and efficient algorithms.

Keywords: Job Scheduling, Dominant Properties, Branch-and-bound Algorithm, Heuristic Algorithms

* Corresponding Author, E-mail: ugjoo@sunmoon.ac.kr

1. INTRODUCTION

This paper considers a revenue maximization problem on a single processor, where some jobs have positive revenue and others have zero revenue according to their processing sequence. The revenue decreases linearly with the reward decreasing rate according to its processing start time till its preferred start time and finally its revenue is zero if it is started the processing after the preferred time. Due to the capacity restriction of the single processor, some jobs may be started their processing after their preferred times and the delayed jobs have no revenue. The starting time penalty problem occurs in several environments. Consider a service facility with clients who want to receive their service till their preferred (predetermined) times and they will depart the service facility if they are not in service till the times. Then, the starting the service after the preferred time has no revenue. As another example, consider a steel manufacturer that faces peak demand and limited capacity. The customers do not place orders if the manufacture cannot complete the orders by the preferred time. So, the manufacturer must choose some orders considering his capacity and complete the accepted orders before their preferred dates to maximize his revenue. We can also see the problem at the perishable goods such as fresh food materials, vegetables, fruits, and ready-mix concrete. The value (quality) of perishable goods decrease continuously once they are arrived till they are transformed (processed) into other goods. Since the value of the goods not transformed into other goods will be zero finally, the sequencing is very critical consideration to maximize the total revenue (value).

Human characteristics such as motivation and reward has a big impact on the job performance, and the reward needs to be considered in scheduling human

tasks as discussed by Lodree *et al*. (2009). There are a few papers on the revenue maximization scheduling problems. To maximize the total revenue, Aspvall *et al*. (1995), Gupta *et al*. (1992), Kyparisis *et al*. (1996), and Rothkopf and Smith (1984) have considered net present values of the total return as the revenue. Rothkopf and Smith (1984) have shown that the total revenue is maximized if the jobs are sequenced in a single priority index rule when the revenue is linear or exponential function. The revenue maximization problems are also considered under the job selection environment. The job selection problem is to decide both of which jobs to accept for processing and how to sequence them. Gupta *et al*. (1992) have developed a dynamic programming to determine the sequence which maximizes the total revenue for the accepted jobs, where they assume that the number of accepted jobs is given. Aspvall *et al*. (1995) have developed a polynomial algorithm for the same problem as Gupta *et al*. (1992. Kyparisis *et al*. (1996) have considered the job selection problem subject to multiple resource constraints and developed a dynamic programming, where they consider the number of selected jobs as a decision variable. But, they consider the same revenue function as Rothkopf and Smith (1984). As another revenue function, Yang (2009) has considered a function that the revenue of a job is decreasing exponentially with its decreasing rate according to its completion time. But, all of the revenues of Aspvall *et al*. (1995), Gupta *et al*. (1992), Kyparisis *et al*. (1996), Rothkopf and Smith (1984), and Yang (2009) are assumed to have nonnegative values regardless job sequences. By the way, our reward function does not require the assumption so that the function can have negative value for the jobs scheduled starting later than their preferred time, and the single priority index of Rothkopf and Smith (1984) is not optimal rule anymore for our problem.

There are some papers regarding the time related penalties such as starting (lead) time penalty. The lead time represents the time between starting the processing the job and the job's arrival time. In many manufacturing environments, an increase in the waiting time usually decreases satisfaction and reduces the revenue if it is started the processing before the preferred time. However, the customer does not accept the processing if his job is started the processing after the preferred time. Kesskinocak *et al*. (2001) have characterized several special cases for polynomial algorithms to maximize revenue when the revenue is a decreasing function with the lead time. However, they consider the jobs with equal processing times or restricted lead times. This paper allows arbitrary processing times and lead times. The benefit of the lead time flexibility is discussed by Charnsirisakskul *et al*. (2004), where they solve a mixed integer program under the job preemption is allowed, and they add the holding cost at their objective function to incorporate the waiting environment of the preempted order. This paper does not allow the job preemption and develops the optimal and heuristic algorithms based upon the characterization the optimal solution. The job selection and scheduling problems are surveyed by Slotnick (2011).

There are some papers on the single machine problems to maximize the total revenue minus weighted tardiness penalties (Cesaret *et al*., 2010; Nobibon and Leus, 2011; Rom and Slotnick, 2009, Slotnick and Morton, 2007). We show that the tardiness penalty is one of special cases of our starting penalty in Section 2.

This paper assumes that the given jobs are processed on a single processor (machine) without preemption, where some jobs may have zero revenues if they are started the processing later than their preferred time. For the problem, we characterize the optimal solution and develop a branch-and-bound algorithm using the characterizations. Five heuristic algorithms are also suggested to provide efficient solutions.

This paper is organized as follows. Section 2 describes the problem detail and characterizes the optimal solution. Section 3 suggests a branch-and-bound algorithm and five heuristic algorithms based upon the characterized properties. Section 4 evaluates the proposed algorithms by using numerical experiments, and some concluding remarks are added in Section 5.

