A note on competing-agent Pareto-scheduling

Yuan Gao[1,2], **Jinjiang Yuan**[1*], **C.T. Ng**[3], **T.C.E. Cheng**[3]

[1]School of Mathematics and Statistics, Zhengzhou University,

Zhengzhou, Henan 450001, People's Republic of China

[2]School of Information Engineering, Zhengzhou University,

Zhengzhou, Henan 450001, People's Republic of China

[3]Logistics Research Centre, Department of Logistics and Maritime Studies,

The Hong Kong Polytechnic University, Hong Kong, People's Republic of China

---

Abstract: We consider Pareto-scheduling with two competing agents A and B on a single machine, in which each job has a positional due index and a deadline. The jobs of agents A and B are called the A-jobs and B-jobs, respectively, where the A-jobs have a common processing time, while the B-jobs are restricted by their precedence constraint. The objective is to minimize a general sum-form objective function of the A-jobs and a general max-form objective function of the B-jobs, where all the objective functions are regular. We show that the problem is polynomially solvable.

Keywords: competing agents; Pareto-scheduling; positional due indices; deadline; precedence constraint

1 Introduction

Agnetis et al. [2] introduced two-agent scheduling. In this model, there are two competing agents A and B, and n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ that are partitioned into two subsets $\mathcal{J}^A =$

*Corresponding author: Jinjiang Yuan. Email address: yuanjj@zzu.edu.cn

$\{J_1^A, J_2^A, \ldots, J_{n_A}^A\}$ (called the A-jobs) and $\mathcal{J}^B = \{J_1^B, J_2^B, \ldots, J_{n_B}^B\}$ (called the B-jobs). Each agent $X \in \{A, B\}$ has a scheduling objective function γ^X to be minimized. Agnetis et al. [1] collected most of the research results on two-agent scheduling.

We consider in this paper Pareto-scheduling with two competing agents to minimize the objective functions γ^A and γ^B. A feasible schedule σ of the n jobs is called *nondominated* if there exists no other feasible schedule π such that

$$(\gamma^A(\pi), \gamma^B(\pi)) \leq (\gamma^A(\sigma), \gamma^B(\sigma)) \text{ and } (\gamma^A(\pi), \gamma^B(\pi)) \neq (\gamma^A(\sigma), \gamma^B(\sigma)).$$

In this case, we also call $(\gamma^A(\sigma), \gamma^B(\sigma))$ a *nondominated pair*. The goal of the problem is to find all the nondominated pairs and, for each nondominated pair, a corresponding nondominated schedule. Following the notation introduced in T'kindt and Billaut [15], we denote the Pareto-scheduling problem with two competing agents to minimize the objective functions γ^A and γ^B as $\alpha|\beta|^{\#}(\gamma^A, \gamma^B)$, where α represents the machine environment and β represents the job characteristics or the feasibility conditions.

Zhao and Yuan [18] introduced scheduling with the *due-index constraint*. In this scheduling model, each job J_j is associated with a position index $k_j \in \{1, 2, \ldots, n\}$, called the *positional due index* (due index) of job J_j, which indicates the latest tolerable position of job J_j in a feasible schedule. Given a schedule σ of the n jobs $J_1, J_2, \ldots, J_n$, the *position number* of a job J_j in schedule σ is denoted by $\sigma[J_j]$. Thus, $\sigma[J_j] = x$ if and only if $x \in \{1, 2, \ldots, n\}$ and job J_j is scheduled at the x-th position in schedule σ. The due-index constraint requires that, in a feasible schedule σ, we must have $\sigma[J_j] \leq k_j$ for all $j = 1, 2, \ldots, n$. Chen et al. [3], Gao and Yuan [5], and Zhao and Yuan [18] documented most of the research results for this new scheduling model.

Although the due-index constraint "k_j" was originally introduced in Zhao and Yuan [18] in the research of rescheduling problems, it has many applications in the real life. In some servicing or production systems, processing sequences of the jobs (which are either the customers waiting for services or the tasks to be processed) must be determined. When the jobs have their requirements for the latest tolerable positions in the processing sequences, the due-index constraint can be used in the formulation of the corresponding scheduling problems. Concrete examples of such applications can be found in Chen et al. [3].

In this paper we study Pareto-scheduling with two competing agents on a single machine to minimize the total scheduling cost of the A-jobs $\sum f_j^A$ and the maximum scheduling cost of the B-jobs $g_{\max}^B$. In the problem, each job $J_j \in \mathcal{J}$ has a positional due index k_j and a deadline $\bar{d}_j$, the A-jobs have a common processing time p^A, and the B-jobs have a precedence constraint, i.e., $J_j^B \prec J_k^B$ implies that job J_j^B must be processed before job J_k^B in any feasible schedule. Then we denote the Pareto-scheduling problem as

$$1|k_j, \bar{d}_j, p_i^A = p^A, \text{prec}^B|^\#(\sum f_i^A, g_{\max}^B), \tag{1}$$

where "k_j" denotes the due-index constraint, "$\bar{d}_j$" denotes the deadline constraint, "$p_i^A = p^A$" denotes the common processing time p^A of the A-jobs, and "prec^B" denotes the precedence constraint of the B-jobs. In particular, we assume that $\sum f_i^A$ is a general sum-form objective function of the A-jobs and $g_{\max}^B$ is a general max-form objective function of the B-jobs under the assumption that f_i^A, $i \in \{1, 2, \dots, n_A\}$, and g_j^B, $j \in \{1, 2, \dots, n_B\}$, are regular functions, where a function $f(t)$, $t \in [0, +\infty)$, is regular if $f(t)$ is nondecreasing in t. Our goal is to find a polynomial-time algorithm to solve problem (1). For simplicity, we use $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$ to denote problem (1), where $\beta^* = \{k_j, \bar{d}_j, p_i^A = p^A, \text{prec}^B\}$.

