J Comb Optim (2016) 32:970–981
DOI 10.1007/s10878-015-9924-4

Efficient approximation schemes for the maximum lateness minimization on a single machine with a fixed operator or machine non-availability interval

Imed Kacem[1] · **Hans Kellerer**[2] · **Maryam Seifaddini**[1]

Published online: 16 July 2015

Abstract In this paper we deal with the single machine scheduling problem with one non-availability interval to minimize the maximum lateness where jobs have positive tails. Two cases are considered. In the first one, the non-availability interval is due to the machine maintenance. In the second case, the non-availability interval is related to the operator who is organizing the execution of jobs on the machine. The contribution of this paper consists in an improved fully polynomial time approximation scheme (FPTAS) for the maintenance non-availability interval case and the elaboration of the first FPTAS for the operator non-availability interval case. The two FPTASs are strongly polynomial.

Keywords Approximation schemes · Scheduling · Maximum lateness · Single machine · Fixed operator Interval · Machine non-availability interval

1 Introduction

In this paper we deal with the single machine scheduling problem with one non-availability interval to minimize the maximum lateness where jobs have positive tails. Two cases are considered. In the first one, the non-availability interval is due to the machine maintenance. In the second case, the non-availibility interval is related to the operator who is organizing the execution of jobs on the machine. An operator

The short version of this paper has been presented at ISCO'2014 conference (2014).

✉ Imed Kacem
imed.kacem@univ-lorraine.fr

[1] LCOMS EA 7306, Université de Lorraine, 57000 Metz, France

[2] ISOR, University of Graz, Graz, Austria

non-availability period is a time interval in which no job can start, and neither can complete. The main difference between machine non-availability (MNA) and operator non-availability (ONA) consists in the fact that a job can be processed but cannot start neither finish during the ONA period. However, the MNA interval is a completely forbidden period. Rapine et al. (2012) have described the applications of this problem in the planning of a chemical experiments as follows: Each experiment is performed by an automatic system (a robot), during a specified amount of time, but a chemist is required to control its start and completion. At the beginning, the chemist launches the process (preparation step). The completion step corresponds to the experimental analysis, which is to be done in a no-wait mode to stop chemical reactions. Here, the automatic system is available all the time, where the chemists may be unavailable due to planned vacations or activities. This induces operator (chemist) non-availability intervals when experiments (jobs) can be performed by the automatic system (machine), but cannot neither start nor complete.

The MNA case of this type of problems has been studied in the literature under various criteria [a sample of these works includes Lee (1996), Kacem (2009), Kubzin and Strusevich (2006), Qi (2007), Qi et al. (1999), Schmidt (2000), He et al. (2006)]. However, few papers studied the problem we consider in this paper. Lee (1996) explored the Jackson's sequence JS and proved that its deviation to the optimal makespan cannot exceed the largest processing time, which is equivalent to state that JS is a 2-approximation. Recently, Yuan et al. developed an interesting PTAS (Polynomial Approximation Scheme) for the studied problem (Yuan et al. 2008). Kacem (2009) presented a first fully polynomial time approximation scheme (FPTAS) for the maximum lateness minimization. That is why this paper is a good attempt to design more efficient approximation heuristics and approximation schemes to solve the studied problem.

For the ONA case, few works have been published. Brauner et al. (2009) considered the problem of single machine scheduling with ONA periods. They analyzed this problem on a single machine with the makespan as a minimization criterion and they showed that the problem is NP-hard with one ONA period. They also considered the problem with K ONA periods such that the length of each ONA period is no more than $\frac{1}{\lambda}$ times the total processing time of all jobs. They introduced a worst-case ratio smaller than $1 + \frac{2K}{\lambda}$ for algorithm LS (list scheduling). They presented an approximation algorithm with a worst-case ratio close to $2 + \frac{K-1}{\lambda}$. The natural case of periods where the duration of the periods is smaller than any processing time of any job, has been considered by Rapine et al. (2012). They proved the problem can be solved in polynomial time, where there exists only one ONA period and they showed the problem is NP-hard if one has $K \geq 2$ small non-availability periods and the worst-case ratio of LS is no more than $\frac{K+1}{2}$ and the problem does not admit an FPTAS for $K \geq 3$ unless P = NP.

Recently, Chen et al. (2013) considered the single machine scheduling with one ONA period to minimize the total completion time. The problem is NP-hard even if the length of the ONA period is smaller than the processing time of any job. They have also presented an algorithm with a tight worst-case ratio of $\frac{20}{17}$. They showed that the worst-case ratio of SPT is at least $\frac{5}{3}$.