2. PROBLEM DESCRIPTION AND SOLUTION PROPERTIES

Let us define some notations for our problem.

n : total number of jobs
p_i : processing time of job i, $i = 1, 2, \cdots, n$
a_i : initial reward of job i at the arrival time, $i = 1, 2 \cdots, n$
b_i : deterioration rate (weight) of job i, $i = 1, 2 \cdots, n$
S_i : processing start time of job i, $i = 1, 2 \cdots, n$
$V_i = a_i - b_i S_i$: reward of job i when the job i is started at the time S_i, $i = 1, 2 \cdots, n$

We assume that all the n jobs are arrived at the same time and we set the arrival time as zero without loss of generality. The reward V_i of job i is linearly decreasing according to its starting time S_i, where the job i could have negative reward if the job is started its processing after a_i / b_i. The time a_i / b_i can be regarded as the preferred start time or due date of the job i. Since the job with negative reward is not valuable, we set the revenue of job i as zero if $V_i < 0$ so that all the jobs have non-negative revenues. Our objective is to find an optimal schedule which maximizes the total revenue on a single processor (machine), which can process at most one job at a time. Since the processing the jobs with zero revenues has no effect on our objective, we can sequence the jobs in any positions after sequencing all the jobs with positive revenues and it has the same effect as rejection the jobs.

Let S denote a sequence ordered according to a job index for the n jobs and $f(S)$ represent the total revenue of S. Then, $f(S)$ is expressed as

$$f(S) = \sum_{i=1}^{n} \max\{V_i, 0\}. \quad (1)$$

Since a schedule without initial and intermediate idle times is obviously dominant schedule, we use the sequence and schedule interchangeably in this paper. Suppose that $b_i = 0$ for some jobs i. Then, $V_i = a_i$ for the jobs regardless of their sequences $\{S_i\}$. Therefore, we can sequence the jobs as late as possible after sequencing all the jobs with $b_i > 0$. So, this paper assume that $b_i > 0$ for all i without loss of generality.

We can derive several dominant rules by using adjacent pairwise comparisons. Suppose a sequence S, where a job j is processed immediately after the completion a job i. Consider another sequence S' which has the same sequence as the sequence S except that the jobs i and j are interchanged in their positions. Let denote the reward of the job i as V_i and V_i' for the schedule S and S', respectively. Then, there are nine cases of reward combinations according to the values of V_i and V_i' as follows:

Case 1: $V_i > 0,\ V_j > 0,\ V_i' > 0,\ V_j' > 0$
Case 2: $V_i > 0,\ V_j > 0,\ V_i' \le 0,\ V_j' > 0$
Case 3: $V_i > 0,\ V_j \le 0,\ V_i' > 0,\ V_j' > 0$
Case 4: $V_i > 0,\ V_j \le 0,\ V_i' \le 0,\ V_j' > 0$
Case 5: $V_i > 0,\ V_j \le 0,\ V_i' > 0,\ V_j' \le 0$
Case 6: $V_i > 0,\ V_j \le 0,\ V_i' \le 0,\ V_j' \le 0$
Case 7: $V_i \le 0,\ V_j > 0,\ V_i' \le 0,\ V_j' > 0$
Case 8: $V_i \le 0,\ V_j \le 0,\ V_i' \le 0,\ V_j' > 0$
Case 9: $V_i \le 0,\ V_j \le 0,\ V_i' \le 0,\ V_j' \le 0$

Please notice that four cases with $V_i \le 0$ and $V_i' > 0$ and three cases with $V_j > 0$ and $V_j' \le 0$ for the jobs i and j are not considered since those seven cases do not occur. For each case, we can derive conditions for dominant sequences.

Property 1: The following conditions characterize dominant sequences.

(1) For Cases 1, 2, and 3, a sequence with the WSPT (Weighted Shortest Processing Time) order of p_i/b_i is better than the others.
(2) For Case 4, the sequence S is better than S' if $a_i - a_j > (b_i - b_j)S_i$, where S_i is the starting time of the job i in the schedule S.
(3) For Cases 5, 6, 7, and 8, a sequencing the job with positive reward first is better than the others.
(4) For Case 9, there is no difference between S and S' in terms of the objective values.

Proof: We can easily show the property by using the following difference of the objective functions between $f(S)$ and $f(S')$ for each case.

$$f(S)-f(S') = \mathrm{Max}\{0, V_i\} + \mathrm{Max}\{0, V_j\} - \mathrm{Max}\{0, V_i'\} - \mathrm{Max}\{0, V_j'\}. \quad (2)$$

