Algorithmica (2018) 80:3825–3843
https://doi.org/10.1007/s00453-018-0417-6

Approximation Schemes for Minimizing the Maximum Lateness on a Single Machine with Release Times Under Non-availability or Deadline Constraints

Imed Kacem[1] · **Hans Kellerer**[2]

Received: 12 April 2017 / Accepted: 7 February 2018 / Published online: 20 February 2018

Abstract In this paper, we consider four single-machine scheduling problems with release times, with the aim of minimizing the maximum lateness. In the first problem we have a common deadline for all the jobs. The second problem looks for the Pareto frontier with respect to the two objective functions maximum lateness and makespan. The third problem is associated with a non-availability constraint. In the fourth one, the non-availability interval is related to the operator who is organizing the execution of jobs on the machine (no job can start, and neither can complete during the operator non-availability period). For each of the four problems, we establish the existence of a polynomial time approximation scheme.

Keywords Single machine scheduling · Release times · Lateness · Deadlines · Approximation algorithms

1 Introduction

The problem we consider is the one-machine scheduling problem with release dates and delivery times. The objective is to minimize the maximum lateness. Formally, the problem is defined in the following way. We have to schedule a set $J = \{1, 2, \ldots, n\}$ of n jobs on a single machine. Each job $j \in J$ has a processing time p_j, a release time

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

Hans Kellerer
hans.kellerer@uni-graz.at

[1] Laboratoire de Conception, Optimisation et Modélisation des Systèmes, LCOMS EA 7306, Université de Lorraine, Metz, France

[2] Institut für Statistik und Operations Research, ISOR, Universität Graz, Graz, Austria

(or head) r_j and a delivery time (or tail) q_j. The machine can only perform one job at a given time. Preemption is not allowed. The problem is to find a sequence of jobs, with the objective of minimizing the maximum lateness $L_{\max} = \max_{1\leq j\leq n}\left\{C_j + q_j\right\}$ where C_j is the completion time of job j. We also define s_j as the starting time of job j, i.e., $C_j = s_j + p_j$, then $P = \sum_{j=1}^{n} p_j$ as the total processing time and $p(H) = \sum_{j\in H} p_j$ as the total processing time for a subset of jobs $H \subset J$. Four scenarios are considered:

- Scenario 1: Every job should be completed before a deadline d, i.e., $C_{\max} = \max_{1\leq j\leq n}\{C_j\} \leq d$. This scenario is denoted as $1|r_j, C_{\max} \leq d|L_{\max}$. For a simpler notation, this variant is also abbreviated as Π_1.
- Scenario 2: We are given the one-machine bicriteria scheduling problem with the two objectives $L_{\max}$ and $C_{\max}$, denoted by $1|r_j|L_{\max}, C_{\max}$. We will speak shortly of problem Π_2.
- Scenario 3: The machine is not available during a given time interval $]T_1, T_2[$. This scenario is denoted by $1, h_1|r_j|L_{\max}$. We will speak shortly of problem Π_3. Here, $]T_1, T_2[$ is a machine non-availability (MNA) interval.
- Scenario 4: In this case, the non-availability interval $]T_1, T_2[$ is related to the operator who is organizing the execution of jobs on the machine. An operator non-availability (ONA) period is an open time interval in which no job can start, and neither can complete. Using an extended notation as in [11], this scenario is denoted by $1, ONA|r_j|L_{\max}$. We will speak shortly of problem Π_4.

Note that the main difference between machine non-availability and operator non-availability consists in the fact that a job can be processed but can neither start nor finish during the ONA period. However, the MNA interval is a completely forbidden period.

All four presented scenarios are generalizations of the well-known problem $1|r_j, q_j|L_{max}$ which has been widely studied in the literature. For the sake of simplicity, this problem will be denoted by Π_0. According to Lenstra et al. [16] problem $1|r_j, q_j|L_{max}$ is NP-hard in the strong sense. Therefore, we are interested in the design of efficient approximation algorithms for our problems. Note that the four studied scenarios are strongly related and as a consequence the different algorithms we will propose later are linked. For example, we will see in the remainder of the paper that the algorithm elaborated for problem Π_4 uses the one developed for problem Π_3, which recalls the algorithm proposed for problem Π_1. All these links are carefully explained in the different proofs of our algorithms.

For self-consistency, we recall some necessary definitions related to the approximation area. A *ρ-approximation algorithm* for a problem of minimizing an objective function φ is an algorithm such that for every instance π of the problem it gives a solution S_π verifying $\varphi(S_\pi)/\varphi(OPT_\pi) \leq \rho$ where OPT_π is an optimal solution of π . The value ρ is also called the *worst-case bound* of the above algorithm.

A class of $(1+\varepsilon)$-approximation algorithms is called a Polynomial Time Approximation Scheme (PTAS), if its running time is polynomial with respect to the length of the problem input for every $\varepsilon > 0$. A class of $(1+\varepsilon)$ -approximation algorithms is called a Fully Polynomial Time Approximation Scheme (FPTAS), if its running time

is polynomial with respect to both the length of the problem input and $1/\varepsilon$ for every $\varepsilon > 0$.

Given the aim of this paper, we summarize briefly related results on scheduling problems with non-availability constraints but no release times. A huge number of papers is devoted to such problems in the literature. This has been motivated by practical and real industrial problems, like maintenance problems or the occurrence of breakdown periods. For a survey we refer to the article by [15]. However, to the best of our knowledge, there is only a limited number of references related to the design of approximation algorithms in the case of maximum lateness minimization. Yuan et al [26] developed a PTAS for the problem without release dates. The paper by Kacem et al. [11] contains an FPTAS for $1, h_1||L_{\max}$ and $1, ONA||L_{\max}$. Kacem and Kellerer [9] consider different close semi-online variants ($1, h_1||C_{\max}$, $1, h_1|r_j|C_{\max}$ and $1, h_1||L_{\max}$) and propose approximation algorithms with effective competitive ratios.

Numerous works address problems with release times but without unavailability constraints. Most of the exact algorithms are based on enumeration techniques. See for instance the papers by Dessouky and Margenthaler [3], Carlier et al. [2], Larson et al. [14], Yin et al. [24,25] and Grabowski et al. [6].

Various approximation algorithms were also proposed. Most of these algorithms are based on variations of the extended Jackson's rule, also called Schrage's algorithm. Schrage's algorithm consists in scheduling ready jobs on the machine by giving priority to the one having the greatest tail. It is well-known that the Schrage sequence yields a worst-case performance ratio of 2. This was first observed by Kise et al. [13]. Potts [21] improves this result by running Schrage's algorithm at most n times to slightly varied instances. The algorithm of Potts has a worst-case performance ratio of $\frac{3}{2}$ and it runs in $O(n^2 \log n)$ time. Hall and Shmoys [7] showed that a modification of the algorithm of Potts has the same worst-case performance ratio under precedence constraints. Nowicki and Smutnicki [19] proposed a faster $\frac{3}{2}$-approximation algorithm with $O(n \log n)$ running time. By performing the algorithm of Potts for the original and the inverse problem and taking the best solution Hall and Shmoys [7] established the existence of a $\frac{4}{3}$-approximation. They also proposed two polynomial time approximation schemes. A more effective PTAS has been proposed by Mastrolilli [18] for the single-machine and parallel-machine cases. For more details the reader is invited to consult the survey by Kellerer [12].

