Computers & Industrial Engineering 56 (2009) 1708–1712

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Short Communication

Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval

Imed Kacem [a,*], A. Ridha Mahjoub [b]

[a] Université de Technologie de Troyes, ICD, LOSI, CNRS 2848, France
[b] Université Paris-Dauphine, LAMSADE, CNRS 7024, France

ARTICLE INFO

Article history:
Received 13 February 2008
Received in revised form 27 September 2008
Accepted 29 September 2008
Available online 7 October 2008

Keywords:
Scheduling
Non-availability constraint
Approximation
Weighted flow-time
FPTAS

ABSTRACT

In a recent paper [*Theoretical Computer Science* 363, 257–265], He, Zhong and Gu considered the non-resumable case of the scheduling problem with a fixed non-availability interval under the non-resumable scenario. They proposed a polynomial time approximation scheme (PTAS) to minimize the total completion time.

In this paper, we propose a fully polynomial-time approximation scheme to minimize the total weighted completion time. The FPTAS has $O(n^2/\varepsilon^2)$ time complexity, where n is the number of jobs and ε is the required error bound. The proposed FPTAS outperforms all the previous approximation algorithms designed for this problem and its running time is strongly polynomial.

1. Introduction

This paper focuses on scheduling a set of jobs on a single machine on which a maintenance task has to be performed under the non-resumable scenario. The objective is to minimize the total weighted completion time. The machine is unavailable during a fixed interval. This type of problems has been studied in the literature under various criteria (Gharbi & Haouari, 2005; Lee & Chen, 2000). Given the aim of our study, we present a brief overview of previous works related to this subject.

The simplest model, that is minimizing total completion time with a single period of unavailability (denoted $1,h_1\|\sum C_i$), was proved to be NP-Hard by Adiri, Bruno, Frostig, and Rinnooy Kan (1989) and Lee and Liman (1992). Several references studied the worst-case performance of heuristic methods (a sample of these papers includes Adiri et al., 1989, Lee & Liman, 1992, Sadfi, Penz, Rapine, Blażewicz, & Formanowicz, 2005 & Breit, 2007). Recently, He, Zhong, and Gu (2006) proposed a polynomial time approximation scheme (PTAS).

Numerous references considered the problem of simultaneously scheduling jobs and maintenance tasks on a single machine (see for example Qi, 2007 and Qi, Chen, & Tu, 1999 who considered the minimization of the sum of completion times, or Chen (2006) who proposed a branch-and-bound algorithm for solving a similar problem). Others numerous references addressed the shop scheduling problems (parallel-machine, flow shop and job shop problems) and they proposed exact and heuristic methods (Aggoune, 2004; Aggoune & Portmann, 2006; Allaoui & Artiba, 2006; Allaoui, Artiba, Elmaghraby, & Riane, 2006; Kubzin & Strusevich, 2006; Lee, 1996; Lee, 2004, chap. 22; Lee & Chen, 2000; Schmidt, 2000).

The resumable version of the studied problem was studied by Wang, Sun, and Chu (2005). They also studied the case with arbitrary number of unavailability periods. Recently, Kacem and Chu (2008) studied the $1,h_1\|\sum w_iC_i$ problem under the non-resumable scenario and showed that both WSPT[1] and MWSPT[2] rules have a tight worst-case performance ratio of 3 under some conditions. They also proposed exact methods to solve this problem (Kacem & Chu, 2008; Kacem, Chu, & Souissi, 2008). Kellerer and Strusevich proposed a 4-approximation by converting the resumable solution of Wang et al. (2005) into a feasible solution for the non-resumable scenario. Kacem proposed a 2-approximation algorithm which can be implemented in $O(n^2)$ time (Kacem, 2008). Kellerer and Strusevich proposed an FPTAS (fully polynomial time approximation scheme) for the weighted case with $O(n^4/\varepsilon^2)$ time complexity (Kellerer & Strusevich, in preparation). For these reasons, this paper is a successful attempt to develop faster FPTAS for the $1,h_1\|\sum w_iC_i$ problem under the non-resumable scenario. In Table 1, we

* Corresponding author.
E-mail addresses: imed.kacem@utt.fr (I. Kacem), mahjoub@dauphine.fr (A.R. Mahjoub).

