Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(5): 3050–3063.
DOI: 10.3934/era.2023154
Received: 02 February 2023
Revised: 16 March 2023
Accepted: 17 March 2023
Published: 22 March 2023

Research article

Single machine Pareto scheduling with positional deadlines and agreeable release and processing times

Shuguang Li[1,*], Yong Sun[1] and Muhammad Ijaz Khan[2,3]

[1] School of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China

[2] Department of Mechanical Engineering, Lebanese American University, Beirut 362060, Lebanon

[3] Department of Mechanics and Engineering Science, Peking University, Beijing 100871, China

* **Correspondence:** Email: sgliytu@hotmail.com; Tel: +8618753509226.

Abstract: This paper studies the problem of scheduling n jobs on a single machine to minimize total completion time and maximum cost, simultaneously. Each job is associated with a positional deadline that indicates the largest ordinal number of this job in any feasible schedule. The jobs have agreeable release and processing times, meaning that jobs with larger release times also have larger processing times. The agreeability assumption is reasonable since both the single-criterion problems (without positional deadline constraints) of minimizing total completion time and maximum lateness on a single machine with arbitrary release and processing times are strongly NP-hard. An $O(n^3)$-time Pareto optimal algorithm is presented. The previously known algorithms only solve two special cases of the agreeability assumption: either the case of equal release times in $O(n^4)$ time, or the case of equal processing times (without positional deadline constraints) in $O(n^3)$ time.

Keywords: scheduling; Pareto optimization; single machine; positional deadlines; release times

1. Introduction

Today, people usually want their needs to be met as quickly as possible, e.g., in make-to-order production systems or in hospitals. In a production system, such as an automobile or semiconductor manufacturing industry, the orders arrive dynamically. It is reasonable to assume that orders that arrive later may have larger processing times (because they need more preparation), i.e., the orders have agreeable release and processing times. A processing sequence of the orders may be naturally formed by the positional requirements from customers. In general, the more advanced in position an order is, the sooner it will be completed. However, a bad arrangement of the orders may lead to events that the producers suffer a high inventory cost (which can be measured by total completion time of the orders) or some important orders delay (which can be measured by maximum cost of the orders). In a hospital, when emergency patients arrive, they always have expectations for the positions by which their requirements are served or processed. Making reasonable arrangements for the operating rooms can reduce operating expenses and improve patients' satisfaction.

In this paper, we study the following scheduling problem. There are n jobs $J_1, J_2, \ldots, J_n$ to be processed non-preemptively on a machine that can process at most one job at a time. Denote by $\mathcal{J}$ the set of the jobs. Each job, $J_j \in \mathcal{J}$, has a positional deadline $\bar{k}_j$ which indicates the largest ordinal number of J_j in any feasible schedule (i.e., if J_j is scheduled at the x-th position in the schedule, then $x \le \bar{k}_j$.), a positive processing time p_j and a non-negative release time r_j before which it cannot be processed. We assume that the release and processing times of the jobs are *agreeable*, denoted by $(r_j, p_j)-agreeable$, i.e., $r_{j_1} \le r_{j_2}$ implies $p_{j_1} \le p_{j_2}$. In addition, job J_j is associated with a cost function $f_j(t)$ which denotes the scheduling cost incurred if J_j is completed at time t. The cost function $f_j(t)$ is assumed to be *regular*, i.e., it is non-decreasing in the job completion times.

A schedule specified for each job when it is executed on the machine. For a given schedule σ, let $S_j(\sigma)$ and $C_j(\sigma)$ denote the start time and completion time of J_j, respectively. Let $f_j(C_j(\sigma))$ denote the scheduling cost of J_j. Let $f_{\max}(\sigma) = \max_j f_j(C_j(\sigma))$ denote the maximum cost of σ. Two important special cases of $f_{\max}$ are the makespan $C_{\max}(\sigma) = \max_j\{C_j(\sigma)\}$ and the maximum lateness $L_{\max}(\sigma) = \max_j\{C_j(\sigma) - d_j\}$, where d_j denotes the due date of J_j. Let $\sum_{j=1}^n C_j(\sigma)$ denote the total completion time of the jobs in σ. When there is no confusion, we can omit the argument σ for the above notations.

We care about the two objective functions: total completion time $\sum_{j=1}^n C_j$ and maximum cost $f_{\max}$. The total completion time measures the total work-in-process inventory cost in a manufacturing system, while the maximum cost measures how close the system is to meeting the customer requirements. Thus, the two objectives represent the usually competing concerns of manufacturing efficiency and customer service.

A feasible schedule σ is *Pareto optimal* for $\sum_{j=1}^n C_j$ and $f_{\max}$, if there is no feasible schedule σ', such that $(\sum_{j=1}^n C_j(\sigma'), f_{\max}(\sigma')) \le (\sum_{j=1}^n C_j(\sigma), f_{\max}(\sigma))$ and at least one of the two strict inequalities $\sum_{j=1}^n C_j(\sigma') < \sum_{j=1}^n C_j(\sigma)$ and $f_{\max}(\sigma') < f_{\max}(\sigma)$ holds. A feasible schedule σ is *weak Pareto optimal* for $\sum_{j=1}^n C_j$ and $f_{\max}$ if there is no feasible schedule σ', such that $\sum_{j=1}^n C_j(\sigma') < \sum_{j=1}^n C_j(\sigma)$ and $f_{\max}(\sigma') < f_{\max}(\sigma)$. The objective vector $(\sum_{j=1}^n C_j(\sigma), f_{\max}(\sigma))$ of a (weak) Pareto optimal schedule σ is called a *(weak) Pareto optimal point* [1].

Our goal is to find Pareto optimal schedules that simultaneously optimize $\sum_{j=1}^n C_j$ and $f_{\max}$. Following the notation schemes of [2,3], the problem is denoted as $1|\bar{k}_j, (r_j, p_j) - agreeable|(\sum_{j=1}^n C_j, f_{\max})$.

Since both problems $1|r_j|\sum_{j=1}^n C_j$ (minimizing total completion time with release times on a single machine) and $1|r_j|L_{\max}$ (minimizing maximum lateness with release times on a single machine) are strongly NP-hard [4], it is reasonable to focus our attention on the case where the jobs have agreeable release and processing times. This would also appear to be a relatively mild restriction in applications. Equal release times or equal processing times are its two important special cases. Moreover, since $1|prec|\sum_{j=1}^n C_j$ (jobs have precedence constraints) is strongly NP-hard [5], we will not consider precedence constraints in this paper.

We obtain an $O(n^3)$-time algorithm for problem $1|\bar{k}_j, (r_j, p_j) - agreeable|(\sum_{j=1}^n C_j, f_{\max})$, which generalizes and improves the two $O(n^4)$-time algorithms presented in [6] for problem $1|\bar{k}_j|(\sum_{j=1}^n C_j, f_{\max})$ (all jobs have equal release times), and extends the $O(n^3)$-time algorithm presented in [7] for problem $1|r_j, p_j = p|(\sum_{j=1}^n C_j, f_{\max})$ (all jobs have equal processing times and no positional deadlines).

The notations used in this paper are described in Table 1. The notations not listed below can be understood naturally.

Table 1. The list of the notations.

