DOI:
10.1016/j.tcs.2024.114673

Document Version
Peer reviewed version

New Bounds for Single-Machine Time-Dependent Scheduling with Uniform Deterioration

Angelos Gkikas[a], Dimitrios Letsios[a], Tomasz Radzik[a], Kathleen Steinhöfel[a]

[a] *Department of Informatics, King's College London, United Kingdom*

Abstract

We consider the single-machine time-dependent scheduling problem with linearly deteriorating jobs arriving over time. Each job i is associated with a release time r_i and a processing time $p_i(s_i) = \alpha_i + \beta_i s_i$, where $\alpha_i, \beta_i > 0$ are parameters and s_i is the job's start time. In this setting, the approximability of both single-machine minimum makespan and total completion time problems remains open. We develop new bounds and approximation results for the special case of the problems with uniform deterioration, i.e. $\beta_i = \beta$, for each i. The main contribution is a $O(1 + 1/\beta)$-approximation algorithm for the makespan problem and a $O(1 + 1/\beta^2)$ approximation algorithm for the total completion time problem. Further, we propose greedy constant-factor approximation algorithms for instances with $\beta = O(1/n)$ and $\beta = \Omega(n)$, where n is the number of jobs. Our analysis is based on an approach for comparing computed and optimal schedules via bounding pseudomatchings.

Keywords: Single-Machine Time-Dependent Scheduling, Linear Deterioration, Approximation Algorithms, Release Times

1. Introduction

Single-machine scheduling problems involve deciding when to process a set $\mathcal{J} = \{1, \dots, n\}$ of n jobs arriving over time, i.e. each job $i \in \mathcal{J}$ is associated with a release time $r_i \in \mathbb{Z}^+$, using a machine that may execute at most one job per time so as to optimize some objective function $f(\vec{C})$, e.g. the makespan $\max_{i \in \mathcal{J}}\{C_i\}$ or the total completion time $\sum_{i \in \mathcal{J}} C_i$, where $\vec{C} = (C_1, \dots, C_n)$ is the vector of job completion times. Prior literature largely assumes that each job $i \in \mathcal{J}$ has a fixed processing time $p_i \in \mathbb{Z}^+$. However, this assumption can be fairly strong. In various contexts, e.g.

production scheduling with machine degradation and delivery scheduling in road networks with varying traffic, the time at which the execution of a job begins significantly affects its processing time. Scheduling problems where the processing time of each job $i \in \mathcal{J}$ is a function $p_i(s_i)$ of its start time s_i are typically referred to as *time-dependent scheduling problems* [8, 9].

Previous work investigates time-dependent scheduling problems with processing time functions $p_i(s_i) = \alpha_i + \beta_i s_i$, where $\alpha_i \in \mathbb{Z}^+$ is the *fixed part* and $\beta_i s_i$ is the *variable part* depending on the deterioration rate $\beta_i \in \mathbb{Z}^+$ and the start time s_i, for each job $i \in \mathcal{J}$. Such problems are referred to as *scheduling with linear deterioration* and model settings where delaying the beginning of a job execution by one unit of time results in an increase of the job's processing time by β_i units of time. In this context, the single-machine problems of minimizing the makespan and the total completion time for jobs with release times are open from an algorithmic viewpoint. Using the standard three-field scheduling notation [8, 9], the problems can be denoted as $1|r_i, p_i(s_i) = \alpha_i + \beta_i s_i|C_{\max}$ and $1|r_i, p_i(s_i) = \alpha_i + \beta_i s_i| \sum C_i$. When all jobs have equal release times, the makespan problem is polynomially solvable [10], while the complexity of the total completion time problem is unknown and conjectured to be $\mathcal{NP}$-hard [9]. When the jobs have arbitrary release times, both problems are known to be strongly $\mathcal{NP}$-hard [4, 7, 16]. The best known algorithms are based on iterative subproblem decomposition, e.g. dynamic programming and branch-and-bound [2, 6, 15], but have exponential running times.

The two problems have attracted attention in the special cases with (1) proportional linear deterioration, i.e. $p_i(t) = \beta_i s_i$ (eq. $\alpha_i = 0$), for each $i \in \mathcal{J}$, and (2) fixed processing times, i.e. $p_i(s_i) = \alpha_i$ (eq. $\beta_i = 0$), for each $i \in \mathcal{J}$. In the former case, the makespan problem $1|r_i, p_i(s_i) = \beta_i s_i|C_{\max}$ is optimally solvable in $O(n \log n)$ time [17, 18], while the best known algorithm for the total completion problem $1|r_i, p_i(s_i) = \beta_i s_i| \sum C_i$ is $(1 + \beta_{\max})$-approximate [3]. In the latter case, makespan problem is polynomially solvable via a greedy algorithm [14], while the total completion time problem is strongly $\mathcal{NP}$-hard, admitting a Polynomial-Time Approximation Scheme (PTAS) and greedy constant-factor approximation algorithms [1, 13].

In addition to the above, there exist various complexity and approximation results for problem generalizations (e.g. multiprocessor environments), relaxations (e.g. preemptive versions), and variants (e.g. step and position-dependent processing time functions). A survey relevant to time-dependent scheduling algorithms can be found in [9]. The most recent description of

Scheduling Problem	Complexity	Best Known Algorithm	Ref.
Linear Deterioration			
$1\|p_i(s_i) = \alpha_i + \beta_i s_i\|C_{\max}$	$\mathcal{P}$	$O(n \log n)$-time	[10]
$1\|r_i, p_i(s_i) = \alpha_i + \beta_i s_i\|C_{\max}$	$\mathcal{NP}$-hard	Exp.	[4]
$1\|p_i(s_i) = \alpha_i + \beta_i s_i\|\sum C_i$	?	Exp.	[9]
$1\|r_i, p_i(s_i) = \alpha_i + \beta_i s_i\|\sum C_i$	$\mathcal{NP}$-hard	Exp.	[16]
Uniform Deterioration			
$1\|r_i, p_i(s_i) = \alpha_i + \beta s_i\|C_{\max}$	$\mathcal{NP}$-hard	$(1 + 1/\beta)$-approx.	[*]
$1\|r_i, p_i(s_i) = \alpha_i + \beta s_i\|\sum C_i$	$\mathcal{NP}$-hard	$(1 + 1/\beta^2)$-approx.	[*]
Proportional Deterioration			
$1\|r_i, p_i(s_i) = \beta_i s_i, pmtn\|C_{\max}$	$\mathcal{P}$	$O(n \log n)$-time	[19]
$1\|r_i, p_i(s_i) = \beta_i s_i\|C_{\max}$	$\mathcal{P}$	$O(n \log n)$-time	[17, 18]
$1\|r_i, p_i(s_i) = \beta_i s_i, pmtn\|\sum C_i$	$\mathcal{P}$	$O(n \log n)$-time	[19]
$1\|r_i, p_i(s_i) = \beta_i s_i\|\sum C_i$	$\mathcal{NP}$-hard	$(1 + \beta_{\max})$-approx	[17]

Table 1: Algorithmic results for single-machine time-dependent scheduling problems. The stars [*] indicate results obtained in the current manuscript.

approximation algorithms and approximation schemes for time-dependent scheduling problems appears in Chap. 14 of [8]. Table 1 summarizes results closely related to this manuscript. Additionally, there exist investigations on interesting time-dependent scheduling applications, including production and delivery scheduling [11], defending aerial threats [12], fire fighting [20], and personel scheduling [22]. A comprehensive survey of time-dependent scheduling applications is given in Sect. 6.6 of the monograph [8].

Contributions and paper organization. Despite the aforementioned literature, the approximability of the single-machine time-dependent scheduling problems with jobs released over time, linear deterioration, the makespan and total completion time objectives remains unsettled. This manuscript focusses on the special case with uniform deterioration, i.e. the problems $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$ and $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|\sum C_i$. To the authors knowledge, no approximation algorithms are known for those. Our main contribution is the analysis of greedy algorithms and the derivation of approximation results based on a new approach for bounding time-dependent scheduling problems using pseudomatchings. The proposed algorithms can also be viewed as scheduling rules. Section 3 is devoted to the makespan problem and Section 4 covers the total completion time problem. The manuscript proceeds as follows.

Section 2 introduces some terminology, expresses the makespan of a feasible schedule as a function of the job processing times and idle periods, introduces our pseudomatching concepts, and demonstrates their bounding properties. Section 3 describes and analyzes three approximation algorithms that we call *Non-Idling* [5], *Non-Interfering* [13], and *Earliest Completion Time First (ECTF)*. Sections 3.1-3.2 prove that the Non-Idling and Non-Interfering algorithms achieve constant approximation ratios for the special cases of the problem with $\beta \leq 1/n$ and $\beta \geq n+1$, respectively. Nevertheless, we show that both algorithms are $\Omega((1+\beta)^n)$-approximate in the worst case. Section 3.4 shows that ECTF achieves an $(3+1/\beta)$-approximation ratio for $1|r_j, p_j(s_j) = \alpha_j + \beta s_j|C_{\max}$. Section 4 extends an approximation equivalence relationship [21] to the time-dependent scheduling context. On one hand, we show that any ρ-approximation algorithm for the total completion time problem $1|r_j, p_j(s_j) = \alpha_j + \beta s_j|\sum C_i$ is $(1+\rho)$-approximate for makespan problem $1|r_j, p_j(s_j) = \alpha_j + \beta s_j|C_{\max}$. On the other hand, we show that ρ-approximation algorithm for the latter is $(1+1/\beta)\rho$-approximate for the former. This last finding implies the existence of a $O(1+1/\beta^2)$-approximation algorithm for $1|r_j, p_j(s_j) = \alpha_j + \beta s_j|\sum C_i$.

2. Preliminaries

Next, we introduce some terminology, express the makespan of a feasible schedule as a weighted sum of the fixed processing times and gap lengths, and present the pseudomatching concepts and their bounding properties.