Note that the two restrictions $p_i^A = p^A$ and prec^B in the β^*-field in problem (1) cannot be further extended for two reasons. First, problem $1||\sum w_j T_j$ is unary NP-hard, as shown in Lawler [12]. Second, problem $1|chains, p_j = 1|\sum U_j$ is unary NP-hard, as shown in Lenstra and Rinnooy Kan [13].

There are fruitful research results on Pareto-scheduling problems in the literature. Due to the page limit, we only summarize in Table 1 the most closely related known results. Note that our problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$, i.e., $1|k_j, \bar{d}_j, p_i^A = p^A, \text{prec}^B|^\#(\sum f_i^A, g_{\max}^B)$, is a generalization of the problem $1|k_j, p_j^A = p^A, \text{prec}^B|^\#(\sum w_j C_j^A, f_{\max}^B)$ studied in Gao and Yuan [5] and the problem $1|p_j = 1|^\#(\sum f_j^A, g_{\max}^B)$ studied in Oron et al. [14]. This motivates us to study problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$.

Table 1: Known results related to this study.

Scheduling problems	Time complexity	References
$1\|\|^{\#}(\sum C_j, f_{\max})$	$O(n^4)$	Hoogeveen [9]
$1\|\|^{\#}(f_{\max}, g_{\max})$	$O(n^4)$	Hoogeveen [10]
$1\|\|^{\#}(\sum C_j^A, f_{\max}^B)$	$O(n_A^2 n_B \log n_A + n_A n_B^3)$	Agnetis et al. [2]
$1\|\|^{\#}(f_{\max}^A, g_{\max}^B)$	$O(n_A^3 n_B + n_A n_B^3)$	Agnetis et al. [2]
$1\|p_j = p\|^{\#}(\sum w_j C_j, f_{\max})$	$O(n^3 \log n)$	He et al. [7]
$1\|k_j\|^{\#}(\sum C_j, f_{\max})$	$O(n^4)$	Gao and Yuan [4]
$1\|p_j = 1\|^{\#}(f_{\max}^A, g_{\max}^B)$	$O(n_A^2 + n_B^2 + n_A n_B \log n_B)$	Oron et al. [14]
$1\|p_j = 1\|^{\#}(\sum f_j^A, g_{\max}^B)$	$O(n_B n \max\{n_A^3, n\})$	Oron et al. [14]
$1\|\|^{\#}(\sum U_j^A, g_{\max}^B)$	$O(\min\{n_A, n_B\} n_A^2 (n \log n + n_B^2))$	Wan et al. [16]
$1\|k_j, \mathrm{prec}\|f_{\max}$	$O(n^2)$	Zhao and Yuan [18]
$1\|k_j, \mathrm{prec}\|^{\#}(f_{\max}, g_{\max})$	$O(n^4)$	Gao and Yuan [5]
$1\|k_j, \mathrm{prec}\|^{\#}(f_{\max}^A : g_{\max}^B)$	$O(n_A^3 n_B + n_A n_B^3)$	Gao and Yuan [5]
$1\|k_j, p_j = p\|^{\#}(\sum w_j C_j, f_{\max})$	$O(n^3 \log n)$	Gao and Yuan [5]
$1\|k_j, p_j^A = p^A, \mathrm{prec}^B\|^{\#}(\sum w_j C_j^A, f_{\max}^B)$	$O(n n_A n_B^2 + n_A^2 n_B \log n_A)$	Gao and Yuan [5]
$1\|k_j, \mathrm{prec}^B\|^{\#}(\sum C_j^A, f_{\max}^B)$	$O(n n_A n_B^2 + n_A^2 n_B \log n_A)$	Gao and Yuan [5]

Recently, Zhang and Yuan [17] presented an $O(n_B^2 + n_A^3)$ algorithm for problem $1|p_i^A = p^A|\sum f_i^A : g_{\max}^B \le Q$. This provides a basic idea for us to solve problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$.

The organization and contributions of this paper, together with the approaches used in this paper, can be stated as follows.

In Section 2, we introduce some notation and a basic lemma on Pareto-scheduling problems.

In Section 3, we establish an important lemma, which shows that problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$ can be reduced in linear time to an auxiliary problem $1|\beta^*|^{\#}(\sum F_i^A, g_{\max}^B)$ on the same job set, where $F_i^A(t)$ is strictly increasing in $t \ge 0$ for $i = 1, 2, \ldots, n_A$. Such a reduction borrows the technique used in Gao and Yuan [5] for dealing with the problem $1|k_j, \mathrm{prec}|^{\#}(f_{\max}, g_{\max})$. This lemma enables us to assume in the followed discussions that each $f_i^A(t)$ is strictly increasing in $t \ge 0$ for $i = 1, 2, \ldots, n_A$. Our discussions in Section 4 depend heavily on this assumption.

