4OR-Q J Oper Res (2009) 7:79–92
DOI 10.1007/s10288-008-0076-6

RESEARCH PAPER

Approximation algorithms for single machine scheduling with one unavailability period

Imed Kacem · Mohamed Haouari

Received: 29 May 2007 / Revised: 3 April 2008 / Published online: 8 May 2008

Abstract In this paper, we investigate the single machine scheduling problem with release dates and tails and a planned unavailability time period. We show that the problem admits a fully polynomial-time approximation scheme when the tails are equal. We derive an approximation algorithm for the general case and we show that the worst-case bound of the sequence yielded by Schrage's algorithm is equal to 2 and that this bound is tight. Some consequences of this result are also presented.

Keywords Scheduling · Single machine · Approximation · Unavailability constraint

MSC classification (2000) 90B35

1 Introduction

We consider the one-machine scheduling problem with heads and tails, provided that the machine is unavailable during a planned time interval, with the aim of minimizing the makespan. Formally, the problem is defined in the following way. We have to schedule a set J of n jobs on a single machine. Each job $j \in J$ has a processing time p_j, a release date (or head) r_j and a tail q_j. The machine can only perform one job at a given time. The preemption is not permitted. The machine is unavailable during a preset time interval $[T_1, T_2)$. The problem is to find a sequence of jobs, with

I. Kacem (✉)
Charles Delaunay Institute, University of Technology of Troyes,
12 rue Marie Curie, B.P. 2060, 10010 Troyes, France
e-mail: imed.kacem@utt.fr

M. Haouari
Faculty of Economics and Administrative Sciences, Ozyegin University, Istanbul, Turkey
e-mail: mohamed.haouari@ozyegin.edu.tr

the objective of minimizing the makespan $C_{\max} = \max_{1 \leq j \leq n} \{C_j + q_j\}$ where C_j is the completion time of job j. We also define t_j as the starting time of job j (i.e., $C_j = t_j + p_j$). Without loss of generality, we consider that $r_j \notin [T_1, T_2)$ for $j \in J$ and that if $r_j < T_1$, then, the inequality $r_j + p_j \leq T_1$ holds (otherwise, in both cases, r_j would be set equal to T_2).

For self-consistency, we recall some necessary definitions related to the approximation area. A *ρ-approximation algorithm* for a problem of minimizing an objective function φ is an algorithm such that for every instance π of the problem it gives a solution S_π verifying $\varphi(S_\pi)/\varphi(\mathrm{OPT}_\pi) \leq \rho$ where OPT_π is the optimal solution of π. Also, ρ is called the *worst-case bound* of the above algorithm. The approximation is *tight* if ρ is the best possible (i.e., the smallest value we can obtain by the algorithm for all the instances of the problem). A class of $(1+\varepsilon)$-*approximation algorithms* is a fully polynomial-time approximation scheme (FPTAS), if its running time is bounded by a polynomial function in $1/\varepsilon$ and the instance size for every $\varepsilon > 0$. A class of $(1+\varepsilon)$-*approximation algorithms* is a PTAS (*Polynomial-Time Approximation Scheme*), if its running time is an arbitrary function in $1/\varepsilon$ and the instance size for every $\varepsilon > 0$.

Scheduling problems with unavailability constraints have attracted numerous researchers. This has been motivated by practical and real industrial problems. It is noteworthy that during the last decade numerous problems of this class have been addressed in the literature (for more details, see the state-of-the-art papers by Lee 1996, 2004; Schmidt 2000). However, to the best of our knowledge, the worst-case analysis investigated in the present paper has never been addressed before. Maugière et al. (2005) considered the single machine and the job-shop problems under unavailability constraints and proposed a branch-and-bound algorithm to minimize the makespan. Souissi et al. (2006) addressed the special case without tails. Recently, Yuan et al. (2007) developed an interesting PTAS for the problem without release dates and Kacem (2007) proposed an FPTAS for the same problem. That is why this paper is a good attempt to design efficient approximation algorithms for the general problem under release dates and tails assumptions. It is noteworthy that the special case with neither release dates nor tails can be reduced to a classical knapsack problem (for this case see Martello and Toth 1990).

Note that numerous works addressed this type of problem without unavailability constraint. Lawler et al. (1993) proved that this special problem is NP-Hard in the strong sense. Dessouky and Margenthaler (1972) proposed an efficient algorithm for solving this problem. Carlier (1982) studied the Schrage's sequence and derived a branch-and-bound algorithm. Potts (1980) reported that the worst-case performance bound of Schrage's algorithm for the problem without unavailability is 2, and this bound is tight. At this point, it is worth noting that Potts (1980) and Hall and Shmoys (1992) have proposed better approximation algorithms having worst-case performances of $3/2$ and $4/3$, respectively.

The paper is organized as follows. In Sect. 2, we consider the case when the tails are equal and we show that this problem admits an FPTAS. In Sect. 3, we consider the general case and we deduce an approximation algorithm. We also recall the main previous results and we provide the worst-case performance analysis of Schrage's sequence when there is an unavailability period. Moreover, some interesting consequences

are presented. Finally, we conclude by giving some perspectives and concluding remarks.

