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

Imed Kacem¹, Myriam Sahnoune¹ and Günter Schmidt²

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 \(T\) 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 \(T\) 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

© EDP Sciences, ROADEF, SMAI 2017
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 $2\cdot h_{1,\infty}|| w_j C_j$, 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 ($2\cdot h_{1,\infty}|| C_j$). The unconstrained version of the problem ($2||w_j C_j$) 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 $O(n \log \max \{ p_j, w_j \})$. It is worthy to note that from the dynamic programming described in Section 3, we can establish that Woeginger’s approach can lead to the existence of an FPTAS of $O(|I|^3 n^3 / \varepsilon^2)$ 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:

$$p_1/w_1 \leq p_2/w_2 \leq \ldots \leq p_n/w_n$$  \hspace{1cm} (2.1)
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: $p_1 = 1$, $w_1 = 2$, $p_2 = 2$, $w_2 = 3$, $p_3 = 3$, $w_3 = 3$, $p_4 = 4$, $w_4 = 4$, $p_5 = 5$, $w_5 = 3$, $p_6 = 3$, $w_6 = 3$, $p_7 = 5$, $w_7 = 2$ and $T_1 = 10$.

In the remainder of the paper, we denote by $Q$ the studied problem, by $F^* (Q)$ the minimal sum of the weighted completion times for problem $Q$ and by $F_S (Q)$ 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 $\rho$-approximation algorithm for a problem of minimizing an objective function $\varphi$ is an algorithm such that for every instance $\pi$ of the problem it gives a solution $S_\pi$ verifying $\varphi (S_\pi) / \varphi (OPT_\pi) \leq \rho$ where $OPT_\pi$ is the optimal solution of $\pi$. Moreover, $\rho$ 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 $(Q)$ 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 $\sigma_1$ 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 $\sigma_1$ is also a $(1+\varepsilon)$-approximation for the original problem $Q$. If $B'_2 \leq T_1$, then by swapping the two machines, a $(1+\varepsilon)$-approximation schedule $\sigma'_1$ is obtained such that $\sigma'_1$ is also feasible for the original problem $Q$. Thus, the proposition is established. □

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

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

(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. 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 \( \mathcal{U}_k := \emptyset \)
- For each state \([t, f]\) in \( \mathcal{U}_{k-1} \):
  1) Add state \([t, f + w_k (\sum_{j=1}^{k} p_j - t)]\) 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). \( F^* (\mathcal{Q}) := \min \{ f | [t, f] \in \mathcal{U}_n \} \).

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 \( O(nT_1Z) \) where \( Z \) 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.**

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \mathcal{U}_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{[0, 0]}</td>
</tr>
<tr>
<td>1</td>
<td>{[1, 2]; [0, 2]}</td>
</tr>
<tr>
<td>2</td>
<td>{[3, 11]; [1, 8]; [2, 8]; [0, 11]}</td>
</tr>
<tr>
<td>3</td>
<td>{[6, 29]; [3, 20]; [4, 20]; [1, 23]; [5, 23]; [2, 20]; [3, 20]; [0, 29]}</td>
</tr>
</tbody>
</table>

**Table 3. Illustration of Algorithm B after reduction.**

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \mathcal{U}_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{[0, 0]}</td>
</tr>
<tr>
<td>1</td>
<td>{[1, 2]; [0, 2]}</td>
</tr>
<tr>
<td>2</td>
<td>{[3, 11]; [1, 8]; [2, 8]; [0, 11]}</td>
</tr>
<tr>
<td>3</td>
<td>{[6, 29]; [3, 20]; [4, 20]; [1, 23]; [5, 23]; [2, 20]; [0, 29]}</td>
</tr>
</tbody>
</table>
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 $F_H(Q)$ 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 $F^*(Q) \geq \frac{F_H(Q)}{2} + \frac{1}{2} \sum_{j=1}^{n} w_j p_j$. Thus, the following inequality holds:

\[
\frac{F_H(Q)}{F^*(Q)} \leq 2
\]

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

\[
q = \left\lceil \frac{4n}{\varepsilon} \right\rceil,
\]

\[
\delta_1 = \frac{F_H(Q)}{q}
\]

and

\[
\delta_{2,k} = \min \left\{ \frac{T_1}{q}, \sum_{j=1}^{k} p_j \right\} \forall k = 1, 2, \ldots, n.
\]