Table 1 Summary of results

	Result	Reference
MNA	2-approximation (Jackson's Rule)	Lee (1996)
MNA	PTAS: $O(n\ln(n) + n.2^{1/\varepsilon})$	Yuan et al. (2008)
MNA	FPTAS: $O(n^3/\varepsilon^2)$	Kacem (2009)
MNA	PTAS: $O(n\ln(n) + (1/\varepsilon)\,2^{O(1/\varepsilon)})$	This work
MNA	FPTAS: $O(n\ln(n) + \min\{n, 3/\varepsilon\}^3/\varepsilon^2)$	This work
ONA	PTAS: $O(n^2/\varepsilon + (n/\varepsilon^2)2^{O(1/\varepsilon)})$	This work
ONA	FPTAS: $O((n^2/\varepsilon) + n\min\{n, 3/\varepsilon\}^3/\varepsilon^3)$	This work

The contribution of this paper consists in an improved FPTAS for the MNA interval case and the elaboration of the first FPTAS for the ONA interval case. The two FPTASs are strongly polynomial. These contributions can be summarized in Table 1 for the two cases. It is worthy to note that the two cases have been addressed together in this paper for two reasons. The first one is methodological. Indeed, we will show in this paper that the design of approximation schemes for the MNA case can lead to derive efficient extended schemes for the other case. The second reason is related to the common domain of applications (resource non-availability).

The paper is organized as follows. Section 2 describes the exact formulation of the MNA interval case and the improved FPTAS. Section 3 is devoted to the ONA interval case and to the presentation of the proposed FPTAS. Finally, Sect. 4 gives some concluding remarks.

2 Case under MNA interval

Here, the studied problem $(\mathcal{P})$ can be formulated as follows. We have to schedule a set J of n jobs on a single machine, where every job j has a processing time p_j and a tail q_j. The machine can process at most one job at a time and it is unavailable between T_1 and T_2 (i.e., (T_1, T_2) is a forbidden interval). Preemption of jobs is not allowed (jobs have to be performed under the non-resumable scenario). All jobs are ready to be performed at time 0. With no loss of generality, we consider that all data are integers and that jobs are indexed according to Jackson's rule (Carlier 1982) (i.e., jobs are indexed in nonincreasing order of tails). Therefore, we assume that $q_1 \geq q_2 \geq \cdots \geq q_n$. The consideration of tails is motivated by the large set of scheduling problems such that jobs have delivery times after their processing (Dessouky and Margenthaler 1972). Let $C_j\,(S)$ denote the completion time of job j in a feasible schedule S for the problem and let $\varphi_S(\mathcal{I})$ be the maximum lateness yielded by schedule S for instance $\mathcal{I}$ of $(\mathcal{P})$:

$$\varphi_S(\mathcal{I}) = \max_{1\leq j\leq n} \left(C_j\,(S) + q_j\right) \tag{1}$$

The aim is to find a feasible schedule S by minimizing the maximum lateness. It is well-known that the Jackson's order is optimal for the same problem without non-

availability constraint. Due to the dominance of Jackson's order, any optimal schedule for $\mathcal{P}$ is composed of two sequences of jobs scheduled in nondecreasing order of their indexes.

If all the jobs can be inserted before T_1, the instance studied ($\mathcal{I}$) has obviously a trivial optimal solution obtained by Jackson's rule. We therefore consider only the problems in which all the jobs cannot be scheduled before T_1. Moreover, we consider that every job can be inserted before T_1 (i.e., $p_j \leq T_1$ for every $j \in J$).

In the remainder of this paper $\varphi^*(\mathcal{I})$ denotes the minimal maximum lateness for instance $\mathcal{I}$.

2.1 New simplifications and PTAS

Now, let us describe our FPTAS. It uses a simplification technique based on merging small jobs Kacem and Kellerer (2014).

1st STEP:

First, we simplify the instance $\mathcal{I}$ as follows. Given an arbitrary $\varepsilon > 0$. We assume that $1/\varepsilon$ is integer. We split the interval $[0, \max_{j\in J}\{q_j\}]$ in $1/\varepsilon$ equal lenght intervals and we round up every tail q_j to the next multiple of $\varepsilon q_{\max}$ ($q_{\max} = \max_{j\in J}\{q_j\}$). The new instance is denoted as $\mathcal{I}'$.

