

Single-machine scheduling with periodic and flexible periodic maintenance to minimize maximum tardiness

Mohammed Sbihi* and Christophe Varnier*

* Laboratoire d'Automatique de Besançon, CNRS-ENSMM-UFC

24, rue Alain Savary,

25000 Besançon, France.

E-mail: msbihi@ens2m.fr, cvarnier@ens2m.fr

Abstract

This paper considers a single machine scheduling problem with several maintenance periods. Specifically, two situations are investigated. In the first one, maintenance periods are periodically fixed: maintenance is required after a periodic time interval. In the second one, the maintenance is not fixed but the maximum continuous working time of the machine which is allowed is determined. The objective is to minimize the maximum tardiness. These problems are known to be strongly NP-hard. We propose some dominance properties and an efficient heuristic. Branch-and-bound algorithms, in which the heuristics, the lower bounds and the dominance properties are incorporated, are proposed and tested computationally.

Keywords: Scheduling; Periodic maintenance; Maximum tardiness; Non resumable job.

1 Introduction

The majority of the studies in machine scheduling literature assumes that machines are available all times. However, this availability may not be true in real industry settings. Unavailability periods often appear in industry due to a machine breakdown (stochastic) or preventive maintenance (deterministic) during the scheduling period. Therefore, a more realistic scheduling model should take into account associated machine maintenance activities. For a survey of scheduling problems with limited machine availability we refer the reader to the survey paper [15]. Recent papers take into account these unavailability periods (see for instance [1], [2], [3], [6], [8], [9], [12], [14] etc...). Most of these papers consider that characteristics of the unavailability periods are known in advance, *i.e.* that the maintenance of a machine is a fixed time interval known beforehand and study how to schedule jobs under the constraint of machine unavailability. However, in some cases (e.g. preventive maintenance), the maintenance of a machine is also controllable. In other words, we can decide when to maintain the machine. The jobs and the maintenances are scheduled simultaneously. In this paper, we study the problem of minimizing maximum tardiness of jobs.

In most of papers, there is only one unavailability or availability period for each machine [15]. As stated earlier, however, maintenance is scheduled regularly, or periodically, in many manufacturing systems. Therefore, there is a need to develop scheduling methods to deal with periodic maintenance, which usually has more than one maintenance period. In our problem, there are several maintenance periods where each maintenance is required after a time interval.

More precisely, in this paper we consider the single machine maximum tardiness problem subject to periodic maintenance and no preemption (*i.e.* once a job is started it must be completed without interruption). We consider two situations. In the first case (periodic

maintenance) the maintenance periods are fixed periodically. This problem was considered in [4] and [11]; and is NP-hard since the problem that minimizes the maximum lateness subject to one unavailability period and non resumable jobs is NP-hard [10]. Chen and Liao [4] and [11] proposed a heuristic and branch-and-bound algorithm to solve this problem. Other works can be found in the literature regarding fixed maintenance period. We can quote Allaoui and Artiba [2]. They consider the problem of more complex environment such as hybrid flowshop. They proposed an hybrid approach, combining simulation and heuristics to solve the problem of minimizing makespan, mean flowtime and total tardiness. They showed that classical heuristics became ineffective if unavailability constraints, such as maintenance, are taking into account. In [3] same authors proposed also a branch and bound procedure for minimizing the makespan in a two stage hybrid flowshop, where the first stage has only one machine and all machines are subject to unavailability constraints. In the second case (flexible periodic maintenance) that we consider, the maintenance is not fixed but the maximum continuous working time allowed of the machine is fixed. This problem is strongly NP-hard because it becomes a bin packing problem when all due dates are 0. It is known that bin packing is a strongly NP-hard problem [5]. Note that the same problem was studied in [13] in order to minimize the total completion time of jobs, where several heuristics and a branch-and-bound algorithm are proposed. Graves and Lee [7] study several variants of the same problem.

The rest of this paper is organized as follows. Section 2 defines the problems and introduces some notation. Sections 3 and 4 are devoted respectively to periodic maintenance and flexible periodic maintenance. Various dominance rules (different from that of [4] and [11]) are derived and used in the heuristic and branch and bound algorithm proposed. Section 5 reports on computational experience with the algorithms. Finally, Section 6 provides a summary of the main results of this paper, and outlines some useful directions for future research.

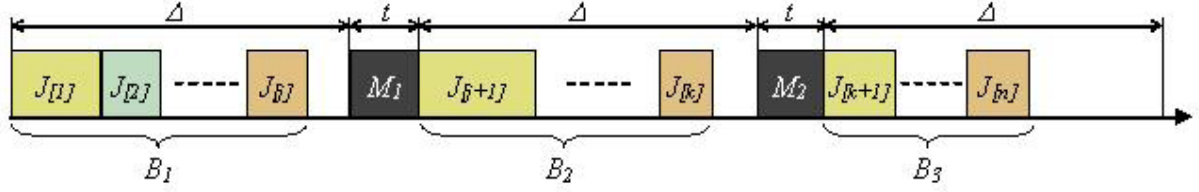


Figure 1: A schedule with periodic maintenance : $J_{[i]}$ is the number of job in i^{th} position and M_i is the i^{th} operation of maintenance

2 Problem definition and notation

We consider the scheduling problem with n jobs J_1, J_2, \dots, J_n to be processed on a single machine. Each job J_i has a processing time p_i and a due date d_i . All jobs are available at time zero and no preemption is allowed. The machine must be maintained periodically. In the periodic version the distance between two consecutive maintenance periods is Δ (see Figure 1) whereas in the flexible periodic version the distance is less than Δ (see Figure 2). The maintenance time is t . We assume that $p_i \leq \Delta$ for all $i \in \{1, \dots, n\}$ otherwise there is no feasible schedule. A sequence giving the processing order of the jobs defines a schedule since there is no advantage in keeping the machine idle when there are jobs to be processed. A schedule $\pi = (J_{[1]}, J_{[2]}, \dots, J_{[n_1]}, M, J_{[n_1+1]}, \dots, J_{[n_2]}, M, \dots)$ contains a sequence of jobs and the maintenance inserted in job sequence. In a schedule, jobs processed continuously form a batch, denoted as B . Thus a schedule π can be denoted as $\pi = (B_1, t, B_2, t, \dots, t, B_L)$ where t is the maintenance and L is the number of batches. Note that L is a decision variable in our problem. Let $J_{[i]}$ be the i th job in a schedule, $C_i(\pi)$ be the completion time of job J_i , and $T_i(\pi) = \max\{C_i(\pi) - d_i; 0\}$ be the tardiness of job J_i . The objective is to find a schedule which minimizes the maximum tardiness of jobs $T_{\max}(\pi) = \max_i T_i(\pi)$. It is worth mentioning that when preemption is allowed the first problem can be solved by the preemptive EDD (PEDD) rule and any optimal solution of the first problem is also optimal for the second one.