In Section 4, we first present an $O(n_B^2 + n_A^3)$ solution algorithm, denoted Algorithm I, for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$, which results a schedule optimal for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$ and nondominated for problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$. Our Algorithm I is derived from the algorithm in Zhang and Yuan [17] for solving problem $1|p_i^A = p^A|\sum f_i^A : g_{\max}^B \le Q$. The main change is that, when we need to schedule an unscheduled B-job in position h for completing at time t, we always pick an available B-job J_j^B to schedule such that $g_j^B(t)$ is as small as possible. This modification guarantees that Algorithm I always yields an optimal and nondominated schedule. Such an approach for determining the B-job to be scheduled has been widely used in bicriteria scheduling research, for example, in Agnetis et al. [2], Gao and Yuan [4, 5], He et al. [7], and Hoogeveen [9, 10]. After this, we present another algorithm, denoted Algorithm II, to generate all the nondominated pairs of problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$. Each iteration of Algorithm II generates a new nondominated pair by running Algorithm 1 based on a new value of Q. Execution of Algorithm II is in fact a general approach for solving Pareto-scheduling problems, with a detail discussion in Geng and Yuan [6]. Thus, from Geng and Yuan [6], the time complexity of Algorithm II depends on the time complexity of Algorithm I and the number of nondominated pairs of problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$. By using a new technique to calculate the number of nondominated pairs, we show that problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$ has at most $n_A n_B + 1$ nondominated pairs. As a result, problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$ is solvable in $O(n_A n_B^3 + n_A^4 n_B) = O(n^5)$ time.

2 Preliminaries

We use the following notation throughout the paper.

- k_i^X is the positional due index of the X-job J_i^X, where $X \in \{A, B\}$.
- $\bar{d}_i^X$ is the deadline of the X-job J_i^X, where $X \in \{A, B\}$.
- $p_i^A = p^A$ is the processing time of the A-job J_i^A.
- p_j^B is the processing time of the B-job J_j^B.
- $P_B = \sum_{j=1}^{n_B} p_j^B$ is the total processing time of the B-jobs.
- $J_{\sigma(i)}$ is the job scheduled in the i-th position in schedule σ.
- $J_{\sigma_A(i)}^A$ is the i-th processed A-job in schedule σ, where we regard σ_A as the restriction of schedule σ on the A-jobs.
- $\sigma[J_i^X]$ is the position number of the X-job J_i^X in schedule σ, where $X \in \{A, B\}$. This means that J_i^X is the $\sigma[J_i^X]$-th job in schedule σ. Then we have $\sigma[J_i^X] \leq k_i^X$ under the due-index constraint.
- $C_i^X(\sigma)$ is the completion time of the X-job J_i^X in schedule σ, where $X \in \{A, B\}$. Then we have $C_i^X(\sigma) \leq \bar{d}_i^X$ under the deadline constraint.

- $f_i^A(t)$ and $g_j^B(t)$ are two functions nondecreasing in $t \geq 0$ for each i with $1 \leq i \leq n_A$ and each j with $1 \leq j \leq n_B$. In this case, we also say that functions $f_i^A(\cdot)$ and $g_j^B(\cdot)$ are *regular*.
- $\sum f_i^A = \sum_{i=1}^{n_A} f_i^A(C_i^A)$ is the total scheduling cost of all the A-jobs under the function vector $(f_1^A, f_2^A, \ldots, f_{n_A}^A)$.
- $g_{\max}^B = \max\{g_j^B(C_j^B) : 1 \leq j \leq n_B\}$ is the maximum scheduling cost of all the B-jobs under the function vector $(g_1^B, g_2^B, \ldots, g_{n_B}^B)$.

We take the following convention throughout the paper.

Convention: *All the parameters in the paper are integer-valued. Especially, the processing time of each job is a positive integer and the functions $f_i^A(\cdot)$ and $g_j^B(\cdot)$ are integer-valued.*

Associated with the Pareto-scheduling problem $1|\beta|^\#(\gamma^A, \gamma^B)$ is its constrained version, denoted as $1|\beta|\gamma^A : \gamma^B \leq Q$, which seeks to find a feasible schedule σ so that $\gamma^A(\sigma)$ is minimized under the condition that $\gamma^B(\sigma) \leq Q$. The following lemma is due to Hoogeveen [8].

Lemma 2.1. *Suppose that schedule σ is optimal for problem $1|\beta|\gamma^A : \gamma^B \leq Q$ and schedule π is optimal for problem $1|\beta|\gamma^B : \gamma^A \leq \gamma^A(\sigma)$. Then $\gamma^A(\pi) = \gamma^A(\sigma)$ and schedule π is nondominated for problem $1|\beta|^\#(\gamma^A, \gamma^B)$.*

3 An important lemma

Consider problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$ on the job set $\{J_1^A, J_2^A, \ldots, J_{n_A}^A, J_1^B, J_2^B, \ldots, J_{n_B}^B\}$, where the functions $f_i^A(t)$ $(i = 1, 2, \ldots, n_A)$ and $g_j^B(t)$ $(j = 1, 2, \ldots, n_B)$ are integer-valued and nondecreasing in $t \geq 0$. For each function $f_i^A(t)$ $(i = 1, 2, \ldots, n_A)$, by adding a common sufficiently large positive integer, we may assume that $f_i^A(t) \geq 1$ for all $t \geq 0$. Suppose that we only consider the schedules in which every job completes at an integral time point. Following Gao and Yuan [5], we use the following notation in our discussion.

- L is a sufficiently large integer so that, for every reasonable schedule σ, we have $C_{\max}(\sigma) \leq L$.
- $M = n_A L + 1$.
- $F_i^A(t) = M f_i^A(t) + t$, $i = 1, 2, \ldots, n_A$.

