Hindawi Publishing Corporation
Journal of Industrial Engineering
Volume 2014, Article ID 568317, 13 pages
http://dx.doi.org/10.1155/2014/568317

Research Article

Minimizing the Number of Tardy Jobs on a Single Machine with an Availability Constraint

Ehsan Molaee[1] and Ghasem Moslehi[2]

[1] *Industrial Management Group, Binaloud Institute of Higher Education, Mashhad 9351991949, Iran*
[2] *Department of Industrial Engineering, Isfahan University of Technology, Isfahan 8415683111, Iran*

Correspondence should be addressed to Ehsan Molaee; e.molaee@yahoo.com

Received 10 May 2014; Accepted 19 August 2014; Published 1 September 2014

Academic Editor: Purushothaman Damodaran

Most scheduling problems are based on the assumption that machines work continuously during the planning horizon. This assumption is not true in many production environments because the machine may not be available during one or more periods such as during breakdowns or maintenance operations. In this paper, the problem of the single machine scheduling with one unavailability period and nonresumable jobs with the aim of minimizing the number of tardy jobs is studied. A number of theorems are proved and a heuristic procedure is developed to solve the problem. A branch-and-bound approach is also presented which includes upper and lower bounds and efficient dominance rules. Computational results for 2680 problem instances show that the branch-and-bound approach is capable of solving 98.7% of the instances optimally, bearing witness to the efficiency of the proposed procedure. Our results also indicate that the proposed approaches are more efficient when compared to other methods.

1. Introduction

Most scheduling problems take the fact that the machine works continuously during the planning horizon for granted. This assumption may be unrealistic in many production environments as the machine becomes unavailable during breakdowns or maintenance operations.

The number of tardy jobs is one of the important objective functions in scheduling problems. This target has wide applications in many production and service environments and reflects factors of external cost based on due dates such as customer satisfaction. The number of tardy jobs can represent orders of customers that are not satisfied. In this paper, the single machine scheduling problem with a known unavailability period and nonresumable jobs with the objective of minimizing the number of tardy jobs is considered. According to Pinedo [1] separation, this problem is denoted by $1, h_1 || \sum U_i$.

So far, a few researchers have studied single machine problems with availability constraints. In these problems, there are three general states for jobs including resumable, nonresumable, and semiresumable states. According to the resumable scenario, job preemption is allowed. This means that if a job cannot be finished before a nonavailability period of the machine, its processing can be resumed when the machine is available again. According to the nonresumable scenario, preemption is undesirable and the whole processing of the job should be repeated if it cannot be finished before the unavailability period. In the semiresumable state, part of the processing should be repeated if it cannot be finished before the nonavailability interval.

Lee [2] showed that the SPT rule, the EDD rule, and the Moore and Hodgson algorithm [3] minimize the targets of sum of completion times, maximum lateness, and the number of tardy jobs, respectively, in the single machine problem with a fixed nonavailability interval and resumable jobs. He also proved that the problem $1, h_1|\text{pmtn}| \sum_{i=1}^{n} w_i C_i$ is NP-hard even if $w_i = p_i$ for all i. In addition, he showed that the problem $1, h_1 || L_{\max}$ is NP-hard, too.

Adiri et al. [4] and Lee and Liman [5] proved that the problem $1, h_1 || \sum_{i=1}^{n} C_i$ is NP-hard. Lee and Liman [5] showed that the worst case boundary of the SPT rule for this problem was 9/7. Sadfi et al. [6] provided a heuristic algorithm called MSPT (modified SPT) to solve the problem and proved that

this heuristic had a tight worst case boundary of 20/17. Breit [7] presented a parametric heuristic procedure with the time complexity $O(n \log n)$ for this problem and showed that the minimized calculated worst case boundary for this algorithm was 1.07.

Kacem and Chu [8] developed a set of lower bounds for the problem $1, h_1 || \sum_{i=1}^{n} w_i C_i$ and compared them analytically. They also proposed three heuristic algorithms H_1 with the time complexity $O(n \log n)$, H_2 with $O(n^3)$, and H_3 with $O(n^2)$. They finally presented a branch-and-bound algorithm to solve the problem in which the best solution of the heuristics H_1, H_2, and H_3 is used as the upper bound. Kacem [9] showed that heuristic H_3 proposed in [8] had a tight worst case boundary of 2. Kacem et al. [10] formulated a mixed integer programming (MIP) model for the problem $1, h_1 || \sum_{i=1}^{n} w_i C_i$. Then, they proposed some new lower bounds including analytic comparisons of them. Next, they presented a branch-and-bound algorithm based on these lower bounds and proposed a dynamic programming procedure to solve the problem optimally. Finally, via computational results, they analyzed the performances of the three exact methods of MIP model, branch-and-bound approach, and the dynamic programming provided in their paper.

Liao and Chen [11] proposed a heuristic algorithm with the time complexity $O(n^2 \sum_{i=1}^{n} p_i)$, for the single machine problem with periodic maintenance operations and nonresumable jobs with the target of maximum tardiness. Then, they presented a branch-and-bound algorithm to achieve optimal schedules and utilized it to evaluate their heuristic.

Ji et al. [12] studied the problem of minimizing makespan on a single machine with periodic maintenance and nonresumable jobs. They showed that the worst case boundary of LPT rule for this problem was 2 and that it was tight. Additionally, they showed that there was no approximation algorithm with polynomial time and worst case performance less than 2 unless $P = NP$.

Chen [13] studied the problem of minimizing total flow time and maximum tardiness subject to periodic maintenance and nonresumable jobs. He provided a heuristic procedure with the time complexity $O(n^2 \sum_{i=1}^{n} p_i)$ to find efficient schedules of the linear target function $\alpha F(S) + (1 - \alpha) T_{\max}(S)$, where $F(S)$ and $T_{\max}(S)$ represent total flow time and maximum tardiness of schedule S, respectively, and $0 \leq \alpha \leq 1$ is the weighting factor. He also showed that the worst case boundary of this heuristic according to values $\alpha = 0$ and $\alpha = 1$ was between 1 and 3/2. He then developed a branch-and-bound algorithm to achieve optimal schedules and utilized it to examine schedules obtained through his heuristic.

Adiri et al. [14] assumed that the unavailable period is unknown but with a probabilistic distribution. They distinguished two cases of a breakdown, that is, the resumable and nonresumable cases. By applying the EDD and MSPT rules for two cases, respectively, the criterion of number of tardy jobs can be minimized. Chen [15] considered the problem of minimizing the number of tardy jobs subject to periodic maintenance and nonresumable jobs. He presented a heuristic algorithm with the time complexity $O(n^2 \sum_{i=1}^{n} p_i)$. He then proposed a branch-and-bound approach to find optimal schedules and applied it for evaluating schedules obtained from his heuristic.

In the literature, the single machine scheduling with the target of minimizing the number of tardy jobs with an availability constraint has not been reported, specially by an efficient heuristic and a binary branch and bound in solving procedure, to the best of our knowledge. Therefore, in this paper, minimizing the number of tardy jobs in the single machine scheduling problem with a fixed nonavailability period, that is, problem $1, h_1 || \sum U_i$, is studied. It is assumed that there is one unavailability interval with known beginning and finishing times, and all the jobs are nonresumable.