Additional Terminology. Given two jobs $i, j \in \mathcal{J}$, if $\alpha_i < \alpha_j$, then we say that i is *shorter* than j and that j is *longer* than i. If job $i \in \mathcal{J}$ is released at time r_i, then it may only begin processing at $s_i \geq r_i$. W.l.o.g. $r_{\min} = \min_{i\in\mathcal{J}}\{r_i\} = 0$. Given a time t, denote by $\mathcal{P}(t)$ the set of *pending jobs*, i.e. the ones which have been released but have not begun processing before t. At each time t that the machine becomes available, a feasible schedule specifies the next job in $\mathcal{P}(t)$ to begin from time t and onward.

2.1. Makespan Expression.

Due to release times, optimal schedules may require *gaps*, i.e. maximal idle time intervals during which no job is processed (Figure 1). Consider a feasible schedule $\mathcal{S}$ and number the jobs in increasing order $s_1 < \ldots < s_n$ of their start times. Denote the gap between jobs $i-1$ and i by $q_i = s_i - C_{i-1}$, for $i \in \{1, \ldots, n\}$, where $C_0 = 0$. If $q_i = 0$, then there is no idle period between jobs $i-1$ and i. Lemma 1 expresses the makespan of a feasible schedule w.r.t. gaps and fixed processing times. This is an adaptation of standard expressions in the time-dependent scheduling literature [9], but now accounts for gaps because release times. Lemma 2 derives an alternative expression of the fixed processing time contributions to the makespan.

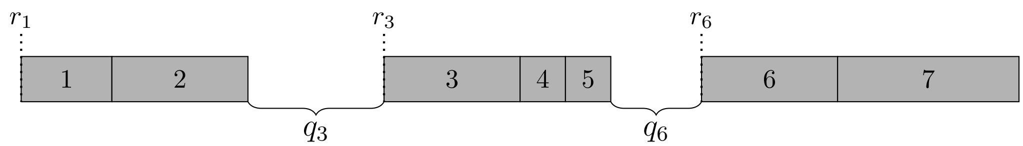

Figure 1: Illustration of feasible schedule with seven jobs and two gaps, e.g. during the time interval $[C_2, s_3)$. There is an optimal schedule such that, if job i begins right after a gap, i.e. $q_i > 0$, then $s_i = r_i$.

Lemma 1. *Consider a feasible schedule $\mathcal{S}$ and suppose that the jobs are numbered in increasing order of their start times in $\mathcal{S}$, i.e. $s_1 \leq \ldots \leq s_n$. Then, the makespan of $\mathcal{S}$ is:*

$$T = \sum_{i=1}^{n} (1+\beta)^{n-i+1} q_i + \sum_{i=1}^{n} (1+\beta)^{n-i} \alpha_i$$

Proof. We show by induction on $k \in \{1, \ldots, n\}$ that $C_k = \sum_{i=1}^{k} (1+\beta)^{k-i}[(1+\beta)q_i + \alpha_i]$. For the induction basis, it clearly holds that $C_1 = (1+\beta)q_1 + \alpha_1$, since job 1 begins at time $s_1 = q_1$. For the induction step, suppose that the equality is true with index $k-1$. Using the fact that $s_k = C_{k-1} + q_k$ and the induction hypothesis:

$$\begin{aligned} C_k &= (1+\beta)[C_{k-1} + q_k] + \alpha_k \\ &= (1+\beta)\left[\sum_{i=1}^{k-1} (1+\beta)^{(k-1)-i}\left[(1+\beta)q_i + \alpha_i\right] + q_k\right] + \alpha_k \\ &= \sum_{i=1}^{k} (1+\beta)^{k-i}\left[(1+\beta)q_i + \alpha_i\right] \end{aligned}$$

□

Lemma 1 has the following implications. First, if all jobs begin at time t and are executed without any gap between them, then they complete at $T = (1+\beta)^n t + \sum_{i=1}^{n}(1+\beta)^{n-i}\alpha_i$. Second, when all jobs have equal release times, there exists always an optimal schedule without gaps and greedily scheduling the jobs in non-decreasing order $\alpha_1 \leq \ldots \leq \alpha_n$ of their fixed processing times is optimal [9]. Third, for any subset $\mathcal{J}' = \{\gamma(1), \ldots, \gamma(k)\}$ of jobs sorted in non-decreasing order $\alpha_{\gamma(1)} \leq \ldots \leq \alpha_{\gamma(k)}$ of their fixed processing times and executed consecutively without gaps starting at time t, we get the lower bound $T \geq (1+\beta)^k t + \sum_{i=1}^{k}(1+\beta)^{k-i}\alpha_{\gamma(i)}$ on the makespan of any feasible schedule.

Lemma 2. *Consider a feasible schedule $\mathcal{S}$ and number the jobs in increasing order $s_1 \leq \ldots \leq s_n$ of their start times in $\mathcal{S}$. Then, $\mathcal{S}$ has fixed processing time cost* $\sum_{i=1}^{n}(1+\beta)^{n-i}\alpha_i = \sum_{i=1}^{n}\alpha_i + \sum_{k=2}^{n}\beta(1+\beta)^{n-k}\left(\sum_{i=1}^{k-1}\alpha_i\right)$.

Proof. Consider a job $i \in \mathcal{J}$. Because i is scheduled in the i-th position of $\mathcal{S}$, its execution increases the start time, and therefore the processing time, of every job in the set $\{i+1, , \ldots, n\}$. Specifically, the processing time of job $i+1$ is increased by $\beta\alpha_i$, of job $i+2$ by $\beta(1+\beta)\alpha_i$ and so on. That is, the processing time of job n is increased by $\beta(1+\beta)^{n-i-1}\alpha_i$. Hence, the overall contribution of α_i to the makespan is $1 + \sum_{j=1}^{n-i}\beta(1+\beta)^{j-1}\alpha_i = (1+\beta)^{n-i}\alpha_i$. Using this geometric series sum and Lemma 1, the fixed processing time distribution to $\mathcal{S}$ can be expressed $A = \sum_{i=1}^{n}\alpha_i + \sum_{k=1}^{n-1}\left(\sum_{i=1}^{n-k}\beta(1+\beta)^{n-k-i}\right)\alpha_k$. Next, we rearrange the sum so that fixed processing time terms α_i with the same weight $\beta(1+\beta)^{n-k}$ are grouped together, for $i \in \mathcal{J}$ and $k \in \{2, \ldots, n\}$. That is, $A = \sum_{i=1}^{n}\alpha_i + \sum_{k=2}^{n}\beta(1+\beta)^{n-k}\left(\sum_{i=1}^{k-1}\alpha_i\right)$. □

2.2. Bounding Pseudomatchings

To analyze the performance of our algorithms for $1|r_j, p_j(t) = \alpha_j + \beta t|C_{\max}$, we need an approach for upper and lower bounding the (fixed processing time) load completed by a feasible and an optimal schedule, respectively, up to any time t. To this end, we introduce the *ρ-pseudomatching* and *weak pseudomatching* concepts that allow bounding sums and geometric series incorporating the β parameter, respectively. Definition 3 defines the so-called *bounding graph* that is used for comparing schedules computed by algorithms with optimal schedules. Definition 4 and Lemma 5 summarize a core argument used for analyzing the Non-Interfering algorithm (Section 4).

Definition 6 and Lemma 7 describe a main argument in the analysis of the Earliest Completion Time First algorithm (Section 5). The main technical difficulty in deriving approximation bounds (Sections 4-5) is showing the existence of these pseudomatchings for a schedule computed by an algorithm.

Definition 3 (Bounding graph)**.** *Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ and $\mathcal{O} = \{o_1, \ldots, o_k\}$ be two equal-cardinality indexed sets of positive real numbers. We refer to the complete bipartite graph $G = (\mathcal{A} \cup \mathcal{O}, \mathcal{A} \times \mathcal{O})$ as the* bounding graph *of $\mathcal{A}$ and $\mathcal{O}$.*

Definition 4 (ρ-pseudomatching)**.** *Given two equal-cardinality indexed sets $\mathcal{A}$ and $\mathcal{O}$ of positive real numbers and their bounding graph G, we say that a subset $M \subseteq \mathcal{A} \cup \mathcal{O}$ of edges is a ρ-*pseudomatching *if the following properties hold:*

4.1 Each node $a_i \in \mathcal{A}$ appears exactly once as an endpoint of a M edge.

4.2 Each node $o_j \in \mathcal{O}$ appears at most ρ times as an endpoint of a M edge.

4.3 For each $(a_i, o_j) \in M$, it holds that $a_i \leq o_j$.

Lemma 5. *Consider two equal-cardinality indexed sets $\mathcal{A}$ and $\mathcal{O}$ of positive real numbers. If the corresponding bounding graph G admits a ρ-pseudomatching M, then:*

$$\sum_{a_i \in \mathcal{A}} a_i \leq \rho \left[\sum_{o_j \in \mathcal{O}} o_j \right].$$

Proof. Denote by $\mathcal{A}_j = \{a_i : (a_i, o_j) \in M\}$ the subset of $\mathcal{A}$ elements matched with element $o_j \in \mathcal{O}$ in M. Because of Property 4.1, each a_i is matched exactly once, thus $\sum_{a_i \in \mathcal{A}} a_i = \sum_{o_j \in \mathcal{O}} \sum_{a_i \in \mathcal{A}_j} a_i$. Due to Properties 4.2-4.3, we have that $\sum_{a_i \in \mathcal{A}_j} a_i \leq \rho \cdot o_j$, for each $o_j \in \mathcal{O}$. Therefore, we conclude that $\sum_{a_i \in \mathcal{A}} a_i \leq \rho [\sum_{o_j \in \mathcal{O}} o_j]$. □

Definition 6 (Weak pseudomatching)**.** *Given two equal-cardinality indexed sets $\mathcal{A}$ and $\mathcal{O}$ of positive real numbers and their bounding graph G, we say that a subset $M \subseteq \mathcal{A} \cup \mathcal{O}$ of edges is a* weak pseudomatching *if the following hold:*