Equation (2) equals $f(S)-f(S') = V_i + V_j - V_i' - V_j'$ for Case 1 and the value is re-expressed as $f(S)-f(S') = b_i p_j - b_j p_i$. Therefore, $f(S)-f(S') > 0$ if $p_j/b_j > p_i/b_i$. That's to say, a schedule S satisfying the WSPT order of p_i/b_i is better than S' for Case 1. The conditions of Case 2 make $f(S)-f(S') = (a_i - S_i) - b_j p_i$. Since $V_i' \le 0$ in Case 2, $f(S)-f(S') \le b_i p_j - b_j p_i$ and $f(S)-f(S') < 0$ if $p_j/b_j < p_i/b_i$. Similarly, $f(S)-f(S') = b_i p_j - (a_j - b_j S_i) \ge b_i p_j - b_j p_i > 0$ if $p_j/b_j > p_i/b_i$ in Case 3. We finish the proof of (1). Case 4 in (2) leads to $f(S)-f(S') = V_i - V_j' = (a_i - a_j) - (b_i - b_j) S_i > 0$, if $a_i - a_j > (b_i - b_j) S_i$. To prove (3), we can show $f(S) > f(S')$ for both Cases 5 and 6, where $V_i > 0$ and $V_j \le 0$. By the way, $f(S) < f(S')$ for both Cases 7 and 8, where $V_i' \le 0$ and $V_j' > 0$. These mean the superiority of a sequencing the job with positive reward first against the sequencing the job with zero reward first. Finally, $f(S) = f(S')$ for Case 9. This completes the proof.

Even though Property 1 shows some dominant rules for the sequencing, we need to check the conditions to use the rules for the optimal sequence. For example, two adjacent jobs i and j should satisfy the conditions of $V_i > 0$, $V_j \le 0$, $V_i' \le 0$, and $V_j' > 0$ to apply the rule of Property 1(2). Therefore, we derive some special properties of Property 1.

Property 2: If $a_i/b_i \ge \sum_{i=1}^{n} p_i - \min\{p_i\}$ for all i, the optimal sequence is obtained if the jobs are ordered as the WSPT of p_i/b_i.

Proof: This is obvious by Property 1(1) because there exists only one case of Case 1 if $a_i/b_i \ge \sum_{i=1}^{n} p_i - \min\{p_i\}$ for all i,

Corollary 3: If a WSPT sequence of p_i/b_i leads to $V_i \ge 0$ for all i, it is an optimal sequence.

Proof: Let S denote a sequence of the jobs according to the WSPT order of p_i/b_i. Suppose that $V_i \ge 0$ for all i in S. Consider another schedule S' which has the same sequence except a pair of adjacent jobs i and j, where the positions of i and j are interchanged in S'. Then, Cases 1 and 2 can occur in S', and the WSPT order of p_i/b_i is an optimal sequence as derived at Property 1(1). This completes the proof.

The other special properties are derived by simplifying the values of parameters. We omit the proofs of the properties since they are special cases of the proof for Property 1.

Property 4: If $a_i = a$ for all i, the WSPT order of p_i/b_i and a nondecreasing order of b_i are optimal sequences for the Cases 1 through 3 and Cases 4 through 8, respectively.

Please notice that the longest first and shortest first

of b_i are optimal sequences for the Cases 1 through 3 and Cases 4 through 8 if $a_i = a$ and $p_i = p$ for all i, respectively.

Property 5: If $b_i = b$ for all i, the SPT(Shortest Processing Time) order of p_i and the nonincreasing order of a_i are optimal sequences for the Cases 1 through 3 and Cases 4 through 8, respectively.

Properties 4 and 5 show that the SPT order of p_i is an optimal sequence if $a_i = a$ and $b_i = b$ for all i. By using the relationship, we can calculate an upper bound (Equation (4)) for a feasible solution as discussed in Section 4.

Property 6: If $p_i = p$ for all i, a nonincreasing order of b_i and a_i / b_i are optimal sequences for the Cases 1 through 3 and Cases 5 through 8, respectively.

It is not easy to find the other dominant conditions in Property 6 for Case 4 without $a_i = a$ and/or $b_i = b$ for all i, However, if $p_i = p$ and $b_i = b$ for all i, a nonincreasing order of a_i leads to an optimal sequence.

Even though we characterize some special properties by restricting the values of parameters, most of them also need to check the conditions for applying the rules. So, we guess the complexity of our problem is not low. Property 7 proves the NP-hardness of our problem.

Property 7: Finding the optimal schedule which maximizes the total revenue (Equation (1)) is NP-hard.

Proof: It is known that the job sequencing for the tardiness objective is NP-hard (Du and Leung, 1990). We can prove the property by showing that our problem is transformed in polynomial time to the tardiness problem. Our objective (Equation (1)) is re-expressed as follow.

$$\begin{aligned}\text{Max } f(S) &= \text{Max} \sum_{i=1}^{n} \max\{V_i, 0\} \\ &= \text{Max} \sum_{i=1}^{n} \max\{a_i - b_i S_i, 0\} \\ &= \text{Min} \sum_{i=1}^{n} \max\{C_i - p_i - d_i', 0\}, \text{ if } a_i = d_i' \text{ and}\end{aligned}$$

$b_i = 1$ for all i, where C_i denotes the completion time of job i,

$$= \text{Min} \sum_{i=1}^{n} \max\{C_i - d_i, 0\}, \text{ if } d_i = d_i' + p_i$$

for all i, where d_i represents the due date of job i.

