Maximum Lateness Minimization on Two-Parallel Machine with a Non-availability Interval

Gais Alhadi[1], Imed Kacem[2], Pierre Laroche[3], and Izzeldin M. Osman[4]

***Abstract*— In this paper, we consider the two-parallel machine scheduling problem with a non-availability interval. We aim to minimize the maximum lateness when every job has a positive tail. We show that the problem has a constant polynomial approximation algorithm. We present a dynamic programming algorithm and we show that the problem has an FPTAS (Fully Polynomial Time Approximation Algorithm). The proposed FPTAS has a strongly polynomial running time. Finally, we present some numerical experiments and we analyze the obtained results.**

I. INTRODUCTION

We consider the two-parallel machine scheduling problem, with the aim of minimizing the maximum lateness. Formally, the problem is defined as follows. We have to schedule a set J of n jobs on two identical parallel machines. Each job $j \in J$ has a processing time p_j, and delivery time q_j. Each machine can only perform one job at a given time. The machines are available at time $t = 0$, which can process at most one job at a time (note that the first machine is available until time T_1 after that machine cannot process any job). The problem is to find a sequence of jobs, with the objective of minimizing the maximum lateness $L_{max} = max_{(1\leq j\leq n)}\{C_j(s) + q_j\}$ where $C_j(s)$ the completion time of job j in sequence S. With no loss of generality, we consider that all data are integers and that jobs are indexed in non-increasing order of their delivery times $q_1 \geq q_2 \geq \ldots \geq q_n$ (i.e., according to Jackson's order). In the remainder of this paper, the studied problem is denoted by π.

For self-consistency, we recall some necessary definitions related to the approximation area. An algorithm A is called a $\rho-$*approximation algorithm* for a problem, if for any instance I of that problem the algorithm A yields a feasible solution with an objective value $A(I)$ such that: $|A(I) - OPT(I)| \leq \varepsilon.OPT(I)$, where $OPT(I)$ is the optimal value for instance I, and ρ is the worst-case ratio of the approximation algorithm A. It can be a real number greater or equal to 1 for the minimization problems $\rho = 1 + \varepsilon$ (that it leads to inequality $A(I) \leq (1+\varepsilon)OPT(I)$ where $\varepsilon > 0$), or it can be real number from the interval $[0,1]$ for the maximization problems $\rho = 1 - \varepsilon$ (that leads to the inequality $A(I) \geq (1-\varepsilon)OPT(I)$). An *approximation scheme* for problem P is a family of $(1+\varepsilon)-$ approximation algorithms A_ε (respectively, $((1-\varepsilon)-$ approximation algorithms A_ε) over all $0 < \varepsilon < 1$. A problem has a *fully polynomial time approximation scheme* (FPTAS) if and only if for all $\varepsilon > 0$ it has $(1+\varepsilon)-$ approximation where the running time is polynomial in $1/\varepsilon$ and polynomial in the size of the instance. For instance, we can refer to the polynomial approximation schemes proposed in [12] for the problem of two-machine flow shop with release dates to minimize the maximum completion time, and also for the single and parallel-machine scheduling problem with release dates to minimize the maximum lateness. In the same reference, the authors also discussed two scheduling problems with precedence constraints and introduced new approximation results.

It is noteworthy that during the last decade numerous scheduling problems with non-availability constraints have attracted several researchers and have been widely studied in the literature. For example, Kacem and Kellerer[2] studied the problem of minimizing the maximum lateness on a single machine with an unexpected machine non-availability interval. They proposed a $(1+\sqrt{2}/2)$-approximation algorithm capable to solve the problem with delivery times but no release dates. This ratio is tight for the proposed algorithm and allows us to establish a precise window for the best possible ratio, which belongs to $[3/2, 1+\sqrt{2}/2]$ (see [2]). Kacem and Levner [1] considered a single machine problem of minimizing the total completion time with non-resumable and proportional deteriorating jobs under a maintenance constraint. They proposed a dynamic programming algorithm and obtained an FPTAS. Kacem et al. [3] studied the two-parallel machines where one of the machines has one non-availability interval, with the aim of minimizing the weighted completion time. They proposed a dynamic programming algorithm and they showed that the problem has an FPTAS with a strongly polynomial time complexity. In [10], Kacem considered the non-resumable single machine scheduling problem with a fixed non-availability interval to minimize the maximum lateness and he proposed a polynomial approximation algorithm with a tight worst-case performance ratio of $3/2$. He also proposed a dynamic programming algorithm and showed that the problem has an FPTAS with strongly polynomial complexity. Kacem et al ([7]-[5]) improved this FPTAS. Kacem and Chu [8] considered a single machine scheduling problem of minimizing the weighted sum of the completion times with one non-availability period. They proved that the WSPT and MWSPT heuristics have the same worst-case performance ratios (3 for the two heuristics under some conditions), and they showed that these worst-

