Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 416–422

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Note

A note on worst-case performance of heuristics for maintenance scheduling problems

Xiangtong Qi

Department of Industrial Engineering and Logistics Management, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Received 24 June 2005; received in revised form 27 June 2006; accepted 29 June 2006
Available online 14 August 2006

Abstract

We study a machine scheduling model in which job scheduling and machine maintenance activities have to be considered simultaneously. We develop the worst-case bounds for some heuristic algorithms, including a sharper worst-case bound of the SPT schedule than the results in the literature, and another bound of the EDD schedule.

Keywords: Scheduling; Maintenance; SPT schedule; EDD schedule; Worst-case bound

1. Introduction

In this note, we will study the worst-case bound analysis of two heuristic algorithms for maintenance scheduling problems. The situation is as follows. We have a set of jobs to be processed on a single machine. The machine has to be interrupted for maintenance from time to time. The problem is to determine the schedule of jobs and the maintenance activities for the machine simultaneously.

When the objective is to minimize total completion time of jobs, the problem has been studied by Qi et al. [14] where they prove that the problem is NP-hard in the strong sense, and propose several heuristic algorithms including the SPT schedule. Meanwhile, the same scheduling model is independently studied by Akturk et al. [1,2] with a different motivation from tool management where they interpret the maintenance activities as the tool changes. Recently, Akturk et al. [3] show that the SPT schedule has a worst-case bound of 2. In this paper, we will present some new results that further improve the bound.

The problem belongs to the field of machine scheduling with non-availability constraints (see [9,13] for reviews). In general, there are two types of models: (1) the non-available machine periods are given and (2) the non-available machine periods are decision variables. Some recent results on the worst-case bound analysis of SPT and WSPT schedules for the former case can be found in Kacem and Chu [11] and Sadfi et al. [12]. Our model falls into the latter case. Some related results for similar models can be found in Cassady and Kutanoglu [4], Chen [5], Graves and Lee [7], and Lee and Lin [10].

E-mail address: ieemqi@ust.hk.

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2006.06.005

In the rest of this paper, we first formally define the problem in Section 2, then study the problem with two different objective functions in Sections 3 and 4, respectively. Our focus is the worst-case analysis for some simple heuristic algorithms.

2. Problem description

The problem is formally defined as follows. There are n jobs to be processed on a single machine. Each job J_j has a processing time p_j, and is available at time 0. The jobs cannot be pre-empted. The machine has to be stopped for maintenance after continuously working for a period of time. Suppose that the longest possible continuously working time for the machine is T time units and it takes t time units for each maintenance. Although mathematically T and t could be any positive numbers, in practice we usually have $T > t$, i.e., the available working time of the machine is longer than the unavailable time. If, on the other hand, $T \leqslant t$, it implies that the machine is unavailable under maintenance for more than 50% of the time, which is at least not common if not unrealistic at all.

A feasible schedule π can then be described by a series of job batches, where each job belongs to a specific batch, the total processing time of the jobs in any batch cannot be more than T, and there is a maintenance between any two consecutive batches. Furthermore, we also need to determine the sequence of jobs within each batch in the schedule. In order to have a feasible schedule, we need to assume that $\max_j \{p_j\} \leqslant T$.

Suppose that there are L batches in a schedule π, and the number of jobs in batch B_i is n_i. Consider the jth job in a schedule, denoted by $J_{[j]}$. If it is in batch B_i, then its completion time is given by $C_{[j]} = \sum_{k=1}^{j} p_{[k]} + (i-1)t$.

We will consider two common objective functions. The first one is to minimize total completion time:

$$f(\pi) = \sum_{j=1}^{n} C_{[j]} = \sum_{j=1}^{n} (n-j+1)p_{[j]} + \sum_{i=2}^{L} (i-1)n_i t.$$

The second objective function is to minimize the maximum lateness $L_{\max}$ where each job J_j has a given due date d_j and its lateness in a schedule π is defined as $L_j = C_j - d_j$. Then the maximum lateness of a schedule is given by $L_{\max} = \max_j \{L_j\}$.