Since the tardiness problem is one of special problems of our problem, our problem is NP-hard. This completes the proof.

The optimal solution for the problem with $b_i = 1$ for all i is a combination of the SPT order of p_i and nonincreasing order of a_i by Property 5. The nonincreasing order of a_i is a *reverse* order to the EDD(Earliest Due Date) sequencesince a_i / b_i in this paper has the similar effect to the due date. Please remind that the optimal sequence for the minimal tardiness is a combination of the SPT and EDD rules (Baker, 1974). Furthermore, our problem cannot transform into the tardiness problem unless $b_i = 1$ for all i. Thus, we cannot use algorithms for the minimal tardiness to solve our problem and we develop our algorithms at the next section.

3. SOLUTION ALGORITHMS

We now develop algorithms based upon the characterization the solution. We can use a branch-and-bound algorithm to find the optimal solution. The branch-and-bound algorithm begins with an empty schedule and adds jobs one by one by branching on which job to select and schedule first. This paper uses the deepest first rule as a branching rule since the dominant (fathoming) rules are more effective when the sequence is generated according to the deepest first rule. For a bounding the optimal solution, we derive a lower bound L as follow.

$$L = \text{Max } \{LB1, LB2\}, \tag{3}$$

where $LB1$ is the objective value resulted from the sequence arranged in the WSPT order of p_i / b_i and the WSPT order is an optimal schedule for Cases 1 through 3 as shown in Property 1. Furthermore, we consider a nonincreasing order of a_i to derive another lower bound $LB2$ by using Property 5. We can use the maximum value between $LB1$ and $LB2$ as a lower bound L. Since the sequence resulted in the lower bound L is obviously a feasible solution, the lower bound L can be utilized to fathom sub-sequences in our branch-and-bound algorithm. Consider a sub-sequence $\hat{S}$ composed of $n1$ jobs at a branch node, $1 \le n1 \le n$. For the unscheduled n-$n1$ jobs at the branch node, we can sequence the unscheduled jobs according to the SPT rule by setting $a_i = \max\{a_i\}$ and $b_i = \min\{b_i\}$ for all i, where the optimality of the SPT rule is discussed at Properties 4 and 5. Let $f2$ denote the revenue of the SPT sequence for n-$n1$ jobs. If $L > f(\hat{S}) + f2$, then we can fathom the branch node. The lower bound L is updated as the current best objective value only when all the n jobs are sequenced and the objective value (Equation (1)) of the sequence is larger than L. The rules of Property 1 are also used to fathom the sub-sequences for the optimal solution. This paper evaluates the number of fathomed sequences to check the usefulness of the derived dominant properties at Section 4.

Since our problem is NP-hard, we need to develop heuristic algorithms for the problems with large number of jobs. We develop five heuristic algorithms to find solutions efficiently.

Algorithms 1 and 2 use the WSPT order of p_i / b_i. Algorithm 1 sequences the jobs with positive reward firstly, then sequences the others. For the algorithms, $T0$ denotes the earliest start time and $|A|$ represents the number of jobs in a set A, where A is a set of jobs scheduled already. The set of jobs to be scheduled is denoted as B.

Algorithm 1
Step 1.1: Let $T0 = 0$ and $A = \emptyset$.
Step 1.2: Find $B = \{i | a_i / b_i > T0\}$ and select a job k with minimum value of p_i / b_i among the set B: $k = \arg\min\{ p_i / b_i | i \in B\}$. Start the processing of the job k at time $T0$. Set $A = A \cup \{k\}$, $B = B-\{k\}$ and $T0 = T0 + p_k$. Repeat Step 1.2 until $B = \emptyset$.
Step 1.3: If $|A|<n$, sequence the $n-|A|$ jobs in an arbitrary sequence.

Step 1.2 of Algorithm 1 uses the results of Property 1 (1) and (3).The optimality of Step 1.2 is also discussed at Property 2 under the condition of $a_i / b_i \geq \sum_{i=1}^{n} p_i - \min\{ p_i \}$ for all i. However, we need Step 1.3 since there may exist some jobs with $a_i / b_i \leq T0$ after Step 1.2.

Similarly to Algorithm 1, Algorithm 2 uses the WSPT order of p_i / b_i and the result of Corollary 3.

Algorithm 2
Step 2.1: Sequence the jobs according to the WSPT order of p_i / b_i.
Step 2.2: Find jobs such that $V_i \leq 0$, and postpone the jobs as late as possible.

Corollary 3 shows the result of Step 2.1 is optimal if $V_i \geq 0$ for all i, and Step 2.2 uses the result of Property 1 (3).

As the other algorithms, we can sequence the jobs by using an $a_i - b_i T0$ index. Properties 4 through 6 show that a nonincreasing order of a_i and nondecreasing order of b_i can be optimal sequences. Thus, Algorithm 3 uses the nonincreasing order of $a_i - b_i T0$.

