RAIRO-Oper. Res. 51 (2017) 1177–1188
DOI: 10.1051/ro/2017044

RAIRO Operations Research
www.rairo-ro.org

STRONGLY FULLY POLYNOMIAL TIME APPROXIMATION SCHEME FOR THE WEIGHTED COMPLETION TIME MINIMIZATION PROBLEM ON TWO-PARALLEL CAPACITATED MACHINES *

Imed Kacem[1], Myriam Sahnoune[1] and Günter Schmidt[2]

Abstract. We consider the total weighted completion time minimization for the two-parallel capacitated machines scheduling problem. In this problem, one of the machines can process jobs until a certain time T_1 after which it is no longer available. The other machine is continuously available for performing jobs at any time. We prove the existence of a strongly Fully Polynomial Time Approximation Scheme (FPTAS) for the studied problem, which extends the results for the unweighted version (see [I. Kacem, Y. Lanuel and M. Sahnoune, Strongly Fully Polynomial Time Approximation Scheme for the two-parallel capacitated machines scheduling problem, *Int. J. Plann. Sched.* **1** (2011) 32–41]). Our FPTAS is based on the simplification of a dynamic programming algorithm. Moreover, we present a set of numerical experiments and we discuss the results.

Mathematics Subject Classification. 90C59, 90C39, 90B35.

Received September 13, 2016. Accepted May 26, 2017.

1. Introduction

Motivated by important real and industrial applications, the problems of scheduling with non-availability constraints have been a challenging subject for many research teams from different fields (Computer Science, Operational Research, Logistics ...). Such applications include especially maintenance activities and non-availability of resources. Different classes of these problems have been recently studied in the literature. Various approaches have been proposed (for instance, the reader is invited to consult the state-of-the-art paper by [17]). In this context, this paper is devoted to study a specific model related to this class of scheduling problems. Indeed, we consider the total weighted completion time minimization for the two-parallel capacitated machines scheduling problem. In this problem, one of the machines can process jobs until a certain time T_1 after which it is no longer available. To the best of our knowledge, the weighted version of the problem studied in this paper has not been addressed in previous references. That is why this paper is a good attempt to study this problem and examine the existence of a strongly Fully Polynomial Time Approximation Scheme (FPTAS) for the above problem.

Keywords. Scheduling, parallel machine, approximation.

* *The short version of this work has been presented at the 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, Troyes, France, 28-30 June 2016.*

[1] Université de Lorraine, LCOMS EA 7306, 57000, Metz, France.
imed.kacem@ univ-lorraine.fr; myriam.sahnoune@ univ-lorraine.fr

[2] Universität des Saarlandes, ORBI, Sarbrucken, Germany. gs@orbi.uni-saarland.de

TABLE 1. Summary of results.

Problem	Result	Reference
$2, h_{1,\infty}\|\|\sum w_jC_j$	FPTAS: $O(\|I\|^3n^3/\varepsilon^2)$	Woeginger [19]
$2, h_{1,\infty}\|\|\sum w_jC_j$	FPTAS: $O(n^4\log\log n + n^4/\varepsilon^2)$	Xu [20]
$2, h_{1,\infty}\|\|\sum C_j$	FPTAS: $O(n^3/\varepsilon^2)$	Kacem et al. [5]
$2, h_{1,\infty}\|\|\sum w_jC_j$	FPTAS: $O(n^3/\varepsilon^2)$	this paper

More precisely, the considered problem consists in scheduling n jobs on two parallel machines where one of these machines is not available after time T_1. The other machine is continuously available for processing jobs at any time. The objective is to elaborate a feasible schedule by minimizing the total weighted completion time. The problem can be denoted as $2, h_{1,\infty}||w_jC_j$, where $h_{1,\infty}$ denotes the non-availability constraint of the first machine. Given the aim of this paper, we recall some related works on the unweighted version of the studied problem. In [14], Lee and Liman elaborated a 3/2-approximation algorithm. In [15], Liao *et al.* introduced some upper and lower bounds and proposed a branch-and-bound procedure for the same problem. It is worth-noting that other general approximation methods can be used to elaborate an FPTAS for close problems (for example, see Kellerer and Strusevich [11], Kovalyov and Kubiak [13], Woeginger[19]). Despite these interesting methods, the time complexity will not be strongly polynomial without adapted upper and lower bounds for the approximate values of some variables used in these algorithms. In contrast to the technique by Woeginger [19], the introduction of these bounds leads to a reduced time complexity. Recently, Kacem *et al.* [5] proposed an FPTAS that can be implemented in a strongly polynomial time for the unweighted case $(2, h_{1,\infty}||C_j)$. The unconstrained version of the problem $(2||w_jC_j)$ has been studied by [16] who established the existence of an FPTAS with a strongly polynomial time.

To summarize, Table 1 compares the results of our paper to the best recent ones of the literature and it shows the interest of our approach.

Remarks: the value $|I|$ mentioned in Table 1 in the time complexity obtained by applying Woeginger's approach [19] represents the input size. In our case, it can be bounded by $O(n\log\max_j\{p_j, w_j\})$. It is worthy to note that from the dynamic programming described in Section 3, we can establish that Woeginger's approach can leads to the existence of an FPTAS of $O(|I|^3n^3/\varepsilon^2)$ time (see [19] for more detail). Moreover, the formulation of the considered problem can be reduced to an instance of the symmetric quadratic knapsack problem for which Kellerer and Strusevich [11] established an FPTAS, improved later by Xu [20] with the complexity mentioned in Table 1 ($O(n^4\log\log n + n^4/\varepsilon^2)$).