3. SPT schedule

The problem of minimizing total completion time is NP-hard [14], and the SPT schedule has been computationally shown to be an effective heuristic algorithm in Qi et al. [14], and Akturk et al. [2,3]. In an SPT schedule, we first sequence jobs in the non-decreasing order of their processing times, then insert certain maintenance activities into the SPT sequence such that each maintenance is as late as possible. In Akturk et al. [3], it is shown that the SPT schedule has a worst-case bound of 2, i.e., $f(SPT)/f(\pi^*) \leqslant 2$, where π^* is the optimal schedule. We can improve this result by the following analysis.

We start with a relaxation of the problem where pre-emption is allowed, i.e., a job can be interrupted by a maintenance, and resumed later when the maintenance is completed. We use $f'(\pi)$ to denote the total completion time for such a pre-emptive version of the problem.

Consider the pre-emptive SPT (P-SPT) schedule where jobs are sequenced in the non-decreasing order of processing times, and maintenance is scheduled as late as possible, i.e., in the following intervals:

$$[T, T+t],\ [2T+t, 2(T+t)], \ldots, [(L-1)T + (L-2)t, (L-1)(T+t)].$$

In such a schedule, the ith maintenance may interrupt a job, which is denoted by J_i^*, as shown in Fig. 1. For the aim of simplicity, if the ith maintenance does not interrupt a job, we use J_i^* to refer to the job immediately following the maintenance.

Lemma 1. *A P-SPT schedule is optimal to the pre-emptive problem of minimizing total completion time.*

Proof. First, we see that there exists an optimal schedule in which if one job J_{j_1} is interrupted by a maintenance, it should be resumed immediately after the maintenance is completed. If this is not true, for example, another job J_{j_2} is scheduled after the maintenance, then we can swap the processed part of J_{j_1} and J_{j_2} so that part of or entire J_{j_2} is

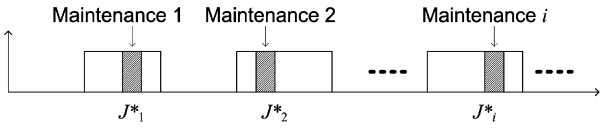

Fig. 1. The P-SPT schedule.

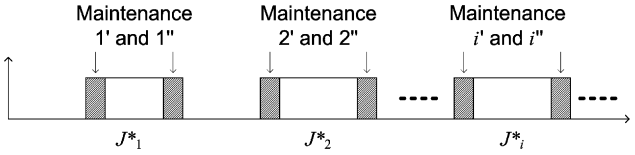

Fig. 2. The N-SPT schedule.

processed before the maintenance. Thus, J_{j_2} will have an earlier completion time, while the completion time for J_{j_1} will not change after such a swap. We also notice that the maintenance activities will not change for any swap of jobs because jobs can be pre-empted.

Now we show the optimality of a P-SPT schedule. Suppose that in an optimal schedule job J_{j1} is immediately followed by J_{j2}, but $p_{j_1} > p_{j_2}$. Then we can swap these two jobs. After swapping, the new completion time of J_{j_2} will be smaller than the original completion time of J_{j_1}, and the new completion time of J_{j_1} will be the same as the original completion time of J_{j_2}. So the total completion time will be reduced after swapping. Therefore, a P-SPT schedule must be optimal. □

Based on Lemma 1, we have

Theorem 1. *The worst-case bound of the SPT schedule is no more than* $1 + t/(t + T)$, *i.e.*,

$$\frac{f(SPT)}{f(\pi^*)} < 1 + \frac{t}{t+T}.$$

Proof. Consider the corresponding pre-emptive version of the problem and a P-SPT schedule. We use $f'(P\text{-}SPT)$ to denote the total completion time for all jobs in the P-SPT schedule. Because the pre-emptive problem is a relaxation of the original problem, it should have a smaller cost than the optimal schedule to the original problem. Formally, we have

$$f'(P\text{-}SPT) \leqslant f(\pi^*).$$

Based on the P-SPT schedule, we can construct another schedule, denoted by N-SPT schedule, which is feasible to the original problem. For each pre-empted job J_i^* in the P-SPT schedule, we first remove the originally scheduled maintenance activity i, then add two maintenance activities i' and i'', one being just before J_i^* and one being just after J_i^*, as shown in Fig. 2. The N-SPT schedule is feasible to the original problem because the distance between any two adjacent maintenance activities is no more than T.

