Improved Fully Polynomial Approximation Schemes for the Maximum Lateness Minimization on a Single Machine with a Fixed Operator or Machine Non-Availability Interval

Imed Kacem[1(✉)] and Hans Kellerer[2]

[1] Université de Lorraine, LCOMS EA 7306, 57000 Metz, France
imed.kacem@univ-lorraine.fr
[2] Institut für Statistik und Operations Research, University of Graz, Universitätsstraße 15, 8010 Graz, Austria
hans.kellerer@uni-graz.at
http://lcoms.univ-lorraine.fr, http://www.uni-graz.at

Abstract. In this paper we consider 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. The contribution of this paper consists in an improved FPTAS for the maintenance non-availability interval case and its extension to the operator non-availability interval case. The two FPTASs are strongly polynomial and outperform the recent ones by Kacem, Kellerer and Seifaddini presented in [12].

Keywords: Scheduling · Approximation Schemes · FPTAS
Maximum lateness minimization · Single machine
Non-availability interval · Dynamic programming

1 Introduction

In this paper we investigate some improvements to the previous work by Kacem, Kellerer and Seifaddini [12]. We consider 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

Supported by the LCOMS EA 7306, a research unit of the Université de Lorraine, and by the University of Graz.

R. Cerulli et al. (Eds.): ICCL 2018, LNCS 11184, pp. 417–427, 2018.
https://doi.org/10.1007/978-3-030-00898-7_28

non-availibility interval is related to the operator who is organizing the execution of jobs on the machine. An operator 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 machine non-availability interval is a completely forbidden period. Rapine *et al.* [18] 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 study of this family of scheduling problems has been motivated by different applications in logistics. First, the production scheduling by integrating the different sources of non-availability (machine and/or operator sources) is an important application (see for example, Brauner *et al.* [1], Rapine *et al.* [18]). Moreover, the maximum lateness minimization in scheduling theory is known as equivalent to the maximum delivery date minimization, where the delivery times can be seen as transportation durations (see for example, Carlier [2], Kacem and Kellerer [9], Dessouky and Margenthaler [4]). Other online applications can be found in Kacem and Kellerer [11].

2 Related Works

The MNA case of this type of problems has been studied in the literature under various criteria (a sample of these works includes Lee [15], Kacem [8], Kacem *et al.* [13], Kubzin and Strusevich [14], Qi *et al.* [16,17], Schmidt [19], He *et al.* [6]). However, few papers studied the problem we consider in this paper. Lee [15] explored the Jackson's sequence and proved that it is a 2-approximation. Recently, Yuan *et al.* developed an interesting PTAS (Polynomial Approximation Scheme) for the studied problem [21]. Kacem [8] presented a first Fully Polynomial Time Approximation Scheme (FPTAS) for the maximum lateness minimization. It is well-known that an FPTAS is the best possible approximation scheme for an NP-hard problem, unless $P = NP$ (see for example, Kacem and Kellerer [10], Gens and Levner [5], Ibarra and Kim [7], Sahni [20]). 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.* [1] 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 the so-called 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.* [18]. They proved that the problem can be solved in polynomial time if there is only one ONA period. It was shown that 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 $\mathrm{P}=\mathrm{NP}$.

Recently, Chen *et al.* [3] 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 $\frac{5}{3}$.

For more details, the previous paper by Kacem, Kellerer and Seifaddini [12] contains an overview on these problems.

3 Contributions

The contribution of this paper consists in an improved FPTAS for the maintenance non-availability interval case and its extension for the ONA interval case. The two FPTASs are strongly polynomial and they have reduced time complexities compared to [12]. These contributions are summarized in Table 1 for both cases.

This note is organized as follows. Section 4 recalls the exact formulation of the maintenance non-availability interval case and the improved FPTAS. Section 5 is devoted to the extension to the operator non-availability interval case and to the presentation of the associated FPTAS. Finally, Sect. 6 concludes the paper.