[1]Gais Alhadi is a member of Faculty of Mathematical and Computer Sciences, University of Gezira, Wad-Madani, Sudan gais.alhadi@uofg.edu.sd

Imed Kacem[2] and Pierre Laroche[3] are members of the LCOMS Laboratory, University of Lorraine, F-57045 Metz, France imed.kacem@univ-lorraine.fr, pierre.laroche@univ-lorraine.fr

Izzeldin M. Osman[4] is from Sudan University of Science and Technology, Khartoum, Sudan izzeldin@acm.org

978-1-5386-5065-3/18/$31.00 ©2018 IEEE

case performance ratios are tight. The same problem has been studied in [11] by Kacem who showed that there is a $2-$approximation algorithm, which can be implemented in $O(n^2)$. Kacem and Mahjoub [4] proposed an efficient FPTAS. In [9], Kacem and Haouari proposed approximation algorithms for the single machine scheduling problem with release dates and tails under one non-availability constraint, with the aim of minimizing the maximum lateness. They showed that the problem has an FPTAS when tails are equal, and a $(2+\varepsilon)-$approximation has been deduced for the general case. Moreover, they showed that the worst-case performance of Schrage's algorithm is not affected by one non-availability interval and established that its tight bound is 2. Recently, Kacem and Kellerer [6] proved the existence of a PTAS for this problem, which represents the best possible result given its NP-hardness in the strong sense.

The organization of this paper is as follows: the next Section II presents the constant approximation heuristics. Section III provides the description and the analysis of the dynamic programming algorithm and the FPTAS. Section IV presents the experimental results and the last Section V concludes the paper.

II. CONSTANT APPROXIMATION HEURISTICS

In this section, we consider two constructive heuristics in the worst-case.

A. The first heuristic

The first heuristic H' is to put all the available jobs as soon as possible on the second machine (not necessary according to the Jackson's order).

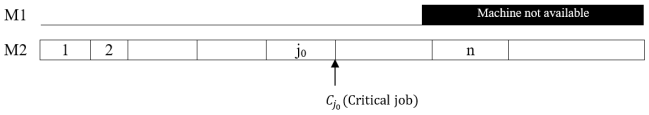


Fig. 1: Illustration of the critical job

Lemma 1.1: Any assignment of jobs to the machines is a constant approximation with a standard ratio no more than 3.

Proof: We define $P=\sum_{j=1}^{n} p_j$, and OPT the optimal value of the maximum lateness.
Now, let j_0 be the critical job in the sequence given by H' such that $C_{j_0}+q_{j_0}=L_{max}(H')$ as it is shown in Fig. 1. Clearly, we have:

$$L_{max}(H') \leq P+q_{j_0} \quad \text{(II-A.1)}$$

Obviously, we have:

$$OPT \geq P/2 \quad \text{(II-A.2)}$$

and

$$OPT \geq q_{j_0} \quad \text{(II-A.3)}$$

Therefore, it can be deduced that $L_{max}(H') \leq 2.(P/2)+q_{j_0} \leq 2.OPT+OPT=3.OPT$ ■

B. The second heuristic

The second heuristic H puts all the jobs as soon as possible on the second machine according to the Jackson's order.
We use again Fig. 1 to refer to the sequence given by H. Here, we define again the critical job such that $C_{j_0}+q_{j_0}=L_{max}(H)$.

Lemma 2.1: Any assignment of jobs to the machines according to the Jackson's order is a constant approximation with a standard ratio no more than 2.