$\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$	The set of n jobs to be scheduled on a single machine
$\bar{k}_j$	The positional deadline of job J_j, where $j \in \{1, 2, \ldots, n\}$
p_j	The processing time of J_j
r_j	The release time of J_j
d_j	The due date of J_j
$\bar{d}_j$	The deadline of J_j
$S_j(\sigma)$	The start time of job J_j in schedule σ
$C_j(\sigma)$	The completion time of job J_j in σ
$w_jC_j(\sigma)$	The weighted completion time of job J_j in σ
$f_j(C_j(\sigma))$	The scheduling cost of job J_j in σ
$\sum_{j=1}^{n} C_j(\sigma)$	The total completion time of the jobs in σ
$\sum_{j=1}^{n} w_jC_j$	The total weighted completion time of the jobs in σ
$f_{\max}(\sigma) = \max_j f_j(C_j(\sigma))$	The maximum cost of σ
$C_{\max}(\sigma) = \max_j\{C_j(\sigma)\}$	The makespan of σ
$L_{\max}(\sigma) = \max_j\{C_j(\sigma) - d_j\}$	The maximum lateness of σ
$(r_j, p_j) - agreeable$	The release and processing times of the jobs are agreeable

The paper is organized as follows: In Section 2, we review the literature. In Section 3, an $O(n^3)$-time algorithm for $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$ is presented. In Section 4, an $O(n^3)$-time algorithm for $1|\bar{k}_j, (r_j, p_j) - agreeable|(\sum_{j=1}^{n} C_j, f_{\max})$ is presented. Finally, some concluding remarks are drawn in Section 5.

2. Literature review

Pareto scheduling has been studied widely in the literature. The methodology and development in this topic can be found in [1, 3, 8, 9]. Here, we only review the results on the combination of total completion time and maximum cost (or maximum lateness) criteria, or/and scheduling with positional deadlines. Table 2 summarizes the time complexities of the directly related problems.

Table 2. Summary of complexity results.

Scheduling problems	Time complexity	References
$1\|r_j\|\sum_{j=1}^{n} C_j$	strongly NP-hard	[4]
$1\|r_j\|L_{\max}$	strongly NP-hard	[4]
$1\|prec\|\sum_{j=1}^{n} C_j$	strongly NP-hard	[5]
$1\|\bar{d}_j\|\sum_{j=1}^{n} w_jC_j$	strongly NP-hard	[4]
$1\|\|(\sum_{j=1}^{n} C_j, L_{\max})$	$O(n^3 \log n)$	[12]
$1\|\|(\sum_{j=1}^{n} w_jC_j, L_{\max})$	strongly NP-hard	[4]
$1\|\|(\sum_{j=1}^{n} C_j, f_{\max})$	$O(n^3)$	[7]
$1\|p_j = p\|(\sum_{j=1}^{n} w_jC_j, f_{\max})$	$O(n^3)$	[7]
$1\|r_j, p_j = p\|(\sum_{j=1}^{n} C_j, f_{\max})$	$O(n^3)$	[7]
$1\|\bar{k}_j\|\sum_{j=1}^{n} C_j$	$O(n \log n)$	[16]
$1\|\bar{k}_j, prec\|f_{\max}$	$O(n^2)$	[17]
$1\|\bar{k}_j\|L_{\max}$	$O(n \log n)$	[17]
$1\|\bar{k}_j, prec^B\|(\sum_{j=1}^{n} C_j^A, f_{\max}^B)$	$O(nn_A{n_B}^2 + {n_A}^2n_B \log n_A)$	[18]
$1\|\bar{k}_j, prec\|(f_{\max}, g_{\max})$	$O(n^4)$	[18]
$1\|\bar{k}_j, prec\|(f_{\max}^A, g_{\max}^B)$	$O({n_A}^3n_B + {n_B}^3n_A)$	[18]
$1\|\bar{k}_j\|(\sum_{j=1}^{n} C_j, f_{\max})$	$O(n^3)$	Theorem 3.7
$1\|\bar{k}_j, (r_j, p_j) - agreeable\|(\sum_{j=1}^{n} C_j, f_{\max})$	$O(n^3)$	Theorem 4.2

Wassenhove and Gelders [10] presented an algorithm for $1||(\sum_{j=1}^{n} C_j, L_{\max})$ (Pareto scheduling for minimizing total completion time and maximum lateness), which finds each Pareto optimal point in $O(n \log n)$ time. John [11] extended the idea to solve $1||(\sum_{j=1}^{n} C_j, f_{\max})$ and obtained an algorithm that finds each Pareto optimal point in $O(n^2)$ time. Hoogeveen and van de Velde [12] proved that, for $1||(\sum_{j=1}^{n} C_j, f_{\max})$, there are at most $n(n-1)/2+1$ Pareto optimal points. Hence, problems $1||(\sum_{j=1}^{n} C_j, L_{\max})$ and $1||(\sum_{j=1}^{n} C_j, f_{\max})$ can be solved in $O(n^3 \log n)$ and $O(n^4)$ time, respectively. Steiner and Stephenson [13] studied $1||(\sum_{j=1}^{n} w_jC_j, L_{\max})$ (Pareto scheduling for minimizing total weighted completion time and maximum lateness). For this strongly NP-hard problem (since $1|\bar{d}_j|\sum_{j=1}^{n} w_jC_j$ is strongly NP-hard [4], where $\bar{d}_j$ denotes the deadline of job J_j by which J_j must be completed), they described several characterizations for the set of Pareto optimal schedules for this problem, and incorporated these results into a branch-and-bound algorithm, which can enumerate all Pareto optimal schedules for the instances with hundreds of Pareto optimal schedules in reasonable time and space. Gao and Yuan [14] gave an $O(n^3 \log \sum_j p_j)$-time algorithm for $1||(\sum_{j=1}^{n} C_j, f_{\max})$. He et al. [15] presented an $O(n^3 \log n)$-time algorithm for $1|p_j = p|(\sum_{j=1}^{n} w_jC_j, f_{\max})$ (Pareto scheduling jobs with equal processing times to minimize total weighted completion time and maximum cost). Liu and Li [7] obtained $O(n^3)$-time algorithms for problems $1||(\sum_{j=1}^{n} C_j, f_{\max})$, $1|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$ and $1|p_j = p|(\sum_{j=1}^{n} w_jC_j, f_{\max})$.

There are also many results on scheduling problems with positional deadlines. Zhao et al. [16] presented an $O(n \log n)$-time algorithm for $1|\bar{k}_j|\sum_{j=1}^{n} C_j$. Zhao and Yuan [17] presented an $O(n^2)$-time algorithms for $1|\bar{k}_j, prec|f_{\max}$ (jobs have positional deadlines and precedence constraints) and an $O(n \log n)$-time algorithm for $1|\bar{k}_j|L_{\max}$. Gao and Yuan [6] presented two $O(n^4)$-time algorithms for $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$. Gao and Yuan [18] presented an $O(nn_A{n_B}^2 + {n_A}^2n_B \log n_A)$-time algorithm for