[1] WSPT: weighted shortest processing time.
[2] MWSPT: modified weighted shortest processing time.

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2008.09.042

Table 1
Approximation algorithms for the studied problem.

Studied case	Reference	WCR	TC
Unweighted case ($w_i = 1\ \forall i$)	Lee and Liman (1992) Acta Informatica	9/7	$n \log n$
Unweighted case ($w_i = 1\ \forall i$)	Sadfi et al. (2005) Eur. J. Oper. Res.	20/17	$n \log n$
Unweighted case ($w_i = 1\ \forall i$)	Breit (2007) Eur. J. Oper. Res.	1.072	$n \log n$
Unweighted case ($w_i = 1\ \forall i$)	He et al. (2006) Theor. Comput. Sci.	$1 + 2/(5 + 2(2k+8)^{1/2})$	n^{2k+1}
Unweighted case ($w_i = 1\ \forall i$)	Kellerer and Strusevich (2008) Algorithmica	1+ε	n^3/ε^2
Weighted case (arbitrary w_i)	Kacem and Chu (2008) Eur. J. Oper. Res.	3	$n \log n$
Weighted case (arbitrary w_i)	Kellerer and Strusevich (2008) Algorithmica	1+ε	n^4/ε^2
Weighted case (arbitrary w_i)	Kacem (2008) Comp. Ind. Eng.	2	n^2
Weighted case (arbitrary w_i)	This paper	1+ε	n^2/ε^2

summarize the best known approximation algorithms designed for the studied problem under different assumptions. For every case, we recall the worst-case performance ratio (WCR) and the time complexity (TC) of the above algorithm. The table shows the obvious effectiveness of the algorithm proposed in this paper.

The paper is organized as follows. In Section 2, we describe the problem and we presents an existing exact algorithm. The improved FPTAS is discussed in Section 3. Finally, Section 4 concludes the paper.

2. Problem formulation and exact algorithm

The problem is to schedule a set J of n jobs on a single machine, with the aim of minimizing the total weighted completion time. Every job i has a processing time p_i and a weight w_i. The machine is unavailable between T_1 and T_2 and it can process at most one job at a time. With no loss of generality, we consider that all data are integers and that jobs are indexed according to the *WSPT* rule (i.e., $(p_1/w_1) \leqslant (p_2/w_2) \leqslant \cdots \leqslant (p_n/w_n)$). Due to the dominance of the *WSPT* order, an optimal schedule is composed of two sequences of jobs scheduled in nondecreasing order of their indexes.

If all the jobs can be inserted before T_1, the studied problem ($\mathscr{P}$) has obviously a trivial optimal solution obtained by the *WSPT* rule (Smith, 1956). We therefore consider only the problems in which all the jobs cannot be scheduled before T_1.

In the remainder of this paper, $\varphi^*(Q)$ denotes the minimal weighted sum of the completion times for problem Q and $\varphi_S(Q)$ is the weighted sum of the completion times of schedule S for problem Q.

The problem can be optimally solved by applying the following exact algorithm A, which is a weak version of the one proposed by Kacem et al. (2008). This algorithm generates iteratively some sets of states. At every iteration k, a set $\mathscr{V}_k$ composed of states is generated ($1 \leqslant k \leqslant n$). Each state $[t,f,w^+]$ in $\mathscr{V}_k$ can be associated to 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 total weighted flow-time of the corresponding schedule. Also, it uses artificially an additional variable w^+ for every state, which denotes the sum of weights of jobs scheduled after T_2 for the corresponding state. Note that this variable can be ignored in Algorithm A, but it will be necessary to explain the FPTAS described in the next section. This algorithm can be described as follows:

Algorithm *A*

(i) Set $\mathscr{V}_1 = \{[0, w_1(T_2 + p_1), w_1], [p_1, w_1 p_1, 0]\}$.
(ii) For $k \in \{2, 3, \ldots, n\}$,
For every state $[t,f,w^+]$ in $\mathscr{V}_{k-1}$:
(1) Put $[t, f + w_k(T_2 + \sum_{i=1}^{k} p_i - t), w^+ + w_k]$ in $\mathscr{V}_k$
(2) Put $[t + p_k, f + w_k(t + p_k), w^+]$ in $\mathscr{V}_k$ if $t + p_k \leqslant T_1$

Remove $\mathscr{V}_{k-1}$
(iii) $\varphi^*(\mathscr{P}) = \min_{[t,f] \in \mathscr{V}_n}\{f\}$.

Let UB be an upper bound on the optimal weighted flow-time for problem ($\mathscr{P}$). If we add the restriction that for every state $[t,f]$ the relation $f \leqslant UB$ must hold, then the running time of A can be bounded by nT_1UB. Indeed, t and f are integers and at each step k, we have to create at most T_1UB states to construct $\mathscr{V}_k$. Moreover, the complexity of A is proportional to $\sum_{k=1}^{n} |\mathscr{V}_k|$.

However, this complexity can be reduced to $O(nT_1)$ as it was done by Kacem et al. (2008), 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, we consider the weak version of Algorithm A, that is to say the dynamic programming based algorithm when $UB = \varphi_H(\mathscr{P})$, where H is the heuristic described later in Section 3.

3. FPTAS

Our FPTAS is based on two steps. First, we use the 2-approximation algorithm by Kacem (2008). Then, we apply a modified dynamic programming algorithm. In the remainder of this section, we describe our algorithm, then, we present the proof that it is an FPTAS.

3.1. Algorithm description

Our algorithm starts by computing the upper bound yielded by Kacem's algorithm (denoted H), which extends the one proposed by Wang et al. (2005) for the resumable version of the problem. For self-consistency, we recall the main steps of this algorithm. At each iteration l, we identify a critical job (the first one scheduled after T_2) and we update $\mathscr{G}_l$ (the subset of critical jobs). It can be summarized as follows:

Algorithm *H*

(i) Let $l = 0$ and $\mathscr{G}_l = \emptyset$.
(ii) Let $\pi(i,l)$ be the ith job in $J - \mathscr{G}_l$ according to the WSPT order. Construct a schedule $\sigma_l = \langle \pi(1,l), \pi(2,l), \ldots, \pi(g(l),l), \mathscr{G}_l, \pi(g(l)+1,l), \ldots, \pi(n - |\mathscr{G}_l|, l)\rangle$ such that $\sum_{i \in \mathscr{G}_l} p_i + \sum_{i=1}^{g(l)} p_{\pi(i,l)} \leqslant T_1$ and $\sum_{i \in \mathscr{G}_l} p_i + \sum_{i=1}^{g(l)+1} p_{\pi(i,l)} > T_1$ where jobs in $\mathscr{G}_l$ are sequenced according to the WSPT order.
(iii) If $\sum_{i \in \mathscr{G}_l} p_i + p_{\pi(g(l)+1,l)} \leqslant T_1$, then: $\mathscr{G}_{l+1} = \{\pi(g(l)+1, l)\} \cup \mathscr{G}_l$; $l = l + 1$; go to step (ii). Otherwise, go to step (iv).
(iv) $\varphi_H(\mathscr{P}) = \min_{0 \leqslant h \leqslant l}\{\varphi_{\sigma_h}(\mathscr{P})\}$.

Note that this algorithm has a worst-case performance ratio of 2 and it can be implemented in $O(n^2)$ time (Kacem, 2008).