The rest of this paper is organized as follows. In Section 2, a definition of the problem is presented along with the notations and assumptions used. In Section 3, some theorems are proved to facilitate the solution of the problem. In Section 4, a heuristic procedure is developed for the problem. A branch-and-bound (BB) approach is proposed in Section 5 that aims at the optimal solution of the problem. Computational results for evaluating the heuristic and the BB algorithm are presented in Section 6. Also in this section, numerical results are used to make comparisons between the procedures proposed in this paper and the algorithms of Chen [15] for the special case of problems in which problems with periodic maintenance are transferred to problems with one nonavailability interval.

2. Problem Definition

The problem $1, h_1 || \sum U_i$ consists of scheduling n jobs in the set $I = \{J_1, J_2, \ldots, J_n\}$ on a single machine. The target is defined as minimizing the number of tardy jobs. Each job has a known processing time and a due date. All jobs are nonresumable and available at time zero. The machine is not available during the time interval of T_1 to T_2 and cannot process any job in this period. Outside this interval, the machine can process only one job at a time. The following notations are used for the problem:

n: number of jobs;

I: the set of jobs to be scheduled, $I = \{J_1, J_2, \ldots, J_n\}$;

p_i: processing time of job J_i, $i = 1, 2, \ldots, n$;

d_i: due date of job J_i, $i = 1, 2, \ldots, n$;

T_1: beginning time of the nonavailability interval;

T_2: finishing time of the nonavailability interval;

C_i: completion time of job J_i, $i = 1, 2, \ldots, n$;

N_T: number of tardy jobs in the complete schedule.

It is assumed that data are integers. If all the jobs can be processed before the nonavailability interval, the problem will be changed to the single machine scheduling problem without any availability constraint whose optimal solution will be obtained by applying the Moore and Hodgson algorithm in a polynomial time. Therefore, this paper considers only problems in which not all the jobs can be inserted before the nonavailability period. In other words, the inequality $\sum_{i=1}^{n} p_i > T_1$ always holds.

3. Theorems of the Problem

The problem $1, h_1 || \sum U_i$ was studied by Lee [2]. He proved that the problem is NP-hard. He also showed that for the resumable case of the problem, that is, $1, h_1 | \text{pmtn} | \sum U_i$, the Moore and Hodgson algorithm can be applied to solve the problem optimally. For this case, the length of the maintenance period, $T_2 - T_1$, is added to completion times of those jobs that are finished after T_2.

Lee [2] also showed that if the Moore and Hodgson algorithm was used for solving the problem $1, h_1 || \sum U_i$, the inequality $P(NR) \leq P^*(NR) + 1$ would always be true, in which $P(NR)$ denotes the number of tardy jobs obtained by the Moore and Hodgson algorithm and $P^*(NR)$ denotes the optimal number of tardy jobs.

The steps in the Moore and Hodgson algorithm are as follows:

Step 1. Put the jobs in the EDD order and index them in the same order.

Step 2. Calculate the tardiness of jobs. Notice that in the completion time of the jobs which are after the nonavailability period, the length of the time interval $T_2 - T_1$ should be added. If there is no tardy job in the sequence obtained, this sequence would be optimal, stop. Otherwise, go to Step 3.

Step 3. Suppose that J_α is the first tardy job in the sequence and select the job J_β as follows:

$$p_\beta = \max_{i=1,2,\ldots,\alpha} \{p_i\}. \quad (1)$$

This job is removed from the sequence and is processed after all the nontardy jobs. Go to Step 2.

It is necessary to note that Lee [2] did not propose any procedure to solve the problem optimally. From the above considerations, we can conclude the following corollary.

Corollary 1. *In view of the fact that the solution obtained from the Moore and Hodgson algorithm has at most one unit difference from the optimal solution, the value of $P(NR) - 1$ can be considered as a lower bound for the problem $1, h_1 || \sum U_i$.*

According to Corollary 1, although the difference between Moore and Hodgson algorithm and optimal solution is at most one, in many practical cases, such as airline scheduling or hospital services scheduling, this difference is very important and it is so critical to recognize the optimal solution.

In the rest of this section, a number of theorems are proposed to solve the problem $1, h_1 || \sum U_i$.

Theorem 2. *In the problem $1, h_1 || \sum U_i$, there is an optimal schedule in which jobs after and before the nonavailability interval are sequenced by the Moore and Hodgson algorithm.*

Proof. Each sequence in the solution of the problem $1, h_1 || \sum U_i$ can be divided into two sets σ_1 and σ_2, such that all the jobs in set σ_1 are completed before T_1 and all the jobs in set σ_2 are completed after T_2. It is clear that the optimal solution of set σ_1 is achieved based on the Moore and Hodgson algorithm.

On the other hand, it can be assumed that the ready times for all the jobs in σ_2 are equal to T_2, which entails that determining the optimal job sequence in this set is similar to minimizing the number of tardy jobs in a single machine with ready times equal to T_2 for all jobs. Therefore, the number of tardy jobs in σ_2 can be minimized by applying the Moore and Hodgson algorithm. □

Corollary 3. *According to Theorem 2, the dominant set of sequences for the problem $1, h_1 || \sum U_i$ contains the ones in which each partial sequence before and after the nonavailability period is arranged by the Moore and Hodgson algorithm. Therefore, to devise a solution procedure for the problem, it suffices to develop algorithms that provide sequences with this property.*

Theorem 4. *In the problem $1, h_1 || \sum U_i$, if there is a job J_i such that $d_i \geq \sum_{i=1}^{n} p_i + (T_2 - T_1) + \max_{1 \leq i \leq n}\{p_i\} - 1$, this job is always placed at the last position of the schedule.*

Proof. Clearly, in the problem $1, h_1 || \sum U_i$, the total completion time of all jobs is not less than $\sum_{i=1}^{n} p_i + (T_2 - T_1)$. But perhaps due to the nonresumption assumption, there is some idle insert for the machine. Surely, the value of this idle insert is not greater than the maximum processing time of jobs minus one. So, completion times of jobs would not be greater than $\sum_{i=1}^{n} p_i + (T_2 - T_1) + \max_{1 \leq i \leq n}\{p_i\} - 1$. Therefore, if the due date of a job is not less than this value, this job will not be a tardy one in any position of the schedule. Thus, placing this job at the last position of the schedule will not increase the number of tardy jobs. □

Corollary 5 (Dominance Rule 1). *At the beginning of each solution algorithm, Theorem 4 can be used as the dominance rule. Each time a job is placed at the last position of the schedule due to this rule, the solution space will be decreased by one unit.*

Theorem 6. *If the Moore and Hodgson algorithm is applied to the problem $1, h_1 || \sum U_i$ and if there is at least one tardy job before the unavailability period in the schedule obtained, this schedule is optimal.*