Table 1. Summary of results

	Result	Reference
MNA	FPTAS: $O(nlog(n)+\min\{n,1/\varepsilon\}^3/\varepsilon^2)$	Kacem et al. [12]
ONA	FPTAS: $O((n/\varepsilon)log(n)+n\min\{n,1/\varepsilon\}^3/\varepsilon^3)$	Kacem et al. [12]
MNA	FPTAS: $O(nlog(n)+\min\{n,1/\varepsilon\}^2/\varepsilon^2)$	This paper
ONA	FPTAS: $O((n/\varepsilon)log(n)+n\min\{n,1/\varepsilon\}^2/\varepsilon^3)$	This paper

4 Case Under MNA Interval

The considered 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 (or delivery time). 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 [15] (i.e., jobs are indexed in nonincreasing order of tails). Therefore, we assume that $q_1 \geq q_2 \geq \ldots \geq q_n$. Let $C_j(S)$ denote the completion time of job j in a feasible schedule S for the problem and let $\varphi_S(\mathcal{P})$ be the maximum lateness (or the delivery date) yielded by schedule S for instance $\mathcal{I}$ of $(\mathcal{P})$:

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

The aim is to find a feasible schedule S by minimizing the maximum lateness. We also denote by $\varphi^*(\mathcal{I})$ the minimal maximum lateness for instance $\mathcal{I}$. Due to the dominance of Jackson's order, an optimal schedule is composed of two sequences of jobs scheduled in nondecreasing order of their indexes [15]. 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 $j \in J$). It is useful to recall that Lee [15] explored the Jackson's sequence JS and proved that its deviation to the optimal maximum lateness cannot exceed the largest processing time, which is equivalent to state that JS is a 2-approximation.

4.1 The Improved Procedure

The proposed FPTAS is based on the modification of the one proposed in [12] by Kacem, Kellerer and Seifaddini.

First, as described in [12], we use the simplification technique based on merging small jobs proposed in [9]. We simplify the input instance $\mathcal{I}$ as follows. Given an arbitrary $\varepsilon > 0$, with the assumption that $1/\varepsilon$ is integer, we split the interval $[0, \max_{j \in J}\{q_j\}]$ in $1/\varepsilon$ equal length 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 $\mathcal{I}'$. Then, 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 less than $\varepsilon P/2$ where $P = \sum_{j=1}^{n} p_j$. 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 $3/\varepsilon$. These reductions are recalled for self-consistency and their details are available in Kacem *et al.* [12].

It is worthy to note that such reductions cannot increase the optimal solution value of $\mathcal{I}$ too much and they can be done in linear time.

We apply a modified dynamic programming algorithm DP_ε to instance $\mathcal{I}''$ using the Jackson's sequence JS to obtain an upper bound for the maximum lateness. The main idea of DP_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. First, we define the following parameters:

$$\overline{n} = \min\{n, 3/\varepsilon\},$$

$$\omega_1 = \left\lceil \frac{4\overline{n}}{\varepsilon} \right\rceil,$$

$$\omega_2 = \left\lceil \frac{2}{\varepsilon} \right\rceil,$$

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

and

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

We split $[0, \varphi_{JS}(\mathcal{I}''))$ into ω_1 equal subintervals $I_m^1 = [(m-1)\delta_1, m\delta_1)_{1 \le m \le \omega_1}$. We also split $[0, T_1)$ into ω_2 equal subintervals $I_s^2 = [(s-1)\delta_2, s\delta_2)_{1 \le s \le \omega_2}$ of length δ_2. Moreover, we define the two singletons $I_{\omega_1+1}^1 = \{\varphi_{JS}(\mathcal{I}'')\}$ and $I_{\omega_2+1}^2 = \{T_1\}$. Our algorithm DP_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 is described in Algorithm 1.