In the second step of our FPTAS, we modify the execution of algorithm A in order to reduce the running time. The main idea is to remove a special part of the states generated by the algorithm. Therefore, the modified algorithm A' 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 FPTAS, was initially proposed by Ibarra and Kim for solving the knapsack problem (Ibarra & Kim, 1975). It is noteworthy that during the last decades numerous combinatorial problems have been addressed by applying such an approach (a sample of these papers includes Gens & Levner, 1981, Sahni, 1976, Kovalyov & Kubiak, 1999, Kacem, 2007, Kellerer & Strusevich, in preparation and Woeginger, 2000, 2005).

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

$$LB = \frac{\varphi_H(\mathscr{P})}{2}, \tag{1}$$

$$q_1 = \left\lceil \frac{4n}{\varepsilon} \right\rceil, \tag{2}$$

$$q_2 = \left\lceil \frac{2}{\varepsilon} \right\rceil, \tag{3}$$

$$\delta_1 = \frac{\varphi_H(\mathscr{P})}{q_1} \tag{4}$$

and

$$\delta_2 = \frac{T_1}{q_2}. \tag{5}$$

We split the interval $[0, \varphi_H(\mathscr{P})]$ into q_1 equal subintervals $I_r^1 = [(r-1)\delta_1, r\delta_1]_{1 \leqslant r \leqslant q_1}$ of length δ_1. We also split the interval $[0, T_1]$ into q_2 equal subintervals $I_s^2 = [(s-1)\delta_2, s\delta_2]_{1 \leqslant s \leqslant q_2}$ of length δ_2. Our algorithm A'_ε generates reduced sets $\mathscr{V}_k^\#$ instead of sets $\mathscr{V}_k$. It can be described as follows:

Algorithm A'_ε

(i) Set $\mathscr{V}_1^\# = \{[0, w_1(T_2 + p_1), w_1], [p_1, w_1 p_1, 0]\}$.
(ii) For $k \in \{2, 3, \ldots, n\}$,
For every state $[t, f, w^+]$ in $\mathscr{V}_{k-1}^\#$:
(1) Put $[t, f + w_k(T_2 + \sum_{i=1}^{k} p_i - t), w^+ + w_k]$ in $\mathscr{V}_k^\#$
(2) Put $[t + p_k, f + w_k(t + p_k), w^+]$ in $\mathscr{V}_k^\#$ if $t + p_k \leqslant T_1$

Remove $\mathscr{V}_{k-1}^\#$
Let $[t, f, w^+]_{r,s}$ be the state in $\mathscr{V}_k^\#$ such that $f \in I_r^1$ and $t \in I_s^2$ with the smallest possible t (ties are broken by choosing the state of the smallest f). Set $\mathscr{V}_k^\# = \{[t, f, w^+]_{r,s} | 1 \leqslant r \leqslant q_1, 1 \leqslant s \leqslant q_2\}$.
(iii) $\varphi_{A'_\varepsilon}(\mathscr{P}) = \min_{[t,f,w^+] \in \mathscr{V}_n^\#} \{f\}$.

As an illustration of the Algorithm A', Fig. 1 shows the set of states obtained after the reduction step compared to the set generated by Algorithm A (for a given iteration k). We can see that the subset of states in each box $I_r^1 \times I_s^2$ is replaced by a single state (corresponding to the smallest t).

3.2. 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'_ε. 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'_ε consists in reducing the cardinal of the state subsets by splitting $[0, \varphi_H(\mathscr{P})] \times [0, T_1]$ into $q_1 q_2$ boxes $I_r^1 \times I_s^2$ and by replacing all the vectors of $\mathscr{V}_k$ belonging to $I_r^1 \times I_s^2$ by a single "approximate" state with the smallest t.

Lemma 1. *For every state $[t, f, w^+]$ in $\mathscr{V}_k$ there exists a state $[t^\#, f^\#, w^{+\#}]$ in $\mathscr{V}_k^\#$ such that*

$$t^\# \leqslant t \tag{6}$$

and

$$f^\# \leqslant f + k\delta_1 + w^+\delta_2 \tag{7}$$

Proof. By induction on k.