Proposition 1 *The obtained instance $\mathcal{I}'$ can be obtained in $O(n)$ time and it can be done with no $(1+\varepsilon)$-loss.*

Proof The modification can be done by setting $q_j := \lceil q_j/\varepsilon q_{\max}\rceil \varepsilon q_{\max}$ for every $j \in J$. Then, it can be done in $O(n)$ time. Moreover, since $\lceil q_j/\varepsilon q_{\max}\rceil \varepsilon q_{\max} \leq q_j + \varepsilon q_{\max}$ then, $\varphi^*(\mathcal{I}') \leq \varphi^*(\mathcal{I}) + \varepsilon q_{\max} \leq (1+\varepsilon)\varphi^*(\mathcal{I})$ since $q_{\max}$ is a lower bound on the optimal maximum lateness. □

2nd STEP:

J is divided into at most $1/\varepsilon$ subsets $J(k)$ ($1 \leq k \leq 1/\varepsilon$) where jobs in $J(k)$ have identical tails of $k\varepsilon q_{\max}$. The second modification consists in reducing the number of small jobs in every subset $J(k)$. Small jobs are those having processing times $< \varepsilon P/2$ where $P = p_1 + p_2 + \cdots + p_n$. The others are called large jobs. The reduction is done by merging the small jobs in each $J(k)$ so that we obtain new greater jobs having processing times between $\varepsilon P/2$ and εP. The small jobs are taken in the order of their index in this merging procedure. At most, for every subset $J(k)$, a single small job remains. We re-index jobs according to nondecreasing order of their tails. The new instance we obtain is denoted as $\mathcal{I}''$. Clearly, the number of jobs remaining in the simplified instance $\mathcal{I}''$ is less than $3/\varepsilon$.

Proposition 2 *This reduction in the second step cannot increase the optimal solution value of $\mathcal{I}'$ by more than $(1+\varepsilon)$-factor. It can be done in $O(n)$ time.*

Proof The proof is based on the comparison of a lower bound lb for $\varphi^*(\mathcal{I}')$ and a feasible solution σ for instance $\mathcal{I}''$. We will demonstrate that $\varphi_\sigma(\mathcal{I}'') \leq (1+\varepsilon)\, lb$ which implies that $\varphi^*(\mathcal{I}'') \leq \varphi_\sigma(\mathcal{I}'') \leq (1+\varepsilon)\, lb \leq (1+\varepsilon)\, \varphi^*(\mathcal{I}')$.

The lower bound lb is the maximum lateness of the optimal solution σ_r of a special preemptive version of problem $\mathcal{I}'$ where the large jobs are supposed to be assigned (before or after the non-availability interval) as in the optimal solution of $\mathcal{I}'$. The principle of this lower bound is based on the splitting idea presented in Kacem (2009). Indeed, we split the small jobs so that the obtained pieces have an identical length of 1 and they keep their tails. It can be demonstrated that the pieces associated to the small jobs must be scheduled in σ_r according to the Jackson's order. At most, one piece of a certain job g will be preempted by the non-availability period. For more details on this lower bound, we refer to Kacem (2009).

It is easy to transform such a relaxed solution σ_r to a close feasible solution σ for $\mathcal{I}''$. Indeed, we can remark that the small jobs of every subset $J(k)$ $(1 \leq k \leq 1/\varepsilon)$, except the subset $k(g)$ containing job g, are scheduled contiguously before or after the non-availibility period. Hence, the associated merged jobs (in $\mathcal{I}''$) for these small ones (in $\mathcal{I}'$) will take the same order in sequence σ. The large jobs are common in the two instances and they will keep the same assignments in σ. The only possible difference between σ and σ_r will consist in the positions of small jobs (from $\mathcal{I}'$) belonging to subset $k(g)$. For these small jobs, we construct the same associated merged jobs as in $\mathcal{I}''$ by scheduling them in σ as close in σ_r as possible. As a consequence, some small jobs (from $\mathcal{I}'$ and belonging to subset $k(g)$) will be moved after the non-availability period. Thus, it is easy to deduce that $\varphi_\sigma(\mathcal{I}'') \leq lb + \varepsilon P \leq (1+\varepsilon)\, lb$. □

Theorem 3 *Problem P has a Polynomial Time Approximation Scheme (PTAS) with a time complexity of* $O(n\ln(n) + (1/\varepsilon)\, 2^{O(1/\varepsilon)})$.

Proof The proof is based on the two previous propositions. The Jackson's order can be obtained in $n\ln(n)$. We construct the optimal solution of $\mathcal{I}''$ by an exhaustive search in $O((1/\varepsilon)\, 2^{O(1/\varepsilon)})$. Then, we derive a feasible solution for $\mathcal{I}$, which can be done in $O(n)$. □

Remark 4 The new PTAS has a lower time complexity compared to the one proposed by Yuan et al. (2008) for which the time complexity is $O(n\ln(n) + n.2^{1/\varepsilon})$.

2.2 Improved FPTAS

The improved FPTAS is similar to the one proposed by Kacem (2009). It uses the same technique but exploits also the modification of the input ($\mathcal{I} \rightarrow \mathcal{I}''$). First, we use the Jackson's sequence JS obtained for the modified instance $\mathcal{I}''$. Then, we apply the modified dynamic programming algorithm APS'_ε introduced in Kacem (2009) on instance $\mathcal{I}''$.