Algorithm 3
Step 3.1: Let $T0 = 0$, $B = \{1, 2, 3, \cdots, n\}$, and $r = 1$.
Step 3.2: Find a job k such that $k = \arg\max\{ a_i - b_i T0 | i \in B\}$, and then sequence the job k at the r-th position. Set $T0 = T0 + p_k$, $r = r+1$, and $B = B-\{k\}$.
Step 3.3: Go to Step 3.2 until $B = \emptyset$.

It is obvious that the sequence for the jobs with zero reward has no effect on the total revenue. Thus, Step 3.2 sequences the jobs with zero rewards only when all the jobs with positive rewards are finished their processing. However, we cannot guarantee the optimality of the nonincreasing order of $a_i - b_i T0$ since the nonincreasing order of b_i can be an optimal sequence by Properties 2 and 3. Algorithm 4 sequences the job with small $a_i - b_i T0$ early. For the algorithm, we find the jobs with positive rewards at the current time $T0$ and denote the set of the jobs as B. The set B of Algorithm 4 changes dynamically according to the selected job at Step 4.2.

Algorithm 4
Step 4.1: Let $T0 = 0$ and $r = 1$.
Step 4.2: Let $B=\{i | a_i - b_i T0 > 0\}$. Find a job k such that $k = \arg\min\{ a_i - b_i T0 | i \in B\}$, and then sequence the job k at the r-th position. Set $T0 = T0 + p_k$, $r = r+1$, and $B = B-\{k\}$.
Step 4.3: Go to Step 4.2 until $B = \emptyset$.
Step 4.4: Sequence the jobs unscheduled in an arbitrary order.

Finally, we can use the sequence resulted in the lower bound (Equation (3)) as an initial solution. For the initial solution, we can apply the rules for the dominant sequences of Property 1 to find an improved solution as the following algorithm, Algorithm 5.

Algorithm 5
Step 5.1: Find a feasible sequence resulted from the lower bound (Equation (3)).
Step 5.2: Apply the rules for dominant sequences to the feasible sequence.

4. NUMERICAL EXPERIMENTS

To evaluate the proposed algorithms, we generate two test problem sets: the one for the optimal solution and the other for the heuristic solutions. The one test problem set is to evaluate the performance of algorithms with respect to the optimal solution, where the optimal solution is obtained by using the branch-and-bound method for the problem set with small number of jobs. The other problem set is to evaluate the heuristic algorithms for the problems with large number of jobs.

We generate the jobs randomly according to uniform distributions. We use three cases of domains for the initial reward a_i: $a_i \sim U[10, 10]$, $a_i \sim U[10, 100]$, and $a_i \sim U[10, 1000]$. Similarly, we generate the deterioration rate b_i randomly according to $b_i \sim U(0,1]$, $b_i \sim U(0, 10]$, or $b_i \sim U(0, 100]$, where we generate b_i to have positive value on the three domains. For the processing time p_i, we use a uniform distribution U[2, 10]. Therefore, there are nine combinations of (a_i, b_i) according to the domains of a_i and b_i. For each combinations of (a_i, b_i), we randomly generate 30 and 100 problems to find the optimal and heuristic solutions, respectively. All the tests are performed on a PC (Intel Core 2 Duo CPU 2.8GHz) with Microsoft visual studio C++ code.

Firstly, we evaluate the mean errors of the heuristic algorithms with respect to the optimal solution. For the optimal solution, we generate 30 problems with n = 3, 5, 7, 9, and 11. Please notice that the computational time

for the optimal solution increases sharply if the number of jobs increases. So, we restrict the number of jobs small for the test as shown in Table 1.

The FR column in Table 1 represents the fathoming ratio(FR) according to the number of jobs for each test environment, where the fathoming ratio is measured as