The paper is organized as follows. In Section 2, the problem formulation is presented. Then, a dynamic programming algorithm is described in Section 3. The proposed FPTAS for the total weighted completion time minimization problem is discussed in Section 4. Section 5 is devoted to present the numerical experiments and to discuss the results. Finally, Section 6 is a conclusion in which some perspectives are introduced.

2. Weighted completion time minimization on capacitated two-parallel machines

The problem consists in scheduling n jobs on two-parallel machines, with the aim of minimizing the total weighted completion time (*i.e.*, the total weighted flow-time since all the jobs are ready for processing at time 0). The first machine is only available for a given period of time $[0, T_1]$ (*i.e.*, after T_1 it can no longer process any job). This parameter T_1 is known in advance. The second machine is continuously available. Every machine can process at most one job at a time. Every job j is characterized by a processing time p_j and a weight w_j. Without loss of generality, we consider that all data are integers and that jobs are indexed according to the $WSPT$ rule:

$$p_1/w_1 \leq p_2/w_2 \leq \ldots \leq p_n/w_n \tag{2.1}$$

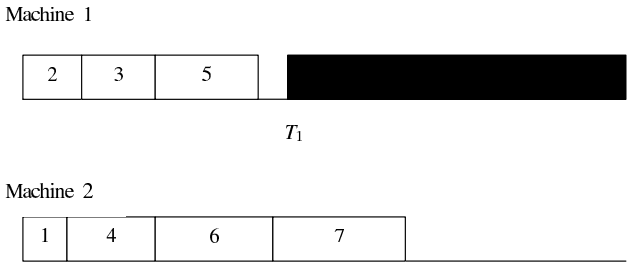

FIGURE 1. Example illustration.

Due to the dominance of the $WSPT$ rule, an optimal solution is composed of two subsets (one subset for each machine) of jobs scheduled in nondecreasing order of their indexes (see [18]). In Figure 1, we present a feasible schedule for a 7-job instance characterized by the following data: $p_1 = 1$, $w_1 = 2$, $p_2 = 2$, $w_2 = 3$, $p_3 = 3$, $w_3 = 3$, $p_4 = 4$, $w_4 = 4$, $p_5 = 4$, $w_5 = 3$, $p_6 = 5$, $w_6 = 3$, $p_7 = 5$, $w_7 = 2$ and $T_1 = 10$.

In the remainder of the paper, we denote by $\mathcal{Q}$ the studied problem, by $\mathcal{F}^*(Q)$ the minimal sum of the weighted completion times for problem Q and by $\mathcal{F}_S(Q)$ the sum of the completion times of schedule S for problem Q. Some necessary standard definitions related to the approximation field are recalled for self-consistency:

Definition 2.1. 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 $\varphi(S_\pi)/\varphi(OPT_\pi) \leq \rho$ where OPT_π is the optimal solution of π. Moreover, ρ is called the *worst-case bound* of the above algorithm.

Definition 2.2. A class of $(1+\varepsilon)$-*approximation algorithms* represents an FPTAS, if its running time is bounded by a polynomial function in $1/\varepsilon$ and the instance size for every $\varepsilon > 0$. It is well-known that an FPTAS is the best possible result for an NP-hard problem unless P = NP.

Definition 2.3. A class of $(1 + \varepsilon)$-*approximation algorithms* is a PTAS (Polynomial Time Approximation Scheme), if its running time is a polynomial function in the instance size and an arbitrary function in $1/\varepsilon$ for every $\varepsilon > 0$.

Proposition 2.4. *If $\sum_{j=1}^{n} p_j \leq 2T_1$, then problem $(\mathcal{Q})$ has an FPTAS with a strongly polynomial time.*

Proof. We relax the non-availability constraint (*i.e.*, we assume that the first machine is continuously available). Then, the relaxed problem has an FPTAS of a strongly polynomial time according to [16]. Let σ_1 be the obtained schedule by applying such an FPTAS for the relaxed problem, B'_1 be the completion time of the last job scheduled on the first machine and B'_2 denote the completion time of the last job scheduled on the second machine. By assumption we know that $B'_1 + B'_2 \leq 2T_1$. Therefore, either $B'_1 \leq T_1$ or $B'_2 \leq T_1$ must hold. If $B'_1 \leq T_1$, then σ_1 is also a $(1+\varepsilon)$-approximation for the original problem $\mathcal{Q}$. If $B'_2 \leq T_1$, then by swapping the two machines, a $(1+\varepsilon)$-approximation schedule σ'_1 is obtained such that σ'_1 is also feasible for the original problem $\mathcal{Q}$. Thus, the proposition is established. □

Based on Proposition 2.4, we limit our investigation to the case where

$$\sum_{j=1}^{n} p_j > 2T_1 \tag{2.2}$$

3. Dynamic programming procedure

In this section, we show that the studied problem can be optimally solved by applying the following standard dynamic programming procedure B. Such a procedure needs $n+1$ iterations in which it creates some sets of states. At every iteration k, a set $\mathcal{U}_k$ composed of states is generated ($0 \leq k \leq n$). Each state $[t, f]$ in $\mathcal{U}_k$ is associated to a feasible schedule for the first k jobs. Variable t denotes the completion time of the last job scheduled on the first machine before T_1 and f is the total weighted completion time of the corresponding schedule. The following algorithm describes the proposed method:

Algorithm B

(i). $\mathcal{U}_0 := \{[0,0]\}$.