There are only a few papers which treat the problem with both release times and non-availability intervals. Leon and Wu [17] present a branch-and-bound algorithm for the problem with release times and several machine non-availability intervals. Gharbi et al. [5] investigate the single machine scheduling problem with job release dates, due dates and multiple planned unavailability time periods. They propose a new lower bound and an exact algorithm for that problem. A 2-approximation has been established by Kacem and Haouari [8] for $1, h_1|r_j|L_{\max}$.

Rapine et al. [22] have described some applications of operator non-availability problems in the planning of a chemical experiments. Brauner et al. [1] have studied single-machine problems under the operator non-availability constraint. Finally, we refer to the book by T'kindt and Billaut [23] for an introduction on multicriteria scheduling.

For each of the four problems Π_1, Π_2, Π_3 and Π_4 we will present a polynomial time approximation scheme. Notice, that the PTAS's in [7] and [18] for $1|r_j, q_j|L_{max}$ use modified instances with only a constant number of release dates. This is not possible in the presence of deadlines or non-availability periods since the feasibility of a solution would not be guaranteed. Notice that the four investigated scenarios are strongly NP-hard since already the problem $1|r_j|L_{\max}$ is strongly NP-hard. Consequently, no FPTAS can be derived unless P = NP and we limit our investigation to the existence of an PTAS.

The paper is organized as follows. Section 2 repeats some notations and results for Schrage's sequence. Section 3 contains the PTAS for problem Π_1, Sect. 4 is devoted to the bicriteria scheduling problem Π_2, Sects. 5 and 6 deal with problems Π_3 and Π_4, respectively. Finally, Sect. 7 concludes the paper.

2 Schrage's Algorithm

For self-consistency we repeat in this section some notations and results for the Schrage's sequence, initially designed for problem Π_0 and later exploited for other extensions (see for instance Kacem and Kellerer [10]). Recall that Schrage's algorithm schedules the job with the greatest tail from the available jobs at each step. At the completion of such a job, the subset of the available jobs is updated and a new job is selected. The procedure is repeated until all jobs are scheduled. For a given instance I the sequence obtained by Schrage shall be denoted by $\sigma_{Sc}(I)$. Assume that jobs are indexed such that $\sigma_{Sc}(I) = (1, 2, \ldots, n)$. The job c which attains the maximum lateness in Schrage's schedule, is called the *critical job*. Then the maximum lateness of $\sigma_{Sc}(I)$ can be given as follows:

$$L_{max}(\sigma_{Sc}(I)) = \min_{j\in\Lambda}\{r_j\} + \sum_{j\in\Lambda} p_j + q_c = r_a + \sum_{j=a}^{c} p_j + q_c,$$

where job a is the first job so that there is no idle time between the processing of jobs a and c, i.e., either there is idle time before a or a is the first job to be scheduled. The sequence of jobs $a, a+1, \ldots, c$ is called the *critical path* in the Schrage schedule (or the *critical bloc* Λ). It is obvious that all jobs j in the critical path have release dates $r_j \geq r_a$.

Let $L^*_{max}(I)$ denote the optimal maximum lateness for a given instance I. We will write briefly L^*_{max} if it is clear from the context. For every subset of jobs $F \subset J$ the following useful lower bound is valid.

$$L^*_{max} \geq \min_{j\in F}\{r_j\} + \sum_{j\in F} p_j + \min_{j\in F}\{q_j\} \tag{1}$$

If c has the smallest tail in $\Lambda = \{a, a+1, \ldots, c\}$, sequence $\sigma_{Sc}(I)$ is optimal by inequality (1). Otherwise, there exists an *interference job* $b \in \Lambda$ such that

$$q_b < q_c \text{ and } q_j \geq q_c \text{ for all } j \in \{b+1, b+2, \ldots, c-1\}. \tag{2}$$

Let $\Lambda_b := \{b+1, b+2, \ldots, c\}$ be the jobs in Λ processed after the interference job b. Clearly, $q_j \geq q_c > q_b$ and $r_j > s_b$ hold for all $j \in \Lambda_b$, where s_b denotes the starting time of the interference job b. Inequality (1) applied to Λ_b gives with (2) the lower bound

$$L^*_{max} \geq \min_{j\in\Lambda_b} r_j + p(\Lambda_b) + \min_{j\in\Lambda_b} q_j > s_b + p(\Lambda_b) + q_c. \tag{3}$$

where $p(\Lambda_b)$ is the sum of processing times of jobs in Λ_b.

Since there is no idle time during the execution of the jobs of the critical sequence, the maximum lateness of Schrage's schedule is $L_{\max}(\sigma_{Sc}(I)) = s_b + p_b + p(\Lambda_b) + q_c$. Subtracting this equation from (3) we get the following upper bound for the absolute error in terms of the processing time p_b of the interference job b

$$L_{max}(\sigma_{Sc}(I)) - L^*_{max} < p_b. \tag{4}$$

Notice that using $L^*_{max} \geq p_b$ and applying inequality (4) shows that Schrage yields a relative performance guarantee of 2.

It is well-known that Schrage's sequence is optimal with respect to the makespan, i.e.,

$$C_{\max}(\sigma_{Sc}(I)) = C^*_{max}. \tag{5}$$

3 PTAS for the First Scenario: $1|r_j, C_j \leq d|L_{\max}$

In this section, we consider problem Π_1 with a common deadline d for the jobs. By (5) the Schrage sequence shows whether a feasible solution for Π_1 exists or not. If $C_{\max}(\sigma_{Sc}(I)) > d$, then the problem has no feasible solution. Hence, we will assume in the following that

$$C_{\max}(\sigma_{Sc}(I)) \leq d \tag{6}$$

Let $\varepsilon > 0$. A job j is called *large* if $p_j \geq \varepsilon L_{\max}(\sigma_{Sc}(I))/2$, otherwise it is called *small*. Let L be the subset of large jobs. Since $L_{\max}(\sigma_{Sc}(I)) \leq 2L^*_{\max}$ (see [13]), it can be observed that $|L| \leq 2/\varepsilon$. Let $k = |L|$. We assume that jobs are indexed such that $L = \{1, 2, \ldots, k\}$.

Our PTAS is based on the construction of a set of modified instances starting from the original instance I. Set $R = \{r_1, \ldots, r_n\}$ and $Q = \{q_1, \ldots, q_n\}$. Let us define the following sets of heads and tails:

$$R(i) = \{r_j \in R | r_j \geq r_i\}, \quad i = 1, 2, \ldots, k,$$
$$Q(i) = \{q_j \in Q | q_j \geq q_i\}, \quad i = 1, 2, \ldots, k.$$

Now, we define the set of all the combinations of couples $(r, q) \in R(i) \times Q(i)$ for every $i = 1, 2, \ldots, k$:

$$W = \{(\widetilde{r}_1, \widetilde{q}_1, \widetilde{r}_2, \widetilde{q}_2, \ldots, \widetilde{r}_k, \widetilde{q}_k) | \widetilde{r}_i \in R(i), \widetilde{q}_i \in Q(i), \quad i = 1, 2, \ldots, k\}$$

Clearly, the cardinality of W is bounded as follows:

$$|W| \leq n^{2k} = O(n^{4/\varepsilon}).$$

Let $w \in W$ with $w = (\widetilde{r}_1, \widetilde{q}_1, \widetilde{r}_2, \widetilde{q}_2, \ldots, \widetilde{r}_k, \widetilde{q}_k)$. Instance I_w is a slight modification of instance I and is defined as follows. I_w consists of large and small jobs. The k large jobs have modified release times $\widetilde{r}_1, \widetilde{r}_2, \ldots, \widetilde{r}_k$ and delivery times $\widetilde{q}_1, \widetilde{q}_2, \ldots, \widetilde{q}_k$. All processing times and the $n - k$ small jobs remain unchanged. Let $\mathcal{I}$ be the set of all possible instances I_w, i.e.,

$$\mathcal{I} = \{I_w | w \in W\}.$$

It is clear that $|\mathcal{I}|$ is in $O(n^{4/\varepsilon})$. By the modification, the processing times of instances in $\mathcal{I}$ are not changed and release times and delivery times are not decreased.