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

**Algorithm $B'_{\varepsilon}$**

1. $\tilde{U}_0 := \{[0,0]\}$.
2. For $k \in \{1, 2, 3, \ldots, n\}$,
   - Initialize $\tilde{U}_k := \emptyset$
   - For every state $[t, f]$ in $\tilde{U}_{k-1}$:
     1) Add state $[t, f + w_k(\sum_{j=1}^{k} p_j - t)]$ to $\tilde{U}_k$
     2) Add state $[t + p_k, f + w_k(t + p_k)]$ to $\tilde{U}_k$ if $t + p_k \leq T_1$
   - Remove $\tilde{U}_{k-1}$
   - Let $[t, f]_{r,s}$ be the state in $\tilde{U}_k$, which verifies: $f \in I^1_r$ and $t \in I^2_{s,k}$ with the smallest possible value $t$ (ties are broken by choosing the state of the smallest $f$).
   - Update $\tilde{U}_k := \{[t, f]_{r,s} | 1 \leq r \leq q, 1 \leq s \leq q \}$. 

\[4.1\]
(iii). $F_{B'\varepsilon}(Q) := \min \{ f | [t,f] \in \tilde{U}_n \}$.

Note that the added step in Algorithm $B'\varepsilon$ 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 $B'\varepsilon$ leads to the following theorem.

**Theorem 4.1.** Algorithm $B'\varepsilon$ 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 $B'\varepsilon$ 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 $B'\varepsilon$, 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 $[t,f] \in \tilde{U}_k (k \in \{0,1,\ldots,n\})$, Algorithm $B'\varepsilon$ creates at least one state $[\tilde{t},\tilde{f}] \in \tilde{U}_k$ verifying the following conditions:

\[
\tilde{t} \leq t,
\]

and

\[
\tilde{f} \leq f + k\delta_1 + \sum_{z=1}^k w_z \eta_z.
\]

**Proof.** We prove the result by induction. For $k = 0$, $\tilde{U}_k = U_k$ 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 $[t,f] \in U_k$. Two cases can be distinguished:

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

---

**Figure 2.** Illustration of the reduction step.

- $F_{B\varepsilon}(Q)$ and $\tilde{U}_k$ indicate the state space and the algorithm's output, respectively.
- $t$ and $T_1$ represent time intervals.
- $\delta_1$ and $\delta_{2,k}$ denote specific intervals for comparison.

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

\[
\theta_1 \leq \tilde{t'} \leq t' = t
\]

and

\[
\theta_2 \leq \tilde{f'} + w_k \left( \sum_{j=1}^{k} p_j - \tilde{t'} \right) + \delta_1
\]

\[
\leq f' + (k - 1) \delta_1 + \sum_{z=1}^{k-1} w_z \eta_z + w_k \left( \sum_{j=1}^{k} p_j - \tilde{t'} \right) + \delta_1
\]

\[
\leq f' + k\delta_1 + \sum_{z=1}^{k-1} w_z \eta_z + w_k \left( \sum_{j=1}^{k} p_j - \tilde{t'} \right) + w_k (t' - \tilde{t'} + \delta_1)
\]

\[
= f' + k\delta_1 + w_k \left( \sum_{j=1}^{k} p_j - \tilde{t'} \right) + w_k (t' - \tilde{t'}) + \sum_{z=1}^{k-1} w_z \eta_z
\]

It is easy to see by induction that \(t' - \tilde{t'} \leq \sum_{h=1}^{k} \delta_{2,h} = \eta_k\). 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.

2\textsuperscript{nd} Case. \([t, f] = [t' + p_k, f' + w_k (t' + p_k)]\) where \([t', f'] \in U_{k-1}\).