4.2 Algorithm DP_ε is an Improved FPTAS

Compared to the previous FPTAS presented in [12], our new algorithm keeps two approximate states in every box $I_m^1 \times I_s^2$ $(1 \le m \le \omega_1 + 1, 1 \le s \le \omega_2 + 1)$ instead of a single approximate state. As a consequence, the loss in terms of variable t will be reduced as it can be shown in the proof of the following theorem. Thus, the interval length δ_2 is taken larger compared to [12]. Moreover, in the following proof we will use a tighter recursive relation on the closeness of the approximate states and those originally generated by the standard dynamic algorithm. As a result, we will show that the new algorithm DP_ε outperforms the one provided in [12] by a linear factor in n or $1/\varepsilon$.

Theorem 1. *Given an arbitrary $\varepsilon > 0$, the modified algorithm DP_ε yields an output $\varphi_{DP_\varepsilon}(\mathcal{I}'')$ such that:*

$$\varphi_{DP_\varepsilon}(\mathcal{I}'') - \varphi^*(\mathcal{I}'') \le \varepsilon \varphi^*(\mathcal{I}''). \tag{2}$$

Algorithm 1. The new proposed FPTAS DP_ε

The inputs of the algorithm are: ε, T_1, T_2, $\overline{n}$ *and simplified instance* $\mathcal{I}''$. *The algorithm returns a feasible schedule with a maximum lateness value less or equal to* $(1+\varepsilon)\varphi^*(\mathcal{I}'')$.

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 $[t + p_j, \max\{f, t + p_j + q_j\}]$ in $\mathcal{X}_j^{\#}$ if $t + p_j \leq T_1$
Remove $\mathcal{X}_{j-1}^{\#}$
Let $[t, f]_{m,s}$ and $[u, g]_{m,s}$ be the states in $\mathcal{X}_j^{\#}$ such that $f, g \in I_m^1$, $t, u \in I_s^2$ and t and u are respectively the smallest and the greatest possible values in subinterval I_s^2.
Set $\mathcal{X}_j^{\#} = \left\{[t, f]_{m,s}, [u, g]_{m,s} \,|1 \leq m \leq \omega_1 + 1, 1 \leq s \leq \omega_2 + 1\right\}$.
iii. $\varphi_{DP_\varepsilon}(\mathcal{I}'') = \min_{[t,f] \in \mathcal{X}_{\overline{n}}^{\#}} \{f\}$.

Proof. First, we recall the idea of the dynamic programming algorithm [8] which is necessary to explain the proof. Indeed, the problem can be optimally solved by applying the following dynamic programming algorithm DP. 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 dynamic programming is given in Algorithm 2.

Algorithm 2. The standard dynamic programming DP [8]

The inputs of the algorithm are: T_1, T_2, $\overline{n}$ *and simplified instance* $\mathcal{I}''$. *The algorithm returns a schedule with an optimal maximum lateness value* $\varphi^*(\mathcal{I}'')$.

(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 $[t + p_j, \max\{f, t + p_j + q_j\}]$ 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 $UB = \varphi_{JS}(\mathcal{T}'')$ be an upper bound on the optimal maximum lateness for problem $(\mathcal{T}'')$ obtained by Jackson's sequence. We add the restriction that for every state $[t, f]$ the relation $f \leq UB$ must hold.

The main idea of the FPTAS is to remove a special part of the states generated by the dynamic programming algorithm. Therefore, the modified algorithm DP_ε becomes faster and yields an approximate solution instead of the optimal schedule. The worst-case analysis of our FPTAS is based on the comparison of the execution of algorithms DP and DP_ε, which can be summarized by the following relations. For every state $[t, f]$ in $\mathcal{X}_j$ there exists a state $\left[t^{\#}, f^{\#}\right]$ in $\mathcal{X}_j^{\#}$ such that:

$$t - \delta_2 \leq t^{\#} \leq t \tag{3}$$

and

$$f^{\#} \leq f + \delta_2 + j\delta_1 \tag{4}$$

The two relations can be proved 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 DP introduces this state into $\mathcal{X}_j$ when job j is added to some feasible state for the first $j-1$ jobs. Let $[t', f']$ be the above feasible state. Two cases can be distinguished: either $[t, f] = [t' + p_j, \max\{f', t' + p_j + q_j\}]$ or $[t, f] = \left[t', \max\left\{f', T_2 + \sum_{i=1}^{j} p_i - t' + q_j\right\}\right]$ must hold. For proving the statement for level j we will distinguish two cases.