(ii). For $k \in \{1, 2, \ldots, n\}$,

Initialize $\mathcal{U}_k := \varnothing$

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

1) Add state $\left[t, f + w_k \left(\sum_{j=1}^{k} p_j - t\right)\right]$ to $\mathcal{U}_k$ (*i.e.*, job k is performed on M_2)

2) Add state $[t + p_k, f + w_k (t + p_k)]$ to $\mathcal{U}_k$ if $t + p_k \leq T_1$ (*i.e.*, job k is performed on M_1)

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

(iii). $\mathcal{F}^* (\mathcal{Q}) := \min \{f | [t, f] \in \mathcal{U}_n\}$.

As an illustration of this dynamic programming algorithm, Table 2 depicts the generated states when we apply this method to the instance previously mentioned for the first 4 iterations.

It is worthy to note that the time complexity of this algorithm can be bounded by $O(nT_1\mathcal{Z})$ where $\mathcal{Z}$ is an upper bound (on the optimal weighted completion time) obtained by Heuristic H (described in the next section). This complexity can be reduced to $O(nT_1)$ if at every iteration k in the algorithm, we only keep for t one state $[t, f]$ with the smallest value of f. By applying this reduction to the example in Table 2, we obtain the results in Table 3.

Table 2. Illustration of Algorithm B.

k	$\mathcal{U}_k$
0	$\{[0,0]\}$
1	$\{[1,2]; [0,2]\}$
2	$\{[3,11]; [1,8]; [2,8]; [0,11]\}$
3	$\{[6,29]; [3,20]; [4,20]; [1,23]; [5,23]; [2,20]; [3,20]; [0,29]\}$

Table 3. Illustration of Algorithm B after reduction.

k	$\mathcal{U}_k$
0	$\{[0,0]\}$
1	$\{[1,2]; [0,2]\}$
2	$\{[3,11]; [1,8]; [2,8]; [0,11]\}$
3	$\{[6,29]; [3,20]; [4,20]; [1,23]; [5,23]; [2,20]; [0,29]\}$

4. FPTAS

4.1. Principle

In this section, we describe the different steps of the proposed FPTAS for solving the studied problem in a strongly polynomial time. Our method extends the one proposed by [5] by using the simplification of the state space generated by Algorithm B.

The FPTAS starts by applying a simple heuristic H to get a feasible schedule in $O(n\log(n))$ time. This heuristic consists in processing all the jobs on the second available machine according to the WSPT order. By comparing $\mathcal{F}_H(Q)$ to the optimal solution of the relaxation in which the first machine is continuously available, and according to the well-known lower bound (introduced in [1] by Eastman), we can deduce that $\mathcal{F}^*(Q) \geqslant \frac{\mathcal{F}_H(Q)}{2} + \frac{1}{4}\sum_{j=1}^{n} w_j p_j$. Thus, the following inequality holds:

$$\frac{\mathcal{F}_H(Q)}{\mathcal{F}^*(Q)} \leq 2 \tag{4.1}$$

The second step of the proposed FPTAS consists in simplifying Algorithm B with the aim of decreasing the running time. The general technique of *modifying the execution of an exact algorithm* to design FPTAS, was initially proposed by Ibarra and Kim [3] for solving the knapsack problem. It is noteworthy that during the last decades various scheduling problems have been studied by using such a technique. A sample of these papers includes Gens and Levner [2], Kacem [4–6], Kacem and Kellerer [7], Kacem and Mahjoub [8], Kacem and Haouari [9], Sahni [16], Kovalyov and Kubiak [12], Kellerer and Strusevich [10] and Woeginger [19].

Let us consider an arbitrary $\varepsilon > 0$ and let us define

$$q = \left\lceil \frac{4n}{\varepsilon} \right\rceil, \tag{4.2}$$

$$\delta_1 = \frac{\mathcal{F}_H(\mathcal{Q})}{q} \tag{4.3}$$

and

$$\delta_{2,k} = \frac{\min\left\{T_1, \sum_{j=1}^{k} p_j\right\}}{q} \quad \forall k = 1, 2, \ldots n. \tag{4.4}$$

The interval $[0, \mathcal{F}_H(\mathcal{Q})]$ is divided into q subintervals of equal-length δ_1. These subintervals are denoted $I_r^1 = [(r-1)\delta_1, r\delta_1]_{1\leq r\leq q}$. Moreover, we divide interval $[0, \min\{T_1, \sum_{j=1}^{k} p_j\}]$ into q subintervals $I_{s,k}^2 = [(s-1)\delta_{2,k}, s\delta_{2,k}]_{1\leq s\leq q}$ of equal length $\delta_{2,k}$ at every iteration k. As a consequence, Algorithm B'_ε will create a reduced set $\widetilde{\mathcal{U}}_k$ instead of sets $\mathcal{U}_k$ at every iteration k. This reduction step is illustrated in Figure 2. The resulted FPTAS can be summarized in the following algorithm:

Algorithm B'_ε

(i). $\widetilde{\mathcal{U}}_0 := \{[0,0]\}$.