Let $\widetilde{\mathcal{I}} \subseteq \mathcal{I}$. An instance $I' \in \widetilde{\mathcal{I}}$ is called *maximal* if there is no other instance $I'' \in \widetilde{\mathcal{I}}$ such that for every $i = 1, 2, \ldots, k$, we have $r'_i \leq r''_i$, $q'_i \leq q''_i$ and at least one inequality is strict. Here, r'_i, q'_i denote the heads and tails in I' and r''_i, q''_i the heads and tails in I'', respectively.

Now, we can introduce our procedure PTAS1 which depends on the instance I, the deadline d and the accuracy ε.

Algorithm $PTAS1(I, d, \varepsilon)$

INPUT: An instance I of n jobs, a deadline d and an accuracy ε.
OUTPUT: A sequence $\sigma(I)$ with $L_{\max}(\sigma(I)) \leq (1+\varepsilon)L^*_{\max}$ and $C_{\max}(\sigma(I)) \leq d$.

1. Run Schrage's algorithm for all instances in $\mathcal{I}$.
2. Select the best feasible solution, i.e., the best solution with $C_{\max} \leq d$. Apply the corresponding sequence to the original instance I.

Let $\sigma^*(I)$ be a sequence for instance I which is optimal for $L_{\max}$ under the constraint $C_{\max}(\sigma^*(I)) \leq d$. Let $\widetilde{I} \in \mathcal{I}$ be an instance which is *compatible* with $\sigma^*(I)$, i.e.,

$$C_{\max}(\sigma^*(\widetilde{I})) \leq d, \tag{7}$$

$$\widetilde{r}_j \leq s_j(\sigma^*(I)) \text{ for } j = 1, 2, \ldots, k, \tag{8}$$

$$L_{\max}(\sigma^*(\widetilde{I})) = L_{\max}(\sigma^*(I)). \tag{9}$$

The set of all instances $\widetilde{I} \in \mathcal{I}$ which are compatible with σ^* is denoted as $\mathcal{I}_{\sigma^*}$. By (6) instance I fulfills the conditions (7)–(9). Thus, $\mathcal{I}_{\sigma^*}$ is nonempty.

Theorem 1 *Algorithm PTAS1 yields a* $(1+\varepsilon)$*-approximation for* Π_1 *and has polynomial running time for fixed* ε*. It can be implemented in* $O\left(\ln(n) \cdot n^{(1+4/\varepsilon)}\right)$.

Proof Let $I_{\max}$ be a maximal instance in $\mathcal{I}_{\sigma^*}$. Applying Schrage's algorithm to $I_{\max}$ gives the sequence $\sigma_{Sc}(I_{\max})$. We show

$$C_{\max}(\sigma_{Sc}(I_{\max})) \leq d \tag{10}$$

and

$$L_{\max}(\sigma_{Sc}(I_{\max})) \leq (1+\varepsilon)L_{\max}(\sigma^*(I)). \tag{11}$$

From (5) and (7) we conclude that $C_{\max}(\sigma_{Sc}(I_{\max})) \leq C_{\max}(\sigma^*(I_{\max})) \leq d$ and (10) follows.

Several cases are distinguished:

- Case 1: $\sigma_{Sc}(I_{\max})$ is optimal, i.e., $L_{\max}(\sigma_{Sc}(I_{\max})) = L_{\max}(\sigma^*(I))$ and by (9) we get (11).
- Case 2: $\sigma_{Sc}(I_{\max})$ is not optimal. Thus, there is an interference job b. Recall that $\Lambda_b = \{b, b+1, \ldots, c\}$ is the set of jobs processed after the interference job until the critical job c. By (4) $L_{\max}(\sigma_{Sc}(I_{\max})) - L_{\max}(\sigma^*(I_{\max})) < p_b$. If b is not large, then (11) follows immediately with (4). Otherwise, two subcases occur.

Subcase 2a: $\sigma_{Sc}(I_{\max})$ is not optimal and at least one job of Λ_b is processed before b in the optimal solution. Thus, b cannot start before $r_{\min} = \min_{i\in\Lambda_b}\{r_i\} > s_b$. Consequently, r_b can be increased to $r_{\min}$ and the new instance fulfills the conditions (7)–(9), which contradicts the maximality of $I_{\max}$.

Subcase 2b: $\sigma_{Sc}(I_{\max})$ is not optimal and all the jobs of Λ_b are processed after b in the optimal solution. Thus, q_b can be increased to q_c and the new instance fulfills the conditions (7)–(9), which again contradicts the maximality of $I_{\max}$.

Thus, we have found a sequence which is a $(1+\varepsilon)$-approximation for Π_1. Since $\mathcal{I}$ has at most $O(n^{4/\varepsilon})$ elements and Schrage's sequence can be computed in $O(n \ln(n))$, algorithm PTAS1 runs in polynomial time. Its overall running time is in $O\left(\ln(n) \cdot n^{(1+4/\varepsilon)}\right)$. □

4 PTAS for the Second Scenario: $1|r_j|L_{\max}, C_{\max}$

Computing a set of solutions which covers all possible trade-offs between different objectives can be understood in different ways. We will define this (as most commonly done) as searching for "efficient" solutions. Given an instance I a sequence $\sigma(I)$ *dominates* another sequence $\sigma'(I)$ if

$$L_{\max}(\sigma(I)) \leq L_{\max}(\sigma'(I)) \text{ and } C_{\max}(\sigma(I)) \leq C_{\max}(\sigma'(I)) \tag{12}$$

and at least one of the inequalities (12) is strict. The sequence $\sigma(I)$ is called *efficient* or *Pareto optimal* if there is no other sequence which dominates $\sigma(I)$. The set $\mathcal{P}$ of efficient sequence for I is called *Pareto frontier*.

We have also to define what we mean by an approximation algorithm with relative performance guarantee for our bicriteria scheduling problem. A sequence $\sigma(I)$ is called a $(1+\varepsilon)$-*approximation algorithm* of a sequence $\sigma'(I)$ if

$$L_{\max}(\sigma(I)) \leq (1+\varepsilon)L_{\max}(\sigma'(I)) \tag{13}$$

and

$$C_{\max}(\sigma(I)) \leq (1+\varepsilon)C_{\max}(\sigma'(I)) \tag{14}$$

hold.

A set $\mathcal{F}_\varepsilon$ of schedules for I is called a $(1+\varepsilon)$-*approximation of the Pareto frontier* if, for every sequence $\sigma'(I) \in \mathcal{P}$, the set $\mathcal{F}_\varepsilon$ contains at least one sequence $\sigma(I)$ that is a $(1+\varepsilon)$-approximation of $\sigma'(I)$.

A *PTAS for the Pareto frontier* is an algorithm which outputs for every $\varepsilon > 0$, a $(1+\varepsilon)$-approximation of the Pareto frontier and runs in polynomial time in the size of the input.