Case 1: $[t, f] = [t' + p_j, \max\{f', t' + p_j + q_j\}]$
Since $[t', f'] \in \mathcal{X}_{j-1}$, there exists $\left[t'^{\#}, f'^{\#}\right] \in \mathcal{X}_{j-1}^{\#}$ such that $t' - \delta_2 \leq t'^{\#} \leq t'$ and $f'^{\#} \leq f' + \delta_2 + (j-1)\,\delta_1$.

Consequently, the state $\left[t'^{\#} + p_j, \max\left\{f'^{\#}, t'^{\#} + p_j + q_j\right\}\right]$ is feasible (since $t'^{\#} + p_j \leq t' + p_j = t \leq T_1$) and it is generated by Algorithm DP_ε at iteration j. However it may be removed when reducing the state subset. Let $[\lambda, \mu]$ and $[\alpha, \beta]$ be the two possible states in set $\mathcal{X}_j^{\#}$ that remain in the same box as the state $\left[t'^{\#} + p_j, \max\left\{f'^{\#}, t'^{\#} + p_j + q_j\right\}\right]$ (with $\lambda \leq t'^{\#} + p_j \leq \alpha$). Hence, we have two situations to consider: $t' + p_j \geq \alpha$ or $t' + p_j < \alpha$.

Subcase 1.a: $t' + p_j \geq \alpha$. In this subcase, we can verify that the state $[\alpha, \beta]$ fulfills (3). Indeed, $\alpha \leq t' + p_j = t$ and by definition $\alpha \geq t'^{\#} + p_j \geq t' - \delta_2 + p_j = t - \delta_2$. Thus, we have

$$t - \delta_2 \leq \alpha \leq t.$$

Subcase 1.b: $t' + p_j < \alpha$. In this subcase, we can verify that the state $[\lambda, \mu]$ fulfills (3). Indeed, $\lambda \leq t'^{\#} + p_j \leq t' + p_j = t$ and by definition $\lambda \geq \alpha - \delta_2 > t' + p_j - \delta_2 = t - \delta_2$. Thus, we have

$$t - \delta_2 \leq \lambda \leq t.$$

On the other hand, the two values μ and β are in the same subinterval as the value $\max\left\{f'^{\#}, t'^{\#} + p_j + q_j\right\}$. Therefore, the kept state will have a maximum

lateness value less or equal to $\max\{\mu, \beta\}$. Then, we conclude that

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

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

Case 2: $[t, f] = \left[t', \max\left\{f', T_2 + \sum_{i=1}^{j} p_i - t' + q_j\right\}\right]$

Since $[t', f'] \in \mathcal{X}_{j-1}$, there exists $\left[t'^{\#}, f'^{\#}\right] \in \mathcal{X}_{j-1}^{\#}$ such that $t' - \delta_2 \le t'^{\#} \le t'$ and $f'^{\#} \le f' + \delta_2 + (j-1)\,\delta_1$.

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 DP_ε in iteration j. However, it may be removed when reducing the state subset. Let $[\lambda', \mu']$ and $[\alpha', \beta']$ be the states in $\mathcal{X}_j^{\#}$ that are kept in the same box as $[t'^{\#}, \max\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\}]$ and having $\lambda' \le t'^{\#} \le \alpha'$. Again, we have two situations to be considered: $t' \ge \alpha'$ or $t' < \alpha'$.