(ii). For $k \in \{1, 2, 3, \ldots, n\}$,

Initialize $\widetilde{\mathcal{U}}_k := \varnothing$

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

1) Add state $\left[t, f + w_k(\sum_{j=1}^{k} p_j - t)\right]$ to $\widetilde{\mathcal{U}}_k$

2) Add state $[t + p_k, f + w_k\,(t + p_k)]$ to $\widetilde{\mathcal{U}}_k$ if $t + p_k \leq T_1$

Remove $\widetilde{\mathcal{U}}_{k-1}$

Let $[t, f]_{r,s}$ be the state in $\widetilde{\mathcal{U}}_k$, which verifies: $f \in I_r^1$ and $t \in I_{s,k}^2$ with the smallest possible value t (ties are broken by choosing the state of the smallest f).

Update $\widetilde{\mathcal{U}}_k := \left\{[t, f]_{r,s} \,|\, 1 \leq r \leq q, 1 \leq s \leq q\right\}$.

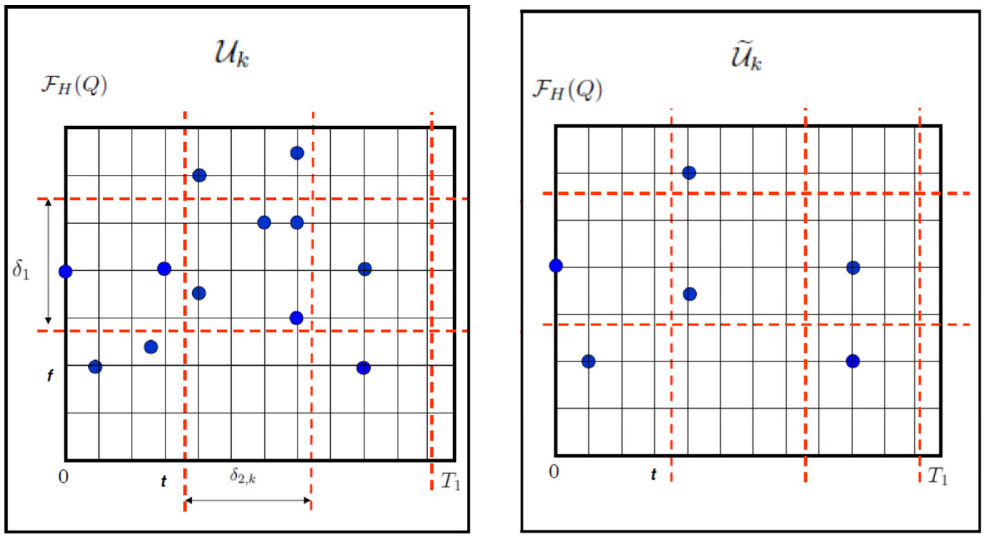

FIGURE 2. Illustration of the reduction step.

(iii). $\mathcal{F}_{B'_\varepsilon}(\mathcal{Q}) := \min\left\{f | [t,f] \in \widetilde{\mathcal{U}}_n\right\}$.

Note that the added step in Algorithm B'_ε with respect to Algorithm B is to reduce the state space and to ensure at the same time the feasibility of solutions by respecting the capacity constraint on the first machine.

A careful comparison of the two algorithms B and B'_ε leads to the following theorem.

Theorem 4.1. *Algorithm B'_ε is an FPTAS and it can be executed in $O(n^3/\varepsilon^2)$ time.*

4.2. Proof of Theorem 4.1

The proof is based on the two following lemmas.

Lemma 4.2. *Algorithm B'_ε can be executed in $O(n^3/\varepsilon^2)$ time.*

Proof. Heuristic H can be computed in $O(n\log(n))$ time. From the construction of Algorithm B'_ε, at each iteration k ($k \in \{1,2,\ldots,n\}$) we have $|U_k| \leq (((4n)/\varepsilon)+1)^2$. Hence, we can establish that $\sum_{k=1}^{n}|U_k| \leq n(((4n)/\varepsilon)+1)^2$. In conclusion, the algorithm can be implemented in $O(n\log(n)+n^3/\varepsilon^2)$ time. □

Lemma 4.3. *Let $\eta_k = \sum_{h=1}^{k}\delta_{2,h}$. For every state $[t,f] \in \widetilde{\mathcal{U}}_k$ ($k \in \{0,1,\ldots,n\}$), Algorithm B'_ε creates at least one state $[\widetilde{t},\widetilde{f}] \in \widetilde{\mathcal{U}}_k$ verifying the following conditions*:

$$\widetilde{t} \leq t,$$

and

$$\widetilde{f} \leq f + k\delta_1 + \sum_{z=1}^{k} w_z\eta_z.$$

Proof. We prove the result by induction. For $k=0, \widetilde{\mathcal{U}}_k = \mathcal{U}_k$ and the lemma is obvious. Let us assume the result holds until level $k-1$ and let us prove it for level k. Consider a state $[t,f] \in \mathcal{U}_k$. Two cases can be distinguished:

1st Case. $[t,f] = [t', f' + w_k(\sum_{j=1}^{k} p_j - t')]$ where $[t',f'] \in \mathcal{U}_{k-1}$.