2 FPTAS for the case when the tails are equal

In this case, tails q_j are equal for all j. Without loss of generality, we assume that $q_j = 0$ for every j and that jobs are sorted in non-decreasing order of release dates (i.e., $r_1 \leq r_2 \leq \cdots \leq r_n$). This special case is denoted as $(\mathcal{P})$. For this configuration, Souissi et al. (2006) showed that the worst case performance bound of the first in first out (FIFO) rule[1] is equal to 2 and that this bound is tight. Due to the dominance of the FIFO order, an optimal solution is composed of two sequences (before and after the unavailability period) of jobs scheduled in nondecreasing order of their indices (see Souissi et al. 2006). Note that if all the jobs can be inserted before T_1, then the FIFO rule is obviously optimal. Therefore, we assume that we cannot insert all jobs before T_1 (consequently, the optimal makespan is strictly greater than T_2).

In the remainder of this paper, $C^*_{\max}(Q)$ denotes the minimal makespan for problem Q and $C^S_{\max}(Q)$ is the makespan of schedule S for problem Q.

2.1 Dynamic programming

The problem can be optimally solved by applying the following dynamic programming algorithm A, which is a weak version of the one proposed in Souissi et al. (2006). This algorithm generates iteratively some sets of states. At every iteration k, a set $\mathcal{V}_k$ composed of states is generated ($1 \leq k \leq n$). Each state $[t, f]$ in $\mathcal{V}_k$ can be associated with a feasible schedule for the first k jobs. Variable t denotes the completion time of the last job scheduled before T_1 and f is the completion time of the last job scheduled after the unavailability period. In a first step, we assume that $r_j < T_1$ for every job j (this assumption is considered to simplify the proofs and we show later that the result holds for arbitrary release dates). This algorithm can be described as follows:

Algorithm A

(i) Set $\mathcal{V}_1 = \{[0, T_2 + p_1], [r_1 + p_1, T_2]\}$.

(ii) For $k \in \{2, 3, \ldots, n\}$,

For every state $[t, f]$ in $\mathcal{V}_{k-1}$:

1. Put $[t, f + p_k]$ in $\mathcal{V}_k$
2. Put $[\max\{t, r_k\} + p_k, f]$ in $\mathcal{V}_k$ if $\max\{t, r_k\} + p_k \leq T_1$

Remove $\mathcal{V}_{k-1}$

(iii) $C^*_{\max}(\mathcal{P}) = \min_{[t,f] \in \mathcal{V}_n} \{f\}$.

Let UB be an upper bound on the optimal makespan for the problem $(\mathcal{P})$. If we add the restriction that for every state $[t, f]$ the relation $f \leq \text{UB}$ must hold, then the running time of A can be bounded by $nT_1\text{UB}$. Indeed, t and f are integers and at each step k, we have to create at most $T_1\text{UB}$ states to construct $\mathcal{V}_k$. Moreover, the complexity of A is proportional to $\sum_{k=1}^{n} |\mathcal{V}_k|$.

[1] FIFO consists in scheduling jobs according to the nondecreasing release dates order.

However, this complexity can be reduced to $O(nT_1)$ as it was done by Souissi et al. (2006), by choosing at each iteration k and for every t the state $[t, f]$ with the smallest value of f.

In the remainder of the paper, algorithm A denotes the weak version of the dynamic programming algorithm by taking $\mathrm{UB} = C_{\max}^{\mathrm{FIFO}}(\mathcal{P})$.

2.2 FPTAS

Our FPTAS is based on two steps and its principle is similar to the one proposed in Kacem (2007). First, we use the 2-approximation algorithm by Souissi et al. (2006). In the second step of our FPTAS, we modify the execution of algorithm A in order to decrease the running time. The principle is to remove a part of the states generated by the algorithm. Therefore, the modified algorithm $A(\varepsilon)$ becomes faster and yields an approximate solution instead of the optimal schedule. Hence, we have to take care when removing such states so that the approximation will be of a good quality.

The approach of *modifying the execution of an exact algorithm* to design an FPTAS, was initially proposed by Ibarra and Kim (1975) for solving the knapsack problem. It is noteworthy that during the last decades numerous scheduling problems have been addressed by applying such an approach (a sample of these papers includes Sahni 1976; Kacem 2007; Kovalyov and Kubiak 1999; Kellerer and Strusevich 2007; Woeginger 2000, 2005).