Proof. Figure 1(a) shows the schedule obtained from the Moore and Hodgson algorithm for the problem $1, h_1 | \text{pmtn} | \sum U_i$. In this Figure, S_1 is the set of nontardy jobs before the unavailability period, J_i is the tardy job before the unavailability period, J_k is the preempted tardy job, and S_2 is the set of other tardy jobs. We know that this schedule is optimal for the problem $1, h_1 | \text{pmtn} | \sum U_i$ [2]. If we apply the Moore and Hodgson algorithm for the problem $1, h_1 || \sum U_i$, a schedule illustrated in Figure 1(b) will be obtained. Clearly, the sequence of jobs in this case is similar to that in Figure 1(a) except for job J_k which could not be finished before T_1 and has to start at time T_2. According to Figures 1(a) and 1(b), the number of tardy jobs will be the same in both problems $1, h_1 || \sum U_i$ and $1, h_1 | \text{pmtn} | \sum U_i$. On the other hand, the optimal number of tardy jobs in problem

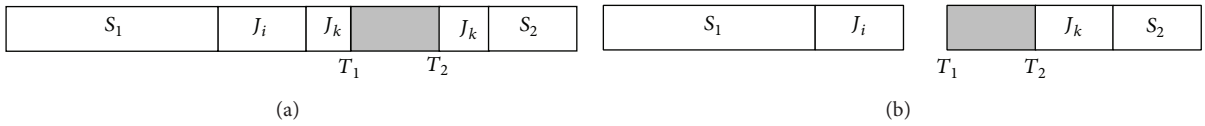

Figure 1: (a) Solution obtained from the Moore and Hodgson algorithm for 1, $h_1|\text{pmtn}|\sum U_i$. (b) Solution obtained from the Moore and Hodgson algorithm for 1, $h_1||\sum U_i$.

$1, h_1|\text{pmtn}|\sum U_i$ is always equal or less than the optimal number of tardy jobs in problem $1, h_1||\sum U_i$. This is because, in the optimal schedule of the problem $1, h_1||\sum U_i$, if all the jobs after the nonavailability interval were shifted to the left so that the idle time before the unavailability period is filled, a feasible solution for the problem $1, h_1|\text{pmtn}|\sum U_i$ would be obtained in which the number of tardy jobs would not be more than the optimal number of tardy jobs in problem $1, h_1||\sum U_i$. Thus, the schedule shown in Figure 1(b) illustrates the optimal number of tardy jobs for the problem $1, h_1||\sum U_i$. □

Corollary 7. *According to Theorem 6, before applying any algorithm, the Moore and Hodgson algorithm can be used and its sequence can be achieved. If there is at least one tardy job before T_1 in this sequence, then this sequence is optimal. So, this approach may yield the optimal solution and there will be no need to apply any other algorithms to solve the problem.*

Corollary 8. *If the condition of Theorem 6 is not true for a problem, it means there is no tardy job before T_1 in the optimal schedule, so that, without losing optimality, the jobs before T_1 can be scheduled according to the EDD order. In other words, without satisfying the condition of Theorem 6, the dominant set would be scheduled in which the jobs before T_1 are not tardy and are arranged in the EDD order.*

4. Heuristic H

In this section, a heuristic procedure is developed to solve the problem $1, h_1||\sum U_i$. In this heuristic, δ_h and $N_T(h)$ are defined as the sequence obtained by heuristic H and the target value of this sequence, respectively. The steps in heuristic H are as follows.

Step 0. Begin.

Set $k = n$, $I = \{J_1, J_2, \ldots, J_n\}$ and $L = I$.

Step 1. Calculate the value of this term:

$$\text{sum} = \sum_{i=1}^{n} p_i + (T_2 - T_1) + \max_{1\le i\le n}\{p_i\} - 1. \tag{2}$$

Step 2. Check Dominance Rule 1.

If there is job J_i such that $d_i \ge$ sum, then go to Step 3. Otherwise, go to Step 4.

Step 3. Put job J_i in position k.

Set $L = L - \{J_i\}$, sum = sum $- p_i$, and $k = k - 1$; if $k = 0$ stop. Otherwise, return to Step 2.

Step 4. Put the jobs in set L in the order of the Moore and Hodgson algorithm. Designate the achieved schedule as δ_{moore} and the target value obtained through this sequence as $N_T(\text{moore})$.

If $N_T(\text{moore}) = 0$, the obtained schedule is optimal. Set $\delta_h = \delta_{\text{moore}}$ and $N_T(h) = N_T(\text{moore})$; stop.

Step 5. Check Corollary 7.

If there is any tardy job before T_1 in the schedule δ_{moore}, this schedule is optimal. Set $\delta_h = \delta_{\text{moore}}$ and $N_T(h) = N_T(\text{moore})$; stop.

Step 6. Designate the set of jobs before T_1 as I' and the set of jobs after T_2 as I''.

Set $k' = |I'|$, $k'' = |I''|$. Job I'_i, $i = 1, 2, \ldots, k'$, is defined as the ith job in set I' and job I''_j, $j = 1, 2, \ldots, k''$, is defined as the jth job in set I''. Put $i = 1$ and $j = 1$.

Step 7. Job I'_i is the candidate for exchanging with job I''_j. If by eliminating job I'_i from the set I', the possibility exists for job I''_j to be assigned before the nonavailability period, and the schedule δ' is obtained by exchanging these two jobs in the schedule δ_{moore}. Otherwise, go to Step 11.

Step 8. In the schedule δ', put the jobs before the nonavailability interval in the EDD order and arrange the set of jobs after the nonavailability interval based on the Moore and Hodgson algorithm.

Step 9. Check Corollary 8.

If there is at least one tardy job before the nonavailability interval, then this schedule will not be in the set of dominant solutions and will not be accepted; go to Step 11.

Step 10. Calculate the target value of schedule δ' and designate it as $N_T(\delta')$.

If $N_T(\delta') < N_T(\text{moore})$, then schedule δ' is optimal. Set $\delta_h = \delta'$ and $N_T(h) = N_T(\delta')$; stop.

Step 11. If $j \ne k''$, set $j = j + 1$ and go to Step 7.

Step 12. If $i = k'$, set $\delta_h = \delta_{\text{moore}}$ and $N_T(h) = N_T(\text{moore})$; stop. Otherwise, set $i = i + 1$ and go to Step 7.

In summary, through Steps 1–3, the Dominance Rule 1 condition is checked for each job; by applying this rule for the jobs with this condition, the solution space is reduced. In Step 4, a feasible schedule is obtained by applying the Moore and Hodgson algorithm. In Step 5, the optimality condition of Corollary 7 is checked. If this condition is true, the schedule

Table 1: The data for Example 9.

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7
p_i	4	10	5	9	2	6	4
d_i	52	26	48	24	25	26	23

Table 2: Applying Dominance Rule 1 for Example 9.

J_i	J_2	J_4	J_5	J_6	J_7	J_1	J_3
p_i	10	9	2	6	4	4	5
d_i	26	24	25	26	23	52	48

Table 3: The initial schedule for Example 9.

J_i	J_7	J_5	J_2	J_6	J_4
p_i	4	2	10	6	9
C_i	4	6	23	29	38
d_i	23	25	26	26	24

Table 4: The second schedule for Example 9.

J_i	J_5	J_6	J_7	J_2	J_4
p_i	2	6	4	10	9
C_i	2	8	17	27	36
d_i	25	26	23	26	24

Table 5: The final schedule for Example 9.