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

\[
\theta_1' \leq \tilde{t'} + p_k \leq t' + p_k = t
\]

(4.5)

and

\[
\theta_2' \leq \tilde{f'} + w_k (\tilde{t'} + p_k) + \delta_1
\]

\[
\leq f' + (k - 1) \delta_1 + \sum_{z=1}^{k-1} w_z \eta_z + w_k (\tilde{t'} + p_k) + \delta_1
\]

\[
\leq f' + (k - 1) \delta_1 + \sum_{z=1}^{k-1} w_z \eta_z + w_k (t' + p_k) + \delta_1
\]

\[
= f' + w_k (t' + p_k) + k\delta_1 + \sum_{z=1}^{k-1} w_z \eta_z
\]

\[
= f + k\delta_1 + \sum_{z=1}^{k-1} w_z \eta_z < f + k\delta_1 + \sum_{z=1}^{k} w_z \eta_z.
\]
Hence, \([\theta'_1, \theta'_2]\) is an approximate state verifying the two conditions. In the two cases, the lemma holds.

The complexity of \(B'_\varepsilon\) 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 \(\mathcal{U}_n\) a state\([\tilde{t}, \tilde{f}]\), which respects the following condition:

\[
\tilde{f} \leq f^* + n\delta_1 + \sum_{z=1}^n w_z \eta_z
\] (4.6)

Thus, the following inequality can be established:

\[
\tilde{f} = \mathcal{F}^*(\mathcal{Q}) + n \frac{\mathcal{F}_H(\mathcal{Q})}{q} + \sum_{z=1}^n w_z \sum_{h=1}^z \delta_{2,h}
\]

\[
\leq \mathcal{F}^*(\mathcal{Q}) + n \frac{\mathcal{F}_H(\mathcal{Q})}{\left\lfloor \frac{4n}{\varepsilon} \right\rfloor} + \sum_{z=1}^n w_z \sum_{h=1}^z \min\left\{ T_1, \sum_{j=1}^h p_j \right\}
\]

\[
< \mathcal{F}^*(\mathcal{Q}) + \varepsilon \frac{\mathcal{F}_H(\mathcal{Q})}{4} + \varepsilon \frac{\mathcal{F}^*(\mathcal{Q})}{4} + \varepsilon \frac{\mathcal{F}^*(\mathcal{Q})}{4} + \varepsilon \sum_{z=1}^n w_z \sum_{h=1}^z \sum_{j=1}^h p_j
\]

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

\[
\tilde{f} \leq \mathcal{F}^*(\mathcal{Q}) + \varepsilon \frac{\mathcal{F}^*(\mathcal{Q})}{2} + \varepsilon \sum_{z=1}^n w_z p_1 + \varepsilon \sum_{z=1}^n w_z (2p_1 + p_2) + \varepsilon \sum_{z=1}^n w_z (3p_1 + 2p_2 + 1.3) + \ldots
\]

\[
+ \varepsilon \sum_{z=1}^n w_z (n.p_1 + (n-1).p_2 + (n-2).p_3 + \ldots + 1.n)
\]

Thus, it can be established that:

\[
\tilde{f} \leq \mathcal{F}^*(\mathcal{Q}) + \varepsilon \frac{\mathcal{F}^*(\mathcal{Q})}{2} + \varepsilon \sum_{z=1}^n w_z p_1 + \varepsilon \sum_{z=1}^n w_z (p_1 + (n-1).p_2 + (n-2).p_3 + \ldots + 1.n)
\]

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,

\[
\tilde{f} \leq (1 + \varepsilon) \mathcal{F}^*(\mathcal{Q})
\]

Finally, we know that \(\mathcal{F}_B(\mathcal{Q}) \leq \tilde{f}\), which leads to the fact that \(B'_\varepsilon\) generates \((1 + \varepsilon)\)-schedule in a strongly polynomial time, which demonstrates the result.
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 $\varepsilon$ 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 $n \times \varepsilon \times 300$ (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 column, the speed ratio in the sixth column and the gap in % in the seventh column.