Given an arbitrary $\varepsilon > 0$, we define

$$\mathrm{LB} = \frac{C_{\max}^{\mathrm{FIFO}}(\mathcal{P})}{2},$$

$$\beta = \left\lceil \frac{2n}{\varepsilon} \right\rceil,$$

and

$$\gamma = \frac{C_{\max}^{\mathrm{FIFO}}(\mathcal{P})}{\beta}.$$

We split the interval $\left[0, C_{\max}^{\mathrm{FIFO}}(\mathcal{P})\right]$ into β equal subintervals $I_h = \left[(h-1)\gamma, h\gamma\right]_{1 \le h \le \beta}$ of length γ. Our algorithm $A(\varepsilon)$ generates reduced sets $\mathcal{V}_k^{\#}$ instead of sets $\mathcal{V}_k$. It can be described as follows:

Algorithm A(ε)

(i) Set $\mathcal{V}_1^{\#} = \{[0, T_2 + p_1], [r_1 + p_1, T_2]\}$.
(ii) For $k \in \{2, 3, \ldots, n\}$,
  For every state $[t, f]$ in $\mathcal{V}_{k-1}^{\#}$:
  1. Put $[t, f + p_k]$ in $\mathcal{V}_k^{\#}$
  2. Put $[\max\{t, r_k\} + p_k, f]$ in $\mathcal{V}_k^{\#}$ if $\max\{t, r_k\} + p_k \le T_1$
  Remove $\mathcal{V}_{k-1}^{\#}$

Let $[t, f]_h$ be the state in $\mathcal{V}_k^{\#}$ such that $f \in I_h$ and $t \leq T_1$ with the smallest possible t (ties are broken by choosing the state of the smallest f). Set $\mathcal{V}_k^{\#} = \left\{[t, f]_h \,|1 \leq h \leq \beta\right\}$.

(iii) $C_{\max}^{A(\varepsilon)}(\mathcal{P}) = \min_{[t,f]\in\mathcal{V}_n^{\#}}\{f\}$.

2.3 Worst-case analysis and complexity

The worst-case analysis of our FPTAS is based on the comparison of the execution of algorithms A and $A(\varepsilon)$. In particular, we focus on the comparison of the states generated by each of the two algorithms. We can remark that the main action of algorithm $A(\varepsilon)$ consists in reducing the cardinality of the state subsets by splitting $[0, T_1] \times \left[0, C_{\max}^{\text{FIFO}}(\mathcal{P})\right]$ into β boxes $[0, T_1] \times I_h$ and by replacing all the vectors of $\mathcal{V}_k$ belonging to $[0, T_1] \times I_h$ by a single *approximate* state with the smallest t.

Lemma 1 *For every state $[t, f]$ in $\mathcal{V}_k$ there exists a state $\left[t^{\#}, f^{\#}\right]$ in $\mathcal{V}_k^{\#}$ such that:*

$$t^{\#} \leq t \tag{1}$$

and

$$f^{\#} \leq f + k\gamma \tag{2}$$

Proof By induction on k.
First, for $k = 1$ we have $\mathcal{V}_1^{\#} = \mathcal{V}_1$. Therefore, the statement is trivial.
Now, assume that the statement holds true up to level $k-1$. Consider an arbitrary state $[t, f] \in \mathcal{V}_k$. Algorithm A introduces this state into $\mathcal{V}_k$ when job k is added to some feasible state for the first $k-1$ jobs. Let $\left[t', f'\right]$ be the above feasible state. Two cases can be distinguished: either $[t, f] = \left[t', f' + p_k\right]$ or $[t, f] = \left[\max\left\{t', r_k\right\} + p_k, f'\right]$ must hold. We will prove the statement for level k in the two cases.

1st case $[t, f] = \left[t', f' + p_k\right]$

Since $\left[t', f'\right] \in \mathcal{V}_{k-1}$, there exists $\left[t'^{\#}, f'^{\#}\right] \in \mathcal{V}_{k-1}^{\#}$ such that $t'^{\#} \leq t'$ and $f'^{\#} \leq f' + (k-1)\gamma$. Consequently, the state $\left[t'^{\#}, f'^{\#} + p_k\right]$ is created by algorithm $A(\varepsilon)$ at iteration k. However it may be removed when reducing the state subset. Let $[\lambda, \mu]$ be the state in $\mathcal{V}_k^{\#}$ that is in the same box as $\left[t'^{\#}, f'^{\#} + p_k\right]$. Hence, we have:

$$\lambda \leq t'^{\#} \leq t' = t$$

and

$$\mu \leq f'^{\#} + p_k + \gamma \leq f' + p_k + (k-1)\gamma + \gamma \leq f' + k\gamma + p_k = f + k\gamma.$$

Consequently, the statement holds for level k in this case.