State $[t', f']$ belongs to $\mathcal{U}_{k-1}$. This implies the existence of $[\widetilde{t}', \widetilde{f}'] \in \widetilde{\mathcal{U}}_{k-1}$ such that $\widetilde{t}' \leq t'$, and $\widetilde{f}' \leq f' + (k-1)\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z$. Hence, the construction of state $[\widetilde{t}', \widetilde{f}' + w_k(\sum_{j=1}^{k} p_j - \widetilde{t}')]$ is considered at iteration k. Such a state can be eliminated and replaced by an approximate one $[\theta_1, \theta_2]$ such that

$$\theta_1 \leq \widetilde{t}' \leq t' = t$$

and

$$\begin{aligned}
\theta_2 &\leq \widetilde{f}' + w_k \left(\sum_{j=1}^{k} p_j - \widetilde{t}' \right) + \delta_1 \\
&\leq f' + (k-1)\,\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z + w_k \left(\sum_{j=1}^{k} p_j - \widetilde{t}' \right) + \delta_1 \\
&\leq f' + k\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z + w_k \left(\sum_{j=1}^{k} p_j - \widetilde{t}' \right) \\
&= f' + k\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z + w_k \left(\sum_{j=1}^{k} p_j - \widetilde{t}' \right) + w_k\,(t' - t') \\
&= f' + k\delta_1 + w_k \left(\sum_{j=1}^{k} p_j - t' \right) + w_k\left(t' - \widetilde{t}'\right) + \sum_{z=1}^{k-1} w_z\eta_z \\
&= f + k\delta_1 + w_k\left(t' - \widetilde{t}'\right) + \sum_{z=1}^{k-1} w_z\eta_z
\end{aligned}$$

It is easy to see by induction that $t' - \widetilde{t}' \leq \sum_{h=1}^{k} \delta_{2,h} = \eta_k$. Then, $\theta_2 \leq f + k\delta_1 + \sum_{z=1}^{k} w_z\eta_z$. Hence, $[\theta_1, \theta_2]$ is an approximate state verifying the two conditions.

2nd Case. $[t, f] = [t' + p_k, f' + w_k\,(t' + p_k)]$ where $[t', f'] \in \mathcal{U}_{k-1}$.
State $[t', f']$ belongs to $\mathcal{U}_{k-1}$. This implies the existence of $[\widetilde{t}', \widetilde{f}'] \in \widetilde{\mathcal{U}}_{k-1}$ such that $\widetilde{t}' \leq t'$ and $\widetilde{f}' \leq f' + (k-1)\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z$. Hence, the construction of state $[\widetilde{t}' + p_k, \widetilde{f}' + w_k(\widetilde{t}' + p_k)]$ is considered at iteration k. Such a state can be eliminated and replaced by an approximate one (θ_1', θ_2') such that

$$\theta_1' \leq \widetilde{t}' + p_k \leq t' + p_k = t \tag{4.5}$$

and

$$\begin{aligned}
\theta_2' &\leq \widetilde{f}' + w_k\left(\widetilde{t}' + p_k\right) + \delta_1 \\
&\leq f' + (k-1)\,\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z + w_k\left(\widetilde{t}' + p_k\right) + \delta_1 \\
&\leq f' + (k-1)\,\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z + w_k\,(t' + p_k) + \delta_1 \\
&= f' + w_k\,(t' + p_k) + k\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z \\
&= f + k\delta_1 + \sum_{z=1}^{k-1} w_z\eta_z < f + k\delta_1 + \sum_{z=1}^{k} w_z\eta_z.
\end{aligned}$$

Hence, $[\theta'_1, \theta'_2]$ is an approximate state verifying the two conditions.
In the two cases, the lemma holds. □

The complexity of B'_ε is strongly polynomial (see Lem. 4.2). Let $[t^*, f^*]$ be a state from $\mathcal{U}_n$ associated to an optimal solution. Thus, by recalling Lemma 4.3, we can deduce that there exists in $\widetilde{\mathcal{U}}_n$ a state$[\widetilde{t}, \widetilde{f}]$, which respects the following condition:

$$\widetilde{f} \leq f^* + n\delta_1 + \sum_{z=1}^{n} w_z \eta_z \tag{4.6}$$

Thus, the following inequality can be established:

$$\begin{aligned}
\widetilde{f} &= \mathcal{F}^*(\mathcal{Q}) + n\frac{\mathcal{F}_H(\mathcal{Q})}{q} + \sum_{z=1}^{n} w_z \sum_{h=1}^{z} \delta_{2,h} \\
&\leq \mathcal{F}^*(\mathcal{Q}) + n\frac{\mathcal{F}_H(\mathcal{Q})}{\lceil \frac{4n}{\varepsilon} \rceil} + \sum_{z=1}^{n} w_z \sum_{h=1}^{z} \frac{\min\left\{T_1, \sum_{j=1}^{h} p_j\right\}}{\lceil \frac{4n}{\varepsilon} \rceil} \\
&< \mathcal{F}^*(\mathcal{Q}) + \varepsilon\frac{\mathcal{F}_H(\mathcal{Q})}{4} + \frac{\varepsilon}{4n} \sum_{z=1}^{n} w_z \sum_{h=1}^{z} \sum_{j=1}^{h} p_j \\
&< \mathcal{F}^*(\mathcal{Q}) + \varepsilon\frac{\mathcal{F}^*(\mathcal{Q})}{2} + \frac{\varepsilon}{4n} w_1(p_1) + \frac{\varepsilon}{4n} w_2(p_1 + (p_1 + p_2)) \\
&\quad + \frac{\varepsilon}{4n} w_3(p_1 + (p_1 + p_2) + (p_1 + p_2 + p_3)) + \ldots \\
&\quad + \frac{\varepsilon}{4n} w_n(p_1 + (p_1 + p_2) + (p_1 + p_2 + p_3) + \ldots \\
&\quad + (p_1 + p_2 + \ldots + p_n))
\end{aligned}$$