First, for $k = 1$ we have $\mathscr{V}_1^\# = \mathscr{V}_1$. Therefore, the statement is trivial.

Now, assume that the statement holds true up to level $k-1$. Consider an arbitrary state $[t, f, w^+] \in \mathscr{V}_k$. Algorithm A introduces this state into $\mathscr{V}_k$ when job k is added to some feasible state for the first $k-1$ jobs. Let $[t', f', w']$ be the above feasible state. Two cases can be distinguished: either $[t, f, w^+] = [t' + p_k, f' + w_k(t' + p_k), w']$ or $[t, f, w^+] = [t', f' + w_k(T_2 + \sum_{i=1}^{k} p_i - t'), w' + w_k]$ must hold. We will prove the statement for level k in the two cases.

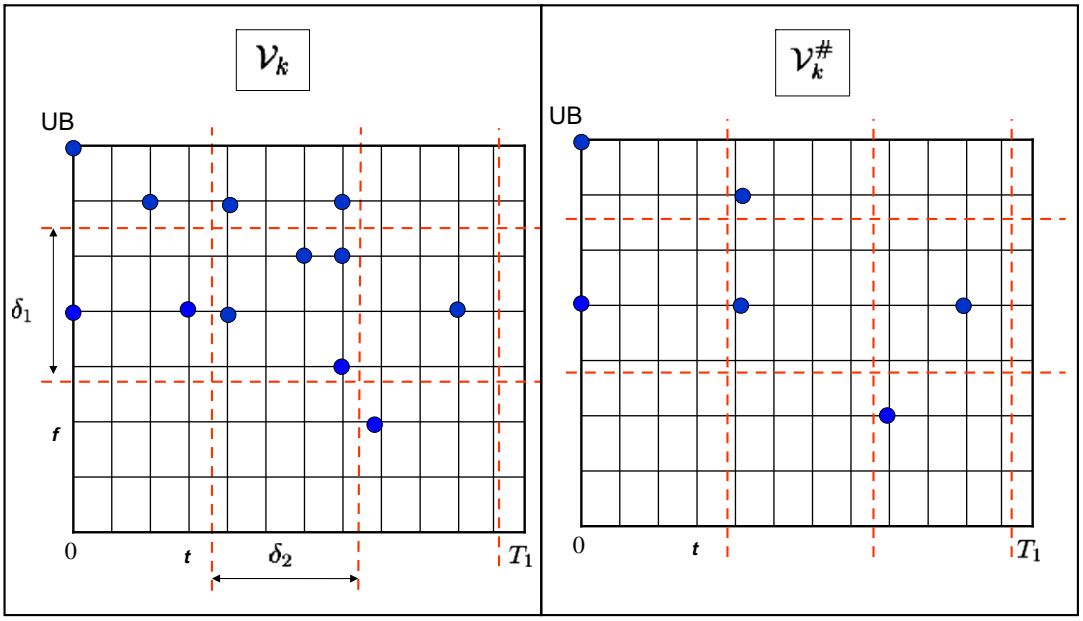

Fig. 1. Illustration of the reduction step of Algorithm A'.

1st Case: $[t,f,w^+] = [t'+p_k, f'+w_k(t'+p_k), w']$

Since $[t',f',w'] \in \mathscr{V}_{k-1}$, there exists $[t'^{\#},f'^{\#},w'^{\#}] \in \mathscr{V}^{\#}_{k-1}$ such that $t'^{\#} \leqslant t'$ and $f'^{\#} \leqslant f' + (k-1)\delta_1 + w'\delta_2$. Consequently, the state $[t'^{\#}+p_k, f'^{\#}+w_k(t'^{\#}+p_k), w'^{\#}]$ is generated by algorithm A'_ε at iteration k. However, it may be removed when reducing the state subset. Let $[\lambda,\mu,\nu]$ be the state in $\mathscr{V}^{\#}_k$ that is in the same box as $[t'^{\#}+p_k, f'^{\#}+w_k(t'^{\#}+p_k), w'^{\#}]$. Hence, we have