Proof: Clearly, we have:

$$L_{max}(H)=\sum_{i=1}^{j_0} p_i+q_{j_0} \quad \text{(II-B.1)}$$

Moreover, it can be established by considering the relaxed problem of scheduling the jobs in $\{1,2,...,j_0\}$ on two identical machines that the following relation holds:

$$OPT \geq (\sum_{i=1}^{j_0} p_i)/2+q_{j_0} \quad \text{(II-B.2)}$$

Indeed, there is at least one job from the subset $\{1,2,...,j_0\}$ for which the completion time will be greater than $(\sum_{i=1}^{j_0} p_i)/2$. Consequently, we deduce the following relation:

$$L_{max}(H)=\sum_{i=1}^{j_0} p_i+q_{j_0} \leq OPT+(\sum_{i=1}^{j_0} p_i)/2 \leq 2.OPT \quad \text{(II-B.3)}$$

■

III. FULLY POLYNOMIAL-TIME APPROXIMATION SCHEME

A. Dynamic programming algorithm

The following dynamic programming algorithm A, can be applied for solving this problem optimally. This algorithm A generates iteratively some sets of states. At every iteration j, a set Q_j of states is generated $(0 \leq j \leq n)$. Each state $[t,f]$ in Q_j can be associated to a feasible schedule for the j first jobs. Here, variable t denotes the completion time of the last job scheduled on the first machine before T_1, and variable f is the maximum lateness of the corresponding schedule.

Algorithm A

1) Set $Q_0=\{[0,0]\}$.
2) For $j \in \{1,2,3,...,n\}$,
   $Q_j=\{\}$
   For every state $[t,f]$ in Q_{j-1}
   a) Put $[t, max\{f, \sum_{i=1}^{j} p_i - t + q_j\}]$ in Q_j
   b) Put $[t+p_j, max\{f, t+p_j+q_j\}]$ in Q_j (if $t+p_j \leq T_1$)
   Remove Q_{j-1}
3) Return $L^*_{max}=min_{[t,f]\in Q_n}\{f\}$

The standard version of this dynamic programming has an exponential complexity since it needs $O(2^n)$ time. Nevertheless, this complexity can be reduced to $O(n.T_1)$ by keeping only one state having the smallest value of f for every value t in Q_j (for every iteration $j \in \{1,2,...,n\}$).

Therefore, A can be reduced to a pseudo-polynomial algorithm, which proves at the same time that the problem is NP-hard only in the ordinary sense.

B. FPTAS

Here, we are interested in the existence of an FPTAS for the studied problem. In the proposed FPTAS we remove a special part of the states generated by Algorithm A, in order to produce an approximation solution instead of the optimal solution. The new algorithm is called A^{ε} ($\varepsilon > 0$) It will be the proposed FPTAS, which uses similar ideas as in Kacem et al. (see for instance [5]-[3]).
By comparing the steps of algorithms A^{ε} and A, we produce the analysis at the worst-case for the proposed FPTAS.
Given an arbitrary $\varepsilon > 0$, we define:

$$\gamma_1 = \frac{2}{\varepsilon}, \tag{III-B.1}$$

$$\gamma_2 = \frac{4.n}{\varepsilon} \tag{III-B.2}$$

Without loss of generality, we assume that γ_1 and γ_2 are integers (otherwise, we round up to the next integer values). Now, we define the following additional parameters, which we will use in the FPTAS:

$$\delta_1 = \frac{T_1}{\gamma_1}, \tag{III-B.3}$$

$$\delta_2 = \frac{L_{max}(H)}{\gamma_2} \tag{III-B.4}$$

We divide the interval $[0, T_1]$ (respectively, the interval $[0, L_{max}(H)]$) into γ_1 equal sub intervals $X_h = [(h-1)\delta_1, h\delta_1]_{1 \le h \le \gamma_1}$ of length δ_1 (respectively, into γ_2 equal sub intervals $Y_k = [(k-1)\delta_2, k\delta_2]_{1 \le k \le \gamma_2}$ of length δ_2). The algorithm A^{ε} described below produces the reduced sets $Q_j^{\#}$ instead of sets Q_j.