In our problem, we may define $L = n_A p^A + P_B$. It can be easily verified that each function $F_i^A(t)$, $i = 1, 2, \ldots, n_A$, is strictly increasing in $t \geq 0$. For problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$ on the job set $\mathcal{J} = \{J_1^A, J_2^A, \ldots, J_{n_A}^A, J_1^B, J_2^B, \ldots, J_{n_B}^B\}$, we call the problem

$$1|\beta^*|^\#(\sum F_i^A, g_{\max}^B) \tag{2}$$

on the same job set $\mathcal{J}$ the *auxiliary problem*. Since the two problems are defined on the same job set, we use schedules of the n jobs for both problems at the same time. According to the definitions of M and function $F_i^A(t)$ $(i = 1, 2, \ldots, n_A)$, we obtain the following two observations.

Observation 3.1. *Suppose that f', f'', x, and y are positive integers, where $x, y \in \{1, 2, \ldots, n_A L\}$. If $Mf' + x \le Mf'' + y$, then $f' \le f''$. Moreover, if $Mf' + x = Mf'' + y$, then $f' = f''$ and $x = y$.*

Observation 3.2. *For every feasible schedule σ, we have $\sum F_i^A(\sigma) = M \sum f_i^A(\sigma) + \sum C_i^A(\sigma) \le M \sum f_i^A(\sigma) + n_A L$.*

The following lemma reveals an essential relation between problem $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$ and its auxiliary problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$.

Lemma 3.1. *Consider the two problems $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$ and $1|\beta^*|\#(\sum F_i^A,\ g_{\max}^B)$ on the same job set $\mathcal{J} = \mathcal{J}^A \cup \mathcal{J}^B$. If (f, g) is a nondominated pair of problem $1|\beta^*|\#(\sum f_i^A,\ g_{\max}^B)$, then $(Mf+x, g)$ is a nondominated pair of problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$ for some $x \in \{1, 2, \ldots, n_A L\}$.*

Proof. Suppose that (f, g) is a nondominated pair of problem $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$ and let σ be a nondominated schedule of the problem corresponding to (f, g). Then $\sum f_i^A(\sigma) = f$ and $g_{\max}^B(\sigma) = g$. From Observation 3.2, we have $\sum F_i^A(\sigma) = M \sum f_i^A(\sigma) + \sum C_i^A(\sigma) = Mf + \sum C_i^A(\sigma) \le Mf + n_A L$. Thus $\sum F_i^A(\sigma) = Mf + x^*$ for some $x^* \in \{1, 2, \ldots, n_A L\}$. It follows that $(Mf + x^*, g)$ is an objective vector of problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$.

Suppose to the contrary that $(Mf + x, g)$ is not a nondominated pair of the auxiliary problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$ for each $x \in \{1, 2, \ldots, n_A L\}$. Then $(Mf + x^*, g)$ is not a nondominated pair of problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$, either. Therefore, there is a nondominated schedule π for problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$ such that $(\sum F_i^A(\pi), g_{\max}^B(\pi)) \le (Mf + x^*, g)$, and either $\sum F_i^A(\pi) < Mf + x^*$ and $g_{\max}^B(\pi) = g$, or $g_{\max}^B(\pi) < g$. From Observation 3.2, we have $\sum F_i^A(\pi) = M \sum f_i^A(\pi) + x'$ for some $x' \in \{1, 2, \ldots, n_A L\}$. Since $(\sum F_i^A(\pi), g_{\max}^B(\pi)) \le (Mf+x^*, g)$, we have $M \sum f_i^A(\pi) + x' \le Mf + x^*$. From Observation 3.1, we have $\sum f_i^A(\pi) \le f$. Since (f, g) is a nondominated pair of problem $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$, we have $g_{\max}^B(\pi) = g$. Thus, $\sum F_i^A(\pi) = M \sum f_i^A(\pi) + x' < Mf + x^*$. From Observation 3.1, we have $\sum f_i^A(\pi) < f$. This contradicts the assumption that (f, g) is a nondominated pair of problem $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$ since $\sum f_i^A(\pi) < f$ and $g_{\max}^B(\pi) = g$. The result follows. □

The result in Lemma 3.1 implies that, if $\mathcal{P}(\sum F_i^A, g_{\max}^B) = \{(Mf^{(i)} + x^{(i)}, g^{(i)}) : 1 \le i \le K\}$ is the set of nondominated pairs of problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$, then the set of all the nondominated pairs of problem $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$, denoted by $\mathcal{P}(\sum f_i^A, g_{\max}^B)$, is a subset of $\{(f^{(i)}, g^{(i)}) : 1 \le i \le K\}$. We may assume that $g^{(1)} > g^{(2)} > \cdots > g^{(K)}$. Then $\mathcal{P}(\sum f_i^A, g_{\max}^B)$ can be obtained from $\{(f^{(i)}, g^{(i)}) : 1 \le i \le K\}$ in $O(K)$ time, which is dominated by the time complexity for solving problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$. Consequently, we can solve problem $1|\beta^*|\#(\sum f_i^A, g_{\max}^B)$ by using the solution for its auxiliary problem $1|\beta^*|\#(\sum F_i^A, g_{\max}^B)$.