6.1 Each node $a_i \in \mathcal{A}$ appears exactly once as an endpoint of a M edge.

6.2 For each $(a_i, o_j) \in M$, it holds that $i > j$ and $a_i \leq o_j$.

Lemma 7. *Consider two equal-cardinality indexed sets $\mathcal{A}$ and $\mathcal{O}$ of positive real numbers. If the corresponding bounding graph G admits a weak pseudo-matching M, then:*

$$\sum_{a_i \in \mathcal{A}} (1+\beta)^{n-i} a_i \leq \left(1 + \frac{1}{\beta}\right) \left[\sum_{o_j \in \mathcal{O}} (1+\beta)^{n-j} o_j\right].$$

Proof. Denote by $\mathcal{A}_j = \{a_i : (a_i, o_j) \in M\}$ the subset of $\mathcal{A}$ elements matched with element $o_j \in \mathcal{O}$ in M. By Property 6.1, it must be the case that $\sum_{a_i \in \mathcal{A}} (1+\beta)^{n-i} a_i = \sum_{o_j \in \mathcal{O}} \sum_{a_i \in \mathcal{A}_j} (1+\beta)^{n-i} a_i$. Due to Property 6.2, we have that $\sum_{a_i \in \mathcal{A}_j} (1+\beta)^{n-i} a_i \leq \sum_{i=j+1}^{n} (1+\beta)^{n-i} \max_{a_i \in \mathcal{A}_j} \{a_i\} \leq \sum_{i=j+1}^{n} (1+\beta)^{n-i} o_j = (1+\frac{1}{\beta})(1+\beta)^{n-j} o_j$, for each $o_j \in \mathcal{O}$, where the last equality follows from a standard geometric series sum calculation. We conclude that $\sum_{a_i \in \mathcal{A}} (1+\beta)^{n-i} a_i \leq \left(1 + \frac{1}{\beta}\right) \left(\sum_{o_j \in \mathcal{O}} (1+\beta)^{n-j} o_j\right)$. □

3. Approximation Algorithms

This section investigates the two greedy *non-interfering* and *non-idling* algorithms that have been proposed for special cases and variants of our problem. We show that the non-interfering algorithm achieves a constant approximation ratio for instances with $\beta > n+1$, but is $\Omega((1+\beta)^n)$-approximate for general instances. Next, we argue that the non-idling algorithm attains a constant factor approximation ratio for instances with $\beta \leq 1/n$, but is $\Omega((1+\beta)^n)$-approximate for arbitrary instances. Finally, we prove that returning the best of the two schedules computes a 2-approximate solution for instances with two distinct release times.

3.1. Non-Interfering Algorithm

Given a feasible schedule $\mathcal{S}$ for an instance $\mathcal{J}$ of the problem and two jobs $i, j \in \mathcal{J}$, we say that *job i interferes with job j* time t in $\mathcal{S}$ if $s_i = t$, $\alpha_i > \alpha_j$ and $t < r_j < (1+\beta)t + \alpha_i$, i.e. the situation where a longer job i begins at a time t before the release time r_j a shorter job j in $\mathcal{S}$ and i completes after r_j, which can be avoided with an idle period during $[t, r_j)$. Clearly, jobs i and j have start times $s_i < s_j$ in $\mathcal{S}$. In such a case, we say that i is an *interfering job* in $\mathcal{S}$. Algorithm 1 constructs a schedule without interfering jobs.

Algorithm 1 (Non-Interfering)**.** *At each time t that the machine becomes available, schedule a pending job $i = \arg\min_{k \in \mathcal{P}(t)}\{\alpha_k\}$ with minimal fixed processing time, unless this job is interfering, i.e. there exists a job j such that $\alpha_j < \alpha_i$ and $t < r_j < (1+\beta)t + \alpha_i$. In this case, introduce an idle period during $[t, r_j)$ and proceed with time $t = r_j$.*

Next, we proceed with Lemma 8 and Observation 9, which simplify the proof of Lemma 10 (as we do not need to account for gaps). Starting from an instance $\mathcal{J}$, Lemma 8 defines another instance $\widetilde{\mathcal{J}}$ such that the non-interfering schedules execute the jobs in the same order in the two instances and the non-interfering schedule for $\widetilde{\mathcal{J}}$ does not contain gaps. Observation 9 shows that the job order uniquely characterizes an optimal schedule.

Lemma 8. *Consider an arbitrary instance $\mathcal{J} = \{1, \ldots, n\}$ for which the non-interfering algorithm produces a schedule $\mathcal{S}$ with gaps. Number the jobs in increasing order $s_1 < \ldots < s_n$ of their start times in $\mathcal{S}$. Starting from $\mathcal{J}$, construct a different instance $\widetilde{\mathcal{J}}$ with the same number of jobs, i.e. $|\mathcal{J}| = |\widetilde{\mathcal{J}}|$. Each job $k \in \widetilde{\mathcal{J}}$ has fixed processing time $\tilde{\alpha}_k = \alpha_k$ and release time $\tilde{r}_k = \min\{r_k, \sum_{i=1}^{k-1}(1+\beta)^{(k-1)-i}\alpha_i\}$, where α_k and r_k are the original parameters of $\mathcal{J}$. The non-interfering algorithm executes the jobs in the same order in $\mathcal{J}$ and $\widetilde{\mathcal{J}}$, and produces a schedule without gaps, i.e. $q_i = 0$, for each $i \in \{1, \ldots, n\}$, for $\widetilde{\mathcal{J}}$.*

Proof. Starting from $\mathcal{S}$, the new problem instance $\widetilde{\mathcal{J}}$ is constructed so that $|\widetilde{\mathcal{J}}| = |\mathcal{J}|$, by rounding release times down. In particular, we set a new release time $\tilde{r}_k = \min\{r_k, \sum_{i=1}^{k-1}(1+\beta)^{(k-1)-i}\alpha_i\}$ and fixed processing time $\tilde{\alpha}_k = \alpha_k$, for each $k \in \widetilde{\mathcal{J}}$. Consider the schedule $\widetilde{\mathcal{S}}$ for $\widetilde{\mathcal{J}}$ obtained by executing the jobs in the same order with $\mathcal{S}$, but without gaps and denote the makespan of $\widetilde{\mathcal{S}}$ by $\widetilde{T}$. By construction, job $k \in \widetilde{\mathcal{J}}$ has start time $\tilde{s}_k = \sum_{i=1}^{k-1}(1+\beta)^{(k-1)-i}\tilde{\alpha}_i \geq \tilde{r}_k$, i.e. the new release times are not violated, in $\widetilde{\mathcal{S}}$. Next, consider any pair $k, \ell \in \widetilde{\mathcal{J}}$ of jobs such that $k < \ell$ and $\tilde{\alpha}_k > \tilde{\alpha}_\ell$. Since $\mathcal{S}$ is non-interfering, we have that $C_k < r_\ell$, which implies that $\sum_{i=1}^{k-1}(1+\beta)^{(k-1)-i}\alpha_i \leq r_\ell$. Given that $k < \ell$, we conclude that $\tilde{C}_k \leq \tilde{r}_\ell$, i.e. $\widetilde{\mathcal{S}}$ is non-interfering for $\widetilde{\mathcal{J}}$. □

Observation 9. *Consider an arbitrary order $\gamma(\cdot)$ of the jobs, where $\gamma(k) \in \mathcal{J}$ is the job in the k-th position of the order, for $k \in \{1, \ldots, n\}$. Suppose that there exists an optimal schedule $\mathcal{S}^*$ executing the jobs according to $\gamma(\cdot)$, i.e. $s^*_{\gamma(1)} < \ldots < s^*_{\gamma(n)}$, where s^*_i is the start time of job $i \in \mathcal{J}$ in $\mathcal{S}^*$. W.l.o.g. it holds that $s_{\gamma(i)} = \max\{r_{\gamma(i)}, C_{\gamma(i-1)}\}$, for $i \in \{1, \ldots, n\}$, where $C_{\gamma(0)} = 0$.*

Number the jobs in increasing order $s_1 < \ldots < s_n$ of their start times in the schedule $\mathcal{S}$ produced by the non-interfering algorithm. Job $k \in \mathcal{J}$ is executed in the k-th position of $\mathcal{S}$. Next, consider an optimal schedule $\mathcal{S}^*$ and let $\gamma(k) \in \mathcal{J}$ be the job executed in the k-th position of $\mathcal{S}^*$, for $k \in \{1, \ldots, n\}$. We say that $k \in \mathcal{J}$ is a *critical job* if $C_k \leq C^*_{\gamma(k)}$, where C^*_i is the completion time of job $i \in \mathcal{J}$ in $\mathcal{S}^*$. Lemma 10 upper bounds the fixed processing times of jobs executed after the last critical job in $\mathcal{S}$ based on pseudomatchings.

Lemma 10. *Consider a non-interfering schedule $\mathcal{S}$ and let $\ell = \max\{k : C_k \leq C^*_{\gamma(k)}, k \in \mathcal{J}\}$ be the last critical job. For each $k \in \{\ell+1, \ldots, n\}$, it holds that $\sum_{i=\ell+1}^{k} \alpha_i \leq 2\left[\sum_{j=1}^{k} \alpha_{\gamma(j)}\right]$.*

Proof. To prove the lemma, we may assume w.l.o.g. that $\mathcal{S}$ does not contain gaps. Otherwise, if $\mathcal{S}$ contains gaps, starting from the original instance $\mathcal{J}$, we may consider the modified instance $\widetilde{\mathcal{J}}$ obtained according to Lemma 8. Using Observations 9 and the orders of the jobs in the non-interfering schedule $\mathcal{S}$ and in an optimal schedule $\mathcal{S}^*$ for $\mathcal{J}$, we can obtain two feasible schedules $\widetilde{\mathcal{S}}$ and $\widetilde{\mathcal{S}}^*$, respectively, for $\widetilde{\mathcal{J}}$. By Lemma 8, $\widetilde{\mathcal{S}}$ is the schedule produced by the non-interfering algorithm for $\widetilde{\mathcal{J}}$ and does not contain any gaps. Therefore, proving the lemma with $\widetilde{\mathcal{S}}$ and $\widetilde{\mathcal{S}}^*$ implies that the lemma holds for the original instance $\mathcal{J}$. We note that the optimal schedule may change for $\widetilde{\mathcal{J}}$, but this does not affect our argument since we compare a non-interfering schedule without gaps with an arbitrary feasible schedule. In the remainder of the proof, assume that $q_i = 0$, for each $i \in \mathcal{J}$, in $\mathcal{S}$.