Algorithm A^{ε}

1) Set $Q_0^{\#} = \{[0,0]\}$.
2) For $j \in \{1,2,3,...,n\}$,
   $Q_j^{\#} = \{\}$
   For every state $[t,f]$ in $Q_{j-1}^{\#}$
   a) Put $[t, max\{f, \sum_{i=1}^{j} p_i - t + q_j\}]$ in $Q_j^{\#}$
   b) Put $[t + p_j, max\{f, t + p_j + q_j\}]$ in $Q_j^{\#}$ (if $t + p_j \le T_1$)
   Remove $Q_{j-1}^{\#}$
   Let $[t,f]_{h,k}$ and $[s,g]_{h,k}$ be the states in $Q_j^{\#}$ such that $t, s \in X_h$ and $f, g \in Y_k$ where t (respectively, s) is the smallest value (respectively, the greatest value) in subinterval X_h. Set $Q_j^{\#} = \{[t,f]_{h,k}, [s,g]_{h,k} | 1 \le h \le \gamma_1, 1 \le k \le \gamma_2\}$.
3) Return $L_{max}^{A^{\varepsilon}} = min_{[t,f] \in Q_n^{\#}}\{f\}$

Lemma 2.1: For every state $[t,f] \in Q_j$ there exists at least one approximate state $[t^{\#}, f^{\#}] \in Q_j^{\#}$ such that:

$$t - \delta_1 \le t^{\#} \le t \tag{III-B.5}$$

and

$$f^{\#} \le f + \delta_1 + j\delta_2 \tag{III-B.6}$$

Proof: By induction on j.
First, for $j = 0$ we have $Q_j^{\#} = Q_j$ and the statement is trivial. Then, assume that the lemma holds true up to level $j-1$. Now, let us consider an arbitrary state $[t,f] \in Q_j$. Algorithm A introduces this state into Q_j when job j is added to some feasible state for the $j-1$ first jobs. Let $[t', f']$ be the above feasible state. Two cases can be distinguished:
Either $[t,f] = [t', max\{f', \sum_{i=1}^{j} p_i - t' + q_j\}]$ or $[t,f] = [t' + p_j, max\{f', t' + p_j + q_j\}]$ must hold. We will prove the statement for level j in the two cases.

Case 1 : $[t,f] = [t', max\{f', \sum_{i=1}^{j} p_i - t' + q_j\}]$

Since $[t', f'] \in Q_{j-1}$, there exists $[t^{\#\prime}, f^{\#\prime}] \in Q_{j-1}^{\#}$, such that $t' - \delta_1 \le t^{\#'} \le t'$ and $f^{\#\prime} \le f' + \delta_1 + (j-1)\delta_2$. Consequently, the state $e = [t^{\#'}, max\{f^{\#'}, \sum_{i=1}^{j} p_i - t^{\#'} + q_j\}]$ is created at iteration j by algorithm A^{ε} and it is feasible. However, it may be removed when reducing the state subset. Let $[\lambda, \mu]$ and $[\alpha, \beta]$ be the states replacing sate e in $Q_j^{\#}$ ($\lambda \le \alpha$). This means that the states $[\lambda, \mu]$ and $[\alpha, \beta]$ belong to the same "box" as the state e. Therefore, we have:

$$\lambda \le t^{\#'} \le \alpha, \tag{III-B.7}$$

Now we will consider two sub-cases to prove the existence of a "close" state in $Q_j^{\#}$ for approximating $[t,f]$. These sub-cases are: $t' \le \alpha$ or $t' > \alpha$.
In the first sub-case ($t' \le \alpha$), we will take $[\lambda, \mu]$ as a close state for $[t,f]$. Indeed, we can observe that $\lambda \le t^{\#'} \le t' = t$ and that $\lambda \ge \alpha - \delta_1 \ge t' - \delta_1 = t - \delta_1$. Moreover,

$$\begin{aligned}
\mu &\le max\{f^{\#'}, \sum_{i=1}^{j} p_i - t^{\#'} + q_j\} + \delta_2 \\
&\le max\{f' + \delta_1 + (j-1)\delta_2, \sum_{i=1}^{j} p_i - t^{\#'} + q_j\} + \delta_2 \\
&\le max\{f' + \delta_1 + (j-1)\delta_2, \sum_{i=1}^{j} p_i - t' + \delta_1 + q_j\} + \delta_2 \\
&\le max\{f', \sum_{i=1}^{j} p_i - t' + q_j\} + \delta_1 + j\delta_2 \\
&= f + \delta_1 + j\delta_2
\end{aligned} \tag{III-B.8}$$

Consequently, $[\lambda, \mu]$ is an approximate state in $Q_j^{\#}$ verifying the two conditions.