By using the definition of the WSPT order, we deduce the following relation:

$$\begin{aligned}
\widetilde{f} \leq \mathcal{F}^*(\mathcal{Q}) + \varepsilon\frac{\mathcal{F}^*(\mathcal{Q})}{2} + \frac{\varepsilon}{4n} w_1(p_1) + \frac{\varepsilon}{4n} w_2(2p_1 + p_2) + \frac{\varepsilon}{4n} w_3(3p_1 + 2p_2 + 1.p_3) + \ldots \\
+ \frac{\varepsilon}{4n} w_n(n.p_1 + (n-1).p_2 + (n-2).p_3 + \ldots + 1.p_n)
\end{aligned}$$

Thus, it can be established that:

$$\begin{aligned}
\widetilde{f} \leq \mathcal{F}^*(\mathcal{Q}) + \varepsilon\frac{\mathcal{F}^*(\mathcal{Q})}{2} + \frac{\varepsilon}{4n}(n.\left(\sum_{z=1}^{n} w_z\right)p_1 + (n-1).\left(\sum_{z=2}^{n} w_z\right)p_2 \\
+ (n-2).\left(\sum_{z=3}^{n} w_z\right)p_3 + \ldots + 1.w_n.p_n) < \mathcal{F}^*(\mathcal{Q}) + \varepsilon\frac{\mathcal{F}^*(\mathcal{Q})}{2} \\
+ \frac{\varepsilon}{4}\left(\left(\sum_{z=1}^{n} w_z\right)p_1 + \left(\sum_{z=2}^{n} w_z\right)p_2 + \left(\sum_{z=3}^{n} w_z\right)p_3 + \ldots + w_n.p_n\right)
\end{aligned}$$

Since $\mathcal{F}_H(\mathcal{Q}) = (\sum_{z=1}^{n} w_z)p_1 + (\sum_{z=2}^{n} w_z)p_2 + \ldots + w_n.p_n$, therefore,

$$\widetilde{f} \leq (1+\varepsilon)\mathcal{F}^*(\mathcal{Q}).$$

Finally, we know that $\mathcal{F}_{B'_\varepsilon}(\mathcal{Q}) \leq \widetilde{f}$, which leads to the fact that B'_ε generates $(1+\varepsilon)-$schedule in a strongly polynomial time, which demonstrates the result.

TABLE 4. Experimental results for the first series.

n	Time(DP)	Time(FPTAS)	F(DP)	F(FPTAS)	Speed Ratio	Gap in %
20	0,0076	0,004	70 584,7	70 649,6	1,9	0,0919
30	0,0239	0,017	158 445,4	158 563,5	1,41	0,0745
40	0,0415	0,0338	29 1474,6	291 678,9	1,23	0,0701
50	0,071	0,061	472 205,6	472 515,8	1,16	0,0657
60	0,1006	0,0935	629 800,9	630 121,8	1,08	0,051
70	0,1489	0,144	845 684,7	846 026,5	1,03	0,0404
80	0,1966	0,198	1 141 700,8	1 142 184,6	0,99	0,0424
90	0,2367	0,2394	1 398 167,7	1 398 684	0,99	0,0369
100	0,2993	0,3048	1 834 911,3	1 835 604,8	0,98	0,0378
110	0,3473	0,3597	2 111 960,5	2 11 2703,4	0,97	0,0352
120	0,4433	0,4716	2 710 292,8	2 711 164,8	0,94	0,0322
130	0,5096	0,5446	2 998 650,4	2 999 510,2	0,94	0,0287
140	0,5946	0,642	3 496 842,5	34 97 851,8	0,93	0,0289
150	0,7043	0,7547	4 070 917,6	4 071 961,5	0,93	0,0256
160	0,7529	0,8227	4302416,9	4 303 540,5	0,92	0,0261
170	0,9084	1,0038	5 273 864	5 275 155,8	0,9	0,0245
180	1,0008	1,1176	5 631 711,3	5 632 977,5	0,9	0,0225
190	1,1156	1,2398	6 373 485,6	6 374 882,3	0,9	0,0219
200	1,2636	1,4154	7 298 691,4	7 300 156,6	0,89	0,0201
210	1,3368	1,5215	7 308 466,7	7309821,5	0,88	0,0185
220	1,4774	1,7005	8 216 537,1	8 217 961,3	0,87	0,0173
230	1,6061	1,8619	9 020 383,3	9 022 003	0,86	0,018
240	1,8074	2,0775	10 652 215,3	10 654 080,8	0,87	0,0175
250	1,8937	2,2056	10 847 259,3	10 848 888,5	0,86	0,015
260	2,1408	2,505	12 173 502,2	12 175 779,6	0,85	0,0187
270	2,1557	2,4978	12 307 261,2	12 309 162,3	0,86	0,0154
280	2,373	2,833	13 337 482,2	13 339 352,6	0,84	0,014
290	2,6094	3,0881	14 243 862,5	14 246 333,7	0,84	0,0173
300	2,8445	3,4073	15 834 329,2	15 836 925	0,83	0,0164