The main idea of APS'_ε is to remove a special part of the states generated by a dynamic programming algorithm. Therefore, the modified algorithm becomes faster and yields an approximate solution instead of the optimal schedule (see Appendix 1). First, we define the following parameters:

$$\overline{n} = \min\{n, 3/\varepsilon\},$$
$$\omega_1 = \left\lceil \frac{2\overline{n}}{\varepsilon} \right\rceil,$$

$$\omega_2 = \left\lceil \frac{\overline{n}}{\varepsilon} \right\rceil,$$

$$\delta_1 = \frac{\varphi_{JS}\left(\mathcal{I}''\right)}{\omega_1}$$

and

$$\delta_2 = \frac{T_1}{\omega_2}.$$

We split $\left[0, \varphi_{JS}\left(\mathcal{I}''\right)\right)$ into ω_1 equal subintervals $I_m^1 = [(m-1)\delta_1, m\delta_1)_{1\leq m\leq \omega_1}$. We also split $[0, T_1)$ into ω_2 equal subintervals $I_s^2 = [(s-1)\delta_2, s\delta_2)_{1\leq s\leq \omega_2}$ of length δ_2. Moreover, we define the two singletons $I_{\omega_1+1}^1 = \left\{\varphi_{JS}\left(\mathcal{I}''\right)\right\}$ and $I_{\omega_2+1}^2 = \{T_1\}$. Our algorithm APS'_ε generates reduced sets $\mathcal{X}_j^{\#}$ of states $[t, f]$ where t is the total processing time of jobs assigned before T_1 in the associated partial schedule and f is the maximum lateness of the same partial schedule. It can be described as follows:

Algorithm APS'_ε

(i). set $\mathcal{X}_1^{\#} = \{[0, T_2 + p_1 + q_1], [p_1, p_1 + q_1]\}$.

(ii). For $j \in \{2, 3, \ldots, \overline{n}\}$,

$\mathcal{X}_j^{\#} = \emptyset$.

For every state $[t, f]$ in $\mathcal{X}_{j-1}^{\#}$:

(1) Put $\left[t, \max\left\{f, T_2 + \sum_{i=1}^{j} p_i - t + q_j\right\}\right]$ in $\mathcal{X}_j^{\#}$

(2) Put $\left[t + p_j, \max\left\{f, t + p_j + q_j\right\}\right]$ in $\mathcal{X}_j^{\#}$ if $t + p_j \leq T_1$

Remove $\mathcal{X}_{j-1}^{\#}$

Let $[t, f]_{m,s}$ be the state in $\mathcal{X}_j^{\#}$ such that $f \in I_m^1$ and $t \in I_s^2$ with the smallest possible t (ties are broken by choosing the state of the smallest f).

Set $\mathcal{X}_j^{\#} = \left\{[t, f]_{m,s} \,|\, 1 \leq m \leq \omega_1 + 1, 1 \leq s \leq \omega_2 + 1\right\}$.

(iii). $\varphi_{APS'_\varepsilon}\left(\mathcal{I}''\right) = \min_{[t,f]\in\mathcal{X}_{\overline{n}}^{\#}}\{f\}$.

Theorem 5 *Given an arbitrary $\varepsilon > 0$, Algorithm APS'_ε yields an output $\varphi_{APS'_\varepsilon}\left(\mathcal{I}''\right)$ such that:*

$$\varphi_{APS'_\varepsilon}\left(\mathcal{I}''\right) - \varphi^*\left(\mathcal{I}''\right) \leq \varepsilon\varphi^*\left(\mathcal{I}''\right). \tag{2}$$

Proof The proof is similar to Kacem (2009). For self-consistency, a sketch proof is given in Appendix 1. □

Lemma 6 *Given an arbitrary $\varepsilon > 0$, algorithm APS'_ε can be implemented in $O\left(\overline{n}\log\overline{n} + \overline{n}^3/\varepsilon^2\right)$ time.*

Proof See Appendix 2. □

The schedule obtained by APS'_ε for instance $\mathcal{I}''$ can be easily converted into a feasible one for instance $\mathcal{I}$. This can be done in $O(n)$ time. From the previous lemma and the proof of Lemma 13, the main result is proved and the following theorem holds.