<table>
<thead>
<tr>
<th>$n$</th>
<th>Time(DP)</th>
<th>Time(FPTAS)</th>
<th>F(DP)</th>
<th>F(FPTAS)</th>
<th>Speed Ratio</th>
<th>Gap in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0076</td>
<td>0.004</td>
<td>70 584.7</td>
<td>70 649.6</td>
<td>1.9</td>
<td>0.0919</td>
</tr>
<tr>
<td>30</td>
<td>0.0239</td>
<td>0.017</td>
<td>158 445.4</td>
<td>158 563.5</td>
<td>1.41</td>
<td>0.0745</td>
</tr>
<tr>
<td>40</td>
<td>0.0415</td>
<td>0.0338</td>
<td>29 147.4</td>
<td>29 678.9</td>
<td>1.23</td>
<td>0.0701</td>
</tr>
<tr>
<td>50</td>
<td>0.071</td>
<td>0.061</td>
<td>472 205.6</td>
<td>472 515.8</td>
<td>1.16</td>
<td>0.0657</td>
</tr>
<tr>
<td>60</td>
<td>0.1006</td>
<td>0.0935</td>
<td>629 800.9</td>
<td>630 121.8</td>
<td>1.08</td>
<td>0.051</td>
</tr>
<tr>
<td>70</td>
<td>0.1489</td>
<td>0.144</td>
<td>845 684.7</td>
<td>846 026.5</td>
<td>1.03</td>
<td>0.0404</td>
</tr>
<tr>
<td>80</td>
<td>0.1966</td>
<td>0.198</td>
<td>1141 700.8</td>
<td>1142 184.6</td>
<td>0.99</td>
<td>0.0424</td>
</tr>
<tr>
<td>90</td>
<td>0.2367</td>
<td>0.2394</td>
<td>1398 167.7</td>
<td>1398 684</td>
<td>0.99</td>
<td>0.0369</td>
</tr>
<tr>
<td>100</td>
<td>0.2993</td>
<td>0.3048</td>
<td>1834 911.3</td>
<td>1835 604.8</td>
<td>0.98</td>
<td>0.0378</td>
</tr>
<tr>
<td>110</td>
<td>0.3473</td>
<td>0.3597</td>
<td>2111 960.5</td>
<td>2112 703.4</td>
<td>0.97</td>
<td>0.0352</td>
</tr>
<tr>
<td>120</td>
<td>0.4433</td>
<td>0.4716</td>
<td>2710 292.8</td>
<td>2711 164.8</td>
<td>0.94</td>
<td>0.0322</td>
</tr>
<tr>
<td>130</td>
<td>0.5096</td>
<td>0.5446</td>
<td>2998 650.4</td>
<td>2999 510.2</td>
<td>0.94</td>
<td>0.0287</td>
</tr>
<tr>
<td>140</td>
<td>0.5946</td>
<td>0.642</td>
<td>3496 842.5</td>
<td>3497 851.8</td>
<td>0.93</td>
<td>0.0289</td>
</tr>
<tr>
<td>150</td>
<td>0.7043</td>
<td>0.7547</td>
<td>4070 917.6</td>
<td>4071 961.5</td>
<td>0.93</td>
<td>0.0256</td>
</tr>
<tr>
<td>160</td>
<td>0.7529</td>