The Pareto frontier of an instance of a multiobjective optimization problem may contain an arbitrarily large number of solutions. On the contrary, for every $\varepsilon > 0$ there exists a $(1+\varepsilon)$-approximation of the Pareto frontier that consists of a number of solutions that is polynomial in the size of the instance and in $\frac{1}{\varepsilon}$ (under reasonable assumptions). An explicit proof for this observation was given by Papadimitriou and Yannakakis [20]. Consequently, a PTAS for a multiobjective optimization problem does not only have the advantage of computing a provably good approximation in polynomial time, but also has a good chance of presenting a reasonably small set of solutions.

Our PTAS for the Pareto frontier of problem Π_2 has even the stronger property that for every sequence $\sigma'(I) \in \mathcal{P}$, the set $\mathcal{F}_\varepsilon$ contains a sequence such that (13) holds and (14) is replaced by the inequality

$$C_{\max}(\sigma(I)) \leq C_{\max}(\sigma'(I)). \tag{15}$$

Algorithm $PTAS2(I, \varepsilon)$

INPUT: An instance I of n jobs and an accuracy ε.
OUTPUT: A PTAS for the Pareto frontier of problem Π_2.

1. Run Schrage's algorithm for all instances in $\mathcal{I}$ and store the solutions in set $\mathcal{F}_\varepsilon$.
2. Remove all sequences from $\mathcal{F}_\varepsilon$ which are dominated by other sequences in $\mathcal{F}_\varepsilon$.

Theorem 2 *Algorithm PTAS2 yields a PTAS for problem* Π_2 *such that for every sequence* $\sigma'(I) \in \mathcal{P}$*, the set* $\mathcal{F}_\varepsilon$ *contains a sequence* $\sigma(I)$ *such that (13) and (15) hold. PTAS2 has polynomial running time for fixed* ε*. It can be implemented in* $O\left(\ln(n) \cdot n^{(1+4/\varepsilon)}\right)$.

Proof The running time of PTAS2 is polynomial because the set $\mathcal{I}$ contains only a polynomial number of instances. More precisely, PTAS2 has the same running time as PTAS1. Let $\sigma'(I)$ be a Pareto optimal sequence for problem Π_2 with $C_{\max}(\sigma'(I)) = d$. Algorithm $PTAS1(I, d, \varepsilon)$ of Sect. 3 outputs a sequence $\sigma(I)$ which fulfills both (13) and (15). Since sequence $\sigma(I)$ is also found in Step 1 of Algorithm PTAS2, the theorem follows. □

5 PTAS for the Third Scenario: 1, $h_1|r_j|L_{\max}$

The third scenario 1, $h_1|r_j|L_{\max}$, denoted as Π_3, is studied in this section. The proposed PTAS for this scenario is related to the first one and it is based on several steps.

In the remainder of this section, I denotes a given instance of Π_3 and ε is the desired accuracy. Since a 2-approximation for $1, h_1|r_j|L_{\max}$ has been established in [8], we consider only $\varepsilon < 1$. Without loss of generality, in any instance I of problem Π_3, we assume that $r_j \notin [T_1, T_2[$ for $j \in J$ and that if $r_j < T_1$, then the inequality $r_j + p_j \leq T_1$ should hold. Otherwise, in both cases, r_j would be set equal to T_2. Thus, jobs in J can be partitioned in two disjoint subsets X and Y:

$$\begin{aligned} X &= \{j \in J | r_j + p_j \leq T_1\} \\ Y &= \{j \in J | r_j \geq T_2\} \end{aligned} \tag{16}$$

Finally, we assume that $1/\varepsilon$ is integer ($1/\varepsilon = f$) and that every instance I of problem Π_3 respects the conditions expressed in the following proposition.

Proposition 3 *With no $(1+\varepsilon)$-loss, every instance I of Π_3 contains at most f different tails from the set $\{\varepsilon\overline{q}, 2\varepsilon\overline{q}, 3\varepsilon\overline{q}, \ldots, \overline{q}\}$, where $\overline{q} = \max_{j\in J}\{q_j\}$.*

Proof We simplify the instance I as follows. Split the interval $[0, \max_{j\in J}\{q_j\}]$ into $1/\varepsilon$ equal length intervals and round up every tail q_j to the next multiple of $\varepsilon\overline{q}$. Clearly, every tail will not be increased by more than $\varepsilon\overline{q} \leq \varepsilon L^*_{\max}(I)$. Then, we obtain that the modified instance has an optimal solution of a maximum lateness less than or equal to $(1 + \varepsilon)\, L^*_{\max}(I)$. □

Since all the jobs of Y should be scheduled after T_2 in any feasible solution of I, our PTAS will focus on subset X. More precisely, this PTAS will be based on guessing the disjoint subsets X_1 and X_2 ($X_1 \subset X$, $X_2 \subset X$ and $X_1 \cup X_2 = X$) such that X_1 (respectively X_2) contains the jobs to be scheduled before T_1 (respectively after T_2). In the optimal solution, we will denote the jobs of X to be scheduled before T_1 (respectively after T_2) by X_1^* (respectively by X_2^*). It is clear that if we are able to guess correctly X_1^* in polynomial time, then it is possible to construct a PTAS for Π_3. Indeed, we can associate the scheduling of the jobs in X_1^* before T_1 with a special instance I_1^* of Π_1 where the jobs to be scheduled are those in subset X_1^* and the deadline is equal to T_1. On the other hand, the scheduling of the other jobs in $X_2^* \cup Y$ after T_2 can be seen as a special instance I_2^* of problem Π_0 , with all release times greater or equal to T_2. Consequently,

$$L^*_{\max}(I) = \max\{L^*_{\max}(I_1^*), L^*_{\max}(I_2^*)\} \tag{17}$$

Clearly, the optimal sequence σ_1^* of I_1^* should satisfy the feasibility condition

$$C_{\max}(\sigma_1^*(I_1^*)) \leq T_1. \tag{18}$$

As a consequence, by applying PTAS1 with $d = T_1$ or another existing PTAS for Π_0 (for example PTAS1 with d set to ∞ or the PTAS by Hall and Shmoys [7]), we would get a PTAS for problem Π_3. Unfortunately, it is not possible to guess the exact subset X_1^* in a polynomial time, since the potential number of the candidate subsets X_1 is exponential. Nevertheless, we will show later that we can guess in a polynomial time another close/approximate subset $X_1^\#$ for which the associated instance $I_1^\#$ of Π_1 and its optimal solution $\sigma_1^\#$ verify the following relations:

$$L_{\max}(\sigma_1^{\#}(I_1^{\#})) \leq (1+\varepsilon)\, L_{\max}(\sigma_1^{*}(I_1^{*})), \tag{19}$$

$$C_{\max}(\sigma_1^{\#}(I_1^{\#})) \leq C_{\max}(\sigma_1^{*}(I_1^{*})). \tag{20}$$

As the core of our problem of designing a PTAS for Π_3 is to schedule correctly the jobs of X, our guessing approach will be based on a specific structure on X. In the next paragraph, we will describe such a structure. Then, we present our PTAS and its proof.

5.1 Defining a Structure for the Jobs in X

In this section, we will focus on the jobs of X and partition them into subsets. Recall that $P = \sum_{j=1}^{n} p_j$. Set

$$\delta = \frac{\varepsilon^2 P}{4}.$$

A job $j \in X$ is called a *big* job if $p_j > \delta$. Let $B \subset X$ denote the set of big jobs. Obviously, $|B| < 4/\varepsilon^2$. The set $S = X \backslash B$ represents the subset of *small* jobs. We partition S into f subsets $S(k)$, $1 \leq k \leq f$, where jobs in $S(k)$ have identical tails of length $k\varepsilon\overline{q}$.