In the second sub-case ($t' > \alpha$), we will take $[\alpha, \beta]$ as a close state for $[t,f]$. Indeed, we can remark that $\alpha < t' = t$ and that $\alpha \ge t^{\#'} \ge t' - \delta_1 = t - \delta_1$. Moreover,

$$\beta \le max\{f^{\#'}, \sum_{i=1}^{j} p_i - t^{\#'} + q_j\} + \delta_2 \tag{III-B.9}$$

By a similar reasoning as in the first sub-case for bounding μ, we can deduce the following relation for β:

$$\beta \leq f + \delta_1 + j\delta_2$$

Therefore, we can verify that $[\alpha, \beta]$ respects all the required conditions.

Case 2 :$[t, f] = [t^{'} + p_j, max\{f^{'}, t^{'} + p_j + q_j\}]$

Since $[t^{'}, f^{'}] \in Q_{j-1}$, there exists $[t^{\#\prime}, f^{\#\prime}] \in Q^{\#}_{j-1}$, such that $t^{'} - \delta_1 \leq t^{\#^{'}} \leq t^{'}$ and $f^{\#\prime} \leq f^{'} + \delta_1 + (j-1)\delta_2$. Consequently, the state $e' = [t^{\#^{'}} + p_j, max\{f^{\#^{'}}, t^{\#^{'}} + p_j + q_j\}]$ is created by algorithm A^{ε} at iteration j. However, it may be removed when reducing the state subset. Again, we have to consider two states $[\lambda', \mu']$ and $[\alpha', \beta']$, which are used to replace e' at iteration j. Similarly to the proof in Case 1, we have to distinguish two sub-cases ($\alpha' \leq t' + p_j$ or $\alpha' > t' + p_j$). For these two sub-cases, we can demonstrate that one of the states $[\lambda', \mu']$ and $[\alpha', \beta']$ can verify the required conditions. The detail is left to the reader.
In conclusion, the statement holds also for level j in the two considered cases, and this completes the proof. ■
We are now ready to establish the existence of an FPTAS with a strongly polynomial time for the studied problem.

Theorem 2.2: Algorithm A^{ε} is an FPTAS for the problem π and it can be implemented in $O(\frac{n^2}{\varepsilon^2})$.

Proof: Let $[t^*, f^*] \in Q_n$ such that $OPT = f^*$. From the previous lemma, it can be deduced that there exists a feasible solution $S^{\#}$ of the problem π associated to a state $[t^{\#}, f^{\#}] \in Q^{\#}_n$ such that:

$$f^{\#} \leq f^* + \delta_1 + n.\delta_2 \quad \text{(III-B.10)}$$

$$= OPT + \varepsilon.T_1/2 + n.\varepsilon.L_{max}(H)/4.n \quad \text{(III-B.11)}$$

$$\leq (1+\varepsilon).OPT \quad \text{(III-B.12)}$$

Regarding the time complexity, the algorithm generates at each iteration a number of states in $O(\gamma_1\gamma_2)$. Indeed, we keep at most $2\gamma_1\gamma_2$ states at every iteration j, leading to a maximum of possible candidates equal to $4\gamma_1\gamma_2$ for the next iteration $j+1$, at which only $2\gamma_1\gamma_2$ states can stand again. In sum, this leads to a global time complexity in $O(\frac{n^2}{\varepsilon^2})$. ■

IV. RESULTS

A. First instances

Figure 2 presents a comparison of FPTAS and Dynamic Programming. For three values of non-availability interval and two ε values, each sub-figure shows the ratio $L^{A^{\varepsilon}}_{max}/L^*_{max}$ for our five sets of instances. It is no surprise that it is easier to obtain close to optimal solutions when the non-availability interval is bigger. Indeed, in these cases, the solutions space is limited: as soon as machine M_1 has reached non-availability, the only choice left is to assign the jobs to machine M_2.

This figure also shows that, regardless of the non-availability interval, FPTAS gives better results when the number of jobs is important. The average gap between FPTAS solutions and the optimal solutions is never more than 0.4%. Considering all the tests we have made, the biggest gap with the optimal solution was 3.2%. The good quality of FPTAS when dealing with big instances can be explained by the way the space search is decomposed, the number of cells given by $\gamma 2$ (see III-B.2) being a function of n.