5. NUMERICAL EXPERIMENTS

In this section, we provide the computational results used to evaluate the performance of the different methods presented above. The tests were carried out on a DELL precision PC, with a processor Intel Core i7- 2820QM CPU @2.30 GHz, in the Windows Seven environment. The following paragraphs describe our data-generation methods, the results obtained and our analysis of these experiments.

5.1. First set of instances

For the first data generation, the value of ε was fixed to 1/3. For a fixed number of jobs (n), ten instances were randomly generated, such that $p_i \in [1, 600]$ and $w_i \in [1, 10]$ according to a discrete uniform distribution. For each instance, the value of T_1 was equal to $n * \varepsilon * 300$ (the value 300 in the last formula is an estimation of the average processing time). Table 4 summarizes the results obtained for this set of instances. It contains the following average data: the number of jobs (n) in the first column, the computation time spent by the DP method in the second column, the computation time spent by the FPTAS in the third column, the objective function F provided by the DP in the fourth column, the objective function provided by the FPTAS in the fifth

TABLE 5. Experimental results for the second series.

n	Time(DP)	Time(FPTAS)	F(DP)	F(FPTAS)	Speed Ratio	Gap in %
20	0,2841	0,0061	4 176 886,1	4 180 397,7	46,57	0,08
30	1,5962	0,0295	9 878 249,1	9 889 467,6	54,11	0,11
40	3,4158	0,0947	15 696 162	15 709 708,9	36,07	0,09
50	5,7226	0,1836	25 056 162	25 078 680,8	31,17	0,09
60	8,308	0,3769	33 648 607,8	33 673 812,6	22,04	0,07
70	11,3333	0,6282	44 129 663,6	44 163 547,1	18,04	0,08
80	14,4667	0,9033	58 842 972,4	58 881 010,4	16,02	0,06
90	19,7863	1,6256	76 604 596,3	76 648 250,5	12,17	0,06
100	24,3176	2,2916	92 656 766,5	92 703 310,8	10,61	0,05
110	27,8984	3,1483	10 4613 739,4	104 668 570,5	8,86	0,05
120	34,4835	3,9207	140 582 827,3	140 653 581,5	8,8	0,05
130	39,239	5,4096	15 366 8598,1	153 739 534,2	7,25	0,05
140	44,1368	7,367	164 976 826,7	165 046 577,2	5,99	0,04
150	49,71	8,9107	191 247 017,2	191 329 646,5	5,58	0,04
160	58,9598	10,8687	235 203 359,7	235 289 619,7	5,42	0,04
170	69,9335	14,0521	276 033 634	276 132 604,1	4,98	0,04
180	77,1604	17,1239	314 180 202,9	314 299 012,4	4,51	0,04
190	83,9298	20,4455	328 874 514,7	328 980 712	4,11	0,03
200	88,0746	24,2299	335 153 282,1	335 259 939,9	3,63	0,03
210	101,0347	29,0111	384 124 791,3	384 253 595,5	3,48	0,03
220	116,4357	36,6883	436 599 988	436 736 941,2	3,17	0,03
230	123,1874	39,7522	475 430 330,2	475 568 636	3,1	0,03
240	132,1277	45,9577	497 762 270,9	497 885 687,6	2,87	0,02
250	147,4051	54,0778	567 895 532,9	568 033 449,2	2,73	0,02
260	156,2205	57,9278	603 780 360,3	603 944 179,8	2,7	0,03
270	170,8563	70,7101	644 494 744,9	644 657 787,6	2,42	0,03
280	183,4643	78,844	709 629 799,1	709 798 088,3	2,33	0,02
290	195,8565	82,4432	774 105 555,4	774 295 067	2,38	0,02
300	226,8268	107,7673	825 770 473,7	825 959 046,5	2,1	0,02

column, the speed ratio by comparing the DP method and the FPTAS in the sixth column and the relative gap of the same methods in the last seventh column.

For this first set of instances it is easy to observe the effectiveness of the DP method compared to the FPTAS. Moreover, we can notice globally that the quality is more or less equivalent between the two methods. This gives the advantage to the DP method thanks to its exact solution. Finally, we can notice that when the number of jobs increases, the FPTAS yields results closer to the optimal solutions.

5.2. Second set of instances

For the second data generation, the value of ε was fixed to $1/3$. For a fixed number of jobs (n), ten instances were randomly generated, such that $p_i \in [1, 300\,000]$ and $w_i \in [1, 10]$ according to a discrete uniform distribution. For each instance, the value of T_1 was equal to $n * \varepsilon * 1500$. Table 5 summarizes the results obtained for this set of instances (with the same parameters as in Tab. 4).