J_i	J_7	J_6	J_2	J_5	J_4
p_i	4	6	10	2	9
C_i	4	10	23	25	34
d_i	23	26	26	25	24

obtained will be optimal and the solution procedure will be stopped. If not, we try to improve the solution of Moore and Hodgson algorithm by a procedure of pairwise exchange. Since, according to Corollary 1, the solution obtained through the Moore and Hodgson algorithm has at most one tardy job in difference with the optimal solution, gaining the first improvement in this heuristic, therefore, and means achieving the optimal solution.

Example 9. As an illustration of the heuristic H, consider a single machine scheduling problem with seven jobs, as given in Table 1, where $T_1 = 10$ and $T_2 = 13$.

Applying Steps 1–3, the jobs J_1 and J_3 are removed and placed at the end of schedule, as given in Table 2. In Step 4, the schedule δ_{moore}, as shown in Table 3, is obtained, in which $N_T(\text{Moore}) = 2$. Since the condition in Step 5 is not true for the achieved solution, according to Step 7, in the schedule δ_{moore}, the job J_7 is exchanged with the job J_6 and the schedule δ', corresponding to Table 4, with $N_T(\delta') = 2$, is achieved. Because the inequality $N_T(\delta') < N_T(\text{moore})$ is not satisfied, the exchanging process is continued. Finally, by exchanging the jobs J_5 and J_6 in δ_{moore}, the schedule δ', as shown in Table 5, is obtained, in which $N_T(\delta') = 1$, and since $N_T(\delta') < N_T(\text{moore})$, we can conclude that the schedule $(J_7 - J_6 - J_2 - J_5 - J_4 - J_1 - J_3)$ is optimal and, therefore, the solving procedure is stopped.

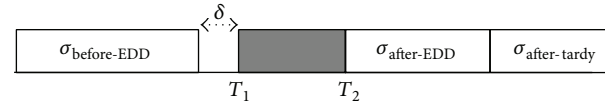

Figure 2: The partial schedule σ.

5. Branch-and-Bound Approach

According to the results stated above, a binary branch-and-bound (BB) is developed in this paper to solve problem $1, h_1 || \sum U_i$. According to this algorithm, the basis of branching is assigning a job before or after the nonavailability interval. After any branching, the sets of jobs before and after the unavailability period are arranged based on the Moore and Hodgson algorithm. The search strategy in this procedure is backtracking. The following notations are used in the BB approach:

σ: a partial sequence consisting of the set of scheduled jobs;

σ': the set of unscheduled jobs which is complementary to σ;

$\sigma_{\text{before-EDD}}$: the set of jobs assigned before the unavailability period from time zero;

$\sigma_{\text{after-EDD}}$: the set of nontardy jobs assigned after the unavailability period from time T_2;

$\sigma_{\text{after-tardy}}$: the set of tardy jobs assigned after the unavailability interval and after the set $\sigma_{\text{after-EDD}}$;

$C(\sigma_{\text{before-EDD}})$: completion time of the set $\sigma_{\text{before-EDD}}$ or the completion time of the last job scheduled before the unavailability interval;

$C(\sigma_{\text{after-tardy}})$: completion time of the set $\sigma_{\text{after-tardy}}$ or the completion time of the last job scheduled after the unavailability period;

δ: idle time immediately before T_1.

Assume that indices of the jobs are based on the EDD order and that this order determines the jobs' priority for entering into the tree. The partial schedule σ consists of three sets as $\sigma_{\text{before-EDD}}$, $\sigma_{\text{after-EDD}}$, and $\sigma_{\text{after-tardy}}$. This schedule is shown in Figure 2. Jobs in the set $\sigma_{\text{before-EDD}}$ are scheduled based on the EDD order before the nonavailability period from time zero. Jobs in the set $\sigma_{\text{after-EDD}}$ are scheduled immediately after the nonavailability period, that is, at time T_2, in the EDD order. Finally, jobs in the set $\sigma_{\text{after-tardy}}$ are placed immediately after the set $\sigma_{\text{after-EDD}}$, that is, at the end of the schedule, in an optional order.

In the proposed BB, each time a job is added to a set of assigned jobs, two branches will be made. One of them refers to assigning the job before the nonavailability interval while the other refers to assigning it after the nonavailability interval.

If a job is assigned before the unavailability period, it will be added to the end of the set $\sigma_{\text{before-EDD}}$, but if it is assigned after the unavailability period, it will be assigned at the end of the jobs in the set $\sigma_{\text{after-EDD}}$. If it becomes a tardy job, by adding it in the set $\sigma_{\text{after-EDD}}$, the job with maximum processing time will be selected between this job and other jobs of the set $\sigma_{\text{after-EDD}}$ and removed from the set $\sigma_{\text{after-EDD}}$ to be added to the jobs in the set $\sigma_{\text{after-tardy}}$. Also, the value of the target function will be increased by one unit. In the rest of this section, the lower and upper bounds and some dominance rules will be presented for the BB procedure.

5.1. Upper Bound. In the proposed BB, heuristic H is executed first. According to the optimality conditions in heuristic H, if we recognize that the sequence obtained from this algorithm is optimal, then it will be considered as the optimal solution of the problem and the solution procedure will be stopped. Otherwise, the main algorithm of BB will be performed. In this case, the solution obtained through heuristic H will be considered as the upper bound of the problem.

5.2. Lower Bound. As mentioned earlier, if heuristic H cannot improve the solution obtained from the Moore and Hodgson algorithm, its value will be considered as an upper bound. In this state, and according to Corollary 1, by subtracting one tardy job from the upper bound, we can use it as a lower bound for problem $1, h_1 || \sum U_i$. To obtain a lower bound for the partial sequence σ, the following theorem is used.

Theorem 10. *In the problem $1, h_1 || \sum U_i$, for a partial sequence σ, a lower bound is obtained from the following expression:*

$$LB(\sigma) = N_T(\sigma) + N_T\left(\sigma'\right) \tag{3}$$

in which $N_T(\sigma)$ is the number of tardy jobs in the schedule σ and $N_T(\sigma')$ is the number of tardy jobs in the set σ', while the jobs in the set σ' are scheduled immediately after $C(\sigma_{\text{before-EDD}})$ by the Moore and Hodgson algorithm assuming that resumption is allowed.