In the next step, jobs in $S(k)$ are sorted in non-decreasing order of release times. In other words, jobs in $S(k)$ are encoded by pairs of integers

$$(k,1), (k,2), \ldots, (k, |S(k)|)$$

such that

$$r_{(k,1)} \leq r_{(k,2)} \leq \cdots \leq r_{(k,|S(k)|)}. \tag{21}$$

Set $m(k) = \lceil p(S(k))/\delta \rceil$. Hence, $m(k)$ is in $O(4/\varepsilon^2)$. For every k and z with $1 \leq k \leq f$ and $1 \leq z \leq m(k)$, the subset $S_{k,z} = \{(k,1), (k,2), \ldots, (k, e(z))\}$ is composed of the $e(z)$ jobs of $S(k)$ with the smallest release times, such that its total processing time $p\left(S_{k,z}\right) = \sum_{l=1}^{e(z)} p_{(k,l)}$ fulfills

$$(z-1)\delta < p\left(S_{k,z}\right) \leq z\delta, \tag{22}$$

and $e(z)$ is the largest integer for which (22) holds. Moreover, define $S_{k,0} = \emptyset$ and $e(0) = 0$. Note that $S_{k,m(k)} = S(k)$.

5.2 Description of the PTAS for Π_3

As we mentioned before, our proposed PTAS for problem Π_3 is based on guessing the subset of jobs $X_1 \subset X$ to be scheduled before T_1 (Subset $X_2 = X \backslash X_1$ will be scheduled after T_2). For every guessed subset $X_1 \subset X$ in our PTAS, we associate an instance I_1 of Π_1 with $d = T_1$. To the complementary subset $X_2 \cup Y$ we associate an instance I_2 of Π_0 after setting all the release times in X_2 to T_2. In order to solve

the generated instances I_2, we use PTAS1 with $d = \infty$. (Alternatively, we could also use the existing PTAS by Hall and Shmoys for Π_0. More details on this PTAS are available in [7]). We will abbreviate $PTAS1(I, \infty, \varepsilon)$ by $PTAS0(I, \varepsilon)$. For instances I_1 we apply $PTAS1(I_1, T_1, 3\varepsilon/5)$ and for instances I_2, we apply $PTAS0(I_2, \varepsilon/3)$. The guessing of X_1 and the details of our PTAS are described in procedure PTAS3.

Algorithm $PTAS3(I, T_1, T_2, \varepsilon)$

INPUT: An instance I of n jobs, integers T_1, T_2 with $T_1 \leq T_2$ and an accuracy ε.
OUTPUT: A sequence $\sigma(I)$ with $L_{\max}(\sigma(I)) \leq (1+\varepsilon)L^*_{\max}(I)$ respecting the non-availability interval $]T_1, T_2[$.

1. The big jobs B_1 to be included in X_1 are guessed from B by complete enumeration. For every guess of B_1, the jobs $B \backslash B_1$ will be a part of X_2.
2. The small jobs to be included in X_1 are guessed for each subset $S(k)$, $1 \leq k \leq f$, by choosing one of the $m(k)+1$ possible subsets $S_{k,z}$, $0 \leq z \leq m(k)$. Thus, $X_1 = B_1 \cup_{k=1}^{f} S_{k,z}$ and $X_2 = (B \backslash B_1) \cup_{k=1}^{f} \left(S(k) \backslash S_{k,z}\right)$. Set all the release times of jobs in X_2 to T_2.
3. With every guessed subset X_1 in Step 2, we associate an instance I_1 of Π_1 with $d = T_1$. With the complementary subset $X_2 \cup Y$ we associate an instance I_2 of Π_0. Apply $PTAS1(I_1, T_1, 3\varepsilon/5)$ for solving instance I_1 of Π_1 and $PTAS0(I_2, \varepsilon/3)$ for solving instance I_2 of Π_0. Let σ_1 and σ_2 be the obtained schedules from $PTAS1(I_1, T_1, 3\varepsilon/5)$ and $PTAS0(I_2, \varepsilon/3)$, respectively. If they are feasible, then merge σ_1 and σ_2 to get σ, which represents a feasible solution for instance I of Π_3.
4. Return $\sigma(I)$, the best feasible schedule from all the complete merged schedules for I in Step 3.

5.3 Proof of the PTAS for Π_3

Denote the jobs from $S(k)$ which are scheduled before T_1 in the optimal solution by $S_{k,*}$. If

$$0 \leq p\left(S_{k,*}\right) < p(S_k) = p\left(S_{k,m(k)}\right),$$

there is an integer $z_k \in \{0, 1, \ldots, m(k)-1\}$ such that

$$z_k \delta \leq p\left(S_{k,*}\right) < (z_k + 1)\delta. \tag{23}$$

We will see in the following lemma that we can get a sufficient approximation by taking the jobs of S_{k,z_k}, $z_k = 0, 1, \ldots, m(k)$, instead of those in $S_{k,*}$. In particular, we will see that it is possible to schedule heuristically before T_1 the jobs in S_{k,z_k} instead of those in $S_{k,*}$ without altering the optimal performance too much. By selecting jobs with minimal release times we maintain the feasibility of inserting these jobs before T_1 instead of $S_{k,*}$. Inequalities (22) and (23) imply that there is an integer $z_k \in \{0, 1, \ldots, m(k)-1\}$ such that

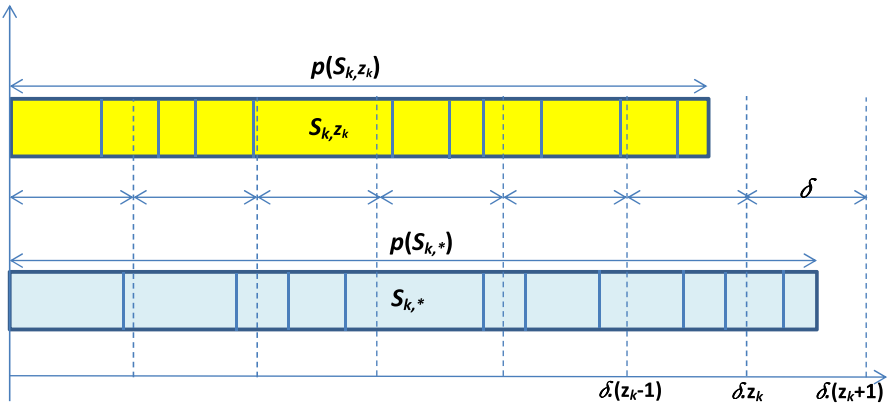

Fig. 1 Relation between S_{k,z_k} and $S_{k,*}$

$$p\left(S_{k,z_k}\right) \leq p\left(S_{k,*}\right) \leq p\left(S_{k,z_k}\right) + 2\delta \tag{24}$$

holds for $0 \leq p\left(S_{k,*}\right) < p\left(S_k\right)$.

Note that if $S_{k,*} = S(k)$ holds, the guessing procedure ensures that for $z_k = m(k)$ we get $S_{k,m(k)} = S(k)$. The choice of S_{k,z_k} is illustrated in Fig. 1.

Recall that X_1^* (respectively $X_2^* \cup Y$) is the subset of jobs to be scheduled before T_1 (respectively after T_2) in the optimal schedule $\sigma^*(I)$ of a given instance I of Π_3. Let I_1^*, I_2^* denote the corresponding instances of problems Π_1 and Π_0, and $\sigma_1^*(I_1^*)$, $\sigma_2^*(I_2^*)$ the associated optimal schedules, respectively. Then, we get.