Theorem 7 *Algorithm APS'_ε is an FPTAS and it can be implemented in $O(n\log n + \min\{n, 3/\varepsilon\}^3/\varepsilon^2)$ time.*

3 Case under operator non-availability interval

Here, the studied problem (Π) can be formulated as follows. An operator has to schedule a set J of n jobs on a single machine, where every job j has a processing time p_j and a tail q_j. The machine can process at most one job at a time if the operator is available at the starting time and the completion time of such a job. The operator is unavailable during (T_1, T_2). Preemption of jobs is not allowed (jobs have to be performed under the non-resumable scenario). All jobs are ready to be performed at time 0. With no loss of generality, we consider that all data are integers and that jobs are indexed according to Jackson's rule and we assume that $q_1 \geq q_2 \geq \cdots \geq q_n$. Let $C_j(S)$ denote the completion time of job j in a feasible schedule S ($C_j(S) \notin (T_1, T_2)$ and $C_j(S) - p_j \notin (T_1, T_2)$) and let $\varphi_S(\mathcal{I})$ be the maximum lateness yielded by schedule S for instance $\mathcal{I}$ of (Π):

$$\varphi_S(\mathcal{I}) = \max_{1 \leq j \leq n} \left(C_j(S) + q_j \right) \tag{3}$$

The aim is to find a feasible schedule S by minimizing the maximum lateness.

If all the jobs can be inserted before T_1, the instance studied ($\mathcal{I}$) has obviously a trivial optimal solution obtained by Jackson's rule. We therefore consider only the problems in which all the jobs cannot be scheduled before T_1. Moreover, we consider that every job can be inserted before T_1 (i.e., $p_j \leq T_1$ for every $j \in J$).

In the remainder of this paper $\varphi^*(\mathcal{I})$ denotes the minimal maximum lateness for instance $\mathcal{I}$.

Proposition 8 *If $p_j < T_2 - T_1$ for every $j \in J$, then problem (Π) has an FPTAS.*

Proof In this case, it is easy to remark that Problem Π is equivalent to Problem $\mathcal{P}$ for which we can apply the FPTAS described in the previous section. □

In the remainder, we consider the hard case where some jobs have processing times greater than $T_2 - T_1$. Let $\mathcal{K}$ be the subset of these jobs. In this case, two scenarios are possible:

- Scenario 1: there exists a job $s \in \mathcal{K}$ such that in the optimal solution it starts before T_1 and completes after T_2 (s is called the stradling job).
- Scenario 2: there is no stradling job in the optimal solution.

It is obvious that Scenario 2 is equivalent to Problem $\mathcal{P}$ for which we have an FPTAS. Thus, the last step necessary to prove the existence of an FPTAS for Problem Π is to construct a special scheme for Scenario 1. Without loss of generality, we assume that the stradling job s is known (indeed, it can be guessed among jobs of $\mathcal{K}$). The following proposition determines the time-window of the starting time of job s in the optimal solution.

Proposition 9 *Let t_s^* be the starting time of s in the optimal schedule. The following relation holds: $t_s^* \in [T_2 - p_s, T_1]$.*

Proof Obvious since the stradling job s has to cover the ONA period in the optimal schedule. □

Proposition 10 *Senario 1 has an FPTAS.*

Proof The stradling job s is assumed to be known. Given an arbitrary $\varepsilon > 0$, we divide the interval $[T_2 - p_s, T_1]$ in $\lceil 1/\varepsilon \rceil$ equal-length sub-intervals $\bigcup_{h=1}^{\lceil 1/\varepsilon \rceil} D_h$ where

$$D_h = \left[T_2 - p_s + (h-1)\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil}, T_2 - p_s + h\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil} \right].$$

We consider a set of $\lceil 1/\varepsilon \rceil + 1$ instances $\{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_{\lceil 1/\varepsilon \rceil + 1}\}$ of Problem $\mathcal{P}$ where in $\mathcal{I}_h$ the stradling job starts at time

$$t_s^h = T_2 - p_s + (h-1)\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil}$$

which is equivalent to an instance of Problem $\mathcal{P}$ with a set of jobs $J - \{s\}$ and a MNA period Δ_h:

$$\Delta_h = \left(T_2 - p_s + (h-1)\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil}, T_2 + (h-1)\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil} \right).$$

For every instance from $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{\lceil 1/\varepsilon \rceil + 1}\}$, we apply the FPTAS described in the previous section for Problem $\mathcal{P}$ and we select the best solution among all the $\lceil 1/\varepsilon \rceil + 1$ instances. It is easy to see that if $t_s^* \in [t_s^h, t_s^{h+1})$ then, delaying s and the next jobs in the optimal schedule of $\mathcal{I}_{h+1}$ $(h = 1, 2, \ldots, \lceil 1/\varepsilon \rceil)$ by setting $t_s^* = t_s^{h+1}$ will not cost more than

$$\begin{aligned} \frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil} &\leq \varepsilon\,(T_1 - T_2 + p_s) \\ &\leq \varepsilon p_s \end{aligned}$$