<td>0.8227</td>
<td>4302 416.9</td>
<td>4303 540.5</td>
<td>0.92</td>
<td>0.0261</td>
</tr>
<tr>
<td>170</td>
<td>0.9084</td>
<td>1.0038</td>
<td>5273 864</td>
<td>5275 155.8</td>
<td>0.9</td>
<td>0.0245</td>
</tr>
<tr>
<td>180</td>
<td>1.0008</td>
<td>1.1176</td>
<td>5631 711.3</td>
<td>5632 977.5</td>
<td>0.9</td>
<td>0.0225</td>
</tr>
<tr>
<td>190</td>
<td>1.1156</td>
<td>1.2398</td>
<td>6373 485.6</td>
<td>6374 882.3</td>
<td>0.9</td>
<td>0.0219</td>
</tr>
<tr>
<td>200</td>
<td>1.2636</td>
<td>1.4154</td>
<td>7288 691.4</td>
<td>7300 156.6</td>
<td>0.89</td>
<td>0.0201</td>
</tr>
<tr>
<td>210</td>
<td>1.3368</td>
<td>1.5215</td>
<td>7308 466.7</td>
<td>7309 821.5</td>
<td>0.88</td>
<td>0.0185</td>
</tr>
<tr>
<td>220</td>
<td>1.4774</td>
<td>1.7005</td>
<td>8216 537.1</td>
<td>8217 961.3</td>
<td>0.87</td>
<td>0.0173</td>
</tr>
<tr>
<td>230</td>
<td>1.6061</td>
<td>1.8619</td>
<td>9020 383.3</td>
<td>9022 003</td>
<td>0.86</td>
<td>0.018</td>
</tr>
<tr>
<td>240</td>
<td>1.8074</td>
<td>2.0775</td>
<td>10652 215.3</td>
<td>10654 080.8</td>
<td>0.87</td>
<td>0.0175</td>
</tr>
<tr>
<td>250</td>
<td>1.8937</td>
<td>2.2056</td>
<td>10847 259.3</td>
<td>10848 888.5</td>
<td>0.86</td>
<td>0.015</td>
</tr>
<tr>
<td>260</td>
<td>2.1408</td>
<td>2.505</td>
<td>12173 502.2</td>
<td>12175 779.6</td>
<td>0.85</td>
<td>0.0187</td>
</tr>
<tr>
<td>270</td>
<td>2.1557</td>
<td>2.4978</td>
<td>12307 261.2</td>
<td>12309 162.3</td>
<td>0.86</td>
<td>0.0154</td>
</tr>
<tr>
<td>280</td>
<td>2.373</td>
<td>2.833</td>
<td>13337 482.2</td>
<td>13339 352.6</td>
<td>0.84</td>
<td>0.014</td>
</tr>
<tr>
<td>290</td>
<td>2.6094</td>
<td>3.0881</td>
<td>14243 862.5</td>
<td>14246 333.7</td>
<td>0.84</td>
<td>0.0173</td>
</tr>
<tr>
<td>300</td>
<td>2.8445</td>
<td>3.4073</td>
<td>15834 329.2</td>
<td>15836 925</td>
<td>0.83</td>
<td>0.0164</td>
</tr>
</tbody>
</table>
Table 5. Experimental results for the second series.