In order to increase the comprehensibility, let us consider in the following a small numerical example. In the job instance, we have $\mathcal{J}^A = \{J_1^A, J_2^A, J_3^A\}$ and $\mathcal{J}^B = \{J_1^B, J_2^B, J_3^B, J_4^B\}$. Each job J_j^X, $X \in \{A, B\}$, has a weight w_j^X, a processing time p_j^X, and a due date d_j^X. For simplicity, the due-index constraint, the deadline constraint, and the precedence constraint are not considered. Table 2 presents all the parameters of the job instance. We may choose $L = 9$, which is the total processing time of all jobs. Then $M = n_A L + 1 = 28$. In a schedule σ of the jobs in $\mathcal{J}^A \cup \mathcal{J}^B$, we use $T_j^X(\sigma) = \max\{0, C_i^X(\sigma) - d_j\}$ to denote the tardiness of the X-job J_j^X. Moreover, the objective functions of the A-jobs and the B-jobs are concretized as $\sum f_j^A = \sum_{j=1}^{3} w_j^A T_j^A$ (the total weighted tardiness) and $g_{\max}^B = \max\{w_j^B T_j^B : j = 1, 2, 3, 4\}$ (the maximum weighted tardiness). Then $\sum F_i^A = 28 \sum f_j^A + \sum C_j^A$. For the auxiliary problem $1|\beta^*|^{\#}(\sum F_i^A, g_{\max}^B)$, the nondominated pairs are given by $\mathcal{P}(\sum F_i^A, g_{\max}^B) = \{(6, 8), (8, 7), (96, 6), (326, 4), (499, 3)\}$ and the corresponding nondominated schedules $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$ are described in Table 3. The nondominated pairs of problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$ are given by $\mathcal{P}(\sum f_i^A, g_{\max}^B) = \{(0, 7), (3, 6), (11, 4), (17, 3)\}$, which are the nondominated pairs in $\{(f^{(i)}, g^{(i)}) : 1 \le i \le 5\} = \{(0, 8), (0, 7), (3, 6), (11, 4), (17, 3)\}$.

	J_1^A	J_2^A	J_3^A
w_j^A	1	2	2
p_j^A	1	1	1
d_j^A	6	3	4

	J_1^B	J_2^B	J_3^B	J_4^B
w_j^B	2	1	3	2
p_j^B	1	2	1	2
d_j^B	3	2	6	1

Table 2. The parameters of agent A and agent B.

$(f^{(i)}, g^{(i)})$	$\mathcal{P}(\sum F_i^A, g_{\max}^B)$	nondominated schedules
$(0, 8)$	$(6, 8)$	$\sigma_1 = (J_1^A, J_2^A, J_3^A, J_4^B, J_1^B, J_3^B, J_2^B)$
$(0, 7)$	$(8, 7)$	$\sigma_2 = (J_1^A, J_2^A, J_4^B, J_3^A, J_1^B, J_3^B, J_2^B)$
$(3, 6)$	$(96, 6)$	$\sigma_3 = (J_2^A, J_3^A, J_4^B, J_1^B, J_3^B, J_2^B, J_1^A)$
$(11, 4)$	$(326, 4)$	$\sigma_4 = (J_2^A, J_4^B, J_1^B, J_2^B, J_3^B, J_3^A, J_1^A)$
$(17, 3)$	$(499, 3)$	$\sigma_5 = (J_4^B, J_1^B, J_2^B, J_3^A, J_3^B, J_2^A, J_1^A)$

Table 3. The nondominated pairs and nondominated schedules.

4 Pareto optimization

From Section 3, we only need to consider problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$ under the assumption that each function $f_i^A(t)$, $i = 1, 2, \ldots, n_A$, is strictly increasing in $t \geq 0$. To check the feasibility of the problem, we define $g_i^A(\cdot) = -\infty$ for each $i = 1, 2, \ldots, n_A$. Let $g_{\max} = \max\{g_{\max}^A, g_{\max}^B\}$. Then problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$ is feasible if and only if the optimal value of problem $1|\beta^*|g_{\max}$, i.e., problem $1|k_j, \bar{d}_j, p_i^A = p^A, \text{prec}^B|g_{\max}$, is a finite number.

Recall that Zhao and Yuan [18] presented an $O(n^2)$ algorithm to solve problem $1|k_j, \text{prec}|f_{\max}$. Their algorithm is also applicable for problem $1|k_j, \bar{d}_j, \text{prec}|f_{\max}$, which is equivalent to problem $1|k_j, \text{prec}|f'_{\max}$, where $f'_j(t) = f_j(t)$ if $t \leq \bar{d}_j$ and $f'_j(t) = +\infty$ if $t > \bar{d}_j$.

So we can use the algorithm in Zhao and Yuan [18] for problem $1|\beta^*|g_{\max}$ to obtain its optimal value, denoted as $G_{\min}$. Clearly, $G_{\min}$ is the minimum value of $g_{\max}^B(\sigma)$ among all the feasible schedules σ. In the sequel, we assume that problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$ is feasible. Then $G_{\min} < +\infty$. Given a number $Q \geq G_{\min}$, we present an algorithm for solving the constrained problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$ as follows:

Algorithm I: *For problem* $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$.

Input: A feasible job set $\{J_1^A, J_2^A, \ldots, J_{n_A}^A, J_1^B, J_2^B, \ldots, J_{n_B}^B\}$.

Preprocessing: Sorting the B-jobs in the topological order under their precedence constraint.

Step 1: Set $t := n_A p^A + P_B$, $h := n$, $l := n_A$, $\mathcal{J}_0^A := \mathcal{J}^A$, and $\mathcal{J}_0^B := \mathcal{J}^B$.