Thus, the solution Ω_{h+1} obtained by APS'_ε for $\mathcal{I}_{h+1} (h = 1, 2, ..., \lceil 1/\varepsilon \rceil)$ is sufficiently close to optimal schedule for Scenario 1 if s is the stradling job and $t_s^* \in [t_s^h, t_s^{h+1})$. As a conclusion, Scenario 1 has an FPTAS. □

Theorem 11 *Problem* Π *admits an FPTAS and this scheme can be implemented in* $O\left(n^2/\varepsilon + n \min\{n, 3/\varepsilon\}^3/\varepsilon^3\right)$ *time.*

Proof The proof is a direct deduction from all the cases mentioned in this section. □

Remark 12 By applying the same approach, a PTAS can be elaborated for Problem Π and it can be implemented in $O(n^2/\varepsilon + (n/\varepsilon^2)2^{O(1/\varepsilon)})$ time.

4 Conclusion

In this paper, we considered the non-resumable case of the single machine scheduling problem with a non-availability interval. Our aim is to minimize the maximum lateness

when every job has a positive tail. Two cases are considered. In the first one, the non-availability interval is due to the machine maintenance. In the second case, the non-availibility interval is related to the operator who is organizing the execution of jobs on the machine. The contribution of this paper consists in an improved FPTAS for the MNA interval case and the elaboration of the first FPTAS for the operator non-availability interval case. The two FPTASs are strongly polynomial.

As future perspectives, we aim to consider other criteria for the single-machine problem as well as the study of multiple operator non-availablility periods.

Acknowledgments The authors would like to thank the referees and the editors for their helpful remarks and suggestions. This work has been funded by the CONSEIL REGIONAL DE LORRAINE (under the Programme "Chercheur d'Excellence 2013").

Appendix 1: Proof of Theorem 5

First, we recall the idea of the dynamic programming algorithm which is necessary to explain the proof. Indeed, the problem can be optimally solved by applying the following dynamic programming algorithm APS. This algorithm generates iteratively some sets of states. At every iteration j, a set $\mathcal{X}_j$ composed of states is generated ($1 \leq j \leq \overline{n}$). Each state $[t, f]$ in $\mathcal{X}_j$ can be associated to a feasible schedule for the first j jobs. Variable t denotes the completion time of the last job scheduled before T_1 and f is the maximum lateness of the corresponding schedule. This algorithm can be described as follows:

Algorithm APS

(i). Set $\mathcal{X}_1 = \{[0, T_2 + p_1 + q_1], [p_1, p_1 + q_1]\}$.

(ii). For $j \in \{2, 3, \ldots, \overline{n}\}$,

$\mathcal{X}_j = \{\}$.

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

(1) Put $\left[t, \max\left\{f, T_2 + \sum_{i=1}^{j} p_i - t + q_j\right\}\right]$ in $\mathcal{X}_j$

(2) Put $\left[t + p_j, \max\left\{f, t + p_j + q_j\right\}\right]$ in $\mathcal{X}_j$ if $t + p_j \leq T_1$

Remove $\mathcal{X}_{j-1}$

(iii). $\varphi^*(\mathcal{P}) = \min_{[t,f] \in \mathcal{X}_{\overline{n}}} \{f\}$.

Let UB be an upper bound on the optimal maximum lateness for problem $\left(\mathcal{T}''\right)$. If we add the restriction that for every state $[t, f]$ the relation $f \leq UB$ must hold, then the running time of APS can be bounded by $\overline{n} T_1 UB$. Indeed, t and f are integers and at each step j, we have to create at most $T_1 UB$ states to construct $\mathcal{X}_j$. Moreover, the complexity of APS is proportional to $\sum_{k=1}^{\overline{n}} |\mathcal{X}_k|$. In the remainder of the paper, Algorithm APS denotes the version of the dynamic programming algorithm by taking $UB = \varphi_{JS}\left(\mathcal{T}''\right)$.

The main idea of the FPTAS is to remove a special part of the states generated by the algorithm. Therefore, the modified algorithm APS'_ε becomes faster and yields an approximate solution instead of the optimal schedule. The approach of *modifying the execution of an exact algorithm* to design FPTAS, was initially proposed by Ibarra and Kim for solving the knapsack problem (Ibarra and Kim 1975). It is noteworthy

that during the last decades numerous combinatorial problems have been addressed by applying such an approach [for instance, see Sahni (1976) and Gens and Levner (1981)]. The worst-case analysis of our FPTAS is based on the comparison of the execution of algorithms APS and APS'_ε as described in the following lemma.