2nd case $[t,\, f] = \left[\max\left\{t', r_k\right\} + p_k,\, f'\right]$

Since $\left[t',\, f'\right] \in \mathcal{V}_{k-1}$, there exists $\left[t'^{\#},\, f'^{\#}\right] \in \mathcal{V}_{k-1}^{\#}$ such that $t'^{\#} \leq t'$ and $f'^{\#} \leq f' + (k-1)\,\gamma$. Consequently, the state $\left[\max\left\{t'^{\#}, r_k\right\} + p_k,\, f'^{\#}\right]$ is created by algorithm $A\,(\varepsilon)$ at iteration k. However it may be removed when reducing the state subset. Let $\left[\lambda',\, \mu'\right]$ be the state in $\mathcal{V}_k^{\#}$ that is in the same box as $\left[\max\left\{t'^{\#}, r_k\right\} + p_k,\, f'^{\#}\right]$. Hence, we have:

$$\lambda' \leq \max\left\{t'^{\#}, r_k\right\} + p_k \leq \max\left\{t', r_k\right\} + p_k = t$$

and

$$\mu' \leq f'^{\#} + \gamma \leq f' + (k-1)\,\gamma + \gamma \leq f' + k\gamma = f + k\gamma.$$

In conclusion, the statement holds also for level k in the second case, and this completes our inductive proof. □

Theorem 1 *Given an arbitrary $\varepsilon > 0$, algorithm $A\,(\varepsilon)$ yields an output $C_{\max}^{A(\varepsilon)}\,(\mathcal{P})$ such that:*

$$\frac{C_{\max}^{A(\varepsilon)}\,(\mathcal{P})}{C_{\max}^{*}\,(\mathcal{P})} \leq (1+\varepsilon)\,. \tag{3}$$

Proof By definition, the optimal solution can be associated with a state $\left[t^*,\, f^*\right]$ in $\mathcal{V}_n$. From Lemma 1, there exists a state $\left[t^{\#},\, f^{\#}\right]$ in $\mathcal{V}_n^{\#}$ such that:

$$t^{\#} \leq t^*$$

and

$$f^{\#} \leq f^* + n\gamma < C_{\max}^{*}\,(\mathcal{P}) + \varepsilon \mathrm{LB}.$$

Clearly, we have $C_{\max}^{*}\,(\mathcal{P}) \geq \mathrm{LB}$. Therefore, we deduce that

$$f^{\#} \leq (1+\varepsilon)\,C_{\max}^{*}\,(\mathcal{P})$$

Since $C_{\max}^{A(\varepsilon)}\,(\mathcal{P}) \leq f^{\#}$, we conclude that inequality (3) holds. □

Lemma 2 *Given an arbitrary $\varepsilon > 0$, algorithm $A\,(\varepsilon)$ can be implemented in $O\left(n^2/\varepsilon\right)$ time.*

Proof The first step consists in applying Heuristic FIFO, which can be implemented in $O\,(n \log(n))$ time. In the second step, algorithm $A\,(\varepsilon)$ generates the state sets $\mathcal{V}_k^{\#}$ ($k \in \{1, 2, \ldots, n\}$). Since $\left|\mathcal{V}_k^{\#}\right| \leq \beta$, we deduce that

$$\sum_{k=1}^{n} \left|\mathcal{V}_k^{\#}\right| \leq n\beta = n\left\lceil \tfrac{2n}{\varepsilon} \right\rceil \leq n\left(\tfrac{2n}{\varepsilon} + 1\right).$$

Note that algorithm $A(\varepsilon)$ generates $\mathcal{V}_k^{\#}$ by associating every new created state to its corresponding box if and only if such a state has a smaller value of t (in this case, the last state associated with this box will be removed). Otherwise, the new created state will be immediately removed. This allows us to generate $\mathcal{V}_k^{\#}$ in $O(\beta)$ time. Hence, our method can be implemented in $O\left(n\log(n)+n^2/\varepsilon\right)$ time and this completes the proof. □

From Lemma 2 and Theorem 1, the result is proved and the following corollary holds.

Corollary 1 *Algorithm $A(\varepsilon)$ is an FPTAS for the non-resumable version of the problem.*

Remark 1 The approach remains valid when some $r_j \geq T_2$. In this case, we divide J in two subsets J_1 and $J - J_1$ where $r_j < T_1$ if $j \in J_1$ and $r_j \geq T_2$ if $j \in J - J_1$. Then, we apply algorithm $A(\varepsilon)$ to the data of J_1. The obtained solution will be completed by scheduling jobs of $J - J_1$ in nondecreasing order of their release dates after jobs of J_1.

3 Approximation algorithms for the general case

In this section, we consider the case when the tails are arbitrary. Such a problem is NP-Hard in the strong sense and therefore it is not possible to design an FPTAS for it (unless $P \neq NP$). Hence, we prefer to investigate the development of polynomial approximations. We propose two main ideas to design approximation algorithms. The first one is based on our previous FPTAS. The second idea uses the Schrage's sequence.

3.1 A $(2+\varepsilon)$-approximation algorithm

The idea of this algorithm is based on the application of the schedule yielded by the previous FPTAS.

First, jobs are sorted according to the nondecreasing order of their release dates. Then, we partition the jobs in two subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ where $r_j < T_1$ for every $j \in \mathcal{X}_1$ and $r_j \geq T_2$ for every $j \in \mathcal{X}_2$. Let $\overline{\sigma}$ be the schedule given by algorithm $A(\varepsilon)$ for data $\left(r_j\right)_{j\in\mathcal{X}_1}$, $\left(p_j\right)_{j\in\mathcal{X}_1}$, T_1 and T_2. We complete $\overline{\sigma}$ by the jobs of $\mathcal{X}_2$ in nondecreasing order of their indices. Let $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be such a schedule.