Lemma 4 *There is a subset $X_1^\# \subset X$, generated in Step 2 of PTAS3, for which we can construct two feasible schedules $\sigma_1^\#(I_1^\#)$ and $\sigma_2^\#(I_2^\#)$ such that:*

$$L_{\max}(\sigma_1^\#(I_1^\#)) \leq L_{\max}(\sigma_1^*(I_1^*)) + f\delta, \tag{25}$$

$$C_{\max}(\sigma_1^\#(I_1^\#)) \leq C_{\max}(\sigma_1^*(I_1^*)), \tag{26}$$

$$L_{\max}(\sigma_2^\#(I_2^\#)) \leq L_{\max}(\sigma_2^*(I_2^*)) + 2f\delta, \tag{27}$$

where $I_1^\#$ is an instance of Π_1 where we have to schedule the jobs of $X_1^\#$ before T_1 and $I_2^\#$ is an instance of Π_0 where we have to schedule the jobs of $(X \backslash X_1^\#) \cup Y$ after T_2.

Proof Let B_1^* be the subset of big jobs contained in X_1^*. The other jobs of X_1^* are small and belong to $\cup_{k=1}^{f} S(k)$. Hence,

$$X_1^* = B_1^* \cup \bigcup_{k=1}^{f} S_{k,*}$$

and $X_1^{\#}$ will have the following structure:

$$X_1^{\#} = B_1^* \cup \bigcup_{k=1}^{f} S_{k,z_k}.$$

Obviously, subset B_1^* can be guessed correctly in Step 2 of PTAS3 since PTAS3 considers all possible subsets of B. Therefore, for proving that subset $X_1^{\#}$ is a good guess, we need to prove that the guessing of each subset S_{k,z_k} is close enough to $S_{k,*}$, but with smaller total processing time.

First, we observe that if $S_{k,*} = \varnothing$ or $S_{k,*} = S(k)$, then the guessing is perfect, i.e. $S_{k,*} = S_{k,z}$.

If $S_{k,*} \neq \varnothing$, we can determine intervals where only jobs of $S_{k,*}$ are scheduled in the optimal schedule $\sigma_1^*(I_1^*)$ and which contain no idle time. Denote those intervals by

$$G_{1,k} = [a_{1,k}^*, b_{1,k}^*], G_{2,k} = [a_{2,k}^*, b_{2,k}^*], \ldots, G_{\gamma_k,k} = [a_{\gamma_k,k}^*, b_{\gamma_k,k}^*], \quad k = 1, \ldots, f,$$

where γ_k is a certain finite integer. Set

$$G_k = \bigcup_{u=1}^{\gamma_k} G_{u,k}, \quad k = 1, \ldots, f.$$

Otherwise, in case $S_{k,*} = \varnothing$, we set $G_k = \emptyset$. Finally, define

$$G = \bigcup_{k=1}^{f} G_k.$$

In addition, we recall that $S_{k,z_k} = \{(k,1), (k,2), \ldots, (k, e(z_k))\}$. In our feasible schedule $\sigma_1^{\#}(I_1^{\#})$, we will schedule the jobs of this subset S_{k,z_k} according to the following heuristic procedure *Schedule*$^{\#}$. It assigns the jobs to the intervals $G_{u,k}$ but throughout the procedure starting and finish times of these intervals may be changed (e.g. when there is some overload of jobs) or they are simply shifted to the right.

Procedure *Schedule*$^{\#}$

1. **For** $k = 1$ to f do:
2. $\alpha_0 := 0$;
3. **For** $u = 1$ to $u = \gamma_k$ do:
   - 3.1 Process remaining jobs from S_{k,z_k} (in non-decreasing order of release dates) starting at time $a_{u,k}^* + \alpha_{u-1}$ until the first job finishes at time $t \geq b_{u,k}^*$ or all jobs from S_{k,z_k} are assigned. Let job (k,l) be the last scheduled job from this set with completion time $C_{(k,l)}$.
   - 3.2 **If** $l = e(z_k)$ set $G_{u,k} = [a_{u,k}^* + \alpha_{u-1},\ b_{u,k}^*]$ and **Stop**: All items from S_{k,z_k} are assigned.

3.3 **Else** set $\alpha_u = C_{(k,l)} - b^*_{u,k}$ and $G_{u,k} = [a^*_{u,k} + \alpha_{u-1},\ b^*_{u,k} + \alpha_u] = [a^*_{u,k} + \alpha_{u-1},\ C_{(k,l)}]$.

3.4 Shift intervals from G which are located between $b^*_{u,k}$ and $a^*_{u+1,k}$ without any modification of their lengths by α_{u+1} to the right.

3.5 End **If/Else**

4. End **For**

5. End **For**

Notice that the value α_u represents the "overload" of the uth interval $G_{u,k}$ in a given set G_k. The next interval $G_{u+1,k}$ is then shortened by the same amount α_u. Consequently, only the intervals between $G_{u,k}$ and $G_{u+1,k}$ are shifted to the right, but no others. Moreover, the total processing time assigned until $b^*_{u+1,k}$ in the kth iteration of *Schedule*$^{\#}$ is not greater than the original total length of the intervals $G_{1,k}, G_{2,k}, \ldots, G_{u+1,k}$. Since for every family k the jobs in S_{k,z_k} have the smallest release times and since by (21) these jobs are sorted in non-decreasing order of release times, no idle time is created during Step 3. Together with the first inequality of (24) this guarantees that all jobs of family k are processed without changing the right endpoint of the last interval $G_{\gamma_k,k}$, i.e., creating no overload in this interval. That ensures the feasibility of the schedule yielded by *Schedule*$^{\#}$ and that $G_{\gamma_k,k} = [a^*_{\gamma_k,k} + \alpha_{\gamma_k-1},\ b^*_{\gamma_k,k}]$.

In each iteration of the procedure the jobs will be delayed by at most $\max_{u \le \gamma_k}\{\alpha_u\} \le \delta$. Since this possible delay may occur for every family $k, k = 1, 2, \ldots, f$, the possible overall delay for every job, after the f applications of procedure *Schedule*$^{\#}$ is therefore bounded by $f\delta$. Hence, (25) follows.

The finishing time of the last interval among the intervals $G_{\gamma_k,k}, k = 1, \ldots, f$, generated by *Schedule*$^{\#}$ determines the makespan of instance $I^{\#}_1$. But this interval is never shifted to the right and we obtain (26).

Recall that if $S_{k,*} = S(k)$, we get with $z_k = m(k)$ that $S_{k,m(k)} = S(k)$. Hence, in the case that no jobs of family k are scheduled after T_2 in $\sigma^*_2(I^*_2)$, also $\sigma^{\#}_2(I^{\#}_2)$ contains no jobs of family f.

Hence, assume in the following that $S(k)\backslash S_{k,*} \neq \emptyset$ and let $[c^*_{1,k}, d^*_{1,k}]$, $[c^*_{2,k}, d^*_{2,k}]$, $\ldots$, $[c^*_{\lambda_k,k}, d^*_{\lambda_k,k}]$ be the intervals in which the jobs of $S(k)\backslash S_{k,*}$ are scheduled in the optimal schedule $\sigma^*_2(I^*_2)$. The schedule $\sigma^{\#}_2(I^{\#}_2)$ will be created similarly to procedure *Schedule*$^{\#}$.