For each $k \in \{1, \ldots, n\}$, define the sets $\mathcal{A}_k = \{1, \ldots, k\}$ and $\mathcal{O}_k = \{\gamma(1), \ldots, \gamma(k)\}$ of jobs executed in the first k positions of $\mathcal{S}$ and $\mathcal{S}^*$, respectively. Further, for each $k > \ell$, denote by $\mathcal{A}_k^- = \{\ell+1, \ldots, k\}$ the subset of the $\mathcal{A}_k$ jobs executed after the last critical job ℓ in $\mathcal{S}$. For simplicity of the presentation, we denote a job in $\mathcal{A}_k$ by its actual index i and a job in $\mathcal{O}_k$ by $\gamma(j)$ (i.e. using the $\gamma(\cdot)$ notation), for $i, j \in \{1, \ldots, k\}$. Deriving the lemma is equivalent to showing that $\sum_{i \in \mathcal{A}_k^-} \alpha_i \leq 2\left[\sum_{\gamma(j) \in \mathcal{O}_k} \alpha_{\gamma(j)}\right]$.

For each $k \in \{1, \ldots, n\}$, consider the bounding (complete bipartite) graph $G_k = (\mathcal{A}_k \cup \mathcal{O}_k, \mathcal{A}_k \times \mathcal{O}_k)$ with $2k$ nodes: a node for each of the k jobs in $\mathcal{A}_k$ and a node for each of the k jobs in $\mathcal{O}_k$. Note that, if there exist $i \in \mathcal{A}_k$ and $\gamma(j) \in \mathcal{O}_k$ such that $i = \gamma(j)$, then we introduce two nodes for job i, i.e. a node in each side of the bipartition. The graph contains all possible k^2 edges with one endpoint in $\mathcal{A}_k$ and the other in $\mathcal{O}_k$. Using standard terminology, a matching in G_k is a subset $M_k \subseteq \mathcal{A}_k \times \mathcal{O}_k$ of edges without a common endpoint. If $(i, \gamma(j)) \in M_k$, then we say that the nodes $i \in \mathcal{A}_k$ and $\gamma(j) \in \mathcal{O}_k$ are *matched* by M_k. By relaxing the notion of a matching, we refer to a set M_k of edges in G_k as a *ρ-pseudomatching* if every node $i \in \mathcal{A}_k$ appears at most once as an endpoint of an edge in M_k and every node $\gamma(j) \in \mathcal{O}_k$ appears at most ρ times as an edge point of an edge in M_k, where $\rho \in \mathbb{Z}^+$ is a positive integer. Let $M_k(\mathcal{A}_k) = \{i : (i, \gamma(j)) \in M_k, i \in \mathcal{A}_k, \gamma(j) \in \mathcal{O}_k\}$ be the subset of the $\mathcal{A}_k$ nodes appearing as an endpoint of an edge in M_k. Next, for each $k \in \{\ell + 1, \ldots, n\}$, we show the existence of a 2-pseudomatching M_k in G_k with the following properties:

1. $M_k(\mathcal{A}_k) = \mathcal{A}_k^-$, i.e. each job $i \in \mathcal{A}_k^-$ appears exactly once as the endpoint of an edge in M_k and no other $\mathcal{A}_k$ node is matched.
2. For every job $i \in \mathcal{A}_k^-$ such that there exists a job $\gamma(j) \in \mathcal{O}_k$ with $i = \gamma(j)$, we have that $(i, \gamma(j)) \in M_k$. That is, every job i which is executed in the $\{\ell + 1, \ldots, k\}$ positions of $\mathcal{S}$ and the first k positions of $\mathcal{S}^*$ must be matched with itself in M_k.
3. Every job $i \in \mathcal{A}_k^-$ which is not executed in the first k positions of $\mathcal{S}^*$, i.e. $\gamma(j) = i$ for some $\gamma(j) \notin \mathcal{O}_k$, must be matched with a job $\gamma(j) \in \mathcal{O}_k \setminus \mathcal{A}_k$ in M_k.
4. Each job $\gamma(j) \in \mathcal{O}_k \setminus \mathcal{A}_k$ is matched with at most one job in $\mathcal{A}_k \setminus \mathcal{O}_k$.
5. If $(i, \gamma(j)) \in M_k$, for some pair of jobs $i \in \mathcal{A}_k$ and $\gamma(j) \in \mathcal{O}_k$, then $\alpha_i \leq \alpha_{\gamma(j)}$.

We refer to a pseudomatching satisfying the above properties as a *2-pseudomatching*. If such a pseudomatching exists, then it clearly holds that $\sum_{i \in \mathcal{A}_k^-} \alpha_i \leq 2 \left[\sum_{\gamma(j) \in \mathcal{O}_k} \alpha_{\gamma(j)}\right]$: each $\mathcal{A}_k^-$ job is matched exactly once with an $\mathcal{O}_k$ job and each $\mathcal{O}_k$ job is matched at most two $\mathcal{A}_k^-$ jobs. We will show its existence by induction on $k \in \{\ell + 1, \ldots, n\}$.

For the induction basis, consider the case $k = \ell + 1$. If $\ell + 1 \in \mathcal{O}_k$, then $\gamma(j) = \ell + 1$, for some $j \in \{1, \ldots, \ell + 1\}$. Clearly, $\mathcal{M}_{\ell+1} = \{(\ell + 1, \gamma(j))\}$ is a 2-pseudomatching, given that $\alpha_{\ell+1} = \alpha_{\gamma(j)}$. If $\ell + 1 \notin \mathcal{O}_k$, then, by using a simple pigeonhole principle argument [a similar, but more elaborate, pigeonhole argument is rigorously presented in the proof of Theorem 15], there exists a job $\gamma(j) > \ell + 1$ such that $j \in \{1, \ldots, \ell + 1\}$. Since $\ell + 1$ is not critical, we have that $\alpha_{\ell+1} \leq \alpha_{\gamma(j)}$. Otherwise, if $\alpha_{\ell+1} > \alpha_{\gamma(j)}$, by the way the non-interfering algorithm works and the fact that $\ell + 1 < \gamma(j)$, we would have $C_{\ell+1} \leq r_{\gamma(j)} < C^*_{\gamma(\ell+1)}$, which would contradict that $\ell + 1$ is not critical. We conclude that $M_{\ell+1} = \{(\ell + 1, \gamma(j))\}$ is a 2-pseudomatching.

For the induction step, assume that G_k admits a 2-pseudomatching M_k. We will convert M_k into a 2-pseudomatching M_{k+1} for G_{k+1}. This update involves the following two steps.

In the first step, we adapt M_k based on the job $\gamma(k+1)$ executed in the $(k+1)$-th position of $\mathcal{S}^*$ to obtain an intermediate 2-pseudomatching $\widetilde{M}_{k+1}$. Suppose that $\gamma(k+1) = i$. If $i \notin \mathcal{A}_k^-$, then we set $\widetilde{M}_{k+1} = M_k$. Otherwise, if $i \in \mathcal{A}_k^-$, i.e. $\mathcal{S}$ completes job i in the positions $\{\ell+1, \ldots, k\}$, then we need to update M_k so as to satisfy Properties 1-2 in the resulting 2-pseudomatching M_{k+1}. Since $\gamma(k+1) \notin \mathcal{O}_k$, by the induction hypothesis, job i is matched with exactly one job $\gamma(j) \in \mathcal{O}_k \setminus \mathcal{A}_k$ in M_k. We set $\widetilde{M}_{k+1} = (M_k \cup \{(i, \gamma(k+1))\}) \setminus \{(i, \gamma(j))\}$, that is we remove $(i, \gamma(j))$ from M_k and add $(i, \gamma(k+1))$ to obtain $\widetilde{M}_{k+1}$. In this way, $i \in \mathcal{A}_k^-$ is now matched with job $\gamma(k+1)$, i.e. itself, in the $\mathcal{O}_k$ side of G_k.

In the second step, we adapt $\widetilde{M}_{k+1}$ so as to have job $k+1$ matched with some $\mathcal{O}_{k+1}$ job in M_{k+1}. We distinguish two cases based on whether $(k+1) \in \mathcal{O}_{k+1}$, or $(k+1) \notin \mathcal{O}_{k+1}$. In the former case, there exists $j \in \{1, \ldots, k+1\}$ s.t. $\gamma(j) = k+1$. Based on Property 2, we set $M_{k+1} = \widetilde{M}_{k+1} \cup \{(k+1, \gamma(j))\}$, i.e. job $(k+1) \in \mathcal{A}_{k+1}^-$ is matched with itself in the $\mathcal{O}_{k+1}$ side. In the latter case, it holds that $k+1 \in \mathcal{A}_{k+1} \setminus \mathcal{O}_{k+1}$. Let $x = |\mathcal{A}_{k+1} \setminus \mathcal{O}_{k+1}|$ be the number of jobs executed in the first $k+1$ positions of $\mathcal{S}$ and after the first $k+1$ positions of $\mathcal{S}^*$. A simple set theoretic argument implies that $|\mathcal{O}_{k+1} \setminus \mathcal{A}_{k+1}| = x$. By the induction hypothesis and Properties 3-4, we conclude that each $\mathcal{O}_k \setminus \mathcal{A}_k$ is matched with at most one $\mathcal{A}_k \setminus \mathcal{O}_k$ job. Therefore, given that $i \in \mathcal{A}_{k+1} \setminus \mathcal{O}_{k+1}$, there exists a job $\gamma(j) \in \mathcal{O}_{k+1} \setminus \mathcal{A}_{k+1}$ which is not matched with any $\mathcal{A}_k$ job in $\widetilde{M}_{k+1}$. We set $M_{k+1} = \widetilde{M}_{k+1} \cup \{(i, \gamma(j))\}$ and guarantee that Properties 1-4 are satisfied. Next, we claim that $\alpha_{k+1} \leq \alpha_{\gamma(j)}$. Otherwise, if $\alpha_{k+1} > \alpha_{\gamma(j)}$, by the way the non-interfering algorithm works and the fact that $k+1 < \gamma(j)$ $(\gamma(j) \in \mathcal{O}_{k+1} \setminus \mathcal{A}_{k+1})$, we would have $C_{k+1} \leq r_{\gamma(j)} < C^*_{\gamma(k+1)}$, which would contradict that $k+1$ is not critical. □

Lemma 11 lower bounds the optimal makespan using release times.