$$\lambda \leqslant t'^{\#} + p_k \leqslant t' + p_k = t \tag{8}$$

and

$$\begin{aligned}\mu &\leqslant f'^{\#} + w_k(t'^{\#}+p_k) + \delta_1 \\ &\leqslant f' + (k-1)\delta_1 + w'\delta_2 + w_k(t'^{\#}+p_k) + \delta_1 \\ &\leqslant f' + k\delta_1 + w'\delta_2 + w_k(t'+p_k) = f + k\delta_1 + w^+\delta_2.\end{aligned} \tag{9}$$

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

2nd Case: $[t,f,w^+] = [t', f' + w_k(T_2 + \sum_{i=1}^{k} p_i - t'), w' + w_k]$

Since $[t',f',w'] \in \mathscr{V}_{k-1}$, there exists $[t'^{\#},f'^{\#},w'^{\#}] \in \mathscr{V}^{\#}_{k-1}$ such that $t'^{\#} \leqslant t'$ and $f'^{\#} \leqslant f' + (k-1)\delta_1 + w'\delta_2$. Consequently, the state $[t'^{\#}, f'^{\#} + w_k(T_2 + \sum_{i=1}^{k} p_i - t'^{\#}), w'^{\#} + w_k]$ is generated by algorithm A'_ε at iteration k. However it may be removed when reducing the state subset. Let $[\lambda',\mu',\nu']$ be the state in $\mathscr{V}^{\#}_k$ that is in the same box as $[t'^{\#}, f'^{\#} + w_k(T_2 + \sum_{i=1}^{k} p_i - t'^{\#}), w'^{\#} + w_k]$. Hence, we have

$$\lambda' \leqslant t'^{\#} \leqslant t' = t \tag{10}$$

and

$$\begin{aligned}\mu' &\leqslant f'^{\#} + w_k\left(T_2 + \sum_{i=1}^{k} p_i - t'^{\#}\right) + \delta_1 \\ &\leqslant f' + (k-1)\delta_1 + w'\delta_2 + w_k\left(T_2 + \sum_{i=1}^{k} p_i - t'^{\#}\right) + \delta_1.\end{aligned} \tag{11}$$

Note that $t'^{\#} \geqslant t' - \delta_2$. Hence,

$$\begin{aligned}\mu' &\leqslant f' + k\delta_1 + w'\delta_2 + w_k\left(T_2 + \sum_{i=1}^{k} p_i - (t' - \delta_2)\right) \\ &= f' + k\delta_1 + (w' + w_k)\delta_2 + w_k\left(T_2 + \sum_{i=1}^{k} p_i - t'\right) \\ &= f + k\delta_1 + w^+\delta_2\end{aligned} \tag{12}$$

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'_ε yields an output $\varphi_{A'_\varepsilon}(\mathscr{P})$ such that*

$$\varphi_{A'_\varepsilon}(\mathscr{P}) - \varphi^*(\mathscr{P}) \leqslant \varepsilon\varphi^*(\mathscr{P}). \tag{13}$$

Proof. By definition, the optimal solution can be associated to a state $[t^*,f^*,w^{+*}]$ in $\mathscr{V}_n$. From Lemma 1, there exists a state $[t^{\#},f^{\#},w^{+\#}]$ in $\mathscr{V}^{\#}_n$ such that

$$t^{\#} \leqslant t^* \tag{14}$$

and

$$\begin{aligned}f^{\#} &\leqslant f^* + n\delta_1 + w^{+*}\delta_2 = f^* + n\frac{\varphi_H(\mathscr{P})}{q_1} + w^{+*}\frac{T_1}{q_2} \\ &= f^* + n\frac{\varphi_H(\mathscr{P})}{\lceil \frac{4n}{\varepsilon} \rceil} + w^{+*}\frac{T_1}{\lceil \frac{2}{\varepsilon} \rceil} \leqslant f^* + n\frac{\varphi_H(\mathscr{P})}{\frac{4n}{\varepsilon}} + w^{+*}\frac{T_1}{\frac{2}{\varepsilon}} \\ &= \varphi^*(\mathscr{P}) + \varepsilon\frac{LB}{2} + w^{+*}\varepsilon\frac{T_1}{2}.\end{aligned} \tag{15}$$