As a consequence of performing possibly less amount of jobs from every $S(k)$ before T_1 in the feasible schedule $\sigma^{\#}_1(I^{\#}_1)$, an additional small quantity of processing time must be scheduled after T_2 compared to the optimal solution. This amount is bounded by 2δ by the second inequality of (24). Therefore, in each iteration k we start by enlarging the last interval by 2δ, i.e., $[c^*_{\lambda_k,k}, d^*_{\lambda_k,k} + 2\delta]$ instead of $[c^*_{\lambda_k,k}, d^*_{\lambda_k,k}]$, and shifting those jobs starting after $d^*_{\lambda_k,k}$ in $\sigma^*_2(I^*_2)$ by the same distance 2δ to the right. Then, all what we have to do in order to continue the construction of our feasible schedule $\sigma^{\#}_2(I^{\#}_2)$ is to insert the remaining jobs from every $S(k)\backslash S_{k,z_k}$ in the corresponding intervals $[c^*_{1,k}, d^*_{1,k}]$, $[c^*_{2,k}, d^*_{2,k}]$, $\ldots$, $[c^*_{\lambda_k,k}, d^*_{\lambda_k,k} + 2\delta]$ like in iteration k of procedure *Schedule*$^{\#}$.

Analogously to *Schedule*$^{\#}$ intervals between $d^*_{u,k}$ and $c^*_{u+1,k}$ are shifted to the right by some distance bounded by δ. Thus, in a certain iteration k jobs are shifted to the right by not more than $\max\{2\delta, \delta\} = 2\delta$. Since the possible delay may occur for every

job family k, the possible overall delay for every job scheduled in $\sigma_2^{\#}(I_2^{\#})$ is bounded by $2f\delta$. This shows that inequality (27) is valid. □

Theorem 5 *Algorithm PTAS3 is a PTAS for problem* Π_3. *The running time of this algorithm is in* $O\left(2^{4/\varepsilon^2}\cdot\left(4/\varepsilon^2\right)^{1/\varepsilon}\cdot\ln(n)\cdot\left(n^{(1+20/3\varepsilon)}+n^{(1+12/\varepsilon)}\right)\right)$.

Proof The big jobs B_1 to be included in X_1 are completely guessed from B in $O(2^{4/\varepsilon^2})$. The small jobs to be included in X_1 are guessed from every subset $S(k)$ by considering only the $m(k)+1$ possible subsets $S_{k,z}$. This can be done for every guessed B_1 in $O((m(k)+1)^f)=O(\left(4/\varepsilon^2\right)^{1/\varepsilon})$ time. In overall, the guessing of all the subsets X_1 will be done in $O(2^{4/\varepsilon^2}.\left(4/\varepsilon^2\right)^{1/\varepsilon})$. Hence, from the computational analysis of the different steps of PTAS3, it is clear that this algorithm runs polynomially in the size of instance I of Π_3 for a fixed accuracy $\varepsilon>0$.

Let us now consider the result of Lemma 4. We established that a subset $X_1^{\#}\subset X$ is generated in Step 2 of PTAS3 and that two feasible schedules $\sigma_1^{\#}(I_1^{\#})$ and $\sigma_2^{\#}(I_2^{\#})$ exist such that (25), (26) and (27) hold. By Theorem 1 we have

$$L_{\max}\left(PTAS1(I_1^{\#},T_1,3\varepsilon/5)\right)\leq(1+3\varepsilon/5)\,L_{\max}(\sigma_1^{\#}(I_1^{\#})).$$

From (25), we then deduce that

$$L_{\max}\left(PTAS1(I_1^{\#},T_1,3\varepsilon/5)\right)\leq(1+3\varepsilon/5)\left(L_{\max}(\sigma_1^{*}(I_1^{*}))+f\delta\right). \tag{28}$$

Moreover, we have

$$L_{\max}\left(PTAS0(I_2^{\#},\varepsilon/3)\right)\leq(1+\varepsilon/3)\,L_{\max}(\sigma_2^{\#}(I_2^{\#})).$$

From (27) we establish that

$$L_{\max}\left(PTAS0(I_2^{\#},\varepsilon/3)\right)\leq(1+\varepsilon/3)\left(L_{\max}(\sigma_2^{*}(I_2^{*}))+2f\delta\right). \tag{29}$$

By using (26) we get

$$C_{\max}(\sigma_1^{\#}(I_1^{\#}))\leq C_{\max}(\sigma_1^{*}(I_1^{*}))\leq T_1. \tag{30}$$

Inequality (30) implies that the algorithm $PTAS1(I_1^{\#},T_1,3\varepsilon/5)$ yields a feasible solution with $C_{\max}(PTAS1(I_1^{\#},T_1,3\varepsilon/5))\leq T_1$.

Let us now define the global solution $\sigma^{\#}(I)$ for instance I (of Π_3) as the union of the two feasible subsolutions $PTAS1(I_1^{\#},T_1,3\varepsilon/5)$ and $PTAS0(I_2^{\#},\varepsilon/3)$. The following relation holds

$$L_{\max}\left(\sigma^{\#}(I)\right)=\max\left\{L_{\max}(PTAS1(I_1^{\#},T_1,3\varepsilon/5)),L_{\max}\left(PTAS0(I_2^{\#},\varepsilon/3)\right)\right\}.$$

Thus, by applying (28) and (29), we deduce that

$$L_{\max}\left(\sigma^{\#}(I)\right) \leq \max\{(1+3\varepsilon/5)\left(L_{\max}(\sigma_1^*(I_1^*)) + f\delta\right), (1+\varepsilon/3)\left(L_{\max}(\sigma_2^*(I_2^*)) + 2f\delta\right)\}. \quad (31)$$

From (31) and by observing that $f\delta \leq \varepsilon L^*_{\max}(I)/4$, we obtain for $\varepsilon \leq 1$ the final relations

$$\begin{aligned} L_{\max}\left(\sigma^{\#}(I)\right) &\leq \max\{L_{\max}(\sigma_1^*(I_1^*)) + \varepsilon L^*_{\max}(I), L_{\max}(\sigma_2^*(I_2^*)) + \varepsilon L^*_{\max}(I)\} \\ &= \max\{L_{\max}(\sigma_1^*(I_1^*)), L_{\max}(\sigma_2^*(I_2^*))\} + \varepsilon L^*_{\max}(I) = (1+\varepsilon)L^*_{\max}(I). \end{aligned}$$

This finishes the proof of Theorem 5. □

6 PTAS for the Fourth Scenario: 1, $ONA|r_j|L_{\max}$

The studied scenario Π_4 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, a release date r_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). As in the previous scenarios, we aim to minimize the maximum lateness.

If for every $j \in J$ we have $p_j < T_2 - T_1$ or $r_j \geqslant T_2$, it can be easily checked that Π_4 is equivalent to an instance of Π_3 for which we can apply the PTAS described in the previous section. Hence, in the remainder we assume that there are some jobs which have processing times greater than $T_2 - T_1$ and can complete before T_1. Let $\mathcal{K}$ denote the subset of these jobs. Similarly to the transformation proposed in [11], the following theorem on the existence of the PTAS for Scenario Π_4 can be proved.

Theorem 6 *Scenario Π_4 admits a PTAS. The running time of this approximation scheme is in $O\left((n/\varepsilon)\cdot 2^{4/\varepsilon^2}\cdot\left(4/\varepsilon^2\right)^{1/\varepsilon}\cdot\ln(n)\cdot\left(n^{(1+20/3\varepsilon)}+n^{(1+12/\varepsilon)}\right)\right)$.*

Proof We distinguish two subcases:

- Subcase 1: There exists a job $s \in \mathcal{K}$, called *straddling job*, such that in the optimal solution it starts not later than T_1 and completes not before T_2.
- Subcase 2: There is no straddling job in the optimal solution.