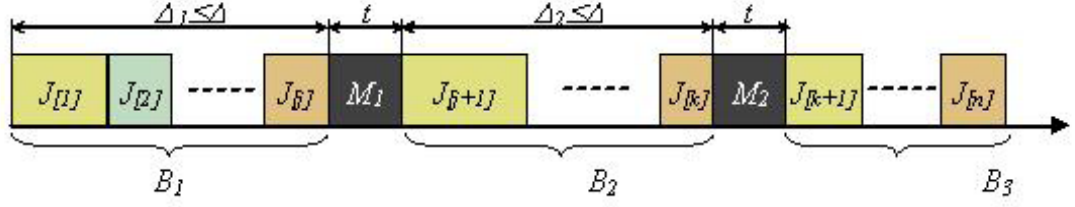


Figure 2: A schedule with flexible periodic maintenance : $J_{[i]}$ is the number of job in i^{th} position and M_i is the i^{th} operation of maintenance

Giving a partial schedule σ we will use the following notations:

$J(\sigma)$: the set of jobs scheduled in σ .

$C_i(\sigma)$: completion time of job $J_i \in J(\sigma)$ in σ .

$C(\sigma)$: completion time of σ .

$T_{\max}(\sigma)$: maximum tardiness of σ .

$B_m(\sigma)$: m th batch in σ .

$q_m(\sigma)$: total processing time of jobs in batch $B_m(\sigma)$.

$m(i, \sigma)$: the index of the batch where job $J_i \in J(\sigma)$ is scheduled *i.e.* $J_i \in B_{m(i, \sigma)}(\sigma)$.

For $J_i \notin J(\sigma)$, $\sigma \circ i$: schedule obtained by adding job i after σ . Note that if job i requires a processing time greater than the slack time in the current (last) batch in σ , it is scheduled in the next time interval Δ .

When there is no ambiguity, $C_i(\sigma)$, $T_{\max}(\sigma)$, $B_m(\sigma)$, $q_m(\sigma)$ and $m(i, \sigma)$ are simplified into C_i , T_{\max} , B_m , q_m and $m(i)$, respectively.

3 Periodic maintenance

We propose here a first approach considering the periodic maintenance case.

3.1 Dominance properties and lower bounds

In this section, we introduce some dominance properties and give lower bounds that will be used in our algorithms. The following property is similar to the EDD schedule in the ordinary T_{\max} problem.

Property 3.1. *There exists an optimal schedule in which in each batch jobs are sequenced in EDD (earliest due date) order.*

Property 3.1 can be proved by the pairwise interchange procedure within the same batch.

The next property states that the length of a job scheduled after batch i is longer than the gap $\Delta - q_i$ between scheduled jobs in batch i and the next scheduled maintenance (we remind that q_i represents the total completion time of batch B_i).

Property 3.2. *There exists an optimal schedule in which for each batch B_i , we have*

$$\Delta - q_i < p_j, \text{ for all } J_j \in B_k, k = i + 1, \dots, L.$$

Proof. Suppose π is an optimal schedule where batch B_i does not satisfy Property 3.2. Let job J_j be the first job satisfying $\Delta - q_i \geq p_j$, $J_j \in B_k$, $k > i$. Consider the sequence π' obtained from π by removing J_j from its original position and inserting it at the end of batch B_i . Then it is clear that π' is also an optimal sequence. Repeating this procedure a finite number of times, we obtain an optimal sequence verifying Property 3.2. \square

In order to give others dominance properties we need to introduce some notation. From now on we restrict ourselves to schedules satisfying Property 3.1. Given a partial schedule σ we denote by $O(\sigma)$ the schedule obtained by appending to σ the optimal partial schedule of the remaining jobs while keeping Property 3.1 satisfied.

To compute lower bounds for $T_{\max}(O(\sigma))$, we start with the following which give a lower bound for the starting date in $O(\sigma)$ for jobs not scheduled in σ . Let J_i denote the last job in σ .

Lemma 3.1. *Let $J_j \notin J(\sigma)$ such that $p_j > \Delta - q_{m(i)}(\sigma)$ or $d_j < d_i$, then*

$$C_j(O(\sigma)) - p_j \geq m(i)(\Delta + t).$$

Proof. If $p_j > \Delta - q_{m(i)}(\sigma)$, i.e. J_j not fit into batch $B_{m(i)}$ then it will not starts before $m(i)(\Delta + t)$.

If $d_j < d_i$, because EDD rule within the batch, J_j cannot be scheduled in $B_{m(i)}$, so it will start after $m(i)(\Delta + t)$. \square

Lower bound 1. To each job J_j not scheduled in σ we associate a ready time as follows:

$$r_j = \begin{cases} m(i)(\Delta + t) & \text{if } p_j > \Delta - q_{m(i)}(\sigma) \text{ or } d_j < d_i; \\ C(\sigma) & \text{otherwise.} \end{cases}$$

We consider then the problem $P_1(\sigma)$: Pursue the scheduling of jobs not scheduled in σ subject to preceding ready times in order to minimize the maximum tardiness while allowing preemption.

We denote by $LB_1(\sigma)$ the maximum tardiness in optimal solution of $P_1(\sigma)$. It is obtained by a dynamic version of Preemptive Earliest Due Date (D-PEDD) rule. Namely, sequencing decisions must be considered both at job completion times and at job ready times as follows:

1. At each job completion the job with minimum d_j among available jobs is selected to begin processing.
2. At each ready time, r_j , the due-date of the newly-available job j is compared to

the due-date of the jobs being processed. If d_j is lower, job j immediately preempts the job being processed, otherwise job j is simply added to the list of available jobs.

Lower bound 2. To calculate the second lower bound $LB_2(\sigma)$ we consider $P_2(\sigma)$ analogous to $P_1(\sigma)$ except in the definition of ready times:

$$r_j = \begin{cases} m(i)(\Delta + t) & \text{if } d_j < d_i; \\ C(\sigma) & \text{otherwise.} \end{cases}$$

It is clear that $LB_2 \leq LB_1$. In fact, the interest of the second lower bound lies in Lemma 3.2 used in the branch and bound algorithm.

Let $LB(\sigma)$ denote any lower bound for $T_{\max}(O(\sigma))$ (for e.g. LB_1 or LB_2), then we have:

Property 3.3. *For any partial schedule σ and any job j not scheduled in σ , schedule $O(\sigma \circ j)$ is dominated if there is a job i scheduled in σ such that*

$$m(i) < m(j), \quad p_i \leq p_j \leq p_i + (\Delta - q_{m(i)}(\sigma)) \quad \text{and} \quad C(\sigma \circ j) - (p_j - p_i) - d_i \leq LB(\sigma),$$

where $m(i) = m(i, \sigma)$ and $m(j) = m(j, \sigma \circ j)$.