Subcase 2.a: $t' \ge \alpha'$. In this subcase, we can verify that the state $[\alpha', \beta']$ fulfills (3). Indeed, $\alpha' \le t' = t$ and by definition $\alpha' \ge t'^{\#} \ge t' - \delta_2 = t - \delta_2$. Thus, we have

$$t - \delta_2 \le \alpha' \le t.$$

Subcase 2.b: $t' < \alpha'$. In this subcase, we can verify that the state $[\lambda', \mu']$ fulfills (3). Indeed, $\lambda' \le t'^{\#} \le t' = t$ and by definition $\lambda' \ge \alpha' - \delta_2 > t' - \delta_2 = t - \delta_2$. Thus, we have

$$t - \delta_2 \le \lambda' \le t.$$

On the other hand, the values μ' and β' are in the same subinterval as $\max\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\}$. Therefore, the kept state will have a maximum lateness value less or equal to $\max\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\} + \delta_1$. Moreover,

$$\max\left\{f'^{\#}, T_2 + \sum_{i=1}^{j} p_i - t'^{\#} + q_j\right\} + \delta_1 \le \max\{X, Y\} + \delta_1$$

where $X = f' + \delta_2 + (j-1)\,\delta_1$ and $Y = T_2 + \sum_{i=1}^{j} p_i - t' + \delta_2 + q_j$. Thus,

$$\begin{aligned}\max\{\mu',\beta'\} &\le \max\left\{f', T_2 + \sum_{i=1}^{j} p_i - t' + q_j\right\} + \max\{\delta_2 + j\delta_1, \delta_2 + \delta_1\} \\ &\le f + \delta_2 + j\delta_1.\end{aligned}$$

In conclusion, the statement holds also for level j in the second case, and this completes our inductive proof. Now, we give the proof of Eq. (2). By definition,

the optimal solution can be associated to a state $[t^*, f^*]$ in $\mathcal{X}_{\overline{n}}$. From Eq. (4), there exists a state $\left[t^{\#}, f^{\#}\right]$ in $\mathcal{X}_{\overline{n}}^{\#}$ such that:

$$\begin{aligned} f^{\#} &\leq f^* + \delta_2 + \overline{n}\delta_1 \\ &= f^* + \frac{T_1}{\omega_2} + \overline{n}\frac{\varphi_{JS}\left(\mathcal{I}''\right)}{\omega_1} \\ &= f^* + \frac{T_1}{\left\lceil \frac{2}{\varepsilon} \right\rceil} + \overline{n}\frac{\varphi_{JS}\left(\mathcal{I}''\right)}{\left\lceil \frac{4\overline{n}}{\varepsilon} \right\rceil} \\ &\leq f^* + \varepsilon\frac{T_1}{2} + \varepsilon\frac{\varphi_{JS}\left(\mathcal{I}''\right)}{4} \\ &\leq (1+\varepsilon)\,\varphi^*\left(\mathcal{I}''\right). \end{aligned}$$

Since $\varphi_{DP_\varepsilon}\left(\mathcal{I}''\right) \leq f^{\#}$, we conclude that Eq. (2) holds.

It can be easily seen that the proposed modified algorithm DP_ε can be implemented in $O\left(\overline{n}\log\overline{n} + \overline{n}^2/\varepsilon^2\right)$ time. The schedule obtained by DP_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.

To summarize, we conclude that Algorithm DP_ε is an FPTAS and it can be implemented in $O\left(n\log n + \min\{n, 1/\varepsilon\}^2/\varepsilon^2\right)$ time.

5 Consequences: An Improved FPTAS for the ONA Case

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. Without loss of generality, we consider that all data are integers and that jobs are indexed according to Jackson's rule. The aim is to find a feasible schedule S by minimizing the maximum lateness. Again, 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 $j \in J$).

As it has been done in Kacem *et al.* [12], an FPTAS can be established for Π. The procedure is based on guessing the so-called *straddling* job and its starting time from a finite set of approximate values, which leads to $O(\frac{n}{\varepsilon})$ auxiliary MNA problems ($\mathcal{P}$). Thus, the application of DP_ε to all these auxiliary problems can lead to an improved FPTAS as the following theorem claims:

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

Proof. The proof is a straightforward from [12] and Theorem 1.