$1|\bar{k}_j, prec^B|(\sum_{j=1}^{n} C_j^A, f_{\max}^B)$ (two-agent scheduling with positional deadlines and B-jobs have precedence constraints), where n_A and n_B denote the numbers of jobs from agents A and B respectively, and an $O(n^4)$-time algorithm for $1|\bar{k}_j, prec|(f_{\max}, g_{\max})$. The latter result also implies an $O({n_A}^3 n_B + {n_B}^3 n_A)$-algorithm for $1|\bar{k}_j, prec|(f_{\max}^A, g_{\max}^B)$. Chen and Yuan [19] studied the scheduling of proportional-linearly deteriorating jobs with deadlines, positional deadlines, release times, and precedence constraints on a single machine. For various single-criterion problems, they proposed polynomial time algorithms or established the NP-hardness results (when processing times of the jobs have no deterioration). Chen et al. [20] studied single machine scheduling problems with due dates, positional due indices, deadlines and positional deadlines (positional due index means a soft restriction, while positional deadline means a hard restriction). For some single-criterion or bicriteria related scheduling problems, they presented polynomial time algorithms or NP-hardness proofs. They also listed several practical applications related to positional due index or positional deadline constraints. Chen et al. [21] studied the ND (Non-Disjoint)-agent scheduling of linear-deteriorating jobs on a single machine with positional deadlines. They presented an $O(n^2)$-time algorithm for the constrained optimization problem of minimizing total completion time of the jobs from one agent, subject to the maximum cost of the jobs from the other agent does not exceed a threshold value. They also presented an $O(n^4)$-time algorithm for the Pareto scheduling problem to minimize total completion time of the jobs from one agent and the maximum cost of the jobs from the other agent simultaneously. If the maximum cost of the agent is a lateness-like criterion and the jobs have no positional deadline constraints, then the time complexity of the two algorithms can be improved to $O(n \log n)$ and $O(n^3 \log n)$, respectively. Gao et al. [22] studied the two-agent Pareto scheduling on a single machine where each job has a deadline and a positional deadline. Moreover, the jobs from one agent have equal processing times, and the jobs from the other agent are restricted by their precedence constraints. They presented an $O(n^5)$-time algorithm for minimizing a general min-sum objective function of the jobs from one agent and the maximum cost of the jobs from the other agent simultaneously.

There are also many results on Pareto scheduling or/and scheduling with positional deadlines combined with batch scheduling (serial-batch or parallel-batch) or/and multi-agent scheduling. Please refer to [23–26] for the recent results on batch scheduling and multi-agent scheduling.

3. Equal release times

In this section, we will present an $O(n^3)$-time algorithm for $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$.

Let $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(n))$ denote a feasible schedule, in which $\sigma(i)$ is the index of the job scheduled at the i-th position in σ, $i = 1, 2, \ldots, n$. Since $\sum_{j=1}^{n} C_j$ and $f_{\max}$ are both regular, we can focus our attention on the schedules without idle times. Therefore, we have:

Lemma 3.1. *Let $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(n))$ be a feasible schedule for $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$. Then, $S_{\sigma(1)} = 0$, $S_{\sigma(i)} = S_{\sigma(i-1)} + p_{\sigma(i-1)}$, $i = 2, \ldots, n$.*

Recall that the well-known SPT (Shortest Processing Time First) rule solves $1|| \sum_{j=1}^{n} C_j$ optimally [27]. To coincide with the following discussion, we apply an adapted version of the SPT rule, called the BRLPT (Backward Restricted Largest Processing Time) rule, to solve $1|\bar{k}_j| \sum_{j=1}^{n} C_j$ optimally in $O(n^2)$ time [6]: In each iteration, let U be the set containing unscheduled jobs. A job $J_j \in U$, which is chosen such that $\bar{k}_j \geq |U|$ and p_j is as large as possible, is scheduled at the $|U|$-th position in the schedule, where $|U|$ denotes the cardinality of U(i.e., the number of the elements in U).

Let $\Omega(\mathcal{J})$ denote the Pareto set, which consists of all Pareto optimal points together with the corresponding schedules. Below, in the algorithm for $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$, we will construct $\Omega(\mathcal{J})$ by repeatedly applying the standard ε-constrained approach for multicriteria scheduling [1, 3].

Lemma 3.2. *([1, 3]) Let y be the optimal value of problem $\alpha|f \le \hat{x}|g$ (minimizing g subject to the constraint $f \le \hat{x}$), and let x be the optimal value of problem $\alpha|g \le y|f$ (minimizing f subject to the constraint $g \le y$). Then, the standard ε-constraint approach tells us that (x, y) is a Pareto optimal point for problem $\alpha||(f, g)$.*

Specifically, we will use the following framework in this section. Suppose that solving the general problem $\alpha|f < x|g$ gives a Pareto optimal schedule (which is true in this section, as shown in Lemma 3.6 below). Then, let $(x^{(1)}, y^{(1)})$ be the first Pareto optimal point obtained by solving $\alpha|f \le \hat{x}|g$, where $\hat{x}$ is an upper bound on f and $(x^{(1)}, y^{(1)})$ is the objective vector of the generated Pareto optimal schedule. Continue to solve $\alpha|f < x^{(i)}|g$ and obtain the next Pareto optimal schedule whose objective vector gives the next Pareto optimal point $(x^{(i+1)}, y^{(i+1)})$ until problem $\alpha|f < x^{(i)}|g$ is infeasible.

Let $\Pi(\mathcal{J})$ denote the set of all feasible schedules for $1|k_j|(\sum_{j=1}^{n} C_j, f_{\max})$. Let $\Pi(\mathcal{J}, y) \subseteq \Pi(\mathcal{J})$ denote the set of the schedules with maximum costs (i.e., $f_{\max}$-values) less than y, where y is a given threshold value. We have $\Pi(\mathcal{J}, +\infty) = \Pi(\mathcal{J})$.

The algorithm maintains a *position index* $k_j^{(h)}$ dynamically for each job J_j when $\sigma^{(h)}$ is adjusted, $h = 0, 1, \ldots$. Initially, the position indices of all the jobs are equal to their positional deadlines, i.e., $k_j^{(0)} = \bar{k}_j$, $j = 1, 2, \ldots, n$. When $\sigma^{(h)}$ is adjusted, the position index of any job will never increase, i.e., it can only decrease or keep unchanged. The ordinal number of J_j in (adjusted) $\sigma^{(h)}$ cannot be larger than $k_j^{(h)}$. In fact, the BRLPT-SC (BRLPT-Smallest Cost) rule is used throughout the algorithm: To assign a job to the currently last position i, always select the unscheduled job whose position index is not less than i and processing time is as large as possible, ties broken in favor of the job with the smallest cost. The BRLPT-SC rule is applicable only for the case of equal release times, because the completion time of the currently last position can be pre-computed and, thus, the job with the smallest cost can be determined. In the next section, we have to apply BRLPT (without SC) rule to deal with unequal release times.

NOREALEASE-TCTFMAX:

Step 1. Initially, set $h = 0$, $y^{(h)} = +\infty$. Set $k_j^{(h)} = \bar{k}_j$, $j = 1, 2, \ldots, n$. Let $\sigma^{(h)} = (\sigma^{(h)}(1), \sigma^{(h)}(2), \cdots, \sigma^{(h)}(n))$ be the schedule which is obtained by the BRLPT-SC rule: For $i = n, n-1, \ldots, 1$, assign the i-th position to the job whose position index is not less than i and processing time is as large as possible in $\mathcal{J} \backslash \{J_{\sigma^{(h)}(n)}, J_{\sigma^{(h)}(n-1)}, \ldots, J_{\sigma^{(h)}(i+1)}\}$; Ties are broken in favor of the job with the smallest cost when completed at time $C_{\sigma^{(h)}(i)} = \sum_{j=1}^{n} p_j - \sum_{l=i+1}^{l=n} p_{\sigma^{(h)}(l)}$. Compute the start and completion times of the jobs as well as the objective values by Lemma 3.1. Let $\Omega(\mathcal{J}) = \{(\sum_{j=1}^{n} C_j(\sigma^{(h)}), f_{\max}(\sigma^{(h)}), \sigma^{(h)})\}$.