Proof. Clearly, in the partial schedule σ, the value $N_T(\sigma)$ is always fixed and independent of scheduling the jobs in the set σ'. Also, we know that the starting time of jobs in the set σ' is not earlier than $C(\sigma_{\text{before-EDD}})$. Now, if we ignore jobs after the unavailability period and schedule jobs in the set σ' at time $C(\sigma_{\text{before-EDD}})$ while assuming that resumption is allowed to obtain $N_T(\sigma')$, the number of tardy jobs will in no way be smaller than $N_T(\sigma')$ no matter how the schedule σ is completed. So, the lower bound of the schedule σ is obtained from the sum of values $N_T(\sigma)$ and $N_T(\sigma')$. □

5.3. Dominance Rules. The following dominance rules are utilized in formulating our BB procedure.

Lemma 11 (Dominance Rule 2). *In the BB tree, if a job assigned before the nonavailability period is tardy, this node will be fathomed and the search process will continue from other nodes and branches.*

Proof. According to Corollary 8, if there is no tardy job in problem $1, h_1 || \sum U_i$ before the unavailability period in the solution obtained from the Moore and Hodgson algorithm, there will neither be any tardy jobs before T_1 in the optimal solution. □

Lemma 12 (Dominance Rule 3). *In the BB tree, if a job is decided to be assigned before the unavailability period, but there is no possibility for such assignment, this node will then be fathomed.*

Proof. In fact, by assigning this job before the unavailability interval, an unfeasible solution will be obtained and there will be no need for continuing the search process from this node. □

Lemma 13 (Dominance Rule 4). *If the inequality condition $T_1 - C(\sigma_{\text{before-EDD}}) < \min_{J_i \in \sigma'}\{p_i\}$ holds in the partial schedule σ, all the jobs in set σ' will be assigned after the nonavailability interval.*

Proof. Regarding the available time before T_1, that is, $T_1 - C(\sigma_{\text{before-EDD}})$, it is clear that even the job with the smallest processing time cannot be inserted before the nonavailability period. So, all the jobs in this set will be added to the set $\sigma_{\text{after-EDD}}$ and the Moore and Hodgson algorithm will be executed for this set. □

6. Computational Results

In this section, a set of instances will be analyzed in order to evaluate the performance of heuristic H and the BB approach. These instances are generated and solved on a PIV PC with 3.4 GB CPU and 1 GB RAM in the Windows XP environment. In the rest of this section, the instance generation method and the analysis of the results will be presented. Then, comparisons will be made between the results obtained through these algorithms and those from Chen's [15] algorithms.

6.1. Data Generation. Processing times for problem instances are randomly generated using the discrete uniform distribution in the range [1, 10]. According to [11, 13, 15], due dates are randomly generated from the discrete uniform distribution in the range $[(1 - C - Q/2)\sum_{k=1}^{n} p_{[k]}, (1 - C + Q/2)\sum_{k=1}^{n} p_{[k]}]$, in which Q represents the range of due date factor and C denotes the tardiness factor. Either of the two values 0.2 and 0.6 is assumed for the two parameters C and Q. According to [8, 10], to evaluate the impact of start times of the non-availability period on the performance of algorithms, the values T_1 and T_2 are generated from the following data sets.

Data set 1: $T_1 = 1/4\sum_i p_i$ and $T_2 = T_1 + 1/n\sum_i p_i$.

Data set 2: $T_1 = 1/2\sum_i p_i$ and $T_2 = T_1 + 1/n\sum_i p_i$.

Data set 3: $T_1 = 3/4\sum_i p_i$ and $T_2 = T_1 + 1/n\sum_i p_i$.

For the number of jobs, n, the values are taken from 10 to 10000. For each possible combination of T_1, C, Q, and n, 20 instances are randomly generated ($np = 20$). So, for all

possible combinations of the start time of the unavailability period, tardiness factor, and due date factor 12 ($2 \times 2 \times 3$) series are generated with a total number of 2680 instances.

6.2. Numerical Results. Heuristic H and the BB procedure are coded in C++ programming language to solve the problem instances. For solving each problem through the BB approach, the time constraint of 4000 seconds is considered such that if the problem cannot be solved optimally within this time limit, the BB approach will be stopped for it.

Numerical results for 12 series are shown in Table 6, the symbol ">0" in the table reveals the 0.00 seconds computational time. In this table, the column "Number of optimal instances BB" shows the number of instances for each n that are optimally solved by the BB approach. As shown here, except for the series S_{11}, S_{13}, and S_{21}, optimal solutions are achieved for all instances in the others.

The column "Number of optimal instances H" shows the number of instances in which the solution obtained from algorithm H is optimal. The results indicate that optimal solutions are obtained for all instances by heuristic H in the series S_{23}, S_{33}, and S_{34} and that this heuristic yields very good results in other series.

The column "Comp. time of BB" represents the minimum, average, and maximum times required to solve instances by the BB approach. It should be mentioned that because heuristic H is applied as an upper bound in the BB approach, its computation time is included in that of the BB approach. Additionally, because the computation time of heuristic H was 0.00 second to two decimals accuracy for all instances, the time required for applying the heuristic algorithm is not shown separately in this table.

Comparison of solution times for all instances reveals that the greatest computation times belong to the series S_{11}, S_{21}, and S_{13} while the smallest ones belong to the series S_{32}, S_{33}, and S_{34}.

From these observations, it can be inferred that the most difficult instances belong to the series S_{11} and S_{21}. Instances in these series have greater average times and fewer optimal solutions than the other series. The reason for this may be that the range for producing due dates in these series are smaller considering the values $C = 0.2$ and $Q = 0.2$ and that there would be accordingly more equal due dates for different jobs. On the other hand, because T_1 is smaller, most jobs have tardiness due to their assignment after the nonavailability period. These will cause the Moore and Hodgson algorithm to be applied more for scheduling the jobs after the unavailability period.

Following the series S_{11} and S_{21}, the most difficult instances belong to the series S_{13}. Considering the values $C = 0.6$ and $Q = 0.2$, the reason for this is that, in this series, firstly, the range of due dates is small; secondly, jobs have smaller due dates; and, finally, T_1 has its smallest value. In the instances of this series, therefore, most of the jobs are scheduled after the nonavailability period and, so, they are tardy ones. Hence, in this series the Moore and Hodgson algorithm is applied more frequently.

Columns related to "Number of optimal states in H" demonstrate the amount of instances in which the solution obtained by heuristic H is identified as optimal, and the solution procedure is, therefore, stopped. The column "Number of optimality states in H for D_1" shows the number of instances in which the condition of Dominance Rule 1 is true for all jobs and, therefore, the optimal schedule is achieved by applying this rule and the solution process is stopped. The comparison of these columns reveals that this rule works rather properly in the instances with $n \geq 40$ in the series S_{12}, S_{22}, and S_{32} such that by applying this rule optimal solutions are obtained. This observation can be justified as follows: based on the values $C = 0.2$ and $Q = 0.6$, first of all, due dates are more scattered in these series and, secondly, the upper bound of the due date range is increased so that with the increasing the number of jobs, more jobs will satisfy the condition of Dominance Rule 1.

The column "Number of optimality states in H for C_4" shows the number of instances in which the solution obtained through the Moore and Hodgson algorithm satisfies the optimality condition of the Moore and Hodgson algorithm in Corollary 7. In these instances, it is recognized that the solution obtained from the Moore and Hodgson algorithm is optimal, and the solution procedure is, therefore, stopped. These columns show that in the series S_{33} and S_{34}, the condition in Corollary 7 is true for all instances. This is because the dispersion of due dates in these series is high and the lower bound of the due date range is decreased. Also, T_1 is as high as possible in these series. In the series S_{32} and S_{23}, Corollary 7 has a quite proper application.

The column "Number of optimality states in H for others" shows the number of instances in which other conditions of optimality H are satisfied and thus the solution procedure is stopped.