Lemma 13 *For every state* $[t, f]$ *in* $\mathcal{X}_j$ *there exists a state* $\left[t^{\#}, f^{\#}\right]$ *in* $\mathcal{X}_j^{\#}$ *such that:*

$$t^{\#} \leq t \leq t^{\#} + j\delta_2 \tag{4}$$

and

$$f^{\#} \leq f + j \max\{\delta_1, \delta_2\} \tag{5}$$

Proof By induction on j.
First, for $j = 1$ we have $\mathcal{X}_1^{\#} = \mathcal{X}_1$. Therefore, the statement is trivial.
Now, assume that the statement holds true up to level $j-1$. Consider an arbitrary state $[t, f] \in \mathcal{X}_j$. Algorithm APS introduces this state into $\mathcal{X}_j$ when job j is added to some feasible state for the first $j-1$ jobs. Let $\left[t', f'\right]$ be the above feasible state. Two cases can be distinguished: either $[t, f] = \left[t' + p_j, \max\left\{f', t' + p_j + q_j\right\}\right]$ or $[t, f] = \left[t', \max\left\{f', T_2 + \sum_{i=1}^{j} p_i - t' + q_j\right\}\right]$ must hold. We will prove the statement for level j in the two cases.

1st case: $[t, f] = \left[t' + p_j, \max\left\{f', t' + p_j + q_j\right\}\right]$
Since $\left[t', f'\right] \in \mathcal{X}_{j-1}$, there exists $\left[t'^{\#}, f'^{\#}\right] \in \mathcal{X}_{j-1}^{\#}$ such that $t'^{\#} \leq t' \leq t'^{\#} + (j-1)\,\delta_2$ and $f'^{\#} \leq f' + (j-1)\max\{\delta_1, \delta_2\}$. Consequently, the state $\left[t'^{\#} + p_j, \max\left\{f'^{\#}, t'^{\#} + p_j + q_j\right\}\right]$ is generated by Algorithm APS'_ε at iteration j. However it may be removed when reducing the state subset. Let $[\lambda, \mu]$ be the state in $\mathcal{X}_j^{\#}$ that is in the same box as $\left[t'^{\#} + p_j, \max\left\{f'^{\#}, t'^{\#} + p_j + q_j\right\}\right]$. Hence, we have:

$$\lambda \leq t'^{\#} + p_j \leq t' + p_j = t \tag{6}$$

Moreover,

$$\lambda + \delta_2 \geq t'^{\#} + p_j \geq t' - (j-1)\,\delta_2 + p_j = t - (j-1)\,\delta_2$$

which implies

$$t \leq \lambda + j\delta_2 \tag{7}$$

Finally,

$$\begin{aligned}
\mu &\leq \max\left\{f'^{\#}, t'^{\#} + p_j + q_j\right\} + \delta_1 \\
&\leq \max\left\{f' + (j-1)\max\{\delta_1, \delta_2\}, t' + p_j + q_j\right\} + \delta_1 \\
&\leq \max\left\{f', t' + p_j + q_j\right\} + (j-1)\max\{\delta_1, \delta_2\} + \delta_1 \\
&< f + j\max\{\delta_1, \delta_2\}.
\end{aligned} \tag{8}$$

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

2nd case: $[t, f] = \left[t', \max\left\{f', T_2 + \sum_{i=1}^{j} p_i - t' + q_j\right\}\right]$
Since $\left[t', f'\right] \in \mathcal{X}_{j-1}$, there exists $\left[t'^{\#}, f'^{\#}\right] \in \mathcal{X}_{j-1}^{\#}$ such that $t'^{\#} \leq t' \leq t'^{\#} + (j-1)\,\delta_2$ and $f'^{\#} \leq f' + (j-1)\max\{\delta_1, \delta_2\}$. Consequently, the state $\left[t'^{\#}, \max\left\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\right\}\right]$ is generated by algorithm APS'_{ε} at iteration j. However it may be removed when reducing the state subset. Let $\left[\lambda', \mu'\right]$ be the state in $\mathcal{X}_j^{\#}$ that is in the same box as $[t'^{\#}, \max\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\}]$. Hence, we have:

$$\lambda' \leq t'^{\#} \leq t' = t \tag{9}$$

Moreover,

$$\lambda' + \delta_2 \geq t'^{\#} \geq t' - (j-1)\,\delta_2 = t - (j-1)\,\delta_2$$

which implies

$$t \leq \lambda + j\delta_2 \tag{10}$$

and

$$\mu' \leq \max\left\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\right\} + \delta_1 \tag{11}$$

$$\leq \max\left\{f' + (j-1)\max\{\delta_1, \delta_2\}, T_2 + \sum_{i=1}^{j} p_i - t' + (j-1)\,\delta_2 + q_j\right\} + \delta_1 \tag{12}$$

$$\leq \max\left\{f', T_2 + \sum_{i=1}^{j} p_i - t' + q_j\right\} + (j-1)\max\{\delta_1, \delta_2\} + \delta_1 \tag{13}$$