Lemma 11. *Assume that the jobs are numbered so that $r_1 \leq \ldots \leq r_n$. Any optimal schedule $\mathcal{S}^*$ has makespan $T^* \geq \sum_{i=1}^{n} \beta^{n-i} r_i$.*

Proof. Denote by C_i and T the completion time of job $i \in \mathcal{J}$ and makespan, respectively, in schedule S. We prove by induction that $C_k \geq \sum_{i=1}^{k} \beta^{k-i} r_i$, for each $k \in \{1, \ldots, n\}$. For $k = 1$, it clearly holds that $C_1 \geq \beta s_1^* \geq r_1$.

Suppose that lemma is true for some $k \in \{1, \ldots, n-1\}$. By the fact that $s_{k+1} \geq \max\{r_{k+1}, C_k\}$ and the induction hypothesis:

$$C_{k+1} = (1+\beta)s_{k+1}+\alpha_{k+1} \geq r_{k+1}+\beta C_k \geq r_{k+1}+\beta\left[\sum_{i=1}^{k}\beta^{k-i}r_i\right] = \sum_{i=1}^{k+1}\beta^{(k+1)-i}r_i.$$

□

Theorem 12 presents bounds on the approximation ratio of the non-interfering algorithm.

Theorem 12. *Algorithm 1 is* $(3+e)$*-approximate for instances of the problem* $1|r_i, p_i(s_i) = \alpha_i + \beta s_i, \beta \geq n+1|C_{\max}$ *and* $\Omega((1+\beta)^n)$*-approximate for instances of* $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$.

Proof. Recall that the jobs are numbered in increasing order $s_1 < \ldots \leq s_n$ of their start times in the schedule $\mathcal{S}$ produced by the non-interfering algorithm and $\gamma(k) \in \mathcal{J}$ is the job executed in the k-th position of an optimal schedule $\mathcal{S}^*$, for $k \in \{1, \ldots, n\}$. Let $\ell = \max\{k : C_k \leq C^*_{\gamma(k)}\}$ be the last critical position. By Lemma 1, we have that $T = (1+\beta)^{n-\ell}C_\ell + \sum_{i=\ell+1}^{n}(1+\beta)^{n-i+1}q_i + \sum_{i=\ell+1}^{n}(1+\beta)^{n-i}\alpha_i$. Using Lemma 2, i.e. expanding the last sum of this expression with geometric series, we get that:

$$T = (1+\beta)^{n-\ell}C_\ell + \sum_{i=\ell+1}^{n}(1+\beta)^{n-i+1}q_i + \sum_{i=\ell+1}^{n}\alpha_i + \sum_{k=\ell+2}^{n}\beta(1+\beta)^{n-k}\left(\sum_{i=\ell+1}^{k-1}\alpha_i\right) \tag{1}$$

Consider job $i \in \mathcal{J}$. If $q_i > 0$, then job i begins at its release time r_i. That is, the gap of length q_i immediately preceding job i occurs exactly during the time interval $[r_i - q_i, r_i)$. Hence, $q_i \leq r_i$. Based on this observation, the obvious fact that $\sum_{i=\ell+1}^{n}\alpha_i \leq \sum_{i=1}^{n}\alpha_{\gamma(i)}$ and Lemma 10, Equation (1) implies that:

$$\begin{aligned} T \leq{}& (1+\beta)^{n-\ell}C^*_{\gamma(\ell)} + \sum_{i=\ell+1}^{n}\left(1+\frac{1}{\beta}\right)^{n-i+1}\beta^{n-i+1}r_i \\ &+2\left[\sum_{i=1}^{n}\alpha_{\gamma(i)} + \sum_{k=\ell+2}^{n}\beta(1+\beta)^{n-k}\left(\sum_{i=1}^{k-1}\alpha_{\gamma(i)}\right)\right] \end{aligned} \tag{2}$$

By the definition of $\gamma(\cdot)$, Lemma 2 and Lemma 11, we get that

$$T^* \geq \max\left\{(1+\beta)^{n-\ell}C^*_{\gamma(\ell)}, \sum_{j=1}^{n}\alpha_{\gamma(j)} + \sum_{k=2}^{n}\beta(1+\beta)^{n-k}\left(\sum_{i=1}^{k-1}\alpha_{\gamma(i)}\right), \sum_{i=1}^{n}\beta^{n-i+1}r_i\right\} \tag{3}$$

For $\beta \geq n+1$, we have that $(1+\frac{1}{\beta})^{n+1} \leq e$. Therefore, Equations (2)-(3) imply that $T \leq (3+e)T^*$.

For the lower bound, consider an instance with n jobs, where $r_{\min} = B$, for some large constant $B = \omega(n)$. Job $j \in \{1, \ldots, n\}$ has $\alpha_j = B+n-j$ and $r_j = \sum_{i=1}^{j}(1+\beta)^{i-1}B$. We show by induction on j that no job begins before r_j in the algorithm's schedule $\mathcal{S}$. Given that $r_1 < \ldots < r_n$, our claim trivially holds for $j = 1$ because no job can be executed before $r_1 = \min_{i\in\mathcal{J}}\{r_i\}$. For the induction hypothesis, assume that our claim is true for some $j \geq 1$, i.e. no job begins before r_j in $\mathcal{S}$. Since $\alpha_1 \geq \ldots \geq \alpha_j$, any job beginning at time r_j would have completion time at least:

$$(1+\beta)r_j + \alpha_j \geq \sum_{i=1}^{j}(1+\beta)^{i-1}B + n - j > r_{j+1}.$$

So, the algorithm will not schedule any job during $[r_j, r_{j+1})$, because otherwise this job would be interfering. Our claim implies that the algorithm schedules all jobs according to shortest fixed processing time first starting from r_n and has makespan:

$$T = (1+\beta)^n r_n + \sum_{j=1}^{n}(1+\beta)^{n-j}B + \sum_{j=1}^{n}(1+\beta)^{n-j}(j-1) = \Omega((1+\beta)^{2n}B).$$

In an optimal schedule $\mathcal{S}^*$, all jobs begin at time $t = 0$ and are consecutively executed according without any idle period between them according to earliest release time first. The makespan of $\mathcal{S}^*$ is:

$$T^* = \sum_{j=1}^{n}(1+\beta)^{n-j}B + \sum_{j=1}^{n}(1+\beta)^{n-j}(n-j) = O((1+\beta)^n).$$

Hence, $T/T^* = \Omega((1+\beta)^n)$. □

3.2. Non-Idling Algorithm

Algorithm 2 constructs a feasible schedule by executing the shortest pending job whenever the machine becomes available.

Algorithm 2 (Non-Idling)**.** *Greedily schedule jobs over time, by initiating a pending job* $\arg\min_{i\in\mathcal{P}(t)}\{\alpha_i\}$ *with minimal fixed procesing time at each time* t *that the machine becomes available.*

Theorem 13. *Algorithm 2 is* $(1+e)$*-approximate for instances of the problem* $1|r_i, p_i(s_i) = \alpha_i + \beta s_i, \beta \leq 1/n|C_{\max}$ *with* $\beta \leq 1/n$ *and* $\Omega((1+\beta)^n)$*-approximate for instances of* $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$*.*

Proof. On the positive side, consider a schedule $\mathcal{S}$ produced by the non-idling algorithm and number the jobs in increasing order $s_1 < \ldots < s_n$ of their start times in $\mathcal{S}$. Let $Q = \sum_{i=1}^{n}(1+\beta)^{n-i+1}q_i$ and $A = \sum_{i=1}^{n}(1+\beta)^{n-i}\alpha_i$ be the gap-dependent and fixed processing time costs of $\mathcal{S}$, respectively. Next, we will show that $Q \leq T^*$ and $A \leq eT^*$, where T^* is the optimal makespan. By Lemma 1, we get that $T = Q + A \leq (1+e)T^*$.

To bound the gap-dependent cost of the algorithm, we show the existence of an optimal schedule $\mathcal{S}^*$ satisfying the property that, for each idle time interval $[t,u)$ in $\mathcal{S}$, the interval $[t,u)$ is also idle in $\mathcal{S}^*$. For simplicity, we prove the lemma for the case where $\mathcal{S}$ contains a single maximal idle time interval, i.e. gap $q_j > 0$, but the argument is naturally extended to an arbitrary number of gaps. We may partition $\mathcal{J}$ into the sets $\mathcal{J}_A = \{i \in \mathcal{J} : s_i \geq t\}$ and $\mathcal{J}_B = \{i \in \mathcal{J} : s_i < t\}$ of jobs beginning after and before, respectively, time t in $\mathcal{S}$. Using this definition and the fact that $[t,u)$ is idle in the algorithm's schedule, we conclude that $r_i \geq u$ for each $i \in \mathcal{J}_A$ and $r_i < t$ for every $i \in \mathcal{J}_B$. Let $\mathcal{S}_A^*$ and $\mathcal{S}_B^*$ be optimal schedules for $\mathcal{J}_A$ and $\mathcal{J}_B$ of makespans T_A^* and T_B^*, respectively. Clearly, the schedule $\mathcal{S}^*$ obtained by merging $\mathcal{S}_A^*$ and $\mathcal{S}_B$ is feasible and optimal for $\mathcal{J}$ given that $T^* = T_A^*$. Thus, $Q \leq T^*$. To bound the algorithm's fixed processing time cost, by using the standard Euler constant inequality $(1+\frac{1}{k})^k \leq e$, for each constant $k \geq 1$, we get that:

$$A = \sum_{i=1}^{n}(1+\beta)^{n-i}\alpha_i \leq (1+\beta)^n\left[\sum_{i=1}^{n}\alpha_i\right] \leq e\left[\sum_{i=1}^{n}\alpha_i\right] \leq eT^*.$$