Note that the jobs in the N-SPT schedule are with the same sequence as the jobs in the P-SPT schedule. Suppose in the P-SPT schedule there are l_i jobs following job J_i^*, including the jobs in all the following batches. Then during the construction of the N-SPT schedule, each time a maintenance activity i is replaced by two maintenance activities i' and i'', there are l_i jobs of which the completion time is increased by t. Therefore, we have

$$f(N\text{-}SPT) = f'(P\text{-}SPT) + t(l_1 + l_2 + \cdots + l_{L-1}).$$

Also, in the P-SPT schedule, from the definition of l_i, we have

$$\begin{aligned} f'(P\text{-}SPT) &> (T+t)(l_1 - l_2) + 2(T+t)(l_2 - l_3) + \cdots + (L-1)(T+t)l_{L-1} \\ &= (T+t)(l_1 + l_2 + \cdots + l_{L-1}). \end{aligned}$$

Hence,

$$f(N\text{-}SPT) < f'(P\text{-}SPT) + \frac{t}{t+T} f'(P\text{-}SPT).$$

Finally, the SPT schedule is better than the N-SPT schedule because (1) they have the same job sequence, and (2) the completion time of any job in the SPT schedule is no later than that in the N-SPT schedule. We have (2) because the SPT schedule is actually obtained from the N-SPT schedule by postponing every maintenance activity as late as possible. So,

$$f(SPT) \leqslant f(N\text{-}SPT) < \left(1 + \frac{t}{t+T}\right) f'(P\text{-}SPT) \leqslant \left(1 + \frac{t}{t+T}\right) f(\pi^*). \qquad \square$$

When $t \to 0$, we have $f(SPT)/f(\pi^*) \to 1$, which is consistent with the fact that an SPT schedule is optimal when there is no maintenance requirement. The bound becomes large as t increases, which is consistent to the computational results [3] that the average SPT performance becomes worse when $t > T$.

Because $t/(t+T) < 1$, our bound dominates the result in [3] where the bound of 2 is found. Moreover, under the reasonable assumption that $t < T$, we have $t/(t+T) < \frac{1}{2}$.

Corollary 1. *When $t < T$, the worst-case bound of the SPT schedule is $\frac{3}{2}$.*

However, we still cannot show the tightness of the bound for $t > 0$. The problem instance with the largest bound that we can find has a relative error of $t/(2t + 2.5T)$, or $f(SPT)/f(\pi^*) \to \frac{11}{9}$ when $t \to T$.

Example 1. Suppose that there are five jobs, and their processing times are $p_1 = 4\varepsilon$, $p_2 = T/2 - 2\varepsilon$, $p_3 = T/2 - \varepsilon$, $p_4 = T/2 + 2\varepsilon$, and $p_5 = T/2 + 2\varepsilon$, respectively, where $0 < \varepsilon < T/12$. In an SPT schedule, three maintenance activities are needed following jobs 2, 3, and 4, respectively; and the total completion time is given by $6t + 5T + 15\varepsilon$. When $t > \varepsilon/2$, there is an optimal schedule where jobs 1 and 3 are processed first, then followed by jobs 2, 4, and 5. In such a schedule, two maintenance activities are needed following jobs 3 and 4, respectively; and the total completion time is $4t + 5T + 16\varepsilon$. Therefore, we have $f(SPT)/f(\pi^*) = 1 + (2t - \varepsilon)/(4t + 5T + 16\varepsilon)$. When $\varepsilon \to 0$, $f(SPT)/f(\pi^*) \to 1 + t/(2t + 2.5T)$; and when $t \to T$, $f(SPT)/f(\pi^*) \to \frac{11}{9}$.

In [3], it is shown that when there are three batches in an SPT schedule, there is a tight bound of $\frac{3}{2}$ as $t \to +\infty$. Under the assumption that $t < T$, we can show that the bound is reduced to at most $\frac{21}{17}$. We first need another lemma.

Lemma 2. *If an SPT schedule has three batches, then the completion time of a job j in the third batch is at least $3T/2$, even when $t = 0$.*

Proof. In an SPT schedule, let job k be the first job in the second batch, and T_i be the sum of processing times of jobs in batch i, $i = 1$ and 2. Then from the definition of an SPT schedule, we have $T_1 + p_k > T$, $T_1 + T_2 > T$, and $T_2 + p_j > T$.