6 Conclusion

In this paper we consider 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. The contribution of this paper consists in an improved FPTAS for the maintenance non-availability interval case and its extension to the operator non-availability interval case. The two FPTASs are strongly polynomial and outperform our previous ones published in the literature.

As a research perspective, we are extending the ideas of this paper in order to improve some existing FPTASs for other scheduling problems.

References

1. Brauner, N., et al.: Operator non-availability periods. 4OR: Q. J. Oper. Res. **7**, 239–253 (2009)
2. Carlier, J.: The one-machine sequencing problem. Eur. J. Oper. Res. **11**, 42–47 (1982)
3. Chen, Y., Zhang, A., Tan, Z.: Complexity and approximation of single machine scheduling with an operator non-availability period to minimize total completion time. Inf. Sci. **251**, 150–163 (2013)
4. Dessouky, M.I., Margenthaler, C.R.: The one-machine sequencing problem with early starts and due dates. AIIE Trans. **4**(3), 214–222 (1972)
5. Gens, G.V., Levner, E.V.: Fast approximation algorithms for job sequencing with deadlines. Discret. Appl. Math. **3**, 313–318 (1981)
6. He, Y., Zhong, W., Gu, H.: Improved algorithms for two single machine scheduling problems. Theor. Comput. Sci. **363**, 257–265 (2006)
7. Ibarra, O., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. J. ACM **22**, 463–468 (1975)
8. Kacem, I.: 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 (2009)
9. Kacem, I., Kellerer, H.: Approximation algorithms for no idle time scheduling on a single machine with release times and delivery times. Discret. Appl. Math. **164**(1), 154–160 (2014)
10. Kacem, I., Kellerer, H.: Approximation schemes for minimizing the maximum lateness on a single machine with release times under non-availability or deadline constraints. Algorithmica (2018) https://doi.org/10.1007/s00453-018-0417-6
11. Kacem, I., Kellerer, H.: Semi-online scheduling on a single machine with unexpected breakdown. Theor. Comput. Sci. **646**, 40–48 (2016)
12. Kacem, I., Kellerer, H., Seifaddini, M.: Efficient approximation schemes for the maximum lateness minimization on a single machine with a fixed operator or machine non-availability interval. J. Comb. Optim. **32**, 970–981 (2016)
13. Kacem, I., Sahnoune, M., Schmidt, G.: Strongly fully polynomial time approximation scheme for the weighted completion time minimisation problem on two-parallel capacitated machines. RAIRO - Oper. Res. **51**, 1177–1188 (2017)

14. Kubzin, M.A., Strusevich, V.A.: Planning machine maintenance in two machine shop scheduling. Oper. Res. **54**, 789–800 (2006)
15. Lee, C.Y.: Machine scheduling with an availability constraints. J. Glob. Optim. **9**, 363–384 (1996)
16. Qi, X.: A note on worst-case performance of heuristics for maintenance scheduling problems. Discret. Appl. Math. **155**, 416–422 (2007)
17. Qi, X., Chen, T., Tu, F.: Scheduling the maintenance on a single machine. J. Oper. Res. Soc. **50**, 1071–1078 (1999)
18. Rapine, C., Brauner, N., Finke, G., Lebacque, V.: Single machine scheduling with small operator-non-availability periods. J. Sched. **15**, 127–139 (2012)
19. Schmidt, G.: Scheduling with limited machine availability. Eur. J. Oper. Res. **121**, 1–15 (2000)
20. Sahni, S.: Algorithms for scheduling independent tasks. J. ACM **23**, 116–127 (1976)
21. Yuan, J.J., Shi, L., Ou, J.W.: Single machine scheduling with forbidden intervals and job delivery times. Asia-Pac. J. Oper. Res. **25**(3), 317–325 (2008)