Step 2: Let $U^h(t)$ be the set of all the B-jobs $J_j^B \in \mathcal{J}_0^B$ with $g_j^B(t) \leq Q$, $k_j^B \geq h$, $\bar{d}_j^B \geq t$, and J_j^B having no successor in $\mathcal{J}_0^B$ under the precedence constraint of the B-jobs. (Then $U^h(t)$ consists of all the B-jobs that are available in position h for completing at time t.) Do the following:

– If $U^h(t) \neq \emptyset$, then go to Step 3.

– If $U^h(t) = \emptyset$, then go to Step 4.

Step 3: (The case that $U^h(t) \neq \emptyset$.) Choose a B-job J_j^B from $U^h(t)$ such that $g_j^B(t)$ is as small as possible. Schedule the B-job J_j^B in the interval $[t - p_j^B, t]$ as the h-th job in the final schedule. Set $t := t - p_j^B$, $h := h - 1$, and $\mathcal{J}_0^B := \mathcal{J}_0^B \setminus \{J_j^B\}$. Do the following:

– If $t > 0$, then go to Step 2.

– If $t = 0$, then go to Step 5.

Step 4: (The case that $U^h(t) = \emptyset$.) Define $T_l = [t - p^A, t]$. (Some uncertain A-job J_i^A with

$k_i^A \geq h$ and $\bar{d}_i^A \geq t$ will be scheduled in the interval T_l acting as the h-th job in the final schedule.) For each $i = 1, 2, \ldots, n_A$, we define

$$c_{il} = \begin{cases} f_i^A(t), & \text{if } k_i^A \geq h \text{ and } \bar{d}_i^A \geq t, \\ +\infty, & \text{otherwise.} \end{cases}$$

Set $t := t - p^A$, $h := h - 1$, and $l := l - 1$. Do the following:

– If $t > 0$, then go to Step 2.

– If $t = 0$, then go to Step 5.

Step 5: Assign the A-jobs to the n_A time intervals $T_1, T_2, \ldots, T_{n_A}$, each of length p^A, by solving the $n_A \times n_A$ Linear Assignment Problem with costs c_{ij}, $i, j \in \{1, 2, \ldots, n_A\}$. $\square$

Algorithm I for solving problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$ is a modification of Algorithm 3.1, which runs in $O(n_B^2 + n_A^3)$ time, for solving problem $1|p_j^A = p^A|\sum f_i^A : g_{\max}^B \leq Q$ in Zhang and Yuan [17]. The main differences between the two algorithms are the following two points: (i) We incorporate the constraints "$k_j, \bar{d}_j, \text{prec}$" into the implementation of Algorithm 1, and (ii) in Step 3 of Algorithm I, when more than one B-job are available for completing at time t, we always choose an available B-job J_j^B such that $f_j^B(t)$ is as small as possible. The following lemma shows that the time complexity $O(n_B^2 + n_A^3)$ is still valid.

Lemma 4.1. *Algorithm I runs in $O(n_B^2 + n_A^3)$ time.*

Proof. In the preprocessing procedure, the B-jobs are sorted in the topological order under their precedence constraint, the time complexity of which is $O(n_B^2)$. Algorithm I has n iterations in total. In each iteration, Steps 2-4 are implemented. Step 2, with the topological order of the B-jobs in hand, runs in $O(n)$ time. Step 3 runs in $O(n_B)$ time to pick the B-job J_j^B from $U^h(t)$ such that $f_j^B(t)$ is as small as possible. Step 4 runs in $O(n_A)$ time to calculate all the values c_{il}, $i = 1, 2, \ldots, n_A$. Thus, the total running time of the preprocessing procedure and Steps 2-4 in Algorithm I is $O(n_B^2 + n^2)$. Finally, from Lawler [11], the $n_A \times n_A$ Linear Assignment Problem in Step 5 is solvable in $O(n_A^3)$ time. Consequently, the time complexity of Algorithm I is $O(n_B^2 + n^2 + n_A^3) = O(n_B^2 + n_A^3)$. The lemma follows. $\square$

Algorithm 3.1 in Zhang and Yuan [17] returns an optimal schedule for problem $1|p_j^A = p^A|\sum f_i^A : g_{\max}^B \leq Q$. But in general their algorithm cannot generate a nondominated schedule for problem $1|p_j^A = p^A|^{\#}(\sum f_i^A, g_{\max}^B)$. Alternatively, in our Algorithm I, when more than one B-job are available for completing at time t, we always choose an available B-job J_j^B such that $f_j^B(t)$ is as small as possible. This guarantees the following stronger result.

Lemma 4.2. *Let $\mathcal{J} = \{J_1^A, J_2^A, \ldots, J_{n_A}^A,\ J_1^B, J_2^B, \ldots, J_{n_B}^B\}$ and let σ be the schedule of the job set $\mathcal{J}$ returned by Algorithm I. Then σ is optimal for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$ and nondominated for problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$.*

Proof. From Lemma 2.1, there is a schedule π of the jobs in $\mathcal{J}$ that is optimal for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$ and nondominated for problem $1|\beta^*|^\#(\sum f_i^A, g_{\max}^B)$. For convenience, we call such a schedule π a *desired schedule*. Then it suffices to show that σ is also a desired schedule.