If $p_k < T/2$, then $T_1 \geqslant T/2$. Thus $C_j \geqslant T_1 + (T_2 + p_j) \geqslant 3T/2$.

If $p_k \geqslant T/2$, then $p_j \geqslant p_k \geqslant T/2$. Thus $C_j \geqslant (T_1 + T_2) + p_j \geqslant 3T/2$. $\square$

Theorem 2. *If an SPT schedule has three batches and $t < T$, then the worst-case bound is no more than $\frac{21}{17}$.*

Proof. For any feasible schedule π, let $F_S(\pi)$ be the cost of π if $t = 0$, and $F_T(\pi) = f(\pi) - F_S(\pi)$, i.e., $F_T(\pi)$ is the cost purely incurred by maintenance activities. From [3], we know that $f_S(SPT) \leqslant F_S(\pi^*)$ and $f_T(SPT) \leqslant \frac{3}{2} F_T(\pi^*)$, so we have

$$\frac{f(SPT)}{f(\pi^*)} = \frac{F_T(SPT) + F_S(SPT)}{F_T(\pi^*) + F_S(\pi^*)} \leqslant \frac{F_T(SPT) + F_S(SPT)}{\frac{2}{3} F_T(SPT) + F_S(SPT)} = 1 + \frac{1}{2 + (3F_S(SPT))/(F_T(SPT))}.$$

Let C_j^0 be the completion time of a job j in an SPT schedule when $t = 0$, i.e., $F_S(SPT) = \sum_{j=1}^{n} C_j^0$. Now consider each job j's contribution to $F_T(SPT)$. (1) If job j is in the first batch, then its contribution to $F_T(SPT)$ is 0 because of no maintenance before it; (2) if job j is in the second batch, then its contribution to $F_T(SPT)$ is t, which is less than C_j^0

Table 1
Example for the EDD schedule

$j=$	1	2	3	4	5	6	7	8	9
p_j	$T/2$	ε	$T/2$	ε	$T/2$	ε	$T/2$	ε	$T/2$
d_j	$T/9$	$2T/9$	$3T/9$	$4T/9$	$5T/9$	$6T/9$	$7T/9$	$8T/9$	T
$m(j)$	1	1	2	2	2	2	3	3	3
$k(j)$	1	1	2	2	3	3	4	4	5

because $C_j^0 > T > t$; and (3) if job j is in the third batch, then its contribution to $F_T(SPT)$ is $2t$, which is less than $\frac{4}{3}C_j^0$ due to Lemma 2. Overall, we have $F_T(SPT) < \frac{4}{3}F_S(SPT)$. So

$$\frac{f(SPT)}{f(\pi^*)} \leqslant 1 + \frac{1}{2 + (3F_S(SPT))/(F_T(SPT))} < 1 + \frac{1}{2 + 9/4} = \frac{21}{17}. \qquad \square$$

We still cannot show the tightness of the above bound. The largest bound that we can find is $\frac{6}{5}$.

Example 2. Let there be four jobs with $p_1 = 2\varepsilon$, $p_2 = T/2 - \varepsilon$, $p_3 = p_4 = T/2 + \varepsilon$, $\varepsilon < t/2$. Then an SPT schedule has three batches, and we have $f(SPT) = 3T + 3t + 8\varepsilon$. In an optimal schedule π^*, jobs 1 and 3 form the first batch, jobs 2 and 4 form the second batch, and $f(\pi^*) = 3T + 2t + 10\varepsilon$. Thus, when $\varepsilon \to 0$ and $t \to T$, we have $f(SPT)/f(\pi^*) \to \frac{6}{5}$.

4. EDD schedule

In this section, we study the problem of minimizing the maximum lateness, which 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 [6]. Without loss of generality, in this section we assume that jobs are indexed in the EDD order, i.e., $d_1 \leqslant d_2 \leqslant \cdots \leqslant d_n$.

When an EDD schedule is used as a heuristic algorithm to solve the problem, jobs are sequenced in the non-decreasing order of due dates, and maintenance is inserted as late as possible. Because in an optimal schedule π^* it is possible that $L_{\max}(\pi^*) = 0$, any small error of a heuristic algorithm may result in an infinite relative error if the error is measured by the ratio over the optimum. To resolve this issue, we use the absolute error bound to express the worse-case analysis of the EDD schedule. Similar approaches have been used in the literature such as Koulamas and Kyparisis [8].