In some series, all instances with problem sizes specified by the symbol "$*$" reach optimal solutions by heuristic H before entering the BB tree; hence, the main BB algorithm is not executed for them.

Columns "Ave. Percent fathomed nodes" show the average number of nodes fathomed because of the dominance rules and the lower bound. In these columns, the terms D_2, D_3, D_4, and LB refer to the Dominance Rules 2, 3, and 4, and the lower bound, respectively. Dominance Rule 2 works properly in the series S_{11} and S_{12}. This rule also has rather proper application in the series S_{13}. The reason may be that, in these series, the value of T_1 is as low as possible and that, in the BB tree, there is a higher probability for the occurrence of states in which a tardy job is placed before the nonavailability interval.

The lower bound, LB(σ), in all the series applied quite satisfactorily. In many instances, the very high efficiency of the lower bound left fewer opportunities for the dominance rules to apply.

Comparisons were made between the proposed algorithms and Chen's [15] procedures to demonstrate the efficiency of our proposed algorithms. In the single machine problem, with periodic maintenance and nonresumable jobs, if the time between two consecutive maintenance periods is considered to be $3/4 \sum_i p_i$ and the time length of each

Table 6: Results of 12 series, $np = 20$.

Series	Number of jobs[1]	Number of optimal instances		Comp. time of BB (s)			Number of optimal states in H			Ave. percent fathomed nodes			
		BB	H	Min	Ave.	Max	D_1	C_4	Others	D_2	D_3	D_4	LB
S_{11} $T_1 = 1/4\sum pi$ $T_2 = T_1 + 1/n\sum p$ $C = 0.2$ $Q = 0.2$	10	20	20		>0		0	0	2	1.09	>0	14.23	41.84
	20	20	20		0.01		0	0	9	0.35	>0	6.12	35.29
	30	20	20		5.57		0	0	2	>0	>0	7.04	44.98
	40	20	20		736.26		0	0	6	1.47	>0	4.83	37.41
	50	11	11		>0		0	0	4	>0	>0	>0	63.64
S_{12} $T_1 = 1/4\sum pi$ $T_2 = T_1 + 1/n\sum pi$ $C = 0.2$ $Q = 0.6$	10	20	19	>0	>0	>0	0	0	0	10.20	>0	9.92	61.45
	20	20	20	>0	>0	0.02	0	0	12	25.21	>0	11.32	24.82
	30	20	19	>0	0.28	5.52	4	0	14	17.58	>0	1.18	57.42
	*40	20	20	>0	>0	>0	15	0	5	—	—	—	—
	*50	20	20	>0	>0	>0	19	0	1	—	—	—	—
	*70	20	20	>0	>0	0.02	19	0	1	—	—	—	—
	*100	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*300	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*500	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*700	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*1000	20	20	>0	>0	0.03	20	0	0	—	—	—	—
	*3000	20	20	>0	0.01	0.06	20	0	0	—	—	—	—
	*5000	20	20	>0	0.01	0.06	20	0	0	—	—	—	—
	*7000	20	20	>0	0.01	0.02	20	0	0	—	—	—	—
	*10000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
S_{13} $T_1 = 1/4\sum pi$ $T_2 = T_1 + 1/n\sum pi$ $C = 0.6$ $Q = 0.2$	10	20	19		>0		0	0	3	1.64	>0	1.57	67.69
	20	20	20		>0		0	0	4	>0	>0	>0	80.00
	30	20	20		0.18		0	0	4	>0	>0	1.16	76.61
	40	20	20		100.23		0	0	7	1.35	>0	3.16	38.99
	50	12	12		160.56		0	0	5	>0	>0	0.41	51.81
S_{14} $T_1 = 1/4\sum pi$ $T_2 = T_1 + 1/n\sum pi$ $C = 0.6$ $Q = 0.6$	10	20	20	>0	>0	>0	0	0	1	1.07	7.92	3.05	72.85
	20	20	19	>0	>0	>0	0	0	5	>0	2.50	0.24	68.69
	30	20	20	>0	>0	0.02	0	0	2	>0	>0	>0	90.00
	40	20	20	>0	>0	0.02	0	0	8	>0	>0	>0	60.00
	50	20	20	>0	0.01	0.02	0	0	6	>0	>0	>0	70.00
	70	20	20	0.02	0.03	0.05	0	0	6	>0	>0	>0	70.00
	100	20	20	0.03	0.10	0.16	0	0	3	>0	>0	>0	85.00
	300	20	20	0.09	5.71	8.23	0	0	6	>0	>0	>0	70.00
	500	20	20	29.92	49.08	79.83	0	0	4	>0	>0	>0	80.00
	700	20	20	130.91	198.40	301.61	0	0	6	>0	>0	>0	70.00
	1000	20	20	503.92	692.73	1178.98	0	0	2	>0	>0	>0	90.00
S_{21} $T_1 = 1/2\sum pi$ $T_2 = T_1 + 1/n\sum pi$ $C = 0.2$ $Q = 0.2$	10	20	20	>0	>0	0.02	0	0	2	3.96	>0	2.29	74.30
	20	20	20	>0	0.02	0.11	0	0	5	5.82	>0	7.93	50.63
	30	20	20	>0	10.48	47.77	0	0	5	9.06	>0	8.49	43.20
	40	10	9	>0	624.11	3815.59	0	0	2	8.59	>0	2.23	54.43
	50	12	12	>0	>0	0.02	0	0	6	>0	>0	>0	50.00

Table 6: Continued.

Series	Number of jobs[1]	Number of optimal instances		Comp. time of BB (s)			Number of optimal states in H			Ave. percent fathomed nodes			
		BB	H	Min	Ave.	Max	D_1	C_4	Others	D_2	D_3	D_4	LB
S_{22} $T_1 = 1/2 \sum pi$ $T_2 = T_1 + 1/n \sum pi$ $C = 0.2$ $Q = 0.6$	10	20	19	>0	>0	>0	0	0	1	1.98	>0	2.81	82.47
	20	20	18	>0	>0	0.02	0	0	8	2.60	>0	1.04	50.00
	30	20	20	>0	0.02	0.47	2	0	16	13.31	>0	2.64	6.17
	40	20	13	>0	>0	>0	14	0	5	>0	>0	>0	5.00
	*50	20	20	>0	>0	>0	17	0	3	—	—	—	—
	*70	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*100	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*300	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*500	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*700	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*1000	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*3000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*5000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*7000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*10000	20	20	>0	0.01	0.02	20	0	0	—	—	—	—
S_{23} $T_1 = 1/2 \sum pi$ $T_2 = T_1 + 1/n \sum pi$ $C = 0.6$ $Q = 0.2$	10	20	20	>0	>0	>0	0	2	0	>0	>0	>0	90.00
	20	20	20	>0	>0	>0	0	8	0	>0	>0	>0	60.00
	30	20	20	>0	>0	>0	0	7	0	>0	>0	>0	65.00
	40	20	20	>0	>0	0.02	0	7	0	>0	>0	>0	65.00
	50	20	20	>0	>0	0.02	0	6	0	>0	>0	>0	70.00
	70	20	20	>0	0.01	0.06	0	7	0	>0	>0	>0	65.00
	100	20	20	>0	0.05	0.13	0	4	0	>0	>0	>0	80.00
	300	20	20	>0	2.54	6.67	0	6	0	>0	>0	>0	70.00
	500	20	20	>0	15.26	55.81	0	9	0	>0	>0	>0	55.00
	700	20	20	>0	96.49	197.95	0	5	0	>0	>0	>0	75.00
	1000	20	20	0.02	135.76	742.70	0	11	0	>0	>0	>0	45.00
S_{24} $T_1 = 1/2 \sum pi$ $T_2 = T_1 + 1/n \sum pi$ $C = 0.6$ $Q = 0.6$	10	20	20	>0	>0	>0	0	0	0	>0	5.99	1.17	87.05
	20	20	18	>0	>0	>0	0	0	2	>0	0.29	0.90	83.31
	30	20	18	>0	>0	0.02	0	0	2	>0	0.96	1.60	73.71
	40	20	20	>0	>0	0.02	0	0	2	>0	>0	>0	90.00
	50	20	19	>0	0.01	0.06	0	0	3	>0	0.11	0.36	78.18
	70	20	20	>0	0.01	0.03	0	0	3	>0	>0	>0	85.00
	100	20	20	0.03	0.06	0.13	0	0	3	>0	>0	>0	85.00
	300	20	20	0.02	3.75	7.00	0	0	7	>0	>0	>0	65.00
	500	20	20	>0	26.79	55.25	0	0	6	>0	>0	>0	70.00
	700	20	20	60.48	124.32	212.69	0	0	5	>0	>0	>0	75.00
	1000	20	20	0.94	514.41	871.05	0	0	6	>0	>0	>0	70.00