It is obvious that Subcase 2 is equivalent to an instance of Π_3 for which we have established the existence of a PTAS. Thus, the last step necessary to prove the existence of a PTAS for an instance of Π_4 is to construct a special scheme for Subcase 1. Without loss of generality, we assume that Subcase 1 holds and that the straddling job s is known (indeed, it can be guessed in $O(n)$ time among the jobs of $\mathcal{K}$). Moreover, the time-window of the starting time t_s^* of this job s in the optimal solution fulfills

$$t_s^* \in [T_2 - p_s, T_1].$$

If $T_2 - p_s = T_1$, i.e., the interval $[T_2 - p_s, T_1]$ consists of a single point T_1, we can apply $PTAS3(I, T_1, T_1, \varepsilon)$. Otherwise, set

$$t_s^h = T_2 - p_s + h\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil} \quad h = 0, 1, \ldots, \lceil 1/\varepsilon \rceil .$$

We consider a set of $\lceil 1/\varepsilon \rceil + 1$ instances $\{\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_{\lceil 1/\varepsilon \rceil}\}$ of Π_4 where in $\mathcal{I}_h$ the straddling job s starts at time t_s^h. Each such instance is equivalent to an instance of Π_3 with a set of jobs $J - \{s\}$ and a machine non-availability interval Δ_h with

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

For every instance from $\{\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_{\lceil 1/\varepsilon \rceil}\}$, we apply PTAS3 and select the best solution among all the $\lceil 1/\varepsilon \rceil + 1$ instances.

If $t_s^* = T_2 - p_s$, PTAS3 applied to $\mathcal{I}_0$ is the right choice. In all other cases, there is an $h \in \{0, 1, \ldots, \lceil 1/\varepsilon \rceil\}$ with $t_s^* \in]t_s^h, t_s^{h+1}]$. Delaying s and the next jobs in the optimal schedule of $\mathcal{I}_{h+1}$, $h = 0, 1, \ldots, \lceil 1/\varepsilon \rceil - 1$, will not cost more than

$$\frac{T_1 - T_2 + p_s}{\lceil 1/\varepsilon \rceil} \leq \varepsilon\,(T_1 - T_2 + p_s) \leq \varepsilon p_s.$$

Thus, the solution Ω_h obtained by PTAS3 for $\mathcal{I}_h$, $h = 1, 2, \ldots, \lceil 1/\varepsilon \rceil)$ is sufficiently close to an optimal schedule for Subcase 1 if s is the straddling job and $t_s^* \in]t_s^h, t_s^{h+1}]$. As a conclusion, Subcase 1 has a PTAS. □

7 Conclusions

In this paper, we considered important single-machine scheduling problems under release times and tails assumptions, with the aim of minimizing the maximum lateness. Four variants have been studied under different scenarios. In the first scenario, all the jobs are constrained by a common deadline. The second scenario consists in finding the Pareto front where the considered criteria are the makespan and the maximum lateness. In the third scenario, a non-availability interval constraint is considered (breakdown period, fixed job or a planned maintenance duration). In the fourth scenario, the non-availability interval is related to the operator who is organizing the execution of jobs on the machine, which implies that no job can start, and neither can complete during the operator non-availability period. For each of these four open problems, we establish the existence of a PTAS.

As a perspective, the study of min-sum scheduling problems in a similar context seems to be a challenging subject.

Acknowledgements Open access funding provided by University of Graz.

References

1. Brauner, N., Finke, G., Kellerer, H., Lebacque, V., Rapine, C., Potts, C., Strusevich, V.: Operator non-availability periods. 4OR Q. J. Oper. Res. **7**, 239–253 (2009)
2. Carlier, J., Moukrim, A., Hermes, F., Ghedira, K.: Exact resolution of the one-machine sequencing problem with no machine idle time. Comput. Ind. Eng. **59**(2), 193–199 (2010)
3. Dessouky, M.I., Margenthaler, C.R.: The one-machine sequencing problem with early starts and due dates. AIIE Trans. **4**(3), 214–222 (1972)
4. Gens, G.V., Levner, E.V.: Fast approximation algorithms for job sequencing with deadlines. Discret. Appl. Math. **3**, 313–318 (1981)
5. Gharbi, A., Labidi, M., Haouari, M.: An exact algorithm for the single machine problem with unavailability periods. Eur. J. Ind. Eng. **9**(2), 244–260 (2015)
6. Grabowski, J., Nowicki, E., Zdrzalka, S.: A block approach for single-machine scheduling with release dates and due dates. Eur. J. Oper. Res. **26**, 278–285 (1986)
7. Hall, L.A., Shmoys, D.B.: Jackson's rule for single machine scheduling: making a good heuristic better. Math. Oper. Res. **17**, 22–35 (1992)
8. Kacem, I., Haouari, M.: Approximation algorithms for single machine scheduling with one unavailability period. 4OR Q. J. Oper. Res. **7**(1), 79–92 (2009)
9. Kacem, I., Kellerer, H.: Semi-online scheduling on a single machine with unexpected breakdown. Theor. Comput. Sci. **646**, 40–48 (2016)
10. 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)
11. 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)
12. Kellerer, H.: Minimizing the maximum lateness. In: Leung, J.Y. (ed.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis, vol. 10, pp. 185–196. Chapman & Hall/CRC, Boca Raton, London, New York
13. Kise, H., Ibaraki, T., Mine, H.: Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times. J. Oper. Res. Soc. Jpn. **22**, 205–224 (1979)
14. Larson, R.E., Dessouky, M.I., Devor, R.E.: A forward-backward procedure for the single machine problem to minimize the maximum lateness. IIE Trans. **17**, 252–260 (1985)
15. Lee, C.Y.: Machine scheduling with an availability constraint, chapter 22. In: Leung, J.Y. (ed.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton (2004)
16. Lenstra, J.K., Rinnooy Kan, A.H.J., Brucker, P.: Complexity of machine scheduling problems. Ann. Oper. Res. **1**, 342–362 (1977)
17. Leon, V.J., Wu, S.D.: On scheduling with ready-times, due-dates and vacations. Naval Res. Logist. **39**, 53–65 (1992)
18. Mastrolilli, M.: Efficient approximation schemes for scheduling problems with release dates and delivery times. J. Sched. **6**(6), 521–531 (2003)
19. Nowicki, E., Smutnicki, C.: An approximation algorithm for single-machine scheduling with release times and delivery times. Discret. Appl. Math. **48**, 69–79 (1994)
20. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS'00), pp. 86–92 (2000)
21. Potts, C.N.: Analysis of a heuristic for one machine sequencing with release dates and delivery times. Oper. Res. **28**, 1436–1441 (1980)
22. Rapine, C., Brauner, N., Finke, G., Lebacque, V.: Single machine scheduling with small operator-non-availability periods. J. Sched. **15**, 127–139 (2012)
23. T'kindt, V., Billaut, J.C.: Multicriteria Scheduling. Springer-Verlag, Berlin, Heidelberg (2006)
24. Yin, Y., Wu, W.-H., Cheng, S.-R., Wu, C.-C.: An investigation on a two-agent single-machine scheduling problem with unequal release dates. Comput. Oper. Res. **39**(12), 3062–3073 (2012)

25. Yin, Y., Wu, W.-H., Wu, W.-H., Wu, C.-C.: A branch-and-bound algorithm for a single machine sequencing to minimize the total tardiness with arbitrary release dates and position-dependent learning effects. Inf. Sci. **256**, 91–108 (2014)
26. 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)