To quantify the worst-case bound for an EDD schedule, we need the following notation.

$m(j)$: the minimum number of batches that are needed for processing jobs $J_1, J_2, \ldots, J_j$. Note that $m(j)$ can be obtained by solving a bin-packing problem for the first j jobs.

$k(j)$: index of the batch where job J_j is in an EDD schedule.

We use an example to demonstrate these two definitions.

Example 3. Consider a problem with $n = 9$ jobs. The processing time and due date for each job are given in Table 1, where $\varepsilon < T/8$.

Note that in Example 3, $m(5) = 2$ because we only need two batches for processing the first five jobs if we want to minimize the total number of batches, but $k(5) = 3$ because job J_5 is in batch B_3 in an EDD schedule.

In general, we have $m(j) \leqslant k(j)$ for any job J_j. The next theorem indicates that the error of an EDD schedule depends on the difference between $m(j)$ and $k(j)$ for the job J_j that has the largest lateness in an EDD schedule.

Theorem 3. *Suppose that job J_{i^*} is the job with the largest lateness in an EDD schedule, $L_{i^*} = \max_j\{L_j\}$, then*

$$L_{\max}(EDD) \leqslant L_{\max}(\pi^*) + (k(i^*) - m(i^*))t.$$

Proof. From the definition the EDD schedule and job J_{i^*}, we have

$$L_{\max}(EDD) = L_{i^*}(EDD) = p_1 + \cdots + p_{i^*} + (k(i^*) - 1)t - d_{i^*}.$$

Consider an optimal schedule π^*. Suppose job J_j is the last job among jobs $J_1, \ldots, J_{i^*}$ in π^*. Then we know that the earliest batch to which J_j can be scheduled is $B_{m(i^*)}$. Thus

$$C_j(\pi^*) \geqslant p_1 + \cdots + p_{i^*} + (m(i^*) - 1)t.$$

Since $d_j \leqslant d_{i^*}$, we have

$$\begin{aligned} L_j(\pi^*) &= C_j(\pi^*) - d_j \\ &\geqslant p_1 + \cdots + p_{i^*} + (k(i^*) - 1)t + (m(i^*) - k(i^*))t - d_{i^*} \\ &= L_{\max}(EDD) + (m(i^*) - k(i^*))t. \end{aligned}$$

So, $L_{\max}(EDD) \leqslant L_j(\pi^*) + (k(i^*) - m(i^*))t \leqslant L_{\max}(\pi^*) + (k(i^*) - m(i^*))t$. □

The tightness of Theorem 3 can be shown by the problem in Example 3. In an EDD schedule for the problem, we have $i^*=9$, $L_{\max}(EDD)=L_9=3T/2+4\varepsilon+4t$. In an optimal schedule π^* we can process jobs 1, 2, 4, 6, and 8 in the first batch, jobs 3 and 5 in the second batch, and jobs 7 and 9 in the third batch, which leads to $L_{\max}(\pi^*) = L_9 = 3T/2 + 4\varepsilon + 2t$. Therefore, $L_{\max}(EDD) = L_{\max}(\pi^*) + 2t = L_{\max}(\pi^*) + (k(9) - m(9))t$.

From Theorem 3, we immediately have

Corollary 2. *The EDD schedule is optimal to the problem of minimizing the maximum lateness if either one of the following conditions is true*:

(1) *the EDD schedule has only two batches*;
(2) *the job with the largest lateness is in batches* 1 *or* 2, *i.e.*, $k(i^*) = 1$ *or* $k(i^*) = 2$;
(3) $k(i^*) = m(i^*)$.

In Theorem 3, the error bound estimation is expressed by $k(i^*)$ and $m(i^*)$, the exact values of which depend on a specific schedule. So it is not a fixed worst-case bound in the traditional meaning. A stronger result is as follows.