Table 6: Continued.

Series	Number of jobs[1]	Number of optimal instances		Comp. time of BB (s)			Number of optimal states in H			Ave. percent fathomed nodes			
		BB	H	Min	Ave.	Max	D_1	C_4	Others	D_2	D_3	D_4	LB
S_{31} $T_1 = 3/4 \sum pi$ $T_2 = T_1 + 1/n \sum pi$ $C = 0.2$ $Q = 0.2$	10	20	20	>0	>0	>0	0	0	0	>0	>0	>0	100.00
	20	20	17	>0	>0	>0	0	0	3	0.87	>0	3.21	70.62
	30	20	19	>0	>0	>0	0	0	4	1.36	>0	0.07	76.25
	40	20	20	>0	>0	>0	0	0	9	>0	>0	>0	55.00
	50	20	20	>0	>0	0.02	0	0	6	>0	>0	>0	70.00
	70	20	20	>0	0.01	0.03	0	0	5	>0	>0	>0	75.00
	100	20	20	>0	0.02	0.05	0	0	6	>0	>0	>0	70.00
	300	20	20	0.02	1.26	2.81	0	0	10	>0	>0	>0	50.00
	500	20	20	0.02	7.04	16.78	0	0	10	>0	>0	>0	50.00
	700	20	20	7.41	37.80	72.25	0	0	6	>0	>0	>0	70.00
	1000	20	20	0.02	149.77	290.13	0	0	6	>0	>0	>0	70.00
S_{32} $T_1 = 3/4 \sum pi$ $T_2 = T_1 + 1/n \sum pi$ $C = 0.2$ $Q = 0.6$	10	20	19	>0	>0	>0	0	0	2	0.99	>0	1.03	90.50
	20	20	18	>0	>0	>0	0	7	4	3.06	>0	0.95	79.33
	30	20	19	>0	>0	>0	3	14	2	10.20	>0	8.16	16.33
	40	20	19	>0	>0	>0	11	7	1	>0	>0	>0	100.00
	*50	20	20	>0	>0	>0	18	2	0	—	—	—	—
	*70	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*100	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*300	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*500	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*700	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*1000	20	20	>0	>0	>0	20	0	0	—	—	—	—
	*3000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*5000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*7000	20	20	>0	>0	0.02	20	0	0	—	—	—	—
	*10000	20	20	>0	0.01	0.02	20	0	0	—	—	—	—
S_{33} $T_1 = 3/4 \sum pi$ $T_2 = T_1 + 1/n \sum pi$ $C = 0.6$ $Q = 0.2$	*10	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*20	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*30	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*40	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*50	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*70	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*100	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*300	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*500	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*700	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*1000	20	20	0.02	0.02	0.03	0	20	0	—	—	—	—
	*3000	20	20	0.14	0.16	0.22	0	20	0	—	—	—	—
	*5000	20	20	0.44	0.45	0.50	0	20	0	—	—	—	—
	*7000	20	20	0.86	0.88	0.92	0	20	0	—	—	—	—
	*10000	20	20	1.78	1.84	2.03	0	20	0	—	—	—	—

Table 6: Continued.

Series	Number of jobs[1]	Number of optimal instances		Comp. time of BB (s)			Number of optimal states in H			Ave. percent fathomed nodes			
		BB	H	Min	Ave.	Max	D_1	C_4	Others	D_2	D_3	D_4	LB
	10	20	20	>0	>0	>0	0	15	0	>0	>0	>0	25.00
	20	20	20	>0	>0	>0	0	18	0	>0	>0	>0	10.00
	*30	20	20	>0	>0	>0	0	20	0	—	—	—	—
	*40	20	20	>0	>0	>0	0	20	0	—	—	—	—
S_{34}	*50	20	20	>0	>0	>0	0	20	0	—	—	—	—
$T_1 = 3/4 \sum pi$	*70	20	20	>0	>0	>0	0	20	0	—	—	—	—
$T_2 = T_1 + 1/n \sum pi$	*100	20	20	>0	>0	>0	0	20	0	—	—	—	—
$C = 0.6$	*300	20	20	>0	>0	0.02	0	20	0	—	—	—	—
$Q = 0.6$	*500	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*700	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*1000	20	20	>0	>0	0.02	0	20	0	—	—	—	—
	*3000	20	20	0.11	>0	0.16	0	20	0	—	—	—	—
	*5000	20	20	0.34	>0	0.38	0	20	0	—	—	—	—
	*7000	20	20	0.70	>0	0.72	0	20	0	—	—	—	—
	*10000	20	20	1.45	0.01	1.50	0	20	0	—	—	—	—

[1]Not reported instances could not be solved by BB approach in 4000 seconds.

maintenance period is taken to be $1/n \sum_i p_i$, this problem will be transferred to single machine problem with a nonavailability interval, with $T_1 = 3/4 \sum_i p_i$ and $T_2 = T_1 + 1/n \sum_i p_i$. So, instances of the series S_{31}, S_{32}, S_{33}, and S_{34} will be solved via Chen's [15] algorithms with the time constraint of 4000 seconds. Table 7 shows the results obtained from problem instances solved by Chen's [15] algorithms. The components in this table are the same as those in Table 6.

It is clear from this table that, for instances with job sizes of 20, 30, 40, and 50 in the series S_{31}, the BB algorithm proposed in [15] was not able to solve all the problems optimally, while the BB algorithm proposed in this paper was capable of optimally solving all the instances with identical sizes.