On the negative side, consider an instance with $n = k+1$ jobs, namely a job of fixed processing time $\alpha_1 = B > 1$, release time $r_1 = 0$, and k jobs with $\alpha_i = 0$ and $r_i = 1$, for $i \in \{2,\ldots,k+1\}$. The non-idling schedule executes the jobs in increasing order of their indices, i.e. the first job completes at $C_1 = B$ and all remaining jobs are consecutively executed starting at C_1. By Lemma 1, $\mathcal{S}$ has makespan $T = (1+\beta)^k B$. In an optimal schedule $\mathcal{S}^*$, all short jobs are consecutively executed during $[1, (1+\beta)^k]$, the long job begins right after and completes at $T^* = (1+\beta)^{k+1} + B$. If $B = (1+\beta)^{k+1}$,

$$\frac{T}{T^*} = \frac{(1+\beta)^k B}{(1+\beta)^{k+1} + B} = \Omega((1+\beta)^k).$$

□

3.3. Best-of-Two Algorithm

The *best-of-two* algorithm returns the best among the non-idling and non-interfering schedules. Theorem 14 shows that this algorithm achieves a 2-approximation ratio when $r_i = \{0, r\}$ for each job $i \in \mathcal{J}$.

Theorem 14. *The best-of-two algorithm is 2-approximate for the problem $1|r_i \in \{0, r\}, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$ with two distinct release times.*

Proof. Consider an optimal schedule $\mathcal{S}^*$ of makespan T^*, in which job $i \in \mathcal{J}$ begins at s_i^* and completes at C_i^*. Further, denote by $k^* = |\{j \in \mathcal{J} : s_j^* < r\}|$ the number of jobs beginning before r and let γ be the order $s_{\gamma(1)}^* < \ldots < s_{\gamma(n)}^*$ of jobs in increasing start times, i.e. job $\gamma(i)$ is executed in the i-th position of $\mathcal{S}^*$. For a given subset $\mathcal{J}' = \{\pi(1), \ldots, \pi(k)\}$ of k jobs which are numbered so that $\alpha_{\pi(1)} \leq \ldots \leq \alpha_{\pi(k)}$, denote by $F(\mathcal{J}') = \sum_{i=1}^{k} (1+\beta)^{k-i} \alpha_{\pi(i)}$ their fixed processing time cost if they are continuously scheduled without gaps and other intermediate jobs in non-decreasing order of their fixed processing times. We distinguish two cases based on whether $C_{\gamma(k^*)}^* \leq r$ and $C_{\gamma(k^*)}^* > r$.

In the former case, since $n - k^*$ jobs begin after r in $\mathcal{S}^*$, Lemma 1 implies that $T^* \geq \max\{(1+\beta)^{n-k^*} r, F(\mathcal{J})\}$. Assume that the algorithm's non-interfering schedule $\mathcal{S}$ has makespan T and suppose that it associates a start time s_j and completion time C_j to each job $j \in \mathcal{J}$. Also, let $k = |\{j \in \mathcal{J} : s_j < r\}|$. We claim that $k \geq k^*$. Assume for contradiction that $k < k^*$. W.l.o.g. we may assume that $\alpha_{\gamma(1)} \leq \ldots \leq \alpha_{\gamma(k^*)}$, i.e. $\mathcal{S}^*$ schedules jobs in non-decreasing order of fixed processing times before r. Because $\mathcal{S}$ schedules the pending job with the shortest fixed processing time at each time that the machine becomes available, it must be the case that $\alpha_i \leq \alpha_{\gamma(i)}$, for $1 \leq i \leq k$. If $C_k < C_{\gamma(k^*)}^*$, then there exists a job $j \in \mathcal{J}$ such that $s_j^* < r \leq s_j$ which can be feasibly executed during $[C_k, r]$ in $\mathcal{S}$, i.e. a contradiction on the definition of k. If $C_k \geq C^*_{\gamma(k^*)}$, then there exist jobs $i, j \in \mathcal{J}$ such that $\alpha_i > \alpha_j$, $s_i < r \leq s_j$ and $s^*_i \geq r > s^*_j$, which contradicts the fact that the algorithm always schedules a pending job with a minimal processing time. Hence, our claim is true. By Lemma 1, if $\mathcal{J}' = \{j \in \mathcal{J} : s_j \geq r\}$, then $T = (1+\beta)^{n-k} r + F(\mathcal{J}') \leq (1+\beta)^{n-k^*} r + F(\mathcal{J}) \leq 2T^*$.

In the latter case, consider the non-idling schedule $\mathcal{S}$ of the algorithm, denote its makespan by T and the execution interval of each job $j \in \mathcal{J}$ by $[s_j, C_j]$. Let $t^* = \max_{j\in\mathcal{J}}\{C^*_j : s^*_j < r\}$ and $t = \max_{j\in\mathcal{J}}\{C_j : s_j < r\}$ be the completion time of the interfering job in $\mathcal{S}^*$ and $\mathcal{S}$, respectively. Similarly before, $T^* \geq \max\{(1+\beta)^{n-k^*} t^*, F(\mathcal{J})\}$. Given that $\mathcal{S}$ executes the jobs with the minimal fixed processing times until it encounters job k with $C_k > r$, we have that $k \geq k^*$. If $t \leq t^*$, then $T = (1+\beta)^{n-k} t + F(\mathcal{J}') \leq (1+\beta)^{n-k^*} t^* + F(\mathcal{J}) \leq 2T^*$, where $\mathcal{J}' = \{j \in \mathcal{J} : s_j \geq t\}$. If $t > t^*$, then $k \geq k^* - 1$, i.e. $T^* \geq (1+\beta)^{n-k-1} r$. Therefore, $T \leq (1+\beta)^{n-k-1} r + F(\mathcal{J}' \cup \{k\}) \leq 2T^*$. □

3.4. Earliest Completion-Time First

Next, we consider the *Earliest Completion Time First (ECTF)* algorithm and show that it is $O(1+\frac{1}{\beta})$-approximate. ECTF produces a schedule satisfying Observation 9. That is, if jobs are numbered in increasing order $s_1 < \ldots < s_n$ of their start times in $\mathcal{S}$, then job $i \in \mathcal{J}$ has start time $s_i = \max\{r_i, C_{i-1}\}$. At every time t, let $\Gamma_i(t) = (1+\beta)\max\{t, r_i\} + \alpha_i$ be the completion time of job $i \in \mathcal{J}$, if i is the next to be executed from time t and onward. In addition, denote by $\mathcal{F}(t) = \{i : i \in \mathcal{J}, C_i \leq t\}$ the set of completed jobs at time t in $\mathcal{S}$.

Algorithm 3 (ECTF)**.** *At each time t that the machine becomes available, schedule the uncompleted job* $\arg\min_{i\in\mathcal{J}\setminus\mathcal{F}(t)}\{\Gamma_i(t)\}$ *with the earliest completion time.*

Theorem 15. *Algorithm 3 achieves an approximation ratio $\rho \in [2, 3+\frac{1}{\beta}]$ for the problem $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$.*

Proof. We initially prove the upper bound. Denote the non-interfering schedule and an optimal schedule by $\mathcal{S}$ and $\mathcal{S}^*$, respectively. Number the jobs in increasing order $s_1 < \ldots < s_n$ of their start times in $\mathcal{S}$. That is, job $i \in \mathcal{J}$ is executed in the i-th position of $\mathcal{S}$. Let $\pi(i) \in \{1, \ldots, n\}$ be the position at which job $i \in \mathcal{J}$ is executed in $\mathcal{S}^*$. Analogously, denote by $\gamma(i) \in \mathcal{J}$ the job executed in the i-th position of $\mathcal{S}$, for $i \in \{1, \ldots, n\}$.

We partition the set $\mathcal{J}$ of jobs into the subset $\mathcal{W} = \{i : i \geq \pi(i)\}$ of *well-ordered jobs* whose position in $\mathcal{S}$ is greater than or equal to their position in $\mathcal{S}^*$ and the subset $\mathcal{I} = \{i : i < \pi(i)\}$ of *inverted jobs* executed at a strictly smaller position in $\mathcal{S}$ compared to their position in $\mathcal{S}^*$. Consider an arbitrary inverted job $i \in \mathcal{I}$ executed in a subsequent position $\pi(i) \in \{i+1, \ldots, n\}$ in $\mathcal{S}^*$. By a simple pigeonhole principle argument, a key observation is that there exists a job j executed after i in $\mathcal{S}$ and not later than the i-th position in $\mathcal{S}^*$, i.e. $\pi(j) \leq i < j$. Clearly, job j is well-ordered, i.e. $j \in \mathcal{W}$.