Step 2. The $(h + 1)$-th iteration:

Set $y^{(h+1)} = f_{\max}(\sigma^{(h)})$. Invoke Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$ to adjust $\sigma^{(h)}$ (by decreasing its cost) to construct $\sigma^{(h+1)} \in \Pi\left(\mathcal{J}, y^{(h+1)}\right)$.

Step 3. If $\sigma^{(h+1)} \ne \emptyset$, then include $(\sum_{j=1}^{n} C_j(\sigma^{(h+1)}), f_{\max}(\sigma^{(h+1)}), \sigma^{(h+1)})$ into $\Omega(\mathcal{J})$. Set $h = h + 1$ and go to Step 2. If $\sigma^{(h+1)} = \emptyset$, then return $\Omega(\mathcal{J})$.

Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$**:**

Step 1. For $i = n, n-1, \ldots, 1$, check the inequality $f_{\sigma^{(h)}(i)}(C_{\sigma^{(h)}(i)}) < y^{(h+1)}$. If there is a job $J_{\sigma^{(h)}(i)}$, such that $f_{\sigma^{(h)}(i)}(C_{\sigma^{(h)}(i)}) \geq y^{(h+1)}$, if $i = 1$ then simply return $\sigma^{(h+1)} = \emptyset$, otherwise remove $J_{\sigma^{(h)}(i)}$ from its position in $\sigma^{(h)}$ and update its position index $k^{(h)}_{\sigma^{(h)}(i)}$ to be $i - 1$. Then, adjust $\sigma^{(h)}$ as follows: Let $E(i) = \{l | 1 \leq l \leq i-1 \wedge k^{(h)}_{\sigma^{(h)}(l)} \geq i \wedge f_{\sigma^{(h)}(l)}(C_{\sigma^{(h)}(i)}) < y^{(h+1)}\}$ denote the set of candidate jobs at time $C_{\sigma^{(h)}(i)}$. If $E(i) = \emptyset$, then return $\sigma^{(h+1)} = \emptyset$. Otherwise, find the job with largest processing time in $E(i)$, say $J_{\sigma^{(h)}(e)}$ (ties broken in favor of the job with the smallest cost when completed at time $C_{\sigma^{(h)}(i)}$). Let $J_{\sigma^{(h)}(e)}$ be scheduled at the i-th position instead of $J_{\sigma^{(h)}(i)}$. Move backward over consecutive positions, starting from $i-1$ and ending by e, to find suitable positions for jobs $J_{\sigma^{(h)}(i)}, J_{\sigma^{(h)}(i-1)}, \ldots, J_{\sigma^{(h)}(e+1)}$. Let c denote the current position. Let J_x denote the job in hand, initially $J_{\sigma^{(h)}(i)}$. When $c > e$, if $p_{\sigma^{(h)}(c)} > p_x$, or $p_{\sigma^{(h)}(c)} = p_x$ and $f_{\sigma^{(h)}(c)}(C_{\sigma^{(h)}(c)}) \leq f_x(C_{\sigma^{(h)}(c)})$, then $J_{\sigma^{(h)}(c)}$ and J_x keep unchanged and we continue with position $c - 1$ and job J_x. Otherwise, let J_x be scheduled at the c-position, and continue with position $c - 1$ and job $J_{\sigma^{(h)}(c)}$ (i.e., update the job in hand J_x to be $J_{\sigma^{(h)}(c)}$). When $c = e$, simply let J_x be scheduled at the c-position.

Step 2. Update the completion times of the jobs by Lemma 3.1. Update the costs accordingly.

Step 3. Repeat Steps 1 and 2 until all inequalities $f_{\sigma^{(h)}(i)}(C_{\sigma^{(h)}(i)}) < y^{(h+1)}$ hold for $i = n, n-1, \ldots, 1$ after Step 1. Let $\sigma^{(h+1)}$ be the final $\sigma^{(h)}$.

Example:

We give an example illustrating Algorithm NOREALEASE-TCTFMAX. We have five jobs $J_1, J_2, \ldots, J_5$ defined as follows: $r_j = 0$ ($j = 1, 2, \ldots, 5$); $p_j = j$ ($j = 1, 2, \ldots, 5$); $d_j = 6 - j$ ($j = 1, 2, \ldots, 5$); $\bar{k}_1 = 2$, $\bar{k}_2 = \bar{k}_3 = 5$, $\bar{k}_4 = 4$, $\bar{k}_5 = 5$. Algorithm NOREALEASE-TCTFMAX works as follows:

1) $\sigma^{(0)} = \{J_1, J_2, J_3, J_4, J_5\}$ with $\sum C_j = 35$ and $L_{\max} = 14$. We get: $\Omega(\mathcal{J}) = \{(\sum_{j=1}^{n} C_j(\sigma^{(0)}), f_{\max}(\sigma^{(0)}), \sigma^{(0)})\}$.

2) Run Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(1)})$ to adjust $\sigma^{(0)} = \{J_1, J_2, J_3, J_4, J_5\}$, where $y^{(1)} = 14$.

The job violating the inequality in $\sigma^{(0)}$ is J_5. The set of the candidate jobs at time C_5 is $E(5) = \{J_1, J_2, J_3\}$. Since J_3 has the largest release date among the jobs in $E(5)$, it is scheduled at the fifth position instead of J_5. Job J_5 becomes the job in hand. We compare J_5 and J_4. Since $p_5 > p_4$, J_5 is scheduled at the fourth position instead of J_4. Job J_4 becomes the job in hand. We continue to consider the third position. The third position is not occupied because J_3 has been moved from this position to the fifth position. Therefore, J_4 is scheduled at the third position. We get the adjusted $\sigma^{(0)} = \{J_1, J_2, J_4, J_5, J_3\}$ with $\sum C_j = 38$ and $L_{\max} = 12$. Hence, $\sigma^{(1)} = \{J_1, J_2, J_4, J_5, J_3\}$. We get: $\Omega(\mathcal{J}) = \{(\sum_{j=1}^{n} C_j(\sigma^{(h)}), f_{\max}(\sigma^{(h)}), \sigma^{(h)}) | h = 0, 1\}$.

3) Run Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(2)})$ to adjust $\sigma^{(1)} = \{J_1, J_2, J_4, J_5, J_3\}$, where $y^{(2)} = 12$.

(i) The job violating the inequality is J_3. The set of the candidate jobs at time C_3 is $E(5) = \{J_1, J_2\}$. Since J_2 has the largest release date among the jobs in $E(5)$, it is scheduled at the fifth position instead of J_3. Job J_3 becomes the job in hand. We compare J_3 and J_5 for the fourth position. Then, compare J_3 and J_4 for the third position, and finally decide to schedule J_3 at the second position. We get the adjusted $\sigma^{(1)} = \{J_1, J_3, J_4, J_5, J_2\}$ with $\sum C_j = 41$ and $L_{\max} = 12$.

(ii) Now, the job violating the inequality is J_5. Let J_5 be the job in hand and we continue to adjust $\sigma^{(1)} = \{J_1, J_3, J_4, J_5, J_2\}$. We get the adjusted $\sigma^{(1)} = \{J_1, J_3, J_5, J_4, J_2\}$ with $\sum C_j = 42$ and $L_{\max} = 11$.