Let z be the index such that

$$C_{\sigma(z)} + q_{\sigma(z)} = \max_{1\leq i\leq n}\left\{C_{\sigma(i)} + q_{\sigma(i)}\right\}.$$

Clearly, we have:

$$\begin{aligned} C_{\sigma(z)} + q_{\sigma(z)} &\leq C_{\sigma(n)} + q_{\sigma(z)} \\ &\leq (1+\varepsilon)\, C^*_{\max} + q_{\sigma(z)} \end{aligned} \tag{4}$$

where $C^*_{\max}$ is the optimal makespan.

Moreover, we have:

$$C^*_{\max} \geq q_{\sigma(z)} \tag{5}$$

Therefore, we deduce that:

$$C_{\sigma(z)} + q_{\sigma(z)} \leq (2+\varepsilon)\, C^*_{\max} \tag{6}$$

Consequently, this method yields a $(2+\varepsilon)$-approximation solution in $O\left(n^2/\varepsilon\right)$ time. In the remainder of this paper we show that a simpler Schrage's sequence yields a 2-approximation solution in $O\,(n \log(n))$ time.

3.2 Worst-case analysis of Schrage's algorithm when the machine is continuously available

First, for the sake of completeness we briefly recall Schrage's algorithm and the previous results pertaining to its worst-case performance in case where the machine is continuously available. The algorithm can be described as follows.

Schrage's algorithm

(i) Set $t = \min_{1\leq j\leq n}\left\{r_j\right\}$; $S = \varnothing$;
(ii) At time t, schedule job j such that $j \in W = \{k \in J - S | r_k \leq t\}$ and $q_j = \max_{k\in W}\{q_k\}$.
(iii) Set $S = S \cup \{j\}$; $t_j = t$; $t = \max\left\{t_j + p_j; \min_{k\in J-S}\{r_k\}\right\}$. If $|S| = n$, then STOP; Else, go to step (ii).

In the sequel, without ambiguity we refer to S as the job sequence delivered by the algorithm.

The following result has been derived by Carlier (1982).

Theorem 2 (Carlier 1982) *Let $C^S_{\max}$ be the makespan obtained for S. If S is not optimal, then there exists a critical job c and a critical subset K such that :* $\min_{j\in K}\left\{r_j\right\} + \sum_{j\in K} p_j + \min_{j\in K}\left\{q_j\right\} > C^S_{\max} - p_c$.

For the sake of clarity, we briefly sketch the basic idea of this theorem. To prove the result, the author considers the conjunctive graph associated with Schrage's schedule. He determines the critical path passing throughout the maximum number of jobs. Then, he reindexes the jobs so that the critical path will pass by jobs $1, 2, 3, \ldots, p$. He considers two cases. In the first one, we have: $C^S_{\max} = \min_{1\leq j\leq p}\left\{r_j\right\} + \sum_{1\leq j\leq p} p_j + \min_{1\leq j\leq p}\left\{q_j\right\}$ (case where Schrage's sequence is optimal). In the second case, $q_p \neq \min_{1\leq j\leq p}\left\{q_j\right\}$. This implies the existence of a job $j < p$ such that $q_j < q_p$. Let c be the largest index such that $q_c < q_p$. The critical set is $K = \{c+1, c+2, \ldots, p\}$. The author shows several properties leading to the theorem: $\forall j \in K, q_c < q_j$ and $r_j > t_c$.

In the rest of this paper we will solely focus on the worst-case analysis of Schrage's algorithm in presence of a single unavailability period. The modified algorithm (hereafter denoted by H) consists in taking the sequence delivered by Schrage's algorithm without unavailability constraint.

Table 1 Data of Example 1

j	1	2	3	4	5	6
r_j	0	9	14	21	30	31
p_j	7	7	6	15	2	3
q_j	18	22	27	23	1	9

Fig. 1 Illustration for Example 1

Example 1 Let us consider the 6-job instance that is described in Table 1. The application of Schrage's algorithm yields the following schedule: $t_1 = 0$; $t_2 = 9$; $t_3 = 16$; $t_4 = 22$; $t_5 = 40$; $t_6 = 37$ and $C^S_{\max} = 60$. The corresponding sequence is $S = (1, 2, 3, 4, 6, 5)$.
Now, we assume that the machine is unavailable during the the time interval $[20, 25)$. The schedule corresponding to the sequence S is the following: $t_1 = 0$; $t_2 = 9$; $t_3 = 25$; $t_4 = 31$; $t_5 = 49$; $t_6 = 46$. The new makespan is 69 (see Fig. 1).