Consider the start times s_i and s_i^* of job $i \in \mathcal{J}$ and the immediately preceding gaps q_i and q_i^* in $\mathcal{S}$ and $\mathcal{S}^*$, respectively. Based on the previous observation, define the set $\mathcal{K}_I = \{i : i \in \mathcal{I}, \exists\ j \in \mathcal{W} \text{ s.t. } \pi(j) \leq i < j, r_j > s_i - q_i\}$ of *critical inverted jobs*. That is, for each job $k \in \mathcal{K}_I$, there exists a well-ordered job ℓ such that $\pi(\ell) \leq k < \ell$ and ℓ is released after $s_k - q_k$. Given that ECTF executes k before ℓ, it must be the case that $\Gamma_k(s_k - q_k) \leq \Gamma_\ell(s_k - q_k)$, i.e. $C_k \leq (1+\beta)r_\ell + \alpha_\ell \leq C_\ell^*$. Thus, we get that $\sum_{i=1}^{k}(1+\beta)^{k-i+1}q_i + \sum_{i=1}^{k}(1+\beta)^{k-i}\alpha_i \leq \sum_{i=1}^{\pi(\ell)}(1+\beta)^{\pi(\ell)-i+1}q_{\gamma(i)}^* + \sum_{i=1}^{\pi(\ell)}(1+\beta)^{\pi(\ell)-i}\alpha_{\gamma(i)}$. By taking into account that $\pi(\ell) \leq k$ and multiplying both sides with $(1+\beta)^{n-k}$:

$$\sum_{i=1}^{k}(1+\beta)^{n-i+1}q_i + \sum_{i=1}^{k}(1+\beta)^{n-i}\alpha_i \leq \sum_{i=1}^{k}(1+\beta)^{n-i+1}q_{\gamma(i)}^* + \sum_{i=1}^{k}(1+\beta)^{n-i}\alpha_{\gamma(i)}. \tag{4}$$

Next, define the set $\mathcal{K}_W = \{i : i \in \mathcal{W}, q_i > 0\}$ of *critical well-ordered jobs*. Consider a job $k \in \mathcal{K}_W$. Given that $q_k > 0$, job k begins at its release time in $\mathcal{S}$, i.e. $s_k = r_k$. That is, $C_k = (1+\beta)r_k + \alpha_k \leq C_k^*$, or equivalently $\sum_{i=1}^{k}(1+\beta)^{k-i+1}q_i + \sum_{i=1}^{k}(1+\beta)^{k-i}\alpha_i \leq \sum_{i=1}^{\pi(k)}(1+\beta)^{\pi(k)-i+1}q_{\gamma(i)}^* + \sum_{i=1}^{\pi(k)}(1+\beta)^{\pi(k)-i}\alpha_{\gamma(i)}$. By taking into account the fact that k is well-ordered, i.e. $\pi(k) \leq k$, and multiplying both sides of the inequality with $(1+\beta)^{n-k}$, we conclude that Eq. (4) holds for each job $k \in K_W$ as well.

Let $\mathcal{K} = \mathcal{K}_I \cup \mathcal{K}_W$ be the set of all critical jobs and consider the maximum index critical job $k = \max\{i : i \in \mathcal{K}\}$. Next, denote by $\mathcal{W}_k = \{i : i > k, i \in \mathcal{W}\}$ and $\mathcal{I}_k = \{i : i > k, i \in \mathcal{I}\}$ the well-ordered and inverted jobs, respectively, of index strictly greater than the one of the maximum index critical job k. For each job $i \in \mathcal{J}$ with $i > k$, either $i \in \mathcal{W}_k$, or $i \in \mathcal{I}_k$. In the former case, since $i \in \mathcal{W}_k$, we have that $i \geq \pi(i)$. Because $i > k$, it also holds that $q_i = 0$. Therefore, $(1+\beta)^{n-i+1}q_i + (1+\beta)^{n-i}\alpha_i \leq (1+\beta)^{n-\pi(i)}\alpha_i$.

By summing over all jobs in $\mathcal{W}_k$, we get that

$$\sum_{i\in\mathcal{W}_k}(1+\beta)^{n-i+1}q_i + \sum_{i\in\mathcal{W}_k}(1+\beta)^{n-i}\alpha_i \le \sum_{i\in\mathcal{W}_k}(1+\beta)^{n-\pi(i)}\alpha_i. \tag{5}$$

In the latter case, i.e. $i \in \mathcal{I}_k$, because $i < \pi(i)$, the pigeonhole principle argument mentioned earlier implies that there exists a well-ordered job $j \in \mathcal{W}$ such that $\pi(j) \le i < j$. Let $t = s_i - q_i$. Because $i > k$, i.e. i is non-critical, it must be the case that $r_j \le t$. Due to the ECTF policy and given that job i is executed before job j in $\mathcal{S}$, we have that

$$\begin{aligned}\Gamma_i(t) \le \Gamma_j(t) &\Rightarrow (1+\beta)\max\{t, s_i\} + \alpha_i \le (1+\beta)\max\{t, r_j\} + \alpha_j\\ &\Rightarrow (1+\beta)q_i + \alpha_i \le \alpha_j.\end{aligned}$$

Taking also into account that $\pi(j) \le i$, $(1+\beta)^{n-i+1}q_i + (1+\beta)^{n-i}\alpha_i \le (1+\beta)^{n-\pi(j)}\alpha_j$. We pick such a well-ordered job j arbitrarily, match it with i, and denote it by $\mu(i) = j$. Let $\mathcal{M}_j = \{i : i \in \mathcal{I}, \mu(i) = j\}$ be the set of inverted jobs matched with a job $j \in \mathcal{W}$. If $i \in \mathcal{M}_j$, then it clearly holds that $\pi(j) \le i$. Thus, based on the weak pseudomatching bound (Lemma 7),

$$\begin{aligned}\sum_{i\in\mathcal{M}_j}[(1+\beta)^{n-i+1}q_i + (1+\beta)^{n-i}\alpha_i] &\le \sum_{i=\pi(j)}^{n}(1+\beta)^{n-i}\alpha_j\\ &= \left[\frac{(1+\beta)^{n-\pi(j)+1}-1}{(1+\beta)-1}\right]\alpha_j\\ &\le \left(1+\frac{1}{\beta}\right)(1+\beta)^{n-\pi(j)}\alpha_j\end{aligned}$$

That is, we get that:

$$\begin{aligned}\sum_{i\in\mathcal{I}_k}[(1+\beta)^{n-i+1}q_i + (1+\beta)^{n-i}\alpha_i] &= \sum_{j\in\mathcal{W}}\sum_{i\in\mathcal{M}_j}[(1+\beta)^{n-i+1}q_i + (1+\beta)^{n-i}\alpha_i]\\ &\le \left(1+\frac{1}{\beta}\right)\left[\sum_{j\in\mathcal{W}}(1+\beta)^{n-\pi(j)}\alpha_j\right] \end{aligned} \tag{6}$$

The algorithm achieves makespan:

$$\begin{aligned}T = \sum_{i=1}^{k}(1+\beta)^{n-i+1}[q_i + (1+\beta)^{n-i}\alpha_i] &+ \sum_{i\in\mathcal{W}_k}(1+\beta)^{n-i}\alpha_i\\ &+ \sum_{i\in\mathcal{I}_k}[(1+\beta)^{n-i+1}q_i + (1+\beta)^{n-i}\alpha_i]\end{aligned} \tag{7}$$

For the optimal makespan, it clearly holds that:

$$T^* \geq \max\left\{\sum_{i=1}^{n}(1+\beta)^{n-i+1}q^*_{\gamma(i)} + \sum_{i=1}^{n}(1+\beta)^{n-i}\alpha_{\gamma(i)}, \sum_{i\in\mathcal{W}}(1+\beta)^{n-\pi(i)}\alpha_i\right\} \tag{8}$$

By Eq. (4)-(8), we conclude that $T \leq (3+\frac{1}{\beta})T^*$.

Lower Bound. Next, we show the lower bound. We consider an instance with $n = 2k$ jobs: k short jobs and k long jobs. The j-th long job has $r_j^L = 0$ and $\alpha_j^L = (1+\beta)B$, for $j \in \{1,\ldots,k\}$. The j-th short jobs has release time $r_j^S = \sum_{i=1}^{j}(1+\beta)^{i-1}B$ and and $\alpha_j^S = 0$, for $j \in \{1,\ldots,k\}$. Let $\mathcal{S}$ be a schedule produced by ECTF. We show by induction that all small jobs are executed before all long jobs in $\mathcal{S}$, i.e. the j-th short job is executed during $[\sum_{i=1}^{j}(1+\beta)^{i-1}B, \sum_{i=1}^{j}(1+\beta)^{i}B)$. For $j = 1$, the small job completes at $(1+\beta)B$ and any long job has completion time $\geq (1+\beta)B$ in any feasible schedule. Next, assume that our claim is true for some $j \in \{1,\ldots,k-1\}$. Since the j-th short job has completion time $C_j = \sum_{i=1}^{j}(1+\beta)^{i}B$, the $(j+1)$-th job begins at its release time $r_{j+1} > C_j$ and completes at $\sum_{i=1}^{j+1}(1+\beta)^{i}B$, while any long job would complete at $\geq C_{j+1}$ if it began at C_j. Therefore,

$$T = \sum_{i=1}^{2k}(1+\beta)^{i}B = \frac{(1+\beta)B}{\beta}\left[(1+\beta)^{2k}-1\right].$$

On the other hand, the optimal solution executes all long jobs before all short jobs and has makespan:

$$T^* = \sum_{i=k+1}^{2k}(1+\beta)^{i}B = \frac{(1+\beta)^{k+1}B}{\beta}\left[(1+\beta)^{k}-1\right].$$

Therefore, we conclude that

$$\begin{aligned}
\beta T &= (1+\beta)^{2k+1}B - (1+\beta)B \\
&= \left[(1+\beta)^{2k+1}B - (1+\beta)^{k+1}B\right] + \left[(1+\beta)^{k+1}B - (1+\beta)B\right] \\
&\leq \left[1 + \frac{1}{(1+\beta)^k}\right]\beta T^*
\end{aligned}$$

Hence, $T/T^* \geq (1+\frac{1}{(1+\beta)^k})$. If $\beta = \omega(1)$, then $\mathcal{S}$ is 2-approximate. □

4. Relation Between Makespan and Total Completion Time

This section explores relationships between the problems of minimizing the makespan $\max_{i\in\mathcal{J}}\{C_i\}$ and the sum of completion times $\sum_{i\in\mathcal{J}} C_i$. Theorem 16 shows that any $O(1)$-approximate schedule for the former is also $O(1)$-approximate for the latter.