Hence, $\sigma^{(2)} = \{J_1, J_3, J_5, J_4, J_2\}$. We get: $\Omega(\mathcal{J}) = \{(\sum_{j=1}^{n} C_j(\sigma^{(h)}), f_{\max}(\sigma^{(h)}), \sigma^{(h)})|h = 0, 1, 2\}$.

4) Run Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(3)})$ to adjust $\sigma^{(2)} = \{J_1, J_3, J_5, J_4, J_2\}$, where $y^{(3)} = 11$.

The jobs violating the inequalities are J_2, J_4. We select J_2 as the job in hand and adjust $\sigma^{(2)} = \{J_1, J_3, J_5, J_4, J_2\}$. Since $E(5) = \varnothing$, Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(3)})$ returns $\sigma^{(3)} = \varnothing$.

Finally, Algorithm NOREALEASE-TCTFMAX returns the Pareto set $\Omega(\mathcal{J}) = \{(\sum_{j=1}^{n} C_j(\sigma^{(h)}), f_{\max}(\sigma^{(h)}), \sigma^{(h)})|h = 0, 1, 2\}$.

Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$ adjusts $\sigma^{(h)}$ to construct $\sigma^{(h+1)}$. During the adjustment of $\sigma^{(h)}$, a series of tentative schedules (all denoted by $\sigma^{(h)}$ except for the last one which is denoted by $\sigma^{(h+1)}$) are obtained. A *tentative schedule* is obtained from the current schedule by moving a job violating its inequality to the left and adjusting the positions of some earlier jobs in the schedule, accordingly. If $\sigma^{(h+1)} = \emptyset$, then it is not a tentative schedule. If $\sigma^{(h+1)} \neq \emptyset$, then $\sigma^{(h+1)} \in \Pi\left(\mathcal{J}, y^{(h+1)}\right)$, since there is no job violating its inequality in $\sigma^{(h+1)}$, implying that $f_{\max}(\sigma^{(h+1)}) < y^{(h+1)}$.

Lemma 3.3. *Let $\sigma^{(h-1)} = (\sigma^{(h-1)}(1), \sigma^{(h-1)}(2), \cdots, \sigma^{(h-1)}(n))$ denote any tentative schedule (the last one is denoted by $\sigma^{(h)}$) during the implementation of Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h)})$ ($h = 0, 1. \ldots$). Let $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(n))$ be any feasible schedule in $\Pi\left(\mathcal{J}, y^{(h)}\right)$. Then, the following properties hold:*

1) $S_{\sigma^{(h-1)}(i)} \leq S_{\sigma(i)}$, $i = 1, 2, \ldots, n$;

2) $C_{\sigma^{(h-1)}(i)} \leq C_{\sigma(i)}$, $i = 1, 2, \ldots, n$.

Proof. We prove the lemma by induction on h.

Consider $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(0)})$. The initial schedule $\sigma^{(0)}$ (described in Step 1 of Algorithm NORELEASE-TCTFMAX) is constructed by the BRLPT rule (in fact, BRLPT-SC rule). Let σ be any schedule in $\Pi\left(\mathcal{J}, y^{(0)}\right) = \Pi(\mathcal{J})$. We will provide a transformation of σ into $\sigma^{(0)}$, without increasing the start or completion time of any job.

Compare $\sigma^{(0)}$ and σ backwardly (right-to-left), looking for a difference between the jobs. Suppose that the first difference occurs at the k-th position, which is occupied by jobs J_i and J_j in $\sigma^{(0)}$ and σ, respectively. By the BRLPT rule, we know that $p_i \geq p_j$. Moreover, both J_i and J_j come from $\mathcal{J} \backslash \{J_{\sigma^{(0)}(n)}, J_{\sigma^{(0)}(n-1)}, \ldots, J_{\sigma^{(0)}(k+1)}\}$, implying that J_i is processed earlier than J_j in σ. We can safely interchange J_i and J_j in σ, obeying their positional deadlines, without increasing the start or completion time of any job.

Repetition of this argument shows that σ can be safely transformed into $\sigma^{(0)}$. Properties (1) and (2) follow naturally, thus proving the base case.

Assume that the lemma holds for $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(0)})$, $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(1)}), \ldots$, $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h)})$. We now consider $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$.

Since $y^{(h+1)} < y^{(h)}$, we have: $\Pi\left(\mathcal{J}, y^{(h+1)}\right) \subseteq \Pi\left(\mathcal{J}, y^{(h)}\right)$. Let σ be any schedule in $\Pi\left(\mathcal{J}, y^{(h+1)}\right)$. Then, σ is also in $\Pi\left(\mathcal{J}, y^{(h)}\right)$. Since the first tentative schedule for $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$ is just the last one for $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h)})$, this tentative schedule and σ satisfy the two properties of the lemma. Let $\sigma^{(h)}$ denote the current (not the last) tentative schedule for $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$. By the inductive assumption, $\sigma^{(h)}$ and σ satisfy the properties of the lemma. But since $f_{\max}(\sigma^{(h)}) \geq y^{(h+1)}$, we have to adjust $\sigma^{(h)}$ as described in Step 1 of $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$. In $\sigma^{(h)}$, there is a job $J_{\sigma^{(h)}(i)}$ such that $f_{\sigma^{(h)}(i)}(C_{\sigma^{(h)}(i)}) \geq y^{(h+1)}$. We update its position index $k^{(h)}_{\sigma^{(h)}(i)}$ to be $i - 1$, ensuring that $J_{\sigma^{(h)}(i)}$ cannot be scheduled later than the $(i - 1)$-th position. By the inductive assumption, we have: $C_{\sigma^{(h)}(i)} \leq C_{\sigma(i)}$. It follows that $f_{\sigma^{(h)}(i)}(C_{\sigma(i)}) \geq f_{\sigma^{(h)}(i)}(C_{\sigma^{(h)}(i)}) \geq y^{(h+1)}$, implying that $J_{\sigma^{(h)}(i)}$ cannot be scheduled later than the $(i-1)$-th position in σ. Generally speaking, a crucial observation is: the position index of any job maintained dynamically in the $(h+1)$-th iteration in Algorithm NORELEASE-TCTFMAX indicates its rightmost position in any schedule in $\Pi\left(\mathcal{J}, y^{(h+1)}\right)$.

Let $\bar{\sigma}^{(h)}$ denote the adjusted $\sigma^{(h)}$. We will provide a transformation of σ into $\bar{\sigma}^{(h)}$, without increasing the start or completion time of any job.

Compare $\bar{\sigma}^{(h)}$ and σ backwardly, looking for a difference between the jobs. Suppose that the first difference occurs at the k-th position, which is occupied by jobs J_i and J_j in $\bar{\sigma}^{(h)}$ and σ, respectively. By the BRLPT rule, we know that $p_i \geq p_j$ (since J_j is scheduled at the k-th position in σ, by the above discussion we know that its position index is not less than k. Hence J_j is also an eligible candidate when we select J_i. It must be true that $p_i \geq p_j$.) Moreover, J_i is processed earlier than J_j in σ. We can safely interchange J_i and J_j in σ, obeying their position indices, without increasing the start or completion time of any job.

Repetition of this argument shows that σ can be safely transformed into $\bar{\sigma}^{(h)}$. Properties (1) and (2) follow naturally, thus completing the proof for $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$.

By the principle of induction, we complete the proof of the lemma.