We have also studied the influence of processing and delivery times. Our experiments showed that delivery time ranges have no influence, neither on the quality of FPTAS, nor on the computing times of our algorithms. At the opposite, Table I shows that instances composed of jobs with various processing times are more difficult to solve. Results are presented for $T_1 = P/3$ and $\varepsilon = 0.9$. It should be noted that the analysis would be the same whatever the considered parameters. This table shows the number of instances solved to the optimal by FPTAS and the ratio between the FPTAS solution and the optimal solution ($L^{A^{\varepsilon}}_{max}/L^*_{max}$). The results are presented for two sets of instances, short ones (from 5 to 25 jobs) and bigger ones (from 100 to 200 jobs).

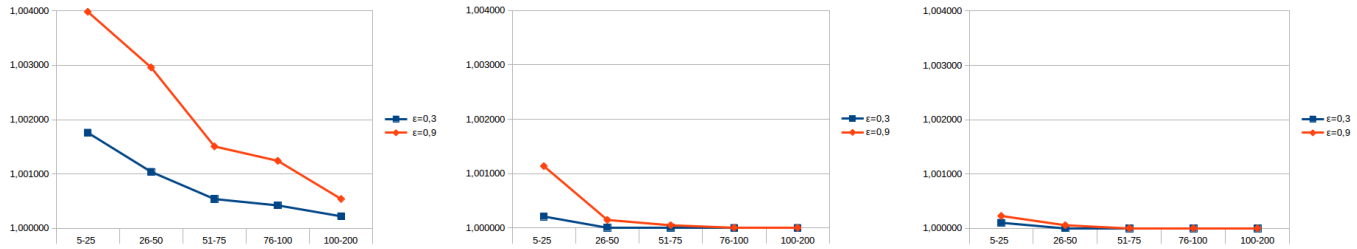
TABLE I: Quality of FPTAS solutions as a function of processing time ranges (for $T_1 = P/3$ and $\varepsilon = 0.9$)

p_i range	% of optimal solutions	$L^{A^{\varepsilon}}_{max}/L^*_{max}$	DP time	FPTAS time
#jobs 5 to 25				
1-20	100	1	0.3	0.15
1-100	80	1.0008	3	0.15
1-500	40	1.0013	37	0.15
#jobs 100 to 200				
1-20	100	1	145	21
1-100	100	1	3068	20
1-500	90.48	1.000006	177449	20

We can see that instances with jobs having a wide range of processing times are more difficult to solve to optimality. For processing times between 1 and 100, the FPTAS gives the optimal solution for 90% (resp. 100%) of the small (resp. big) instances, whereas the optimality is not reached for 40% (resp. 90.48%) of the small (resp. big) instances. The ratios between FPTAS solutions and optimal solutions show that, even when the optimality is rarely reached, the gap remains very small.

This table also mentions the computing times, in milliseconds. We can see that the range of processing times has no influence on the FPTAS, whereas it is an important parameter for the Dynamic Programming algorithm. This is consistent w.r.t. the complexity results given in previous section.

More computing times are also given in Tables II and III. They compare our Dynamic Programming algorithm and our FPTAS, considering two different values for T_1. As earlier, all values are in milliseconds. The Dynamic Programming algorithm is slower when the non-availability interval is decreasing and the number of states is growing, as expected with a complexity $O(n.T_1)$ (see Section III-A). We can remark that computing times are not in a $2/7$ ratio, as expected. This is due to computational effort made to keep only one state for each t value, which is not considered by



$T_1 = P/2$ $T_1 = P/3$ $T_1 = P/7$

Fig. 2: Quality of solutions for our instances with various T_1 and ε values

the theoretical study.

As expected, FPTAS outperforms Dynamic Programming, especially with big values of ε and small values of T_1. The computing times are consistent with the complexity $O(\frac{n^2}{\varepsilon^2})$.

TABLE II: Average computing times (ms) for $T_1 = P/2$

#jobs	DP	FPTAS	
		$\varepsilon = 0.3$	$\varepsilon = 0.9$,
5-25	67	0.9	0.3
26-50	881	3	1
51-75	5903	10	3
76-100	21963	22	7
100-200	138431	74	23

TABLE III: Average computing times (ms) for $T_1 = P/7$

#jobs	DP	FPTAS	
		$\varepsilon = 0.3$	$\varepsilon = 0.9$,
5-25	1.5	0.2	0.1
26-50	37	1.5	0.5
51-75	290	4	1.2
76-100	975	8	2.5
100-200	8118	28	10

B. Big instances

We also tested bigger instances, with 1000 jobs. We generated 12 instances, with 1-20 and 1-100 processing times. The results are consistent with the ones of smaller instances. Table IV presents the quality of FPTAS and computing times for both Dynamic Programming and FPTAS. For these instances, as the computing times are quite big, they have been expressed in seconds. We can see that the FPTAS still managed to find very good solutions very quickly, as we expected after previous experiments.