Clearly, we have $\varphi^*(\mathscr{P}) \geqslant LB$. Since every job scheduled after T_2 in the optimal schedule has a completion time greater than T_2, it is obvious that

$$\varphi^*(\mathscr{P}) > w^{+*}T_1. \tag{16}$$

Therefore, we deduce that

$$f^{\#} < \varphi^*(\mathscr{P}) + \varepsilon\frac{\varphi^*(\mathscr{P})}{2} + \varepsilon\frac{\varphi^*(\mathscr{P})}{2} = (1+\varepsilon)\varphi^*(\mathscr{P}). \tag{17}$$

Since $\varphi_{A'_\varepsilon}(\mathscr{P}) \leqslant f^{\#}$, we conclude that Eq. (13) holds. □

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

Proof. The first step consists in applying heuristic H, which can be implemented in $O(n^2)$ time. In the second step, algorithm A'_ε generates the state sets $\mathscr{V}^{\#}_k$ ($k \in \{1,2,\ldots,n\}$). Since $|\mathscr{V}^{\#}_k| \leqslant q_1q_2$, we deduce that

$$\sum_{k=1}^{n} |\mathscr{V}^{\#}_k| \leqslant nq_1q_2 = n\left\lceil\frac{4n}{\varepsilon}\right\rceil\left\lceil\frac{2}{\varepsilon}\right\rceil \leqslant n\left(\frac{4n}{\varepsilon}+1\right)\left(\frac{2}{\varepsilon}+1\right). \tag{18}$$

Note that algorithm A'_ε generates $\mathscr{V}^{\#}_k$ 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 to this box will be removed). Otherwise, the new created state will be immediately removed. This allows us to generate $\mathscr{V}^{\#}_k$ in $O(q_1q_2)$ time. Hence, our method can be implemented in $O(n^2 + n^2/\varepsilon^2)$ time and this completes the proof. □

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

Corollary 1. *Algorithm A'_ε is an FPTAS for the non-resumable version of problem $1,h_1\|\sum w_iC_i$.*

Remark 1. The approach of Woeginger, 2005 can also be applied to obtain FPTAS for this problem. However, this needs an implementation in $O(|I|^3n^3/\varepsilon^3)$, where $|I|$ is the input size.

4. Conclusion

In this paper, we considered the non-resumable version of the scheduling problem on a single machine with one availability constraint. We studied the criterion of the weighted sum of the completion times. We proposed an FPTAS to solve the problem. The proposed FPTAS has a strongly polynomial running time and outperforms all the previous algorithms for this problem. In our future works, we hope to extend these results to other variants of this problem. The development of better approximation algorithms is also a challenging subject.

References

Adiri, I., Bruno, J., Frostig, E., & Rinnooy Kan, A. H. G. (1989). Single machine flow-time scheduling with a single breakdown. *Acta Informatica, 26*, 679–696.

Aggoune, R. (2004). Minimizing the makespan for the flow shop scheduling problem with availability constraints. *European Journal of Operational Research, 153*, 534–543.

Aggoune, R., & Portmann, M.-C. (2006). Flow shop scheduling problem with limited machine availability: a heuristic approach. *International Journal of Production Economics, 99*, 4–15.

Allaoui, A., & Artiba, A. (2006). Scheduling two-stage hybrid flow shop with availability constraints. *Computers & Operations Research, 33*, 1399–1419.