$$\leq f + j\max\{\delta_1, \delta_2\}. \tag{14}$$

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

Now, we give the proof of Eq. (2) in Theorem 5. By definition, the optimal solution can be associated to a state $\left[t^*, f^*\right]$ in $\mathcal{X}_{\overline{n}}$. From Lemma 13, there exists a state $\left[t^{\#}, f^{\#}\right]$ in $\mathcal{X}_{\overline{n}}^{\#}$ such that:

$$\begin{aligned} f^{\#} &\leq f^* + \overline{n}\max\{\delta_1, \delta_2\} \\ &= f^* + \overline{n}\max\left\{\frac{\varphi_{JS}\left(\mathcal{I}''\right)}{\omega_1}, \frac{T_1}{\omega_2}\right\} \\ &= f^* + n\max\{\frac{\varphi_{JS}\left(\mathcal{I}''\right)}{\left\lceil\frac{2\overline{n}}{\varepsilon}\right\rceil}, \frac{T_1}{\left\lceil\frac{\overline{n}}{\varepsilon}\right\rceil}\} \\ &\leq f^* + \max\left\{\varepsilon\frac{\varphi_{JS}\left(\mathcal{I}''\right)}{2}, \varepsilon T_1\right\} = (1+\varepsilon)\,\varphi^*\left(\mathcal{I}''\right). \end{aligned} \tag{15}$$

Since $\varphi_{APS'_{\varepsilon}}\left(\mathcal{I}''\right) \leq f^{\#}$, we conclude that Equation (5) holds.

Appendix 2: Proof of Lemma 6

The first step consists in applying heuristic JS, which can be implemented in $O(\overline{n}\ln\overline{n})$ time. In the second step, algorithm APS'_ε generates the state sets $\mathcal{X}_j^{\#}$ ($j \in \{1, 2, \ldots, \overline{n}\}$). Since $\left|\mathcal{X}_j^{\#}\right| \leq (\omega_1 + 1)(\omega_2 + 1)$, we deduce that

$$\sum_{j=1}^{\overline{n}} \left|\mathcal{X}_j^{\#}\right| \leq \overline{n}(\omega_1 + 1)(\omega_2 + 1) = \overline{n}\left(\left\lceil\frac{\overline{n}}{\varepsilon}\right\rceil + 1\right)\left(\left\lceil\frac{2\overline{n}}{\varepsilon}\right\rceil + 1\right)$$
$$\leq \overline{n}\left(\frac{\overline{n}}{\varepsilon} + 2\right)\left(\frac{2\overline{n}}{\varepsilon} + 2\right). \tag{16}$$

Note that algorithm APS'_ε generates $\mathcal{X}_j^{\#}$ by associating every new created state to its corresponding box if and only if such a state has a smaller value of t (in this case, the last state associated to this box will be removed). Otherwise, the new created state will be immediately removed. This allows us to generate $\mathcal{X}_j^{\#}$ in $O(\omega_1\omega_2)$ time. Hence, our method can be implemented in $O\left(\overline{n}\ln\overline{n} + \overline{n}^3/\varepsilon^2\right)$ time and this completes the proof.

References

Brauner N, Finke G, Kellerer H, Lebacque V, Rapine C, Potts C, Strusevich V (2009) Operator non-availability periods. 4OR 7:239–253

Carlier J (1982) The one-machine sequencing problem. Eur J Oper Res 11:42–47

Chen Y, Zhang A, Tan Z (2013) Complexity and approximation of single machine scheduling with an operator non-availability period to minimize total completion time. Inf Sci 251:150–163

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

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

He Y, Zhong W, Gu H (2006) Improved algorithms for two single machine scheduling problems. Theor Comput Sci 363:257–265

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

Kacem I (2009) Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. J Comb Optim 17(2):117–133

Kacem I, Kellerer H (2014) Approximation algorithms for no idle time scheduling on a single machine with release times and delivery times. Discret Appl Math 164(1):154–160

Kubzin MA, Strusevich VA (2006) Planning machine maintenance in two machine shop scheduling. Oper Res 54:789–800

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

Qi X (2007) A note on worst-case performance of heuristics for maintenance scheduling problems. Discret Appl Math 155:416–422

Qi X, Chen T, Tu F (1999) Scheduling the maintenance on a single machine. J Oper Res Soc 50:1071–1078

Rapine C, Brauner N, Finke G, Lebacque V (2012) Single machine scheduling with small operator-non-availability periods. J Sched 15:127–139

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

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

Yuan JJ, Shi L, Ou JW (2008) Single machine scheduling with forbidden intervals and job delivery times. Asia-Pac J Oper Res 25(3):317–325