Table 1. Test Results of the Branch-and-Bound and Heuristic Algorithms

n	a_i	b_i	B&B		Relative errors of heuristic algorithms									
			FR	CPUtime	A1		A2		A3		A4		A5	
					Avg	U5	Avg	U5	Avg	U5	Avg	U5	Avg	U5
3	[10, 10]	(0, 1]	0.767	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.267	0	0.056	0.767	0.113	0.633	0.110	0.633	0.105	0.7	0.034	0.867
		(0, 100]	0.011	0	0	1	0	1	0.017	0.967	0.017	0.967	0	1
	[10, 100]	(0, 1]	0.772	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.661	0	0.054	0.633	0.049	0.8	0.107	0.433	0.230	0.333	0.001	0.933
		(0, 100]	0.456	0	0.124	0.533	0.214	0.433	0.041	0.867	0.461	0.133	0.001	0.96
	[10, 1000]	(0, 1]	0.772	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.8	0	0.007	1	0	1	0.021	0.833	0.026	0.833	0	1
		(0, 100]	0.561	0	0.162	0.367	0.093	0.533	0.084	0.6	0.322	0.167	0.010	0.933
5	[10, 10]	(0, 1]	0.973	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.360	0	0.099	0.567	0.206	0.2	0.166	0.333	0.160	0.3	0.130	0.4
		(0, 100]	0.052	0	0.016	0.933	0.032	0.9	0.020	0.933	0.020	0.933	0.029	0.9
	[10, 100]	(0, 1]	0.974	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.890	0	0.148	0.233	0.090	0.467	0.113	0.4	0.362	0.067	0.018	0.967
		(0, 100]	0.540	0	0.221	0.4	0.305	0.2	0.060	0.667	0.567	0.033	0.047	0.7
	[10, 1000]	(0, 1]	0.974	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.986	0	0.021	0.9	0.001	1	0.049	0.633	0.038	0.667	0.001	1
		(0, 100]	0.84	0	0.193	0.233	0.171	0.367	0.140	0.267	0.476	0.033	0.024	0.767
7	[10, 10]	(0, 1]	0.998	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.526	0.002	0.112	0.33	0.263	0.167	0.180	0.2	0.160	0.233	0.186	0.233
		(0, 100]	0.109	0.004	0.033	0.867	0.046	0.867	0.016	0.9	0.021	0.9	0.033	0.867
	[10, 100]	(0, 1]	0.998	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.940	0.001	0.142	0.233	0.202	0.2	0.144	0.2	0.449	0.033	0.033	0.633
		(0, 100]	0.540	0.001	0.404	0.167	0.461	0	0.067	0.7	0.658	0	0.085	0.633
	[10, 1000]	(0, 1]	0.998	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	1	0	0.019	0.967	0.001	1	0.059	0.333	0.063	0.5	0.002	1
		(0, 100]	0.916	0.001	0.213	0.067	0.291	0.067	0.126	0.4	0.426	0.033	0.041	0.667
9	[10, 10]	(0, 1]	1	0.001	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.593	0.158	0.144	0.233	0.265	0.033	0.232	0.067	0.215	0.067	0.236	0.033
		(0, 100]	0.145	0.304	0.083	0.7	0.105	0.667	0.073	0.733	0.067	0.767	0.105	0.667
	[10, 100]	(0, 1]	1	0.001	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.967	0.017	0.206	0.1	0.235	0.167	0.174	0.233	0.501	0	0.036	0.6
		(0, 100]	0.578	0.153	0.402	0.133	0.447	0.033	0.099	0.567	0.702	0	0.108	0.533
	[10, 1000]	(0, 1]	1	0	0	1	0	1	0	1	0	1	0	1
		(0, 10]	1	0.001	0.027	0.9	0.003	1	0.075	0.333	0.083	0.267	0.002	1
		(0, 100]	0.918	0.033	0.263	0.067	0.376	0.1	0.171	0.133	0.577	0	0.046	0.667
11	[10, 10]	(0, 1]	1	0.004	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.664	18.474	0.188	0.167	0.304	0.1	0.242	0.1	0.243	0.133	0.257	0.133
		(0, 100]	0.183	41.129	0.080	0.733	0.116	0.7	0.075	0.733	0.073	0.733	0.116	0.7
	[10, 100]	(0, 1]	1	0.005	0	1	0	1	0	1	0	1	0	1
		(0, 10]	0.966	1.986	0.230	0.067	0.359	0.033	0.214	0.067	0.560	0	0.100	0.367
		(0, 100]	0.612	19.806	0.358	0.167	0.580	0	0.116	0.433	0.638	0	0.168	0.367
	[10, 1000]	(0, 1]	1	0.004	0	1	0	1	0	1	0	1	0	1
		(0, 10]	1	0.003	0.026	0.867	0.003	1	0.076	0.233	0.106	0.1	0.012	1
		(0, 100]	0.968	2.132	0.350	0.033	0.448	0	0.198	0	0.600	0	0.110	0.367

the total number of fathomed sequences to $n!$ in the branch-and-bound algorithm. We denote FR = 1 in Table 1 if the number of fathomed sequences is more than $0.999n!$. Table 1 shows that the fathoming ratio increases as the number of jobs increases or the variance (range of the domain) of b_i decreases. We can also notice that the fathoming ratio increases as the variance of a_i increases. Therefore, we can conclude the dominant rules in Property 1 have big effects on efficiency of branch-and-bound algorithm. The CPU time column in Table 1 denotes the computational time of the branch-and-bound algorithm. The computational time is almost zero second till the number of jobs is 7. However, the branch-and-bound algorithm needs 41.129 seconds in average to find the optimal solution to the problem with n = 11, a_i ~U[10, 10], and b_i ~U(0, 100]. We infer the reason of the long computational time that FRs of the problems is smaller than FRs of the other problems.

Additionally, Table 1 measures the performance of the heuristic algorithms by using the relative errors which is calculated as (optimal solution-heuristic solution)/(optimal solution), where we use the branch-and-bound algorithm for the optimal solution. We denote Ax in the tables for Algorithm x. For example, A1 represents Algorithm 1 in the tables. The Avg column in Table 1 denotes the mean relative error (MRE) and U5 represents the ratio of problems with less than 5% MRE. Even though the error performance slightly varies according to the test parameters, we can notice that the MREs increase as the number of jobs increases, and Algorithm 5 has the best performance with respect to MRE. Please remind that Algorithm 5 uses the dominant rules of Property 1 to find an improved schedule for an initial feasible schedule. Table 1 shows that Algorithms 1 and 3 have nearly similar performance as the second best algorithms. The U5 column shows our heuristic algorithms have good performance if the variance of b_i is small. However the ratio U5 decreases as the number of jobs increases. Table 1 also shows that Algorithm 5 is the best and Algorithm 3 is the second best one with respect to U5.