Proof. Consider the sequence π obtained from $O(\sigma \circ j)$ by putting job i at the place of job j and by putting the latter at the end of batch $B_{m(i)}$ which is possible thanks to the condition $p_i \leq p_j \leq p_i + (\Delta - q_{m(i)}(\sigma))$. Conditions $m(i) < m(j)$ and $p_i \leq p_j$ ensure that only job i is completed later in π . But since $C_i(\pi) - d_i = C(\sigma \circ j) - (p_j - p_i) - d_i$, we deduce that $T_i(\pi) \leq LB(\sigma) \leq T_{\max}(O(\sigma))$. By re-sequencing batches $B_{m(i)}$ and $B_{m(j)}$ in the EDD order we obtain a better schedule π' satisfying Property 3.1, and $T_{\max}(\pi') \leq T_{\max}(O(\sigma))$.

□

Property 3.4. *Let σ be a partial schedule, and i the last job scheduled in σ . We suppose that there are no unscheduled jobs which fit into last batch of σ . Then schedule $O(\sigma)$ is dominated if there is an unscheduled job j such that*

$$p_i \leq p_j \leq p_i + (\Delta - q_{m(i)}(\sigma)) \quad \text{and} \quad d_i \geq d_j - (p_j - p_i).$$

Proof. Consider the sequence π obtained from $O(\sigma)$ by putting job i at the place of job j and by putting the latter at the end of batch $B_{m(i)}$ which is possible thanks to the condition $p_i \leq p_j \leq p_i + (\Delta - q_{m(i)}(\sigma))$. Conditions $m(i) < m(j)$ and $p_i \leq p_j$ ensure that only job i is completed later in π . Since $T_i(\pi) = \max\{C_i(\pi) - d_i; 0\} = \max\{C_j(O(\sigma)) - (p_j - p_i) - d_i; 0\} \leq \max\{C_j(O(\sigma)) - d_j; 0\} = T_j(O(\sigma))$, we conclude that $T_{\max}(\pi) \leq T_{\max}(O(\sigma))$. Finally, by re-sequencing each $B_{m(i)}$ and $B_{m(j)}$ in EDD order we obtain a new schedule π' such that $T_{\max}(\pi') \leq T_{\max}(\pi) \leq T_{\max}(O(\sigma))$. \square

3.2 Heuristic algorithm

Based on the preceding dominance properties, we propose, in this section, a heuristic to provide a near-optimal schedule for the stated problem. The heuristic is denoted **H1**, its steps are outlined as follows:

Heuristic H1

Step 1. Sort jobs in EDD order with ties broken by nonincreasing order of p_i : J_1, J_2, \dots, J_n .

Step 2. Calculate a lower bound \underline{T}_{\max} for the problem by P-EDD.

Step 3. Let $i = 1, j = 1, B_i = \emptyset, q_i = 0, (J, p, d) = (J_j, p_j, d_j)$ and $T_{\max} = 0$.

Step 4. Is there at least one batch B_m such that $\Delta - q_m \geq p_j$. If so, among them choose

the batch with the smallest index, say k , put job J_j at the end of batch B_k , $q_k = q_k + p_j$, $T_{\max} = \max\{T_{\max}; (k-1)(\Delta+t) + q_k - d_j\}$, and moreover $(J, p, d) = (J_j, p_j, d_j)$ if $k = i$, goto Step 8.

Step 5. If $q_i - p + p_j > \Delta$ or $p_j < p$, goto Step 6. Otherwise, goto Step 7.

Step 6. Let $i = i + 1$, insert a maintenance and let job J_j be the first job in the new batch B_i , $q_i = p_j$, $T_{\max} = \max\{T_{\max}; (i-1)(\Delta+t) + q_j - d_j\}$, $(J, p, d) = (J_j, p_j, d_j)$, goto Step 8.

Step 7. If $i(\Delta+t) + p - d \leq \max\{i(\Delta+t) + p_j - d_j; T_{\max}; \underline{T}_{\max}\}$, $B_i = (B_i \setminus \{J\}) \cup \{J_j\}$, $q_i = q_i - p + p_j$, $i = i + 1$, insert a maintenance and let job J be the first job the new batch B_i , $q_i = p$, $T_{\max} = \max\{T_{\max}; (i-1)(\Delta+t) + p - d; (i-2)(\Delta+t) + q_{i-1} - d_j\}$. Otherwise, goto Step 6.

Step 8. If $j = n$, stop. Otherwise, $j = j + 1$, return to Step 4.

Remark 1. Steps 1, 4 and 7 are motivated respectively by Properties 3.1, 3.2 and 3.3. The computational time complexity of Heuristic H1 is $O(n^2)$.

As an illustration of the heuristic, consider the following two examples from [11] and [4] respectively.

Table 1 The data for example 1 (in hours); $\Delta = 8$ and $t = 2$

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
p_i	1	5	3	5	2	2	3	4	4
d_i	1	13	2	30	10	13	20	12	14

Table 2 The schedule obtained by the heuristic in [11]

J_i	J_1	J_5	J_2	t	J_3	J_8	t	J_6	J_9	t	J_7	J_4
T_i	0	0	0		11	5		9	12		13	8

Table 3 The schedule obtained by our heuristic

J_i	J_1	J_3	J_8	t	J_5	J_2	t	J_6	J_9	t	J_7	J_4
T_i	0	2	0		2	4		9	12		13	8

Table 4 The data for example 2 (in hours); $\Delta = 12$ and $t = 3$

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_{11}
p_i	3	4	4	4	7	2	4	5	4	3	3
d_i	5	10	32	12	32	16	18	36	19	20	40

Table 5 The schedule obtained by the heuristic in [4]

J_i	J_1	J_2	J_6	J_{10}	t	J_4	J_5	t	J_7	J_9	t	J_3	J_8	J_{11}
T_i	0	0	0	0		7	0		16	19		17	18	17

Table 6 The schedule obtained by the our heuristic

J_i	J_1	J_2	J_4	t	J_6	J_7	J_9	t	J_{10}	J_5	t	J_3	J_8	J_{11}
T_i	0	0	0		1	3	6		13	8		17	18	17

The algorithm will be studied by computational experiments later. It will also be compared to the heuristic proposed by Liao and Chen [11].

3.3 Branch and Bound algorithm

The proposed branch and bound algorithm uses a usual schema. During the computation, we keep a list of unexplored nodes arranged in increasing order according to the lower bounds of nodes, with ties broken by nonincreasing number of scheduled jobs. Each node represents a partial schedule. The algorithm always try to develop the head of the list. The branching from a node consists of creating child nodes by adding an unscheduled job to the end of the partial schedule. We will calculate a lower bound for each node that cannot be eliminated by the dominance proprieties. In particular we use the following comparison lemma.

Lemma 3.2. *Let σ be a partial schedule, and i_1 and i_2 two jobs not scheduled in σ . If $d_{i_1} \leq d_{i_2}$ then at least one of the following holds:*

1. $O(\sigma \circ i_1)$ is dominated.
2. $O(\sigma \circ i_2)$ is dominated.
3. $LB_2(\sigma \circ i_1) \leq LB_2(\sigma \circ i_2)$.