3.3 Worst-case analysis of Heuristic H when there is an unavailability period

Theorem 3 *Heuristic H has a worst-case performance of 2 and this bound is tight.*

Proof First, we prove that the worst-case performance bound of Heuristic H is larger than or equal to 2. To that aim, we consider the two-job instance such that $r_1 = 0$; $p_1 = 2\varepsilon$; $q_1 = \varepsilon$; $r_2 = \varepsilon$; $p_2 = q - \varepsilon$; $q_2 = \varepsilon$; $T_1 = q$; $T_2 = q + \varepsilon$ (with $\varepsilon << q$). The sequence yielded by H is $(1, 2)$ with a makespan equal to $C^H_{\max} = 2q + \varepsilon$ while the optimal sequence is $(2, 1)$ with $C^*_{\max} = q + 4\varepsilon$ (see Fig. 2). Thus, $C^H_{\max}/C^*_{\max} = (2q + \varepsilon) / (q + 4\varepsilon) \underset{q \to +\infty}{\to} 2$.

Now, we show that the worst-case performance bound of H is less than or equal to 2. Let t_j refers to the starting time of job $j \in J$ in the schedule produced by H. Several cases may occur:

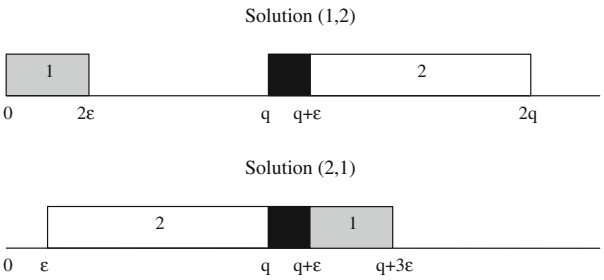

Fig. 2 Worst case example for Heuristic H in the general case

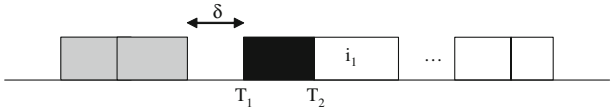

Fig. 3 Definition of δ

(a) Case 1: $\max_{1\leq j\leq n}\left\{t_j + p_j\right\} \leq T_1$: then all the jobs are completed before T_1. In this case, and according to Theorem 2, if H is not optimal, then $C^H_{\max} - C^*_{\max} < p_c$. In addition, we have $C^*_{\max} > p_c$, thus, $C^H_{\max}/C^*_{\max} \leq 2$.

(b) Case 2: a subset of jobs is scheduled after T_2. In this case, let i_1 denote the first job scheduled after T_2. Two sub-cases may occur:

(b.1) $r_{i_1} \geq T_2$. In this case, the unavailability time period does not modify the schedule of Schrage (without unavailability period). In this case, and according to Theorem 2, if the sequence is not optimal, then $C^H_{\max} - C^*_{\max} < p_c$. Hence, we have $C^H_{\max}/C^*_{\max} \leq 2$.

(b.2) $r_{i_1} < T_1$. In this case, the unavailability period leads to delaying the starting time of job i_1 after T_2. Two situations can be distinguished:

(b.2.1) Schrage's sequence S, for the problem without unavailability constraint, is optimal. In this case, let δ denote the time lag between the completion of the last job scheduled prior to the unavailability time period and T_1 (see Fig. 3). Clearly, we have $C^H_{\max} - C^*_{\max} < \delta + \Delta T$ (where, $\Delta T = T_2 - T_1$). On the other hand, $C^*_{\max} > T_2 > \delta + \Delta T$. Therefore, $C^H_{\max}/C^*_{\max} \leq 2$.

(b.2.2) Schrage's sequence S, for the problem without unavailability period, is not optimal. In this case, let c denote the critical job. Two situations need to be considered:

(b.2.2.1) If $i_1 \neq c$, then we have $C^*_{\max} > p_c + p_{i_1} + \Delta T > p_c + \delta + \Delta T$. On the other hand, $C^H_{\max} - C^*_{\max} < \delta + \Delta T + p_c$. Therefore, we get $C^H_{\max}/C^*_{\max} \leq 2$.

(b.2.2.2) If $i_1 = c$, then the critical job is the first job scheduled after the unavailability period (see Fig. 4). Hence, the critical sub-set $K = \{c+1, c+2, \ldots, p\}$ will be scheduled immediately after the critical job c. Thus, $C^H_{\max} = T_2 + \sum_{j\in\{c\}\cup K} p_j + q_p$. Two different cases are considered:

(b.2.2.2.1) If $\Delta T \geq q_p$, then we have: $C^H_{\max} - C^*_{\max} < C^H_{\max} - (t_{c-1} + p_{c-1} + \Delta T + \sum_{j\in\{c\}\cup K} p_j) \leq \delta + q_p \leq$

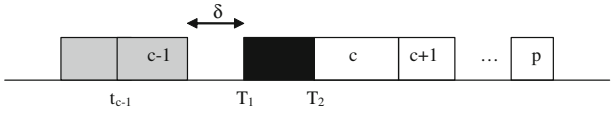

Fig. 4 Critical sub-set

$\delta + \Delta T$. On the other hand, $C^*_{\max} > \delta + \Delta T$. Hence, $C^H_{\max}/C^*_{\max} \leq 2$.