Through the test, we can conclude the dominant (fathoming) rules of Property 1 are valuable to find a good solution. However, the performance for the first test problem set is worse as the number of jobs increases and, thus we need to evaluate the performance for the problems with larger number of jobs.

We test the heuristic algorithms for the problems with large number of jobs, for the problems, the branch-and-bound algorithm is not practical due to huge computational time. We evaluate the mean relative deviations (MRD) of heuristic algorithms with respect to the upper bound U, where the relative deviation is calculated as (U-heuristic solution)/U and the upper bound U is obtained by calculating the objective value of an optimal schedule for the problem with $a_i = \max\{a_i\}$ and $b_i = \min\{b_i\}$ for all i as Equation (4).

Table 2. Numerical Experiments of Heuristic Algorithms for Large-Sized Problems

n	a_i	b_i	MRD				
			A1	A2	A3	A4	A5
10	[10, 10]	(0, 1]	0.000	0.000	0.000	0.000	0.000
		(0, 10]	0.572	0.656	0.614	0.613	0.622
		(0, 100]	0.157	0.182	0.163	0.160	0.171
	[10, 100]	(0, 1]	0.406	0.406	0.406	0.406	0.406
		(0, 10]	0.711	0.752	0.717	0.816	0.675
		(0, 100]	0.695	0.740	0.568	0.872	0.586
	[10, 1000]	(0, 1]	0.438	0.438	0.438	0.438	0.438
		(0, 10]	0.507	0.493	0.532	0.543	0.491
		(0, 100]	0.756	0.780	0.719	0.851	0.687
30	[10, 10]	(0, 1]	0.000	0.000	0.000	0.000	0.000
		(0, 10]	0.836	0.854	0.851	0.843	0.848
		(0, 100]	0.410	0.456	0.401	0.403	0.454
	[10, 100]	(0, 1]	0.436	0.436	0.436	0.436	0.436
		(0, 10]	0.854	0.887	0.856	0.902	0.837
		(0, 100]	0.852	0.913	0.791	0.927	0.826
	[10, 1000]	(0, 1]	0.477	0.477	0.477	0.477	0.477
		(0, 10]	0.651	0.649	0.691	0.760	0.638
		(0, 100]	0.874	0.930	0.859	0.932	0.850
50	[10, 10]	(0, 1]	0.000	0.000	0.000	0.000	0.000
		(0, 10]	0.864	0.874	0.870	0.866	0.868
		(0, 100]	0.577	0.625	0.570	0.575	0.622
	[10, 100]	(0, 1]	0.438	0.438	0.438	0.438	0.438
		(0, 10]	0.888	0.914	0.893	0.924	0.882
		(0, 100]	0.910	0.949	0.870	0.957	0.903
	[10, 1000]	(0, 1]	0.477	0.477	0.477	0.477	0.477
		(0, 10]	0.714	0.733	0.754	0.849	0.710
		(0, 100]	0.906	0.956	0.895	0.952	0.901
100	[10, 10]	(0, 1]	0.001	0.000	0.001	0.001	0.000
		(0, 10]	0.889	0.893	0.894	0.891	0.892
		(0, 100]	0.807	0.852	0.814	0.816	0.850
	[10, 100]	(0, 1]	0.448	0.448	0.448	0.448	0.448
		(0, 10]	0.914	0.929	0.921	0.935	0.911
		(0, 100]	0.934	0.975	0.919	0.966	0.947
	[10, 1000]	(0, 1]	0.489	0.489	0.489	0.489	0.489
		(0, 10]	0.795	0.836	0.824	0.897	0.793
		(0, 100]	0.935	0.977	0.929	0.966	0.942
200	[10, 10]	(0, 1]	0.001	0.000	0.001	0.001	0.000
		(0, 10]	0.893	0.896	0.896	0.894	0.895
		(0, 100]	0.911	0.935	0.911	0.917	0.932
	[10, 100]	(0, 1]	0.452	0.451	0.452	0.452	0.451
		(0, 10]	0.931	0.940	0.940	0.943	0.929
		(0, 100]	0.965	0.986	0.957	0.982	0.974
	[10, 1000]	(0, 1]	0.494	0.494	0.494	0.494	0.494
		(0, 10]	0.852	0.897	0.872	0.922	0.853
		(0, 100]	0.957	0.988	0.953	0.978	0.966
300	[10, 10]	(0, 1]	0.002	0.000	0.002	0.002	0.000
		(0, 10]	0.894	0.897	0.896	0.895	0.896
		(0, 100]	0.951	0.962	0.952	0.954	0.962
	[10, 100]	(0, 1]	0.452	0.452	0.452	0.452	0.452
		(0, 10]	0.934	0.941	0.942	0.944	0.934
		(0, 100]	0.984	0.992	0.983	0.990	0.988
	[10, 1000]	(0, 1]	0.494	0.494	0.494	0.494	0.494
		(0, 10]	0.875	0.912	0.894	0.929	0.880
		(0, 100]	0.968	0.990	0.967	0.983	0.978

$U = f(S)$, where S is the SPT order for the n jobs with $a_i = \max\{a_i\}$ and $b_i = \min\{b_i\}$ for all i. (4)

Please notice that the smaller MRD implies the better performance since U is one of ideal objective value. For the test problem, we generated 100 problems with n = 10, 30, 50, 100, 200, and 300.