Proof. Let J_i be the last job in σ .

- If $\min\{p_{i_1}; p_{i_2}\} \leq \Delta - q_{m(i)}(\sigma) < \max\{p_{i_1}; p_{i_2}\} = p_j$ for $j \in \{i_1, i_2\}$, then by Property 3.3 $O(\sigma \circ j)$ is dominated.
- If $\max\{p_{i_1}; p_{i_2}\} \leq \Delta - q_{m(i)}(\sigma)$. Let r_j^1 (respectively r_j^2) the ready time of job J_j in $P_2(\sigma \circ i_1)$ (respectively in $P_2(\sigma \circ i_2)$). Since $d_{i_1} \leq d_{i_2}$ then $r_j^1 \leq r_j^2$ for $J_j \notin J(\sigma) \cup \{J_{i_1}; J_{i_2}\}$; $r_{i_2}^1 = C(\sigma \circ i_1)$ and $r_{i_1}^2 \geq C(\sigma \circ i_2)$. Let π_2 the optimal schedule for the problem $P_2(\sigma \circ i_2)$ obtained, as said above, by D-PEDD rule. Let A be the set of jobs scheduled in π_2 in the interval $[C(\sigma); C(\sigma) + p_{i_1}]$. Consider the schedule π_1 obtained by interchanging A with J_{i_1} . This interchange will not delay jobs except possibly jobs in A . But because $d_{i_1} \leq d_{i_2} \leq d_j$ for $J_j \in A$ we have $C_j(\pi_1) - d_j \leq \max\{C_{i_1}(\pi_2) - d_{i_1}; C_j(\pi_2) - d_j\}$ for $J_j \in A$, and consequently $T_j(\pi_1) \leq \max\{T_j(\pi_2); T_{i_1}(\pi_2)\}$. This shows that π_1 is better than π_2 . On the other hand, π_1 is feasible for $P_2(\sigma \circ i_1)$ since $r_j^1 \leq r_j^2$. Finally, $LB_2(\sigma \circ i_1) \leq T_{\max}(\pi_1) \leq T_{\max}(\pi_2) = LB_2(\sigma \circ i_2)$.
- We use the same arguments to tackle the case $\Delta - q_{m(i)}(\sigma) < \min\{p_{i_1}; p_{i_2}\}$. □

In fact if $\sigma \circ i$ is eliminated, then we never consider $\sigma \circ j$ for all J_j such that $d_j \geq d_i$. The computation time of the B&B algorithm is reported in Section 5.

4 Flexible periodic maintenance

In this section we deal with the flexible periodic version. Recall that in this case maintenance periods are not fixed but the maximum continuous working time allowed of the machine is Δ .

4.1 Dominance properties and lower bounds

It is clear that Property 3.1 remains true in this case. However we have to make some modifications on the others properties.

Property 4.1. *There exists an optimal schedule in which for each batch B_i , we have*

$$\Delta - q_i < p_j \text{ for the first job } J_j \text{ in the next batch } B_{i+1}.$$

Proof. Suppose π is an optimal schedule which does not satisfy Property 4.1. Consider the sequence π' obtained from π by removing J_j from its original position and inserting it at the end of batch B_i . Then it is clear that π' is also an optimal sequence. Repeating this procedure a finite number of times, we obtain an optimal sequence verifying Property 4.1.

□

There exists an optimal schedule satisfying at the same time Properties 3.1 and 4.1. More precisely, we can prove that every schedule not satisfying simultaneously 3.1 and 4.1 is dominated. Thus from now on we consider only partial schedules satisfying Properties 3.1 and 4.1. Given a partial schedule σ , $O(\sigma)$ will denote the schedule obtained by appending to σ the optimal partial schedule of the remaining jobs while keeping Properties 3.1 and 4.1 satisfied.

Before stating other dominance properties let us give, as in the periodic version, a lower bound for $T_{\max}(O(\sigma))$. For that purpose we establish first a minoration for $C_j(O(\sigma))$ ($J_j \notin$

$J(\sigma)$). Let σ be a partial schedule, J_i the last job in σ and $s := \sum_{m=1}^{m(i)-1} q_m(\sigma) + (m(i)-1)t$ be the starting date of batch $B_{m(i)}$. To simplify, we set $q = q_{m(i)}(\sigma)$.

Lemma 4.1. *Let $J_j \notin J(\sigma)$. If $p_j > \Delta - q$ or $d_j < d_i$, then*

$$C_j(O(\sigma)) \geq \max\{s + \Delta + t - p_j + 1; s + q + t\} + p_j. \quad (1)$$

Proof. We are going to prove that J_j starts in $O(\sigma)$ after $\max\{s + \Delta + t - p_j + 1; s + q + t\}$. Keep in mind that the processing times and due dates are integers. Two cases must be examined.

Case 1: If $\Delta - q < p_j$, i.e. $\max\{s + \Delta + t - p_j + 1; s + q + t\} = s + q + t$. It is not possible to place job J_j in the batch $B_{m(i)}$ and it is necessary to insert at least one maintenance period before processing J_j . So J_j starts after $s + q + t$.

Case 2: If $\Delta - q \geq p_j$ and $d_j < d_i$, in which case $\max\{s + \Delta + t - p_j + 1; s + q + t\} = s + \Delta + t - p_j + 1$. Assume for the sake of contradiction that J_j starts before the latter date. Then Property 3.1 implies that $m(j) > m(i)$, i.e. between jobs J_i and J_j there is at least one maintenance period. Let A be the set of jobs which are sequenced after J_i and before J_j . We have

$$s + q + \sum_{J_k \in A} p_k + t + p_j \leq (s + \Delta + t - p_j + 1) - 1 + p_j = s + \Delta + t.$$

Consequently $q + \sum_{J_k \in A} p_k + p_j \leq \Delta$; which contradicts the validity of Property 4.1. \square

Now we propose two lower bounds for $T_{\max}(O(\sigma))$. The first one is based on the relaxation of the "ready time" (1) and the second is based on the relaxation of the constraint on the maintenance.

Lower Bound 1: We consider the problem $P_1(\sigma)$: we fix maintenance as late as possible

i.e. at dates $s + \Delta$, $s + 2\Delta + t$, $s + 3\Delta + 2t, \dots$ and for each J_j not scheduled in σ we associate a ready time:

$$r_j = \begin{cases} s + \Delta - p_j + 1 & \text{if } d_j < d_i, \\ C(\sigma) & \text{otherwise.} \end{cases} \quad (2)$$

Moreover, we allow the preemption. This problem is solved optimally by D-PEDD, we denote by $LB_1(\sigma)$ the resulting optimal value. We have $LB_1(\sigma) \leq T_{\max}(O(\sigma))$. Indeed, let us construct from $O(\sigma)$ a feasible schedule for $P_1(\sigma)$.