Since σ is a feasible schedule for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$, we have

$$\sum f_i^A(\pi) \le \sum f_i^A(\sigma). \tag{3}$$

Suppose that

$$\mathcal{J}^B = \{J_{\sigma(i_1)}, J_{\sigma(i_2)}, \ldots, J_{\sigma(i_{n_B})}\} = \{J_{\pi(j_1)}, J_{\pi(j_2)}, \ldots, J_{\pi(j_{n_B})}\} \tag{4}$$

such that

$$i_1 < i_2 < \cdots < i_{n_B} \text{ and } j_1 < j_2 < \cdots < j_{n_B}. \tag{5}$$

We first prove the following claim.

Claim 1. *There is a desired schedule π such that $i_x = j_x$ and $J_{\sigma(i_x)} = J_{\pi(j_x)}$ for $x = 1, 2, \ldots, n_B$.*

Suppose to the contrary that the result in Claim 1 is not valid. For each desired schedule π, we define

$$\lambda(\sigma, \pi) = \max\{x \in \{1, 2, \ldots, n_B\} : \text{either } i_x \ne j_x \text{ or } J_{\sigma(i_x)} \ne J_{\pi(j_x)}\}. \tag{6}$$

For our purpose, we may choose the desired schedule π such that $\lambda(\sigma, \pi)$ is as small as possible.

Let $z = \lambda(\sigma, \pi)$. The definition of $\lambda(\sigma, \pi)$ in (6) implies that $i_x = j_x$ and $J_{\sigma(i_x)} = J_{\pi(j_x)}$ for $x = z+1, z+2, \ldots, n_B$ but either $i_z \ne j_z$ or $J_{\sigma(i_z)} \ne J_{\pi(j_z)}$. We distinguish the following three cases.

Case 1: $i_z > j_z$. Since the A-jobs have a common processing time p^A, the definition of $\lambda(\sigma, \pi)$ implies that $C_{\sigma(i_z)}(\sigma) = C_{\pi(i_z)}(\pi)$, $J_{\pi(i_z)}$ is an A-job, and $J_{\sigma(i_z)}$ is scheduled before $J_{\pi(i_z)}$ in π. Since σ is a feasible schedule for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$, $J_{\sigma(i_z)} \in \mathcal{J}^B$ is available (subject to the constraints in field β^* and subject to the restriction $g_{\max}^B \le Q$) in position i_z and time $C_{\sigma(i_z)}(\sigma)$. Let π' be a new schedule obtained from π by shifting $J_{\sigma(i_z)}$ to the i_z-th position for completing at time $C_{\sigma(i_z)}(\sigma) = C_{\pi(i_z)}(\pi)$. Then π' is also a feasible schedule for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$. Moreover, we have $C_{\pi(i_z)}(\pi') < C_{\pi(i_z)}(\pi)$ and $C_j(\pi') \le C_j(\pi)$ for all the A-jobs $J_j \in \mathcal{J}^A \setminus \{J_{\pi(i_z)}\}$. Since each function $f_i^A(t)$, $i = 1, 2, \ldots, n_A$, is strictly increasing in $t \ge 0$, we have $\sum f_i^A(\pi') < \sum f_i^A(\pi)$. This contradicts the fact that π is an optimal schedule for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$. Hence, Case 1 does not occur.

Case 2: $i_z < j_z$. Since the A-jobs have a common processing time p^A, the definition of $\lambda(\sigma, \pi)$ implies that $C_{\sigma(j_z)}(\sigma) = C_{\pi(j_z)}(\pi)$, $J_{\sigma(j_z)}$ is an A-job, and $J_{\sigma(i_z)}$ is scheduled before $J_{\sigma(j_z)}$ in σ. Since π is a feasible schedule for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \le Q$, $J_{\pi(j_z)} \in \mathcal{J}^B$ is available (subject to the constraints in field β^* and subject to the restriction $g_{\max}^B \le Q$) in position j_z

and time $C_{\sigma(j_z)}(\sigma)$. Thus, in Step 3 of Algorithm I, some B-job should be scheduled in position j_z for completing at time $C_{\sigma(j_z)}(\sigma)$. This contradicts the assumption that σ is the schedule generated by Algorithm I. Hence, Case 2 does not occur.

Case 3: $i_z = j_z$ and $J_{\sigma(i_z)} \neq J_{\pi(j_z)}$. Since the A-jobs have a common processing time p^A, the definition of $\lambda(\sigma,\pi)$ implies that $C_{\sigma(j_z)}(\sigma) = C_{\pi(j_z)}(\pi)$ and $J_{\sigma(j_z)}$ is scheduled before $J_{\pi(j_z)}$ in π. Since both σ and π are feasible schedules for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$, the two B-jobs $J_{\sigma(j_z)}$ and $J_{\pi(j_z)}$ are both available (subject to the constraints in field β^* and subject to the restriction $g_{\max}^B \leq Q$) in position j_z and time $t := C_{\pi(j_z)}(\pi)$. Assume that $J_{\sigma(j_z)} = J_x^B$ and $J_{\pi(j_z)} = J_y^B$. From the implementation of Step 3 of Algorithm I, we have $g_x^B(t) \leq g_y^B(t) \leq g_{\max}^B(\pi)$. Let π'' be a new schedule obtained from π by shifting $J_x^B = J_{\sigma(j_z)}$ to the j_z-th position for completing at time $t = C_{\pi(j_z)}(\pi)$. Then π'' is also a feasible schedule for problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$. Moreover, we have $C_j(\pi') \leq C_j(\pi)$ for all the A-jobs $J_j \in \mathcal{J}^A$ and all the B-jobs $J_j \in \mathcal{J}^B \setminus \{J_x^B\}$. By noting that $g_x^B(\pi'') = g_x^B(t) \leq g_{\max}^B(\pi)$, we have $\sum f_i^A(\pi'') \leq \sum f_i^A(\pi)$ and $g_{\max}^B(\pi'') \leq g_{\max}^B(\pi)$. From the fact that π is a nondominated schedule for problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$, we have $\sum f_i^A(\pi'') = \sum f_i^A(\pi)$ and $g_{\max}^B(\pi'') = g_{\max}^B(\pi)$. Thus, π'' is also a desired schedule. Since $J_{\pi''(j_z)} = J_x^B = J_{\sigma(j_z)}$, we have $\lambda(\sigma,\pi'') < z = \lambda(\sigma,\pi)$. This contradicts the choice of the desired schedule π. The claim follows.