In this set of instances we clearly did the choice of the large processing times. According to the obtained results, we can clearly conclude that the FPTAS is very efficient for this set of instances. Here, we have evidence

that the approximate solutions yielded by the FPTAS are very close to the optimum. Moreover, the average computation time is more interesting than for the DP method. Finally, as for the first set of instances, we can notice that the FPTAS has an excellent average solution quality and when the number of jobs increases, such a quality increases.

To conclude, the two sets of instances show clearly the interest of the FPTAS and its complementarity to the DP method for a part of possible instances.

6. Conclusion

In this paper, we study the total weighted completion time minimization for the two-parallel capacitated machines scheduling problem. In this problem, one of the machines can process jobs until a certain time T_1 after which it is no longer available. The other machine is continuously available for processing jobs at every time. In this paper, we prove the existence of a strongly FPTAS. Such an FPTAS is based on the simplification of a dynamic programming algorithm. The result represents an extension of a previous FPTAS proposed to solve the unweighted version by [5]. Moreover, we presented a set of numerical experiments and we discussed the results. One of the important results is the fact that the FPTAS is an effective alternative for instances where the processing times and the availability period length are large, which leads to limit the effectiveness of the dynamic programming algorithm in this case. It is worthy to note that the dynamic programming algorithm remains an efficient method for solving instances with moderate values of processing times and availability period length.

As a possible extension of this work, we hope to improve our results for the studied problem (*i.e.*, to decrease the time complexity). The elaboration of more effective approximation algorithms by combining the proposed technique with other methods (in particular the one proposed by [13]) seems to be also an interesting perspective.

Acknowledgements. The authors thank Dr. Christian Minich for his help on the experimental study presented in this paper. They also thank Prof. Eugene Levner for several discussions on the approximation schemes' design for combinatorial optimization problems. This work has been supported by the Université de Lorraine (France) and the Universität des Saarlandes (Germany) during the year 2015. The short version of this work has been presented at the 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, Troyes, 28 to 30 June 2016, France.

References

[1] W.L. Eastman, S. Even and I.M. Issacs, Bounds for the optimal scheduling of n jobs on m processors. *Manag. Sci.* **11** (1964) 268–279.

[2] G.V. Gens and E.V. Levner, Fast approximation algorithms for job sequencing with deadlines. *Discrete Appl. Math.* **3** (1981) 313–318.

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

[4] I. Kacem, Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. *J. Combin. Optim.*. **17** (2009) 117–133.

[5] I. Kacem, Y. Lanuel and M. Sahnoune, Strongly Fully Polynomial Time Approximation Scheme for the two-parallel capacitated machines scheduling problem. *Int. J. Plann. Scheduling* **1** (2011) 32–41.

[6] I. Kacem, Fully Polynomial-Time Approximation Scheme for the Weighted Total Tardiness Minimization with a Common Due Date. *Discrete Appl. Math.* **158** (2010) 1035–1040.

[7] I. Kacem and H. Kellerer, Fast approximation algorithms to minimize a special weighted flow-time criterion on a single achine with a non-availability interval and release dates. *J. Sched.* **14** (2011) 257–265.

[8] I. Kacem and R.A. Mahjoub, Fully Polynomial Time Approximation Scheme for the Weighted Flow-time Minimization on a Single Machine with a Fixed Non-Availability Interval. *Comput. Ind. Eng.* **56** (2009) 1708–1712.

[9] I. Kacem and M. Haouari, Approximation algorithms for single machine scheduling with one unavailability period. *4OR: A Quarterly J. Oper. Res.* **7** (2009) 79–92.

[10] H. Kellerer and V.A. Strusevich, A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. *Theor. Comput. Sci.* **369** (2006) 230–238.

[11] H. Kellerer and V.A. Strusevich, Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica* **57** (2010) 769–795.

[12] M.Y. Kovalyov and W. Kubiak, A fully polynomial approximation scheme for weighted earliness-tardiness problem. *Oper. Res.* **47** (1999) 757–761.

[13] M.Y. Kovalyov and W. Kubiak, Fully polynomial approximation schemes for decomposable partition problems. Working Paper 98-15 of the Faculty of Business Administration, Memorial University of Newfoundland. *Presentation in Operations Research Proceedings 1999, Selected papers of the Sympos. Oper. Res. (SOR 99), Magdeburg* (1999) 397–401.

[14] C.Y. Lee and S.D. Liman, Capacitated two-parallel machines sceduling to minimize sum of job completion times. *Discrete Appl. Math.* **41** (1993) 211–222.

[15] C.-J. Liao, C-W. Chao and C.-H. Lin, Minimizing the sum of job completion times on capacitated two-parallel machines. *Eur. J. Oper. Res.* **197** (2009) 475–481.

[16] S. Sahni, Algorithms for scheduling independent tasks. *J. ACM* **23** (1976) 116–127.

[17] G. Schmidt, Scheduling with limited machine availability. *Eur. J. Oper. Res.* **121** (2000) 1–15.

[18] W.E. Smith, Various optimizers for single stage production. *Nav. Res. Log. Quarterly* **3** (1956) 59–66.

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

[20] Z. Xu, A strongly polynomial FPTAS for the symmetric quadratic knapsack problem. *Eur. J. Oper. Res.* **218** (2012) 377–381.