<table>
<thead>
<tr>
<th>n</th>
<th>Time(DP)</th>
<th>Time(FPTAS)</th>
<th>F(DP)</th>
<th>F(FPTAS)</th>
<th>Speed Ratio</th>
<th>Gap in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.2841</td>
<td>0.0061</td>
<td>4176.8861</td>
<td>4180.3977</td>
<td>46.57</td>
<td>0.08</td>
</tr>
<tr>
<td>30</td>
<td>1.5962</td>
<td>0.0295</td>
<td>9878.2491</td>
<td>9889.4676</td>
<td>54.11</td>
<td>0.11</td>
</tr>
<tr>
<td>40</td>
<td>3.4158</td>
<td>0.0947</td>
<td>15696.162</td>
<td>15709.7089</td>
<td>36.07</td>
<td>0.09</td>
</tr>
<tr>
<td>50</td>
<td>5.7226</td>
<td>0.1836</td>
<td>25056.162</td>
<td>25078.6808</td>
<td>31.17</td>
<td>0.09</td>
</tr>
<tr>
<td>60</td>
<td>8.308</td>
<td>0.3769</td>
<td>33648.6078</td>
<td>33673.8126</td>
<td>22.04</td>
<td>0.07</td>
</tr>
<tr>
<td>70</td>
<td>11.3333</td>
<td>0.6282</td>
<td>44129.6636</td>
<td>44163.5471</td>
<td>18.04</td>
<td>0.08</td>
</tr>
<tr>
<td>80</td>
<td>14.4667</td>
<td>0.9033</td>
<td>58842.9724</td>
<td>58881.0104</td>
<td>16.02</td>
<td>0.06</td>
</tr>
<tr>
<td>90</td>
<td>19.7863</td>
<td>1.6256</td>
<td>76604.5963</td>
<td>76648.2505</td>
<td>12.17</td>
<td>0.06</td>
</tr>
<tr>
<td>100</td>
<td>24.3176</td>
<td>2.2916</td>
<td>92656.7665</td>
<td>92703.3108</td>
<td>10.61</td>
<td>0.05</td>
</tr>
<tr>
<td>110</td>
<td>27.8984</td>
<td>3.1483</td>
<td>104613.7394</td>
<td>104668.5705</td>
<td>8.86</td>
<td>0.05</td>
</tr>
<tr>
<td>120</td>
<td>34.4835</td>
<td>3.9207</td>
<td>140582.8273</td>
<td>140653.5815</td>
<td>8.8</td>
<td>0.05</td>
</tr>
<tr>
<td>130</td>
<td>39.239</td>
<td>5.4096</td>
<td>153668.5981</td>
<td>153739.5342</td>
<td>7.25</td>
<td>0.05</td>
</tr>
<tr>
<td>140</td>
<td>44.1368</td>
<td>7.367</td>
<td>164976.8267</td>
<td>165046.7727</td>
<td>5.99</td>
<td>0.04</td>
</tr>
<tr>
<td>150</td>
<td>49.71</td>
<td>8.9107</td>
<td>191247.0172</td>
<td>191329.6465</td>
<td>5.58</td>
<td>0.04</td>
</tr>
<tr>
<td>160</td>
<td>58.9598</td>
<td>10.8687</td>
<td>235203.3597</td>
<td>235289.6197</td>
<td>5.42</td>
<td>0.04</td>
</tr>
<tr>
<td>170</td>
<td>69.9335</td>
<td>14.0521</td>
<td>276033.6344</td>
<td>276132.6041</td>
<td>4.98</td>
<td>0.04</td>
</tr>
<tr>
<td>180</td>
<td>77.1604</td>
<td>17.1239</td>
<td>314180.2029</td>
<td>314299.0124</td>
<td>4.51</td>
<td>0.04</td>
</tr>
<tr>
<td>190</td>
<td>83.9298</td>
<td>20.4455</td>
<td>328874.5147</td>
<td>328980.712</td>
<td>4.11</td>
<td>0.03</td>
</tr>
<tr>
<td>200</td>
<td>88.0746</td>
<td>24.2299</td>
<td>335153.2821</td>
<td>335259.9399</td>
<td>3.63</td>
<td>0.03</td>
</tr>
<tr>
<td>210</td>
<td>101.0347</td>
<td>29.0111</td>
<td>384124.7913</td>
<td>384253.5955</td>
<td>3.48</td>
<td>0.03</td>
</tr>
<tr>
<td>220</td>
<td>116.4357</td>
<td>36.6883</td>
<td>436599.988</td>
<td>436736.9412</td>
<td>3.17</td>
<td>0.03</td>
</tr>
<tr>
<td>230</td>
<td>123.1874</td>
<td>39.7522</td>
<td>475430.3302</td>
<td>475568.636</td>
<td>3.1</td>
<td>0.03</td>
</tr>
<tr>
<td>240</td>
<td>132.1277</td>
<td>45.9577</td>
<td>497762.2709</td>
<td>497885.6876</td>
<td>2.87</td>
<td>0.02</td>
</tr>
<tr>
<td>250</td>
<td>147.4051</td>
<td>54.0778</td>
<td>567895.5329</td>
<td>568033.4492</td>
<td>2.73</td>
<td>0.02</td>
</tr>
<tr>
<td>260</td>
<td>156.2205</td>
<td>57.9278</td>
<td>603780.3603</td>
<td>603944.1798</td>
<td>2.7</td>
<td>0.03</td>
</tr>
<tr>
<td>270</td>
<td>170.8563</td>
<td>70.7101</td>
<td>644494.7449</td>
<td>644657.7876</td>
<td>2.42</td>
<td>0.03</td>
</tr>
<tr>
<td>280</td>
<td>183.4643</td>
<td>78.844</td>
<td>709629.7991</td>
<td>709798.0883</td>
<td>2.33</td>
<td>0.02</td>
</tr>
<tr>
<td>290</td>
<td>195.8565</td>
<td>82.4432</td>
<td>774105.5554</td>
<td>774295.067</td>
<td>2.38</td>
<td>0.02</td>
</tr>
<tr>
<td>300</td>
<td>226.8268</td>
<td>107.7673</td>
<td>825770.4737</td>
<td>825959.0465</td>
<td>2.1</td>
<td>0.02</td>
</tr>
</tbody>
</table>

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 set of instances, the value of $\varepsilon$ 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 $n \times \varepsilon \times 1500$. 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