TABLE IV: Results for big instances (times in seconds)

T_1	DP time	FPTAS			
		$\varepsilon = 0.3$		$\varepsilon = 0.9$	
		time	$L^{A^\varepsilon}_{max}/L^*_{max}$	time	$L^{A^\varepsilon}_{max}/L^*_{max}$
$P/2$	1883	3.2	1.000007	1.6	1.00003
$P/3$	980	3.5	1	1.2	1
$P/7$	137	1.1	1	0.6	1

V. CONCLUSION

In this paper, we considered the two-parallel machine scheduling problem without release dates assumption, with the aim of minimizing the maximum lateness. In this problem, instead of allowing both machines to be continuously available we consider that the first machine is available for a specified period of time (available until time T_1). After that, this machine cannot process any job. We proposed an approximation heuristic which has the performance ratio of the worst case not more than $3/2$. We also have proved the existence of an FPTAS for the problem. The results of the numerical experiments showed that the proposed algorithms for the considered problem are very efficient, especially for big instances composed of a lot of jobs. Moreover, the proposed FPTAS which has a strongly polynomial running time has been shown to be very efficient. We have also shown the importance of the values of processing times, non-availability interval, and ε value.

In our future works, we look forward to extend these results to other variants of this problem, considering for instances a fixed number of machines (more than two). We will also investigate some multi-objective versions of this problem, trying to optimize not only the maximum lateness, but also other criteria, e.g. the maximum completion time.

REFERENCES

[1] I. Kacem and E. Levner, *An improved approximation scheme for scheduling a maintenance and proportional deteriorating jobs*. Journal of Industrial and Management Optimization, 12:3, 811-817, 2015.

[2] I. Kacem, H. Kellerer, *Semi-online scheduling on a single machine with unexpected breakdown*. Theoretical Computer Science, 646, 40-48, 2016.

[3] I. Kacem, M. Sahnoune, G. Schmidt, *Strongly Fully Polynomial Time Approximation Scheme for the weighted completion time minimization problem on two-parallel capacitated machines*, RAIRO - Operations Research, 51:4, 1177-1188, 2017.

[4] I. Kacem and A. R. Mahjoub, *Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval*, Computers & Industrial Engineering, 56:4, 1708-1712, 2009.

[5] I. Kacem, H. Kellerer, *Improved Fully Polynomial Approximation Schemes for the Maximum Lateness Minimization with a Single Machine with a Fixed Operator or Non-Availability Interval*, Technical Report, Universit de Lorraine, April 2017.

[6] I. Kacem, H. Kellerer, *Approximation Schemes for Minimizing the Maximum Lateness on a Single Machine with Release Times under Non-Availability or Deadline Constraints*, CoRR abs/1708.05102 (2017).

[7] I. Kacem, H. Kellerer, M. Seifaddini, *Efficient approximation schemes for the maximum lateness minimization on a single machine with a fixed operator or machine non-availability interval*, Journal of Combinatorial Optimization, 32:3, 970-981, 2016.

[8] I. Kacem, C. Chu, *Worst-case analysis of the WSPT and MWSPT rules for single machine scheduling with one planned setup period*, European Journal of Operational Research, 187:3, 1080-1089, 2008.

[9] I. Kacem and M. Haouari, *Approximation algorithms for single machine scheduling with one unavailability period*, 4OR: A Quarterly Journal of Operations Research, 7:1, 79-92, 2009.

[10] I. Kacem, *Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval*, Journal of Combinatorial Optimization, 17:2, 117-133, 2009.

[11] I. Kacem, *Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval*, Computers & Industrial Engineering, 54:3, 401-410, 2008.

[12] L. A. Hall and D. B. Shmoys, *Approximation schemes for constrained scheduling problems*, In Foundations of Computer Science, 30th Annual Symposium on ,pp. 134-139, IEEE, 1989.