In the series S_{32}, Chen's [15] BB algorithm failed to optimally solve all the instances with job sizes of 20, 30, and 40. For instances, with $n \geq 70$, based on the assumption that the condition of Dominance Rule 1 holds true for all jobs in most instances, instances were solved more desirably by our proposed algorithm. For the series S_{33} and S_{34} our BB algorithm was clearly more efficient.

Comparison of Tables 6 and 7 reveals that heuristic H had a higher efficiency than the heuristic proposed by Chen [15].

It should be mentioned that, for the problems with multiple unavailability periods, most of the dominance rules proposed in [15] apply properly to eliminate nodes. For this reason, all the instances in this paper in all series by up to a job size of 32 are solved optimally. In problems with a single unavailability period, however, these rules find fewer applications.

7. Conclusion

In this paper, the problem of single machine scheduling with one nonavailability period and the target of minimizing the number of tardy jobs was investigated. In this problem, the starting and finishing times of the nonavailability period are assumed to be given. It is further assumed that all the jobs are nonresumable and are available at time zero.

A number of theorems were proved for this problem, from which a heuristic and a branch-and-bound algorithm with bounds and some dominance rules were proposed. These procedures were tested on 12 series of problem instances. Computational results showed that, in most series, the branch-and-bound approach was capable of solving the problems up to a size of 1000 jobs. It was also observed that the dominance rules and the bounds of the proposed BB algorithm had satisfactory efficiency to eliminate nodes. Based on these results, it can be found that the heuristic H has a high efficiency to solve the problem. This efficiency is because of two reasons; firstly, this algorithm has achieved nearly optimal solutions in small-scale problems and has a little difference with BB approach in obtained results. Secondly, the algorithm has a very low response time to produce good solutions. The results of evaluating the efficiency of the proposed approach as compared with other procedures confirm the greater success of our proposed procedure.

For future study, it is suggested that other techniques of solving the problem optimally such as dynamic programming be investigated. Additionally, solving the problem in more general cases can be considered.

Table 7: Results of instances solved by Chen [15], $np = 20$.

Series	Number of jobs	Number of optimal instances		Comp. time of BB (s)		
		BB	H	Min	Ave.	Max
S_{31}	10	20	20	>0	>0	>0
	20	15	14	>0	17.34	260.13
	30	15	15	>0	>0	>0
	40	9	9	>0	>0	>0
	50	13	13	>0	>0	0.03
S_{32}	10	20	15	>0	0.02	0.38
	20	19	13	>0	166.10	1950.53
	30	17	17	>0	>0	>0
	40	17	17	>0	>0	>0
	50	20	20	>0	>0	>0
	70	20	20	>0	>0	>0
	100	20	20	>0	>0	>0
	300	20	20	>0	>0	0.02
	500	20	20	>0	>0	0.02
	700	20	20	>0	>0	0.02
	1000	20	20	>0	0.01	0.02
	3000	20	20	0.06	0.07	0.11
	5000	20	20	0.17	0.18	0.19
	7000	20	20	0.36	0.36	0.38
	10000	20	20	0.73	0.75	0.83
S_{33}	10	20	20	>0	>0	>0
	20	20	20	>0	>0	0.02
	30	20	20	>0	>0	0.02
	40	20	20	>0	>0	0.02
	50	20	20	>0	>0	0.02
	70	20	20	>0	0.01	0.02
	100	20	20	0.02	0.02	0.03
	300	20	20	0.38	0.40	0.45
	500	20	20	1.80	1.93	2.33
	700	20	20	4.78	5.05	5.61
	1000	20	20	14.03	14.57	15.47
	3000	20	20	388.59	399.03	409.45
	5000	20	20	1858.27	1892.57	1915.16
	7000	0	—	—	—	—
S_{34}	10	20	20	>0	>0	0.05
	20	20	20	>0	>0	0.02
	30	20	20	>0	>0	>0
	40	20	20	>0	>0	>0
	50	20	20	>0	>0	0.02
	70	20	20	>0	>0	0.02
	100	20	20	>0	0.01	0.02
	300	20	20	0.28	0.31	0.34
	500	20	20	1.31	1.41	1.52
	700	20	20	3.69	3.85	4.45
	1000	20	20	10.80	11.25	11.88
	3000	20	20	303.16	309.80	319.99
	5000	20	20	1435.11	1481.24	1618.86
	7000	0	—	—	—	—

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, New Jersey, NJ, USA, 2002.

[2] C.-Y. Lee, "Machine scheduling with an availability constraint," *Journal of Global Optimization*, vol. 9, no. 3-4, pp. 395–416, 1996.

[3] J. B. Sidney, "An extension of Moore's due date algorithm," in *Symposium on the Theory of Scheduling and Its Applications*, S. E. Elmaghrebi, Ed., Lecture Notes in Economics and Mathematical Systems, pp. 393–398, Springer, New York, NY, USA, 1973.

[4] I. Adiri, J. Bruno, E. Frostig, and A. H. G. Rinnooy Kan, "Single machine flow-time scheduling with a single breakdown," *Acta Informatica*, vol. 26, no. 7, pp. 679–696, 1989.

[5] C. Y. Lee and S. D. Liman, "Single machine flow-time scheduling with scheduled maintenance," *Acta Informatica*, vol. 29, no. 4, pp. 375–382, 1992.

[6] C. Sadfi, B. Penz, C. Rapine, J. Blazewicz, and P. Formanowicz, "An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints," *European Journal of Operational Research*, vol. 161, no. 1, pp. 3–10, 2005.

[7] J. Breit, "Improved approximation for non-preemptive single machine flow-time scheduling with an availability constraint," *European Journal of Operational Research*, vol. 183, no. 2, pp. 516–524, 2007.

[8] I. Kacem and C. Chu, "Efficient branch-and-bound algorithm for minimizing the weighted sum of completion times on a single machine with one availability constraint," *International Journal of Production Economics*, vol. 112, no. 1, pp. 138–150, 2008.

[9] I. Kacem, "Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 401–410, 2008.

[10] I. Kacem, C. Chu, and A. Souissi, "Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times," *Computers & Operations Research*, vol. 35, no. 3, pp. 827–844, 2008.

[11] C. J. Liao and W. J. Chen, "Single-machine scheduling with periodic maintenance and nonresumable jobs," *Computers and Operations Research*, vol. 30, no. 9, pp. 1335–1347, 2003.

[12] M. Ji, Y. He, and T. C. E. Cheng, "Single-machine scheduling with periodic maintenance to minimize makespan," *Computers and Operations Research*, vol. 34, no. 6, pp. 1764–1770, 2007.

[13] W.-J. Chen, "An efficient algorithm for scheduling jobs on a machine with periodic maintenance," *International Journal of Advanced Manufacturing Technology*, vol. 34, no. 11-12, pp. 1173–1182, 2007.

[14] I. Adiri, E. Frostig, and A. H. G. Rinnooy Kan, "Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs," *Naval Research Logistics*, vol. 38, no. 2, pp. 261–271, 1991.

[15] W.-J. Chen, "Minimizing number of tardy jobs on a single machine subject to periodic maintenance," *Omega*, vol. 37, no. 3, pp. 591–599, 2009.