Table 2 shows the MRDs of heuristic algorithms. We omit the computational time of the heuristic algorithms in Tables 1 and 2 since we cannot measure the time in the second scale for almost every problems except the problems under n = 200, a_i ~U[10, 10], and b_i ~U(0, 100], in which the mean computation time of Algorithm 2 is 0.0003 seconds. Table 2 shows that Algorithm 5 has the best performance and Algorithm 3 has the second best performance with respect to MRD. We can see that Algorithm 1 has the best performance often as the number of jobs increases to 100, 200, and 300. However, Algorithm 4 has the worst performance on the overall test problems. Therefore, we can conclude that a sequencing to accept more jobs with positive rewards is not good decision for maximization the total revenue.

In conclusion, Algorithm 5 is the best choice as a heuristic algorithm and we can conclude the fathoming(dominant) rules of Property 1 are valuable to find a good schedule. Additionally, we find that Algorithms1 and 3 have good performance with respect to MRE and MRD.

5. CONCLUSIONS

This paper considers a single processor scheduling problem which maximizes the total revenue for the given jobs. The revenue of a job depends on its starting (lead) time, and the revenue is zero if the job is started its processing after its preferred start time. Since the job with zero revenue is meaningless to the manufacturer, it has the same effect as the rejection of the job.

We show the problem is NP-hard and characterize some dominant properties to derive a branch-and-bound and five heuristic algorithms. Based upon the numerical tests, we show the dominant rules are effective and efficient to find the optimal solution in the branch-and-bound algorithm and to find good solutions in the proposed heuristic algorithms. Especially Algorithm 5 is efficient to find the good solution.

As a further study, a problem under the other scheduling environment such as multiple processors or other measures is considerable. For example, we can derive similar dominant rules to Property 1 for the reward function $a_i - b_i C_i$. A problem with nonlinear decreasing rate also needs to be studied further. Additionally, development of a meta-heuristic algorithm is another valuable further subject. However, it may require more computational time.

REFERENCES

Aspvall, B., S. D. Flam, and K. P. Villanger, "Selecting among scheduled projects," *Operations Research Letter* 17 (1995), 37-40.

Baker, K. R., *Introduction to sequencing and scheduling*, John Wiley and Sons, Inc., 1974.

Cesaret, B., C. Oguz, and F. S. Salman, "A tabu search algorithm for order acceptance and scheduling," *Computers and Operations Research*, 2010, doi:10.1016/j.cor.2010.09.18.

Charnsirisakskul, K., P. M. Griffin, and P. Keskinocak, "Order selection and scheduling with leadtime flexibility," *IIE Transactions* 36 (2004), 697-707.

Du, J. and J. Y. T. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of Operations Research* 15 (1990), 483-495.

Gupta, S. K., J. Kyparisis, and C. M. Ip, "Project selection and sequencing to maximize net present value of the total return," *Management Science* 38 (1992), 751-752.

Kesskinocak, P., R. Ravi, and S. Tayur, "Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues," *Management Science* 47 (2001), 264-279.

Kyparisis, G. J., S. K. Gupta, and C. M. Ip, "Project selection with discounted returns and multiple constraints," *European Journal of Operational Research* 94 (1996), 87-96.

Lodree, Jr. E. J., C. D. Geiger, and X. Jiang, "Taxonomy for integrating scheduling theory and human factors: review and research opportunities," *International Journal of Industrial Ergonomics* 36 (2009), 39-51.

Nobibon, F. T. and R. Leus, "Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment," *Computers and Operations Research* 38 (2011), 367-378.

Rom, W. O. and S. A. Slotnick, "Order acceptance using genetic algorithms," *Computers and Operations Research* 36 (2009), 1758-1767.

Rothkopf, M. H. and S. A. Smith, "There are no undiscovered priority index sequencing rules for minimizing total delay costs," *Operations Research* 32 (1984), 451-456.

Slotnick, S. A. and T. E. Morton, "Order acceptance with weighted tardiness," *Computers and Operations Research* 34 (2007), 3029-3042.

Slotnick, S. A., "Order acceptance and scheduling: a taxonomy and review," *European Journal of Operational Research* 212 (2011), 1-11

Yang, W. H., "Scheduling jobs on a single machine to maximize the total revenue of jobs," *Computers and Operations Research* 36 (2009), 565-583.