Lemma 3.4. *Let $\sigma^{(h)}$ denote the last schedule upon the completion of Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h)})$ $(h = 0, 1 \ldots)$. If $\sigma^{(h)} = \emptyset$, then $\Pi\left(\mathcal{J}, y^{(h)}\right) = \emptyset$; Otherwise, $\sigma^{(h)}$ has minimum total completion time among all schedules in $\Pi\left(\mathcal{J}, y^{(h)}\right)$.*

Proof. Suppose that in implementing $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h)})$, we find a job $J_{\sigma^{(h)}(i)}$ violating its inequality. If $i = 1$ or $E(i) = \emptyset$, then we conclude that $\Pi\left(\mathcal{J}, y^{(h)}\right) = \emptyset$, since $J_{\sigma^{(h)}(i)}$ cannot be scheduled with its cost less than $y^{(h)}$. Therefore, we simply return $\sigma^{(h)} = \emptyset$.

On the other hand, if $\sigma^{(h)} \neq \emptyset$, then by property (2) of Lemma 3.3, $\sigma^{(h)}$ has minimum total completion time among all schedules in $\Pi\left(\mathcal{J}, y^{(h)}\right)$.

Lemma 3.5. *Let $\sigma_l = (\sigma_l(1), \sigma_l(2), \cdots, \sigma_l(n))$ and $\sigma_{l+1} = (\sigma_{l+1}(1), \sigma_{l+1}(2), \cdots, \sigma_{l+1}(n))$ denote two adjacent tentative schedules (either in the same iteration or adjacent iterations) during the implementation of Algorithm NORELEASE-TCTFMAX. Then, $C_{\sigma_l(i)} \leq C_{\sigma_{l+1}(i)}$, $i = 1, 2, \ldots, n$. That is, the completion times of all the positions never decrease during the entire implementation of the algorithm.*

Proof. The proof comes from the BRLPT rule.

Suppose that σ_{l+1} is obtained by adjusting σ_l in an iteration. Let $J_{\sigma_l(i)}$ be the job violating its inequality in this adjustment. As described in Procedure $P_{NORELEASE-TCTFMAX}$, we find a job $J_{\sigma_l(e)} \in E(i)$ with largest processing time. Job $J_{\sigma_l(e)}$ is scheduled at the i-th position instead of $J_{\sigma_l(i)}$. By the BRLPT rule, for $e + 1 \leq q \leq i$, we have $p_{\sigma_l(q)} \geq p_{\sigma_l(e)}$. After assigning jobs $J_{\sigma_l(i)}, J_{\sigma_l(i-1)}, \ldots, J_{\sigma_l(e)}$ to the suitable positions, the completion time of the i-th position keeps unchanged. Only the processing time at the i-th position may decrease. The processing times at all the other positions keep unchanged or increase. Therefore, during the adjustment of σ_l, none of $C_{\sigma_l(1)}, C_{\sigma_l(2)}, \cdots, C_{\sigma_l(n)}$ can decrease.

The proof of the following lemma is very similar to that in [12] and so we slide it to Appendix.

Lemma 3.6. *Let $\sigma^{(h)}$ denote the last schedule upon the completion of Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h)})$ $(h = 0, 1 \ldots)$. If $\sigma^{(h)} \neq \emptyset$, then it is a Pareto optimal schedule for $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$.*

We get the following theorem.

Theorem 3.7. *Algorithm NORELEASE-TCTFMAX solves* $1|\bar{k}_j|(\sum_{j=1}^{n} C_j, f_{\max})$ *in* $O(n^3)$ *time.*

Proof. The correctness of the algorithm follows from Lemmas 3.2 and 3.6.

Step 1 of Algorithm NORELEASE-TCTFMAX can be implemented in $O(n^2)$ time. In Step 1 of Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$, it takes $O(n)$ time to generate a tentative schedule (to find a job violating its inequality, to move it to the left, and to adjust the positions of some earlier jobs in the schedule accordingly). Since each job violating its inequality at a position can only be moved to the left and never comes back to this position (by Lemma 3.5), it goes through at most $n-1$ positions. Therefore, the total number of tentative schedules is $O(n^2)$. Step 2 of Algorithm NORELEASE-TCTFMAX requires $O(n^3)$ time for all iterations. Step 3 of Algorithm NORELEASE-TCTFMAX can be implemented in $O(n^2)$ time, since the total number of tentative schedules is $O(n^2)$. Hence, the overall running time of Algorithm NORELEASE-TCTFMAX is $O(n^3)$.

4. Agreeable release and processing times

In this section, we will present an $O(n^3)$-time algorithm for $1|\bar{k}_j, (r_j, p_j) - agreeable|(\sum_{j=1}^{n} C_j, f_{\max})$.

We will re-sue the notations developed in the preceding section, such as $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(n))$, $\Omega(\mathcal{J})$, $\Pi(\mathcal{J}, y)$, $k_j^{(h)}$, to name a few. We apply BRLPT rule in this section, meaning that Backward Restricted Largest Processing Time and Release Time (ties broken arbitrarily).

Lemma 4.1. *Let* $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(n))$ *be a feasible schedule for* $1|\bar{k}_j, (r_j, p_j) - agreeable|(\sum_{j=1}^{n} C_j, f_{\max})$*. Then,* $S_{\sigma(1)} = r_{\sigma(1)}$*,* $S_{\sigma(i)} = \max\{r_{\sigma(i)}, S_{\sigma(i-1)} + p_{\sigma(i-1)}\}$*,* $i = 2, \ldots, n$*.*

Specifically, we will use the following framework in this section. Suppose that the general problem $\alpha|f < x|g$ is efficiently solvable (which is true in this section, as shown in Lemma 3.4 with a slight modification). Then, let $(x^{(1)}, y^{(1)})$ be the first weak Pareto optimal point obtained by solving $\alpha|f \leq \hat{x}|g$, where $\hat{x}$ is an upper bound on f and $(x^{(1)}, y^{(1)})$ is the objective vector of the generated weak Pareto optimal schedule. Continue to solve $\alpha|f < x^{(i)}|g$ and obtain the next weak Pareto optimal schedule whose objective vector gives the next weak Pareto optimal point $(x^{(i+1)}, y^{(i+1)})$ until problem $\alpha|f < x^{(i)}|g$ is infeasible. Select the Pareto optimal points from among the obtained weak Pareto optimal points.

REALEASE-TCTFMAX:

Step 1. Initially, set $\Omega(\mathcal{J}) = \emptyset$, $z = 0$. Set $h = 0$, $y^{(h)} = +\infty$. Set $k_j^{(h)} = \bar{k}_j$, $j = 1, 2, \ldots, n$. Let $\sigma^{(h)} = (\sigma^{(h)}(1), \sigma^{(h)}(2), \cdots, \sigma^{(h)}(n))$ be the schedule that is obtained by the BRLPT rule: For $i = n, n-1, \ldots, 1$, assign the i-th position to the job whose position index is not less than i and processing time (as well as release time) is as large as possible in $\mathcal{J} \backslash \{J_{\sigma^{(h)}(n)}, J_{\sigma^{(h)}(n-1)}, \ldots, J_{\sigma^{(h)}(i+1)}\}$, ties broken arbitrarily. Compute the start and completion times of the jobs, as well as the objective values by Lemma 4.1.

Step 2. The $(h+1)$-th iteration:

Set $y^{(h+1)} = f_{\max}(\sigma^{(h)})$. Invoke Procedure $P_{RELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$ to adjust $\sigma^{(h)}$ (by decreasing its cost) to construct $\sigma^{(h+1)} \in \Pi\left(\mathcal{J}, y^{(h+1)}\right)$.