The EDD schedule is actually an implementation of the NEXT FIT algorithm for the bin packing problem where $k(j)$ is the number of bins to contain the first j items in the NEXT FIT solution. It is known that the worst-case bound of the NEXT FIT solution is $2 - \varepsilon$ where ε is an arbitrarily small positive number (see [6] for details). Therefore, for any job J_{i^*} defined in Theorem 3, we have $k(i^*) < 2m(i^*)$, and hence $k(i^*) - m(i^*) < m(i^*)$. Because $k(i^*) - m(i^*)$ is integral, we have $k(i^*) - m(i^*) \leqslant m(i^*) - 1$, and $m(i^*) - 1$ achieves the maximum value when $i^* = n$. So we have

Corollary 3. *The worst-case bound of the EDD schedule is* $(m(n) - 1)t$, *i.e.*,

$$L_{\max}(EDD) \leqslant L_{\max}(\pi^*) + (m(n) - 1)t.$$

The tightness of Corollary 3 can also be shown by the problem in Example 3 where we have $L_{\max}(EDD) \leqslant L_{\max}(\pi^*) + 2t = L_{\max}(\pi^*) + (m(9) - 1)t$.

The worst-case bound given in Corollary 3 is solely determined by the problem input rather than a specific schedule, but it is still difficult to calculate because $m(n)$ is actually the optimal solution to bin packing, a classical NP-hard problem. To obtained an easy-to-calculate bound, we can use an upper bound of $m(n)$, denoted by m_n^+, in the worst-case performance estimate of an EDD schedule such that $L_{\max}(EDD) \leqslant L_{\max}(\pi^*) + (m_n^+ - 1)t$.

It is easy to find m_n^+ by any heuristic algorithm for bin packing problems. For example, using BEST FIT [6] algorithm for the above example, we can find a solution with three bins, i.e., $m_n^+ = 3$. So the bound estimation based on m_n^+ is still tight.

Acknowledgments

The author appreciates the comments from two anonymous reviewers. One detailed step in the proof of Theorem 1 is suggested by one reviewer.

References

[1] M.S. Akturk, J.B. Ghosh, E.D. Gunes, Scheduling with tool changes to minimize total completion time, Technical Report, Department of Industrial Engineering, Bilkent University, Ankara, Turkey, 1999.
[2] M.S. Akturk, J.B. Ghosh, E.D. Gunes, Scheduling with tool changes to minimize total completion time: a study of heuristics and their performance, Naval Res. Logistics 50 (2003) 15–30.
[3] M.S. Akturk, J.B. Ghosh, E.D. Gunes, Scheduling with tool changes to minimize total completion time: basic results and SPT performance, European J. Oper. Res. 157 (2004) 784–790.
[4] C.R. Cassady, E. Kutanoglu, Minimizing job tardiness using integrated preventive maintenance planning and production scheduling, IIE Trans. 35 (2003) 503–513.
[5] W.J. Chen, Minimizing total flow time in the single-machine scheduling problem with periodic maintenance, J. Oper. Res. Soc. 57 (2006) 410–415.
[6] E.G. Coffman, M.R. Garey, D.S. Johnson, Approximation algorithms for bin-packing: a survey, in: D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Company, Boston, MA, 1995.
[7] G.H. Graves, C.-Y. Lee, Scheduling maintenance and semiresumable jobs on a single machine, Naval Res. Logistics 46 (1999) 845–863.
[8] C. Koulamas, G.J. Kyparisis, Scheduling on uniform parallel machines to minimize maximum lateness, Oper. Res. Lett. 26 (2000) 175–179.
[9] C.-Y. Lee, Machine scheduling with availability constraints, in: J.Y.-T. Leung (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, Boca Raton, FL, 2004.
[10] C.-Y. Lee, C.S. Lin, Single-machine scheduling with maintenance and repair rate-modifying activities, European J. Oper. Res. 135 (2001) 493–513.
[11] I. Kacem, C. Chu, Worst case analysis of WSPT and MWSPT for 1, $h||\sum w_i C_i$, International Conference on Industrial Engineering and System Management, Marrakech, Maroc, May 16–19, 2005.
[12] C. Sadfi, B. Penz, C. Rapine, J. Blazewicz, P. Formanowicz, An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints, European J. Oper. Res. 161 (2005) 3–10.
[13] G. Schmidt, Scheduling with limited machine availability, European J. Oper. Res. 121 (2000) 1–15.
[14] X. Qi, T. Chen, F. Tu, Scheduling the maintenance on a single machine, J. Oper. Res. Soc. 50 (1999) 1071–1078.