Theorem 16. *Any ρ-approximation algorithm for $1|r_i, p_i(s_i) = \alpha_i+\beta s_i|\sum C_i$ is $(1+\rho)$-approximate for $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$.*

Proof. Suppose that $\mathcal{S}$ and $\mathcal{S}^*$ are a ρ-approximate schedule for minimizing the sum of completion times and an optimal schedule for minimizing makespan, while T and T^* are the makespans of the two schedules. Assuming that s_i, C_i, and q_i the start time, completion time, and gap associated with job $i \in \mathcal{J}$ in $\mathcal{S}$, we similarly define s_i^*, C_i^*, and q_i^* for $\mathcal{S}^*$. Given that $\mathcal{S}$ is ρ-approximate for minimizing $\sum_{i=1}^n C_i$, the job start times in the two schedules can be related as follows:

$$\sum_{i=1}^n C_i \leq \rho\left[\sum_{i=1}^n C_i^*\right] \Rightarrow \sum_{i=1}^n[(1+\beta)s_i+\alpha_i] \leq \rho\left[\sum_{i=1}^n[(1+\beta)s_i^*+\alpha_i]\right]$$
$$\Rightarrow \sum_{i=1}^n s_i \leq \rho\left[\sum_{i=1}^n s_i^*\right] + \frac{\rho-1}{1+\beta}\left[\sum_{i=1}^n \alpha_i\right]$$

Observe that $\sum_{i=1}^n q_i \leq r_{\max}$, where $r_{\max} = \max_{i=1}^n\{r_i\}$ is the maximum release time, because gaps may only occur before release times in a canonical schedule. To upper bound the makespan of $\mathcal{S}$:

$$\begin{aligned}
T &= \sum_{i=1}^n [q_i + p_i(s_i)] \\
&= \sum_{i=1}^n q_i + \sum_{i=1}^n \alpha_i + \beta\left[\sum_{i=1}^n s_i\right] \\
&\leq r_{\max} + \frac{1+\rho\beta}{1+\beta}\left[\sum_{i=1}^n \alpha_i\right] + \rho\beta\left[\sum_{i=1}^n s_i^*\right] \\
&\leq r_{\max} + \rho\left[\sum_{i=1}^n(\beta s_i^* + \alpha_i)\right]
\end{aligned}$$

The last inequality follows from the fact that $\rho \geq 1$. Given that $T^* \geq \max\{\sum_{i=1}^n(\beta s_i^*+\alpha_i), r_{\max}\}$, we conclude that $T \leq (1+\rho)T^*$. □

Theorem 17 shows that any $O(1)$-approximate schedule for $\max_{i\in\mathcal{J}}\{C_i\}$ is $O(1+\frac{1}{\beta})$-approximate for $\sum_{i\in\mathcal{J}} C_i$. This result and Theorem 15 directly imply that there exists an $O(1)$-approximation algorithm for minimizing the sum of completion times when $\beta = \Omega(1)$.

Theorem 17. *Any ρ-approximation algorithm for $1|r_i, p_i(s_i) = \alpha_i+\beta s_i|C_{\max}$ is $(1+\frac{1}{\beta})\rho$-approximate for $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|\sum C_i$.*

Proof. Suppose that $\mathcal{S}$ and $\mathcal{S}^*$ are a ρ-approximate schedule for minimizing the makespan and an optimal schedule for minimizing the sum of completion times, respectively. Assuming that s_i, C_i, and q_i are the start time, completion time, and gap associated with job $i \in \mathcal{J}$ in $\mathcal{S}$, we similarly define s_i^*, C_i^*, and q_i^* for $\mathcal{S}^*$. Given that $\mathcal{S}$ is ρ-approximate for the makespan objective and the fact that $\sum_{i=1}^n q_i^* \leq r_{\max}$, we get that:

$$T \leq \rho T^* \Rightarrow \sum_{i=1}^n [\beta s_i + \alpha_i] \leq \rho \left[\sum_{i=1}^n q_i^* + \sum_{i=1}^n [\beta s_i^* + \alpha_i]\right]$$
$$\Rightarrow \sum_{i=1}^n s_i \leq \rho \left[\sum_{i=1}^n s_i^*\right] + \frac{\rho}{\beta}\left[r_{\max} + \sum_{i=1}^n \alpha_i\right]$$

Since $\rho \geq 1$, we can upper bound the sum of completion times of $\mathcal{S}$ as follows:

$$\sum_{i=1}^n C_i = (1+\beta)\left[\sum_{i=1}^n s_i\right] + \sum_{i=1}^n \alpha_i$$
$$\leq \rho\left[\sum_{i=1}^n [(1+\beta)s_i^* + \alpha_i]\right] + \frac{\rho}{\beta}\left[r_{\max} + \sum_{i=1}^n \alpha_i\right]$$

Because $\sum_{i=1}^n C_i^* = \sum_{i=1}^n [(1+\beta)s_i^* + \alpha_i] \geq r_{\max} + \sum_{i=1}^n \alpha_i$, we get that $\sum_{i=1}^n C_i \leq (1+\frac{1}{\beta})\rho\left[\sum_{i=1}^n C_i^*\right]$. □

Corollary 18. *Algorithm 3 is $O(1+\frac{1}{\beta^2})$-approximate for $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|\sum C_i$.*

5. Concluding Remarks

We obtained approximation results for time-dependent scheduling of jobs with uniformly deteriorating processing times. On the negative side, we showed that the non-interfering and non-idling algorithms are $\Omega((1+\beta))^n$-approximate for $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$. On the the positive side, we proved that ECTF computes $O(1+1/\beta)$-approximate solutions for the same problem. Our analysis relied on the concept of bounding pseudomatchings. Finally, we derived an approximation relationship between the makespan problem $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|C_{\max}$ and the total completion time problem $1|r_i, p_i(s_i) = \alpha_i + \beta s_i|\sum C_i$, which implies that ECTF is $O(1+1/\beta^2)$-approximate for the latter.

The approximability of the investigated time-dependent scheduling problems remains an intriguing future direction. In particular, it is unknown whether any of the two problems admits a Polynomial-Time Approximation Scheme (PTAS) or is $\mathcal{APX}$-hard. We expect our bounding framework to be useful for follow-up work. Our analysis used the existence of bounding pseudomatchings for analyzing the proposed algorithms theoretically. An interesting question is whether algorithms actually computing bounding pseudomatchings could achieve better ratios than ECTF. An answer to this question could be useful for determining whether the more general problems $1|r_i, p_i(s_i) = \alpha_i + \beta_i s_i|C_{\max}$ and $1|r_i, p_i(s_i) = \alpha_i + \beta_i s_i|\sum C_i$ with arbitrary deterioration rates admit constant-factor approximation algorithms.

References

[1] Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M., 1999. Approximation schemes for minimizing average weighted completion time with release dates, in: Symposium on Foundations of Computer Science (FOCS), IEEE. pp. 32–43.

[2] Bosio, A., Righini, G., 2009. A dynamic programming algorithm for the single-machine scheduling problem with release dates and deteriorating processing times. Mathematical Methods of Operations Research 69, 271–280.

[3] Chai, X., Li, W., Yuan, H., Wang, L., 2020. Online scheduling on a single machine with linear deteriorating processing times and delivery times. Journal of Combinatorial Optimization , 1–13.

[4] Cheng, T.C.E., Ding, Q., 1998. The complexity of scheduling starting

time dependent tasks with release times. Information Processing Letters 65, 75–79.

[5] Chrétienne, P., 2016. On scheduling with the non-idling constraint. Annals of Operations Research 240, 301–319.

[6] De Souza, E.A.G., Nagano, M.S., Rolim, G.A., 2022. Dynamic programming algorithms and their applications in machine scheduling: A review. Expert Systems with Applications 190, 116180.

[7] Gawiejnowicz, S., 2007. Scheduling deteriorating jobs subject to job or machine availability constraints. European Journal of Operational Research 180, 472–478.

[8] Gawiejnowicz, S., 2020a. Models and algorithms of time-dependent scheduling. Springer.

[9] Gawiejnowicz, S., 2020b. A review of four decades of time-dependent scheduling: Main results, new topics, and open problems. Journal of Scheduling 23, 3–47.

[10] Gupta, J.N.D., Gupta, S.K., 1988. Single facility scheduling with nonlinear processing times. Computers & Industrial Engineering 14, 387–393.

[11] He, L., de Weerdt, M., Yorke-Smith, N., 2020. Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm. Journal of Intelligent Manufacturing 31, 1051–1078.

[12] Helvig, C.S., Robins, G., Zelikovsky, A., 1998. Moving-target TSP and related problems, in: European Symposium on Algorithms (ESA), Springer. pp. 453–464.

[13] Karger, D., Stein, C., Wein, J., 2010. Scheduling algorithms (chapter 20), in: Algorithms and Theory of Computation Handbook: Special Topics and Techniques, pp. 1–34.

[14] Lawler, E.L., 1973. Optimal sequencing of a single machine subject to precedence constraints. Management Science 19, 544–546.

[15] Lee, W.C., Wu, C.C., Chung, Y.H., 2008. Scheduling deteriorating jobs on a single machine with release times. Computers & Industrial Engineering 54, 441–452.

[16] Lenstra, J.K., Kan, A.R., Brucker, P., 1977. Complexity of machine scheduling problems. Annals of Discrete Mathematics 1, 343–362.

[17] Liu, M., Zheng, F., Wang, S., Huo, J., 2012. Optimal algorithms for online single machine scheduling with deteriorating jobs. Theoretical Computer Science 445, 75–81.

[18] Miao, C., Zhang, Y., Wu, C., 2012. Scheduling of deteriorating jobs with release dates to minimize the maximum lateness. Theoretical Computer Science 462, 80–87.

[19] Ng, C.T., Li, S., Cheng, T.C.E., Yuan, J., 2010. Preemptive scheduling with simple linear deterioration on a single machine. Theoretical Computer Science 411, 3578–3586.

[20] Pappis, C.P., Rachaniotis, N.P., 2010. Scheduling a single fire fighting resource with deteriorating fire suppression times and set-up times. Operational Research 10, 27–42.

[21] Stein, C., Wein, J., 1997. On the existence of schedules that are near-optimal for both makespan and total weighted completion time. Operations Research Letters 21, 115–122.

[22] Xu, S., Hall, N.G., 2021. Fatigue, personnel scheduling and operations: Review and research opportunities. European Journal of Operational Research 295, 807–822.