- In each interval $[s + m(\Delta + t); s + (m + 1)(\Delta + t)]$ ($m \geq 0$) we keep only one maintenance period (in fact there is at least one) and we place it at the date $s + m(\Delta + t) + \Delta$ possibly by interchanging it with jobs situated between it and $s + m(\Delta + t) + \Delta + t$. This is possible since each job having a due date lower than d_i starts after $s + \Delta + t - p_j + 1$ in $O(\sigma)$ (see Lemma 4.1) and advancing it by t will not alter the constraint (2).

- We advance jobs while respecting the ready time.

Clearly the obtained schedule π is better than $O(\sigma)$ and is feasible for $P_1(\sigma)$. Thus, $LB_1(\sigma) \leq T_{\max}(\pi) \leq T_{\max}(O(\sigma))$. \square

Lower Bound 2: We consider the problem $P_2(\sigma)$, where we fix maintenance at dates $s + 2\Delta + t$, $s + 3\Delta + 2t, \dots$ (note that unlike the preceding case, the machine is not maintained between s and $s + 2\Delta + t$) and we associate to every J_j a ready time r_j

$$r_j = \begin{cases} \max\{s + \Delta + t - p_j + 1; s + q + t\} & \text{if } p_j > \Delta - q \text{ or } d_j < d_i, \\ C(\sigma) & \text{otherwise.} \end{cases}$$

Again, we solve the resulting problem by allowing the preemption of jobs and by using the D-PEDD rule. We let $LB_2(\sigma)$ the optimal value corresponding to $P_2(\sigma)$.

Let us prove $LB_2(\sigma) \leq T_{\max}(O(\sigma))$. As above, we are going to construct from $O(\sigma)$ another better schedule π as follows:

- We delete maintenance periods situated in the interval $[s; s + \Delta + t]$
- In each interval $[s + m(\Delta + t); s + (m + 1)(\Delta + t)]$ ($m \geq 1$) we keep only one (in fact there is at least one) maintenance period and we insert it at the date $s + m(\Delta + t) + \Delta$ (possibly by interchanging it with jobs situated between it and $s + m(\Delta + t) + \Delta + t$, which is possible thanks to the inequality $r_j \leq s + \Delta + t$).
- We advance jobs while respecting the ready time.

Clearly π is better than $O(\sigma)$ and is feasible for $P_2(\sigma)$. So, $LB(\sigma) \leq T_{\max}(\pi) \leq T_{\max}(O(\sigma))$.

□

We put $LB(\sigma) = \max\{LB_1(\sigma); LB_2(\sigma)\}$.

Property 4.2. *For any partial schedule σ and any j not scheduled in σ , schedule $O(\sigma \circ j)$ is dominated in both following cases:*

1. *There is a batch B_k where $k < m(j, \sigma \circ j)$, $\Delta - q_k(\sigma) \geq p_j$ and $C_i(\sigma) + p_j - d_i \leq LB(\sigma)$ for all jobs J_i in σ scheduled after B_k .*
2. *There is a batch B_k where $k < m(j, \sigma \circ j)$, $\Delta - q_k(\sigma) \geq p_s$ and $C_i(\sigma \circ j) + p_s - d_i \leq LB(\sigma \circ j)$ for all jobs J_i in $\sigma \circ j$ scheduled after B_k , where J_s is the job with the smallest due date among jobs not in $J(\sigma \circ j)$ i.e. $d_s = \min\{d_i; J_i \notin J(\sigma \circ j)\}$.*

Proof. 1. Let A be the set of jobs which is sequenced after batch B_k and before job J_j in $O(\sigma \circ j)$. A new feasible schedule π can be obtained from $O(\sigma \circ j)$ by placing job J_j at the end of batch B_k and not increasing the completion times of jobs except jobs in A . The completion time of each job in A will increase by p_j , so $C_i(\pi) - d_i = C_i(O(\sigma)) + p_j - d_i$ for $i \in A$. Therefore, $T_i(\pi) \leq LB(\sigma) \leq T_{\max}(O(\sigma \circ j))$, and $T_{\max}(\pi) \leq T_{\max}(O(\sigma \circ j))$.

2. Let A be the set of jobs which is sequenced after batch B_k and before job J_s in $O(\sigma \circ j)$. A new feasible schedule π can be obtained by placing job J_s at the end of batch B_k and

not increasing the completion times of jobs except jobs in A . If J_i is in A and is scheduled in $\sigma \circ j$, then

$$\begin{aligned} T_i(\pi) &= \max\{C_i(\pi) - d_i; 0\} \\ &\leq \max\{C_i(O(\sigma)) + p_s - d_i; 0\} \\ &\leq LB(\sigma \circ j) \\ &\leq T_{\max}(O(\sigma \circ j)). \end{aligned}$$

If J_i is in A but not scheduled in $\sigma \circ j$, then

$$\begin{aligned} T_i(\pi) &= \max\{C_i(\pi) - d_i; 0\} \\ &= \max\{C_i(O(\sigma \circ j)) + p_s - d_i; 0\} \\ &\leq \max\{C_s(O(\sigma \circ j)) - d_s; 0\} \\ &\leq T_s(O(\sigma \circ j)). \end{aligned}$$

Finally, we get $T_{\max}(\pi) \leq T_{\max}(O(\sigma \circ j))$. \square

Property 4.3. *For any partial schedule σ and any job j not scheduled in σ , schedule $O(\sigma \circ j)$ is dominated if there is a job i scheduled in σ such that $m(i, \sigma) < m(j, \sigma \circ j)$, $p_i \leq p_j \leq p_i + (\Delta - q_{m(i)}(\sigma))$, $C(\sigma \circ j) - d_i \leq LB(\sigma)$ and $C_v(\sigma) + (p_j - p_i) - d_v \leq LB(\sigma)$ for all jobs J_v in σ scheduled after $B_{m(i)}$.*

Proof. Let A be the set of jobs which is sequenced after batch $B_{m(i)}$ in σ . A new feasible schedule π can be obtained by replacing job J_i by J_j and inserting J_j at the end of batch $B_{m(i)}$. Only job J_i and jobs in A are completed later in π . We have $C_v(\pi) - d_v = C_v(O(\sigma \circ j)) + (p_j - p_i) - d_v$ for $J_v \in A$. So, $T_v(\pi) \leq LB(\sigma) \leq T_{\max}(O(\sigma \circ j))$. Moreover, $T_i(\pi) = \{C(\sigma \circ j) - d_i; 0\} \leq LB(\sigma)$. Therefore, $T_{\max}(\pi) \leq T_{\max}(O(\sigma \circ j))$. \square

Property 4.4. *Let σ be a partial schedule. Let j be a job not scheduled in σ which does not fit into the last batch of σ . We denote by i the last job in σ . Then $O(\sigma \circ j)$ is dominated if $p_i \leq p_s \leq p_i + (\Delta - q_{m(i)}(\sigma))$, $d_i \geq d_s$ and $C(\sigma \circ j) + (p_s - p_i) - d_j \leq LB(\sigma \circ j)$, where J_s is the job having the smallest due date among jobs not scheduled in $\sigma \circ j$, i.e. $d_s = \min\{d_v; J_v \notin J(\sigma \circ j)\}$.*