Step 3. If $\sigma^{(h+1)} = \emptyset$, then set $\pi^* = \sigma^{(h)}$. Let $\Omega(\mathcal{J}) = \Omega(\mathcal{J}) \cup \{(\sum_{j=1}^n C_j(\pi^*), f_{\max}(\pi^*), \pi^*)\}$ and return $\Omega(\mathcal{J})$. Otherwise, if $\sum_{j=1}^n C_j(\sigma^{(h+1)}) > \sum_{j=1}^n C_j(\sigma^{(h)})$, then set $z = z + 1$ and $\pi_z = \sigma^{(h)}$, let $\Omega(\mathcal{J}) = \Omega(\mathcal{J}) \cup \{(\sum_{j=1}^n C_j(\pi_z), f_{\max}(\pi_z), \pi_z)\}$.

Step 4. Set $h = h + 1$ and go to Step 2.

Procedure $P_{RELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$**:**

Step 1. The same as Step 1 of Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$ except that: Replace "ties broken in favor of the job with the smallest cost when completed at time $C_{\sigma^{(h)}(i)}$" by "ties broken arbitrarily". Replace "When $c > e$, if $p_{\sigma^{(h)}(c)} > p_x$, or $p_{\sigma^{(h)}(c)} = p_x$ and $f_{\sigma^{(h)}(c)}(C_{\sigma^{(h)}(c)}) \leq f_x(C_{\sigma^{(h)}(c)})$" by "When $c > e$, if $p_{\sigma^{(h)}(c)} \geq p_x$".

Step 2. Update the completion times of the jobs by Lemma 4.1. Update the costs accordingly.

Step 3. The same as Step 3 of Procedure $P_{NORELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$.

Lemma 3.2 certainly holds in this section. Lemmas 3.3–3.5 still hold if "NOREALEASE-TCTFMAX" is replaced by "REALEASE-TCTFMAX". The proofs are almost the same as their counterparts in the preceding section, and, thus, are omitted for brevity. Note that we elaborately use the BRLPT (rather than BRLPT-SC) rule in the proofs. Lemma 3.6 relies on BRLPT-SC rule and, thus, does not hold in this section.

Similarly to the preceding section, we can analyze the time complexity of Algorithm REALEASE-TCTFMAX. Step 1 of Algorithm RELEASE-TCTFMAX can be implemented in $O(n^2)$ time. In Step 1 of Procedure $P_{RELEASE-TCTFMAX}(\mathcal{J}, y^{(h+1)})$, it takes $O(n)$ time to generate a tentative schedule. Since each job violating its inequality at a position can only be moved to the left, it goes through at most $n-1$ positions. Therefore, the total number of tentative schedules is $O(n^2)$. Step 2 of Algorithm RELEASE-TCTFMAX requires $O(n^3)$ time for all iterations. Step 3 of Algorithm RELEASE-TCTFMAX can be implemented in $O(n^2)$ time, since the total number of tentative schedules is $O(n^2)$. Hence, Algorithm REALEASE-TCTFMAX runs in $O(n^3)$ time, too.

Algorithm REALEASE-TCTFMAX works in a different way from Algorithm NOREALEASE-TCTFMAX. As shown in Lemma 3.6, the convenience of Algorithm NOREALEASE-TCTFMAX is that each iteration (one call for Procedure $P_{NORELEASE-TCTFMAX}$) determines a Pareto optimal schedule. Consequently, there are exactly $|\Omega(\mathcal{J})| + 1$ iterations in Algorithm NOREALEASE-TCTFMAX. It finds only Pareto optimal schedules. On the other hand, during the implementation of Algorithm REALEASE-TCTFMAX, several schedules belonging to different iterations may have different $f_{\max}$-values but the same $\sum_{j=1}^n C_j$-value. Therefore, Algorithm REALEASE-TCTFMAX may perform more iterations than $|\Omega(\mathcal{J})| + 1$, but the total number of iterations is still bounded by $O(n^2)$. It finds all weak Pareto optimal schedules, and, therefore, will not miss any Pareto optimal schedule.

Based on Lemmas 3.2 and 3.4 with slight modifications, we get the following main result. The proof is omitted, since it is standard and can be found, e.g., [1].

Theorem 4.2. *Algorithm RELEASE-TCTFMAX solves* $1|\bar{k}_j, (r_j, p_j)-agreeable|(\sum_{j=1}^n C_j, f_{\max})$ *in* $O(n^3)$ *time.*

5. Conclusions

In this paper, we studied the bicriteria problem of scheduling jobs with positional deadlines and agreeable release and processing times on a single machine to minimize total completion time and maximum cost simultaneously. We presented an $O(n^3)$-time Pareto optimal algorithm for this problem, which generalizes and improves the previously known two $O(n^4)$-time algorithms for the case of equal release times and an $O(n^3)$-time algorithm for the case of equal processing times (without positional deadline constraints). For future research, it is interesting to design algorithms with better time complexity for the problem. The extensions to batch scheduling (serial-batch or parallel-batch), multi-agent scheduling, or to the case of arbitrary release and processing times for optimal or approximation algorithms, is encouraged. We can also consider more general min-sum objective functions instead of total completion time, in combination with a min-max or min-sum objective function.

Acknowledgments

We thank the editor and reviewers for their helpful suggestions. This work is supported by Natural Science Foundation of Shandong Province China (No. ZR2020MA030), and Shandong Soft Science Project (No. 2021RKY02040).

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. H. Hoogeveen, Multicriteria scheduling, *Eur. J. Oper. Res.*, **167** (2005), 592–623. https://doi.org/10.1016/j.ejor.2004.07.011
2. P. Brucker, *Scheduling Algorithms*, fifth Edition, Springer, 2007. https://doi.org/10.1007/978-3-540-69516-5
3. V. T'kindt, J. C. Billaut, *Multicriteria Scheduling: Theory, Models and Algorithms*, Springer Verlag, Berlin, 2006.
4. J. K. Lenstra, A. R. Kan, P. Brucker, Complexity of machine scheduling problems, in *Annals of Discrete Mathematics*, Elsevier, (1977), 343–362. https://doi.org/10.1016/S0167-5060(08)70743-X
5. J. K. Lenstra, A. R. Kan, Complexity of scheduling under precedence constraints, *Oper. Res.*, **26** (1978), 22–35. https://doi.org/10.1287/opre.26.1.22
6. Y. Gao, J. Yuan, Pareto minimizing total completion time and maximum cost with positional due indices, *J. Oper. Res. Soc. China*, **3** (2015), 381–387.
7. Z. Liu, S. Li, Pareto optimal algorithms for minimizing total (weighted) completion time and maximum cost on a single machine, *Math. Biosci. Eng.*, **19** (2022), 7337–7348. https://doi.org/10.3934/mbe.2022346

8. P. Perez-Gonzalez, J. M. Framinan, A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems, *Eur. J. Oper. Res.*, **235** (2014), 1–16. https://doi.org/10.1016/j.ejor.2013.09.017

9. A. Herzel, S. Ruzika, C. Thielen, Approximation methods for multiobjective optimization problems: A survey, *INFORMS J. Comput.*, **33** (2021), 1284–1299. https://doi.org/10.1287/ijoc.2020.1028