(b.2.2.2.2) If $\Delta T < q_p$. In this case, we have $C^H_{\max} - C^*_{\max} < C^H_{\max} - (t_{c-1} + p_{c-1} + \Delta T + \sum_{j\in\{c\}\cup K} p_j) \leq \delta + q_p$. Now, assume that $C^*_{\max} < \delta + q_p$. To satisfy this assumption, the jobs of $K = \{c+1, c+2, \ldots, p\}$ should be scheduled before the start of the unavailability period and prior to job c. Note that from Theorem 2, we have $r_j > t_{c-1} + p_{c-1}$, $\forall j \in K$. Scheduling the jobs of K before T_1 leads to the impossibility of scheduling c before T_1 and after the jobs of K. This implies that c should be scheduled after T_2 and therefore, this implies that $C^*_{\max} > \delta + \Delta T + p_c$. Hence, we have $C^*_{\max} > \delta + q_p$ or $C^*_{\max} > \delta + \Delta T + p_c$. Consequently, we can deduce that $C^H_{\max}/C^*_{\max} \leq 2$.

Therefore, the result is proven. □

3.4 Consequences

3.4.1 Case where release dates are equal

Now, we turn our attention to the investigation of the worst-case performance of H in case where all jobs have equal release dates. Interestingly, in the following we show that even for this apparently much simpler case, H has a worst-case performance bound equal to 2.

Without loss of generality, we assume that $r_j = 0 \forall j \in J$. Therefore, the sequence obtained by H corresponds to the sequence of jobs ranked in the non-increasing order of their tails.

Proposition 1 *If all the release dates are equal, then the worst-case performance bound of H is larger or equal to* 2.

Proof Consider the two-job instance such that $r_1 = 0$; $p_1 = \varepsilon$; $q_1 = 2\varepsilon$; $r_2 = 0$; $p_2 = q$; $q_2 = \varepsilon$; $T_1 = q$; $T_2 = q + \varepsilon$ (with $\varepsilon << q$). The sequence obtained by H is $(1, 2)$ and has a makespan equal to $C^H_{\max} = 2q + 2\varepsilon$ while the optimal sequence is $(2, 1)$ and $C^*_{\max} = q + 4\varepsilon$ (see Fig. 5). Thus, $C^H_{\max}/C^*_{\max} = (2q + 2\varepsilon) / (q + 4\varepsilon) \underset{q\to+\infty}{\to} 2$. Therefore, the result holds.

In this special case, the worst-case bound is less or equal to 2 (from Theorem 3). Consequently, we have the following theorem.

Theorem 4 *If for every $j = 1, 2, \ldots, n$ we have $r_j = 0$, then the worst-case performance bound of Heuristic H is equal to* 2 *and this bound is tight.*

Note that Lee (1996) considered an equivalent problem with due dates instead of tails in order to minimize the maximum lateness. He proved that the deviation of the Earliest Due Date sequence (EDD) to the optimal one is less than the maximum of

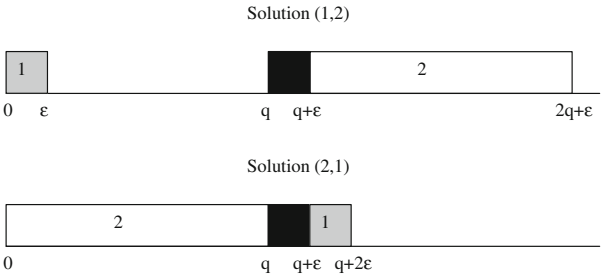

Fig. 5 Worst case example in the case where $r_j = 0$

the processing times. Note that the EDD sequence is exactly the same yielded by Heuristic H and that Lee's result remains valid for the makespan minimization with tails. Therefore, the last result of Theorem 4 refines the one proposed by Lee for the analysis of the EDD sequence.

3.4.2 Asymptotical worst case bound

Theorem 5 *The asymptotical worst-case performance bound of Heuristic H is equal* 2.

Proof First, the asymptotical bound is less than or equal to the absolute bound. This implies that the asymptotical bound is less or equal to 2. On the other hand, consider the following n-job instance: $r_j = 0$; $p_j = 2/(n-1)$; $q_j = 1$; $\forall j \in J \setminus \{n\}$, $r_n = 1$; $p_n = n-1$; $q_n = 1/2$; $T_1 = n$; $T_2 = n+1$. For this instance, we have $C^H_{\max}/C^*_{\max} = (2n+1/2)/(n+2+\lceil \frac{n-1}{2} \rceil . \frac{2}{n-1}) \underset{n \to +\infty}{\to} 2$ (see Fig. 6). Therefore, the result follows. □

3.4.3 Worst case analysis for an improved heuristic

Now, we consider an enhanced variant of H that we denote by H'. This latter could be described in the following way:

(i) Use Heuristic H to get a sequence $S = ([1], [2], \ldots, [n])$,

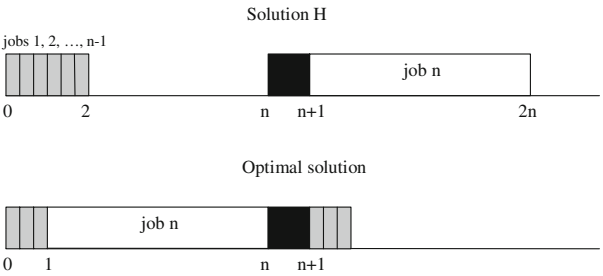

Fig. 6 Example for the asymptotical worst case

(ii) Identify job $[g+1]$ that is the very first job scheduled after T_2,
(iii) Select among the subset $\{[g+2], [g+3], \ldots, [n]\}$ the first job $[l]$ that can be inserted prior to T_1 (i.e. satisfying $r_{[l]} \leq T_1 - p_{[l]}$ and $t_{[g]} + p_{[g]} + p_{[l]} \leq T_1$). If $[l]$ exists, then schedule this job at the position $(g+1)$, update the sequence S and go to step (ii). Else, STOP.

Theorem 6 *The absolute and asymptotical worst case bounds of H' are both equal to 2.*

Proof Clearly, these bounds are less or equal to 2 since H' outperforms H. On the other hand, by considering the two examples used in the previous proofs, it can be established that H and H' yield the same solutions for such examples. Therefore the result follows. □

4 Conclusion

In this paper, we have proposed some approximation algorithms for the single machine scheduling problem under one unavailability constraint. We have shown the existence of an FPTAS for the problem when the tails are equal. A $(2+\varepsilon)$-approximation algorithm has been proposed for the general case. Moreover, we have established that the worst-case performance of Schrage's algorithm is not affected by one unavailability period. We believe that this analysis constitutes a first step toward the investigation of more sophisticated approximation algorithms for machine scheduling with unavailability constraints. An interesting issue that deserves future investigation is whether the worst-case performance of the algorithms of Potts and/or Hall and Shmoys would be affected in the presence of an unavailability period. More generally, the existence of an approximation algorithm, for the one-machine problem with an unavailability period, having a worst-case performance strictly less than 2 remains an open question.

Acknowledgments This work is supported in part by the Conseil Général Champagne-Ardenne, France (Project OCIDI, grant UB902-CR20122-289E). The authors would like to thank two anonymous referees for their helpful suggestions which have significantly improved the presentation of this paper.

References

Carlier J (1982) The one-machine sequencing problem. Eur J Oper Res 11:42–47 (Erratum: Nowicki E, Zdrzalka S (1986) A note on minimizing maximum lateness in a one machine sequencing problem with release dates. Eur J Oper Res 23(2):266–267)

Dessouky MI, Margenthaler CR (1972) The one-machine sequencing problem with early starts and due dates. AIIE Trans 4(3):214–222

Gens GV, Levner EV (1981) Fast approximation algorithms for job sequencing with deadlines. Discr Appl Math 3:313–318

Hall LA, Shmoys DB (1992) Jackson's rule for single machine scheduling: making a good heuristic better. Math Oper Res 17:22–35

Ibarra O, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problems. J ACM 22:463–468

Kacem I (2007) Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. J Comb Optim doi:10.1007/s10878-007-9102-4

Kellerer H, Strusevich VA (2007) Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. Working paper (submitted)

Kovalyov MY, Kubiak W (1999) A fully polynomial approximation scheme for weighted earliness-tardiness problem. Oper Res 47:757–761

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: algorithms and complexity, in logistics of production and inventory. In: Graves SC, Rinnooy Kan AHG, Zipkin PH (eds) Handbooks of operation research management science, vol 4. North-Holland, Amsterdam, pp 445–522

Lee CY (1996) Machine scheduling with an availability constraints. J Global Optim 9:363–384

Lee CY (2004) Machine scheduling with an availability constraint. In: Leung JYT (eds) Handbook of scheduling: algorithms, models, and performance analysis, chap 22, Boca Raton

Martello S, Toth P (1990) Knapsack problems algorithms and computer implementations. Wiley, New York

Maugière Ph, Billaut JC, Bouquard JL (2005) New single machine and job-shop scheduling problems with availability constraints. J Schedul 8:211–231

Potts CN (1980) Analysis of a heuristic for one machine sequencing with release dates and delivery times. Oper Res 28:1436–1441

Sahni S (1976) Algorithms for scheduling independent tasks. J ACM 23:116–127

Schmidt G (2000) Scheduling with limited machine availability. Eur J Oper Res 121:1–15

Souissi A, Kacem I, Chu C (2006) Minimiser le makespan avec prise en compte de contrainte d'indisponibilité sur une seule machine (in French). Valensciences 5:191–206

Woeginger GJ (2000) When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? INFORMS J Comput 12:57–75

Woeginger GJ (2005) A comment on scheduling two machines with capacity constraints. Discr Optim 2:269–272

Yuan JJ, Shi L, Ou JW (2007) Single machine scheduling with forbidden intervals and job delivery times (submitted)