Proof. Let A be the set of jobs which is sequenced after job J_j and before job J_s in $O(\sigma \circ j)$. A new feasible schedule π can be obtained by interchanging J_i and J_s . This will not increase the completion times of jobs except jobs J_j and J_i , and jobs in A . We have $C_j(\pi) - d_j = C_j(\sigma \circ j) + (p_s - p_i) - d_j$, so $T_j(\pi) \leq LB(\sigma \circ j) \leq T_{\max}(O(\sigma \circ j))$. On the other hand, for $J_v \in A$, we have $C_v(\pi) - d_v = C_v(O(\sigma \circ j)) + (p_s - p_i) - d_v \leq C_s(O(\sigma \circ j)) - d_v \leq C_s(O(\sigma \circ j)) - d_s$, so $T_v(\pi) \leq T_s(O(\sigma \circ j))$. Moreover, $T_i(\pi) = \max\{C_i(\pi) - d_i; 0\} \leq \max\{C_s(O(\sigma \circ j)) - d_s; 0\} = T_s(O(\sigma \circ j))$. Thus we have the required result that $T_{\max}(\pi) \leq T_{\max}(O(\sigma))$. \square

4.2 Heuristic algorithm

To approximate the solution of the problem we propose the following heuristic which is an adaptation of Heuristic H1 in light of the new properties.

Heuristic H2

Step 1. Sort jobs in EDD order with ties broken by non-increasing order of p_i : J_1, J_2, \dots, J_n .

Step 2. Calculate a lower bound \underline{T}_{\max} for the problem by putting maintenance as late as possible i.e. at the dates $\Delta, 2\Delta + t, 3\Delta + 2t, \dots$, and scheduling jobs according PEDD rule.

Step 3. Let $i = 1, j = 1, B_i = \emptyset, q_i = 0, (J, p, d) = (J_j, p_j, d_j), C = 0$ and $T_{\max} = 0$.

Step 4. Is there at least one batch B_m such that $\Delta - q_m \geq p_j$ and $T_l \leq \max\{\underline{T}_{\max} -$

$p_j; T_{\max} - p_j; 0\}$, $\forall T_l \in B_k, k > m$. If so, among them choose the Batch with the smallest index, say k , put job J_j at the end of batch B_k , $q_k = q_k + p_j$, $C = C + p_j$, up date T_{\max} , and moreover $(J, p, d) = (J_j, p_j, d_j)$ if $k = i$, goto Step 8.

Step 5. If $q_i - p + p_j > \Delta$ or $p_j < p$, goto Step 6. Otherwise, go to Step 7.

Step 6. Let $i = i + 1$, insert a maintenance task and let job J_j be the first job in the new batch B_i , $q_i = p_j$, $C = C + p_j + t$, $T_{\max} = \max\{T_{\max}; C - d_j\}$, $(J, p, d) = (J_j, p_j, d_j)$, go to Step 8.

Step 7. If $C + t + p_j - d \leq \max\{C + t + p_j - d_j; T_{\max}; \underline{T}_{\max}\}$, $B_i = (B_i \setminus \{J\}) \cup \{J_j\}$, $q_i = q_i - p + p_j$, $i = i + 1$, insert a maintenance task and let job J be the first job the new batch B_i , $q_i = p$, $C = C + t + p_j$, $T_{\max} = \max\{T_{\max}; C - d; C - p - t - d_j\}$. Otherwise, goto Step 6.

Step 8. If $j = n$, stop. Otherwise, $j = j + 1$, return to Step 4.

4.3 Branch and Bound algorithm

The algorithm is similar to that developed in Section 3.3. Note however, for this case we do not have a Lemma analogous to that of Lemma 3.2.

5 Computational results

In this section, we report computational results to evaluate the effectiveness of the heuristics algorithms and the computation time of the B&B. The computational experiments, are done using Visual C++ compiler and on a duo T5600 processor with 1Go of memory. We generate the parameter values as in [11] : processing times were selected from a discrete uniform distribution (DU) over $[1, 10]$. The due dates were selected from

another DU over $[(1 - C - Q/2) \sum_{i=1}^n p_i, (1 - C + Q/2) \sum_{i=1}^n p_i]$, where $Q \in \{0.2, 0.6\}$ and $C \in \{0.2; 0.6\}$ denote the due date range and tardiness factor, respectively.

Experiment 1. Comparison between H1 and Heuristic in [11] for the periodic version. For each combination of n , C , Q , Δ and t , 50 problems are randomly generated. Each problem is solved by H1 and the heuristic proposed in [11] which we denote here by HCL. Table 7, corresponding to $\Delta = 18$ and $t = 4$, shows in column 'better' the number of problems (#HCL) for which Heuristic H1 is better than HCL, and the number of problems (#H1) for which Heuristic HCL is better than H1. In column DH1 Max (respectively Mean) denotes the maximum (respectively the average) deviation of the heuristic HCL solution from the solution obtained by H1 for problems where H1 is better than HCL. Accordingly, in column DHCL Max (respectively Mean) denotes the maximum (respectively the average) deviation of the heuristic H1 solution from the solution obtained by HCL for problems where HCL is better than H1. Table 8 is similar to Table 7 with $\Delta = 10$ and $t = 2$. We can see that for all combinations of C , Q , Δ and t our heuristic outperforms HCL especially for large n . In fact there cases where our heuristic is 100% better. Moreover, the deviation is more important for the case where H1 is better. This is because HCL is very dependent in the manner that jobs are initially ordered (no initial order of jobs is assumed in Heuristic HCL).

From the computational point of view, H1 is better than HCL: H1 spends much less computation time than HCL.

Experiment 2. The relative error of the heuristic algorithms and the average computation time of B&B (in milliseconds).