10. L. N. V. Wassenhove, L. F. Gelders, Solving a bicriterion scheduling problem, *Eur. J. Oper. Res.*, **4** (1980), 42–48. https://doi.org/10.1016/0377-2217(80)90038-7

11. T. C. John, Tradeoff solutions in single machine production scheduling for minimizing flow time and maximum penalty, *Comput. Oper. Res.*, **16** (1989), 471–479. https://doi.org/10.1016/0305-0548(89)90034-8

12. J. Hoogeveen, S. L. van de Velde, Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time, *Oper. Res. Lett.*, **17** (1995), 205–208. https://doi.org/10.1016/0167-6377(95)00023-D

13. G. Steiner, P. Stephenson, Pareto optima for total weighted completion time and maximum lateness on a single machine, *Discrete Appl. Math.*, **155** (2007), 2341–2354. https://doi.org/10.1016/j.dam.2007.06.012

14. Y. Gao, J. Yuan, A note on pareto minimizing total completion time and maximum cost, *Oper. Res. Lett.*, **43** (2015), 80–82. https://doi.org/10.1016/j.orl.2014.12.001

15. C. He, H. Lin, X. Wang, Single machine bicriteria scheduling with equal-length jobs to minimize total weighted completion time and maximum cost, *4OR-Q. J. Oper. Res.*, **12** (2014), 87–93. https://doi.org/10.1007/s10288-013-0244-1

16. Q. Zhao, L. Lu, J. Yuan, Rescheduling with new orders and general maximum allowable time disruptions, *4OR-Q. J. Oper. Res.*, **14** (2016), 261–280. https://doi.org/10.1007/s10288-016-0308-0

17. Q. Zhao, J. Yuan, Rescheduling to minimize the maximum lateness under the sequence disruptions of original jobs, *Asia Pac. J. Oper. Res.*, **34** (2017), 1750024. https://doi.org/10.1142/S0217595917500245

18. Y. Gao, J. Yuan, Bi-criteria pareto-scheduling on a single machine with due indices and precedence constraints, *Discrete Optim.*, **25** (2017), 105–119. https://doi.org/10.1016/j.disopt.2017.02.004

19. R. Chen, J. Yuan, Single-machine scheduling of proportional-linearly deteriorating jobs with positional due indices, *4OR-Q. J. Oper. Res.*, **18** (2020), 177–196. https://doi.org/10.1007/s10288-019-00410-4

20. R. Chen, J. Yuan, L. Lu, Single-machine scheduling with positional due indices and positional deadlines, *Discrete Optim.*, **34** (2019), 100549. https://doi.org/10.1016/j.disopt.2019.06.002

21. R. Chen, J. Yuan, Z. Geng, Nd-agent scheduling of linear-deteriorating tasks with positional due indices to minimize total completion time and maximum cost, *Appl. Math. Comput.*, **365** (2020), 124697. https://doi.org/10.1016/j.amc.2019.124697

22. Y. Gao, J. Yuan, C. Ng, T. Cheng, A note on competing-agent pareto-scheduling, *Optim. Lett.*, **15** (2021), 249–262. https://doi.org/10.1007/s11590-020-01576-1

23. C. He, C. Xu, H. Lin, Serial-batching scheduling with two agents to minimize makespan and maximum cost, *J. Sched.*, **23** (2020), 609–617. https://doi.org/10.1007/s10951-020-00656-5
24. Z. Geng, J. Liu, Single machine batch scheduling with two non-disjoint agents and splitable jobs, *J. Comb. Optim.*, **40** (2020), 774–795. https://doi.org/10.1007/s10878-020-00626-9
25. Y. Gao, J. Yuan, C. Ng, T. Cheng, Pareto-scheduling with family jobs or nd-agent on a parallel-batch machine to minimize the makespan and maximum cost, *4OR-Q. J. Oper. Res.*, **20** (2022), 273–287. https://doi.org/10.1007/s10288-021-00480-3
26. S. Li, Z. Geng, Bicriteria scheduling on an unbounded parallel-batch machine for minimizing makespan and maximum cost, *Inf. Process. Lett.*, **180** (2023), 106343. https://doi.org/10.1016/j.ipl.2022.106343
27. W. E. Smith, Various optimizers for single-stage production, *Nav. Res. Log.*, **3** (1956), 59–66. https://doi.org/10.1002/nav.3800030106

Appendix

Proof of Lemma 3.6:

Proof. By Lemma 3.4, $\sigma^{(h)}$ is optimal for $1|\bar{k}_j, f_{\max} < y^{(h)}|\sum_{j=1}^n C_j$. By Lemma 3.2, to prove that $\sigma^{(h)}$ is Pareto optimal for $1|\bar{k}_j|(\sum_{j=1}^n C_j, f_{\max})$, we only need to show that it is optimal for $1|\bar{k}_j, \sum_{j=1}^n C_j \le \sum_{j=1}^n C_j(\sigma^{(h)})|f_{\max}$.

Suppose that $\pi \neq \sigma^{(h)}$ is optimal for $1|\bar{k}_j, \sum_{j=1}^n C_j \le \sum_{j=1}^n C_j(\sigma^{(h)})|f_{\max}$. We have: $\sum_{j=1}^n C_j(\pi) \le \sum_{j=1}^n C_j(\sigma^{(h)})$ and $f_{\max}(\pi) \le f_{\max}(\sigma^{(h)}) < y^{(h)}$. Therefore, π is a feasible schedule for $1|\bar{k}_j, f_{\max} < y^{(h)}|\sum_{j=1}^n C_j$. We get $\sum_{j=1}^n C_j(\pi) \ge \sum_{j=1}^n C_j(\sigma^{(h)})$. It follows that $\sum_{j=1}^n C_j(\pi) = \sum_{j=1}^n C_j(\sigma^{(h)})$.

We compare $\sigma^{(h)}$ and π backwardly looking for a difference between the jobs. Suppose that the first difference occurs at the k-th position, which is occupied by jobs J_i and J_j in $\sigma^{(h)}$ and π, respectively. From the proof of Lemma 3.3, we do not worry about the position indices of the jobs, since they do not affect the feasibility of any schedule in $\Pi\left(\mathcal{J}, y^{(h)}\right)$. Hence, by the BRLPT-SC rule, we get $p_i \ge p_j$.

We claim that $p_i = p_j$ must hold. (Otherwise, we interchange J_i and J_j in π to get π', which is feasible for $1|\bar{k}_j, f_{\max} < y^{(h)}|\sum_{j=1}^n C_j$ and $\sum_{j=1}^n C_j(\pi') < \sum_{j=1}^n C_j(\pi) = \sum_{j=1}^n C_j(\sigma^{(h)})$, contradicting the fact that $\sigma^{(h)}$ is optimal for $1|\bar{k}_j, f_{\max} < y^{(h)}|\sum_{j=1}^n C_j$.) Moreover, by the BRLPT-SC rule, $f_i(C_i(\sigma^{(h)})) \le f_j(C_j(\pi))$. Thus, we can safely interchange J_i and J_j in π without affecting the cost of the schedule.

Repetition of this argument shows that π can be safely transformed into $\sigma^{(h)}$, without affecting the cost of the schedule. Therefore, $\sigma^{(h)}$ is also optimal for $1|\bar{k}_j, \sum_{j=1}^n C_j \le \sum_{j=1}^n C_j(\sigma^{(h)})|f_{\max}$.

Hence, $\sigma^{(h)}$ is Pareto optimal for $1|\bar{k}_j|(\sum_{j=1}^n C_j, f_{\max})$.