Allaoui, A., Artiba, A., Elmaghraby, S. E., & Riane, F. (2006). Scheduling of a two-machine flowshop with availability constraints on the first machine. *International Journal of Production Economics, 99*, 16–27.

Breit, J. (2007). Improved approximation for non-preemptive single machine flow-time scheduling with an availability constraint. *European Journal of Operational Research, 183*, 516–524.

Chen, W. J. (2006). Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. *Journal of the Operational Research Society, 57*, 410–415.

Gens, G. V., & Levner, E. V. (1981). Fast approximation algorithms for job sequencing with deadlines. *Discrete Applied Mathematics, 3*, 313–318.

Gharbi, A., & Haouari, M. (2005). Optimal parallel machines scheduling with availability constraints. *Discrete Applied Mathematics, 148*, 63–87.

He, Y., Zhong, W., & Gu, H. (2006). Improved algorithms for two single machine scheduling problems. *Theoretical Computer Science, 363*, 257–265.

Ibarra, O., & Kim, C. E. (1975). Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM, 22*, 463–468.

Kacem, I. (2007). Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed nonavailability interval. *Journal of Combinatorial Optimization* (in press). doi:10.1007/s10878-007-9102-4.

Kacem, I. (2008). Approximation algorithm for the weighted flowtime minimization on a single machine with a fixed non-availability interval. *Computers & Industrial Engineering, 54*, 401–410.

Kacem, I., & Chu, C. (2008). Worst-case analysis of the WSPT and MWSPT rules for single machine scheduling with one planned setup period. *European Journal of Operational Research, 187*, 1080–1089.

Kacem, I., & Chu, C. (2008). Efficient branch-and-bound algorithm for minimizing the weighted sum of completion times on a single machine with one availability constraint. *International Journal of Production Economics, 112*, 138–150.

Kacem, I., Chu, C., & Souissi, A. (2008). Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times. *Computers & Operations Research, 35*, 827–844.

Kellerer, H., & Strusevich, V. A. (in preparation). Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica*.

Kovalyov, M. Y., & Kubiak, W. (1999). A fully polynomial approximation scheme for weighted earliness-tardiness problem. *Operations Research, 47*, 757–761.

Kubzin, M. A., & Strusevich, V. A. (2006). Planning machine maintenance in two machine shop scheduling. *Operations Research, 54*, 789–800.

Lee, C. Y. (1996). Machine scheduling with an availability constraints. *Journal of Global Optimization, 9*, 363–384.

Lee, C. Y. (2004). Machine scheduling with an availability constraint. In J. Y. T. Leung (Ed.), *Handbook of scheduling: Algorithms, models, and performance analysis*, Boca Raton, FL, USA.

Lee, C. Y., & Chen, Z. L. (2000). Scheduling of jobs and maintenance activities on parallel machines. *Naval Research Logistics, 47*, 145–165.

Lee, C. Y., & Liman, S. D. (1992). Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica, 29*, 375–382.

Qi, X. (2007). A note on worst-case performance of heuristics for maintenance scheduling problems. *Discrete Applied Mathematics, 155*, 416–422.

Qi, X., Chen, T., & Tu, F. (1999). Scheduling the maintenance on a single machine. *Journal of the Operational Research Society, 50*, 1071–1078.

Sadfi, C., Penz, B., Rapine, C., Blażewicz, J., & Formanowicz, P. (2005). An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. *European Journal of Operational Research, 161*, 3–10.

Sahni, S. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM, 23*, 116–127.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research, 121*, 1–15.

Smith, W. E. (1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly, 3*, 59–66.

Wang, G., Sun, H., & Chu, C. (2005). Preemptive scheduling with availability constraints to minimize total weighted completion times. *Annals of Operations Research, 133*, 183–192.

Woeginger, G. J. (2000). When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing, 12*, 57–75.

Woeginger, G. J. (2005). A comment on scheduling two machines with capacity constraints. *Discrete Optimization, 2*, 269–272.