For different combinations of C , Q , Δ and t 25 problems are randomly generated and solved by heuristic and branch and bound algorithms subject to both periodic and flexible

periodic constraint. Table 9 gives the relative error of our heuristic algorithms *i.e.*

$$\frac{T_{\max}(\text{Hi}) - T_{\max}(\text{B\&B})}{T_{\max}(\text{B\&B})} \quad (i = 1 \text{ or } 2) \quad (3)$$

and the computation time (in milliseconds) of the branch and bound algorithms. For our choice of C and Q , namely $(C, Q) = (0.2, 0.2)$ and $(C, Q) = (0.6, 0.6)$, we are sure that the maximum tardiness is superior to $0.1 \sum_{j=1}^n p_j$ and $0.3 \sum_{j=1}^n p_j$, respectively. To see this, consider the last job, say J_i , in an optimal schedule whose completion time is clearly $C_i \geq \sum_{j=1}^n p_j$. If $(C, Q) = (0.2, 0.2)$ then $d_i \leq 0.9 \sum_{j=1}^n p_j$, and consequently $T_{\max} \geq C_i - d_i \geq (\sum_{j=1}^n p_j) - (0.9 \sum_{j=1}^n p_j) = 0.1 \sum_{j=1}^n p_j$. If $(C, Q) = (0.6, 0.6)$ the same argument gives $T_{\max} \geq 0.3 \sum_{j=1}^n p_j$. So the error relative (3) is well defined. We can see that the B&B is not an efficient algorithm with large problems especially for the flexible periodic version. Of course, we have not reported a computation time of B&B in the last case for $n = 20$ because we have fixed the maximum computational time to be ten minutes and many problems exceed 10 minutes. As we can see in Table 9, the main interesting results is the effectiveness of both heuristics H1 and H2. The error for all treated problem does not exceed the average value of 17% for H1 and 3% for H2. Moreover, the heuristics always find at least once the best solution for all problem sizes (this represents at least 1 problem over 20). This is true for both fixed and flexible maintenance cases.

6 Conclusion

The importance of maintenance has been gradually accepted by the decision maker. Therefore, it has become a common practice to schedule maintenance periodically in many manufacturing systems. Unfortunately, most papers discussing maintenance assume there is only one maintenance period. In this paper, we have addressed the single machine

maximum tardiness problem subject to periodic maintenance and nonresumable jobs. We have considered two versions which are both NP-hard. The first one corresponds to a periodic maintenance. It consists of several maintenance periods where each maintenance is required after a periodic time interval. In the second situation, the maintenance is not fixed but the maximum allowed continuously working time of machine is fixed. We have proposed for each of them a heuristic and a branch and bound algorithm. Computational experiments have been done to evaluate the effectiveness of the algorithms. Moreover, for the first version an extensive empirical comparison of the proposed heuristic with Liao and Chen [11] heuristic shows the superiority of the former. One future research direction is to extend these scheduling problems from the single machine case to the parallel machine case and flow-shop problem.

References

- [1] Adiri I, Frostig E. Rinnooy Kan AHG. Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs. *Naval Research Logistics* 1991;38:261-271.
- [2] Allaoui H, Artiba A. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers & Industrial Engineering* 2004; 47:431-450.
- [3] Allaoui H, Artiba A. Two stage hybrid flow shop scheduling with availability constraints. *Computers & Operations Research* 2006; 33:1399-1419.
- [4] Chen WJ. Scheduling of jobs and maintenance in a textile company. *Journal The International Journal of Advanced Manufacturing Technology* 31 2007;31:737-742.

- [5] Coffman EG, Garey MR, Johnson DS. Approximation algorithms for bin-packing: a survey. In: D.S. Hochbaum (Ed.), *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, Boston, MA, 1995.
- [6] Espinouse ML, Formanowicz P, Penz B. Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. *Comput. Ind. Eng.* 1999;32:497-500.
- [7] Graves GH, Lee CY. Scheduling maintenance and semi-resumable jobs on a single machine. *Nav. Res. Log.* 1999;46:845-863.
- [8] Lee CY, Liman SD. Capacitated two-parallel machine scheduling to minimize sum of job completion time. *Discrete Applied Mathematics* 1993;41:211-222.
- [9] Lee CY. Parallel machines scheduling with non-simultaneous machine available time. *Discrete Applied Mathematics* 1991;30:53-61.
- [10] Lee CY. Machine scheduling with an availability constraint. *Journal of Global Optimization* 1996;9:395-416.
- [11] Liao CJ, Chen WJ. Single-machine scheduling with periodic maintenance and nonresumable jobs. *Comput. Oper. Res.* 2003;30: 1335-1347.
- [12] Mosheiov G. Minimizing the sum of job completion times on capacitated parallel machines. *Mathematical and Computer Modeling* 1994;20:91-99.
- [13] Qi X, Chen T, Tu F. Scheduling the maintenance on a single machine. *Journal of Operational Research Society* 1999;50:1059-1074.
- [14] Sadfi C, Penz B, Rapine C, Błażewicz J, Formanowicz P. An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. *Eur. J. Oper. Res.* 2005;161:3-10.

- [15] G. Schmidt, Scheduling with limited machine availability, *Eur. J. Oper. Res.* 2000;121:1-15.

Table 7 Comparison between H1 and Heuristic in [11] for the periodic version ($\Delta = 18$ and $t = 4$)

C	Q	n	Better		DH1		DHCL		Comp. time (ms)	
			#H1	#HCL	Max	Mean	Max	Mean	H1	HCL
0.2	0.2	10	15	15	15	5.80	13	4.53	0.32	0.60
		20	22	11	25	7.55	22	5.00	0.00	1.86
		30	28	10	19	7.61	5	2.30	1.26	2.18
		40	28	8	22	8.04	15	5.25	1.26	3.76
		50	34	12	24	9.32	19	7.67	1.26	5.30
		80	34	8	44	13.82	24	8.25	1.56	11.26
		110	42	4	42	14.17	17	8.25	3.14	18.42
		150	47	3	67	19.60	13	6.00	5.98	32.12
		170	44	3	76	21.82	20	13.67	7.74	37.56
		200	40	7	54	21.23	17	8.14	9.36	46.90
0.2	0.6	10	15	6	17	7.47	7	3.50	0.00	0.94
		20	33	6	35	8.33	16	7.17	0.32	2.50
		30	35	8	36	12.43	17	6.13	0.32	4.38
		40	41	4	45	15.15	8	4.50	1.24	6.90
		50	42	2	41	16.64	7	4.00	1.24	10.00
		80	47	3	55	26.34	12	6.67	2.54	27.14
		110	49	1	70	40.08	15	15.00	3.12	49.06
		150	50	0	103	54.74	0	0.00	3.12	99.06
		170	50	0	104	59.06	0	0.00	5.36	139.32
		200	50	0	129	79.44	0	0.00	8.40	190.34
0.6	0.2	10	19	12	10	3.79	14	4.33	0.00	0.64
		20	28	10	22	7.18	18	5.30	0.00	2.18
		30	24	12	18	7.04	22	6.58	0.32	3.14
		40	38	7	31	10.05	16	7.71	1.56	3.12
		50	25	12	36	8.84	17	5.50	0.94	5.96
		80	36	10	37	12.31	8	3.90	1.24	12.52
		110	40	6	46	12.55	22	12.17	4.40	17.78
		150	41	8	40	17.15	12	5.88	6.54	33.16
		170	40	9	61	22.60	22	6.78	5.34	39.02
		200	45	3	54	22.24	18	11.00	8.80	51.82
0.6	0.6	10	19	5	13	4.68	8	4.60	0.00	0.94
		20	35	2	23	9.46	16	8.50	0.00	2.18
		30	37	4	35	11.38	22	10.50	0.62	3.74
		40	41	3	61	18.15	7	3.33	1.26	6.26
		50	41	6	44	16.27	11	4.83	0.62	10.94
		80	47	1	51	24.85	3	3.00	3.12	23.10
		110	50	0	104	39.24	0	0.00	5.60	48.16
		150	50	0	91	56.74	0	0.00	6.88	94.34
		170	50	0	102	63.38	0	0.00	5.66	124.64
		200	50	0	118	81.62	0	0.00	8.78	166.86

Table 8 Comparison between H1 and Heuristic in [11] for the periodic version ($\Delta = 10$ and $t = 2$)

C	Q	n	Better		DH1		DHCL		Comp. time (ms)	
			#H1	#HCL	Max	Mean	Max	Mean	H1	HCL
0.2	0.2	10	14	6	11	3.00	9	5.33	0.32	0.92
		20	25	6	12	6.32	12	7.50	0.00	1.86
		30	23	9	16	7.43	11	6.78	0.92	2.20
		40	27	9	17	8.81	12	6.11	1.56	3.76
		50	36	4	24	10.50	11	7.00	1.88	4.68
		80	40	6	24	11.85	12	8.67	2.18	12.20
		110	41	7	37	15.56	13	9.71	6.00	17.44
		150	46	2	44	20.07	6	4.50	9.38	28.44
		170	48	1	49	22.10	1	1.00	11.32	36.54
		200	47	2	48	25.94	10	9.00	13.86	47.08
0.2	0.6	10	18	2	12	6.78	12	12.00	0.32	0.62
		20	20	5	14	6.85	12	5.60	0.00	2.18
		30	33	4	32	11.12	12	5.25	0.64	2.48
		40	34	8	26	13.29	12	8.75	1.24	3.44
		50	30	5	42	14.97	12	9.40	0.92	6.58
		80	35	0	48	17.43	0	0.00	2.50	11.26
		110	39	2	49	21.23	12	10.00	3.72	19.40
		150	43	3	48	21.95	12	8.00	8.52	30.54
		170	42	4	45	21.76	17	11.50	11.62	33.68
		200	47	3	77	27.87	12	8.00	13.80	47.80
0.6	0.2	10	21	6	12	4.81	12	5.50	0.00	0.62
		20	21	5	12	5.90	8	6.60	0.60	1.58
		30	25	6	18	8.56	12	9.00	0.32	3.74
		40	28	4	19	8.50	12	8.75	0.96	3.44
		50	28	6	24	9.96	23	10.33	1.54	5.00
		80	36	7	27	13.44	20	11.57	2.20	11.56
		110	38	4	36	18.11	24	10.75	4.08	18.70
		150	42	3	41	20.21	24	13.67	8.94	30.10
		170	43	2	48	23.47	12	8.50	11.30	37.14
		200	45	3	54	26.24	12	10.00	13.66	51.34
0.6	0.6	10	15	4	18	7.07	12	8.25	0.32	0.94
		20	23	3	22	10.00	6	5.33	0.96	1.54
		30	29	1	22	9.69	12	12.00	0.62	2.52
		40	30	2	24	11.27	6	3.50	1.28	4.04
		50	37	4	35	11.30	12	6.00	2.20	5.30
		80	34	5	42	18.03	12	4.80	4.68	10.62
		110	43	1	39	19.21	9	9.00	5.02	18.72
		150	44	1	54	22.20	12	12.00	9.72	28.10
		170	48	1	50	21.67	12	12.00	8.96	38.22
		200	47	1	63	30.83	2	2.00	14.12	49.32

Table 9 The relative error of the heuristic algorithms and the average computation time of B&B (in milliseconds)

					Fixed periodic maintenance			Flexible periodic maintenance								
C	Q	n	Δ	t	Comp. time (ms)	Relative error			Comp. time (ms)	Relative error						
						Min	Max	Mean		Min	Max	Mean				
0.2	0.2	10	10	2	58	0.0000	0.1538	0.0145	238	0.0000	0.3333	0.0267				
				4	65	0.0000	0.3600	0.0180	217	0.0000	0.2353	0.0258				
			15	2	58	0.0000	0.6471	0.0798	170	0.0000	0.3750	0.0407				
				4	57	0.0000	0.8667	0.0565	166	0.0000	0.2667	0.0245				
		20	2	2	50	0.0000	1.1429	0.1669	142	0.0000	0.5714	0.0463				
				4	66	0.0000	0.5294	0.1341	111	0.0000	0.2941	0.0178				
			15	10	2	1243	0.0000	0.0000	0.0000	21732	0.0000	0.0000	0.0000			
					4	2058	0.0000	0.0000	0.0000	46532	0.0000	0.0909	0.0067			
		20	15	2	2	2471	0.0000	0.0000	0.0000	12517	0.0000	0.1176	0.0174			
					4	2891	0.0000	0.6154	0.0370	21265	0.0000	0.1379	0.0217			
			20	2	2	2944	0.0000	0.3529	0.0689	6189	0.0000	0.1176	0.0129			
					4	5246	0.0000	0.2593	0.0536	6481	0.0000	0.1538	0.0098			
		20	10	2	2	22507	0.0000	0.0000	0.0000							
					4	33888	0.0000	0.0000	0.0000							
				15	2	2	42522	0.0000	0.0000	0.0000						
						4	59153	0.0000	0.0000	0.0000						
				20	2	2	99231	0.0000	0.7083	0.0959						
						4	153958	0.0000	0.5000	0.0396						
				0.6	0.6	10	10	2	39	0.0000	0.0000	0.0000	143	0.0000	0.0000	0.0000
								4	42	0.0000	0.3784	0.0501	138	0.0000	0.1081	0.0065
15	2	2	28				0.0000	0.5200	0.0483	91	0.0000	0.2800	0.0140			
		4	25				0.0000	0.4242	0.0630	100	0.0000	0.1429	0.0157			
20	2	2	18			0.0000	0.3043	0.0328	57	0.0000	0.0435	0.0022				
		4	16			0.0000	0.5806	0.0565	47	0.0000	0.1290	0.0144				
	15	10	2			929	0.0000	0.0652	0.0033	12065	0.0000	0.1026	0.0094			
			4			1575	0.0000	0.1633	0.0161	13713	0.0000	0.1628	0.0162			
20	15	2	2			691	0.0000	0.3269	0.0572	4696	0.0000	0.0968	0.0065			
			4			893	0.0000	0.3171	0.0450	7253	0.0000	0.1026	0.0196			
	20	2	2			640	0.0000	0.2000	0.0245	2143	0.0000	0.0571	0.0059			
			4			621	0.0000	0.2708	0.0303	1781	0.0000	0.0638	0.0073			
20	10	2	2			20907	0.0000	0.0000	0.0000							
			4			22515	0.0000	0.0000	0.0000							
		15	2			2	25207	0.0000	0.3182	0.0356						
						4	27043	0.0000	0.0167	0.0008						
		20	2			2	21600	0.0000	0.2200	0.0659						
						4	18181	0.0000	0.2059	0.0500						