From Claim 1, we suppose in the following that π is a desired schedule such that $i_x = j_x$ and $J_{\sigma(i_x)} = J_{\pi(j_x)}$ for $x = 1, 2, \ldots, n_B$. Since all the A-jobs have a common processing time p^A and σ is generated by Algorithm I, we have (i) $C_j(\sigma) = C_j(\pi)$ for every B-job $J_j \in \mathcal{J}^B$, and (ii) in both σ and π, the n_A A-jobs are scheduled in the n_A intervals $T_1, T_2, \ldots, T_{n_A}$ (generated in Algorithm I), each of length p^A, respectively. From property (i), we have $g_{\max}^B(\sigma) = g_{\max}^B(\pi)$. Note that in schedule σ, the n_A A-jobs are optimally scheduled in the n_A intervals $T_1, T_2, \ldots, T_{n_A}$ in Step 5 of Algorithm I. Thus, from property (ii), we have $\sum f_i^A(\pi) \geq \sum f_i^A(\sigma)$. Taking (3) into consideration, we conclude that $\sum f_i^A(\sigma) = \sum f_i^A(\pi)$ and $g_{\max}^B(\sigma) = g_{\max}^B(\pi)$. Consequently, σ is a desired schedule. The lemma follows. $\square$

The result in Lemma 4.2 implies that we can solve problem $1|\beta^*|^{\#}(\sum f_i^A, g_{\max}^B)$ by using Algorithm I iteratively in the following way.

Algorithm II: Initially set $Q := +\infty$ and apply Algorithm I to solve problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$ to obtain the first nondominated schedule $\sigma^{(1)}$ and the corresponding nondominated pair $(\sum f_i^A(\sigma^{(1)}), g_{\max}^B(\sigma^{(1)}))$. Generally, if the k-th nondominated schedule $\sigma^{(k)}$ and the corresponding nondominated pair $(\sum f_i^A(\sigma^{(k)}), g_{\max}^B(\sigma^{(k)}))$ have been generated and $g_{\max}^B(\sigma^{(k)}) > G_{\min}$, we reset $Q := g_{\max}^B(\sigma^{(k)}) - 1$ and apply Algorithm I to solve problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$ to obtain the next nondominated schedule $\sigma^{(k+1)}$ and the corresponding nondominated pair $(\sum f_i^A(\sigma^{(k+1)}), g_{\max}^B(\sigma^{(k+1)}))$. Repeat this procedure until we meet an index K such that $g_{\max}^B(\sigma^{(K)}) = G_{\min}$.

The above Algorithm II has K iterations and each iteration executes Algorithm I for solving a corresponding problem $1|\beta^*|\sum f_i^A : g_{\max}^B \leq Q$. To estimate the value K, for a schedule σ, we define $N_A(\sigma) = \sigma[J_1^A] + \sigma[J_2^A] + \cdots + \sigma[J_{n_A}^A]$, which is the sum of the position indices of the A-jobs in schedule σ.

Lemma 4.3. $K \le n_A n_B + 1$.

Proof. Let $\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(K)}$ be the nondominated schedules generated by Algorithm II in the given order. Then we have $g^B_{\max}(\sigma^{(1)}) > g^B_{\max}(\sigma^{(2)}) > \cdots > g^B_{\max}(\sigma^{(K)}) = G_{\min}$.

Claim 2. For each index k with $1 \le k \le K-1$, we have $N_A(\sigma^{(k)}) < N_A(\sigma^{(k+1)})$.

In order to prove Claim 2, it suffices to show that the B-jobs processed before the A-job $J^A_{\sigma_A^{(k)}(i)}$ in schedule $\sigma^{(k)}$ are still processed before the A-job $J^A_{\sigma_A^{(k+1)}(i)}$ in schedule $\sigma^{(k+1)}$ for all $i \in \{1, 2, \ldots, n_A\}$.

Suppose to the contrary that there exists an index $i \in \{1, 2, \ldots, n_A\}$ such that some B-job J^B_x is processed before $J^A_{\sigma_A^{(k)}(i)}$ in schedule $\sigma^{(k)}$ and processed after $J^A_{\sigma_A^{(k+1)}(i)}$ in schedule $\sigma^{(k+1)}$. We may choose the B-job J^B_x such that $C^B_x(\sigma^{(k+1)})$ is as large as possible. Since the A-jobs have a common processing time p^A, we have $\bar{d}^B_x \ge C^B_x(\sigma^{(k+1)}) \ge C^A_{\sigma_A^{(k)}(i)}(\sigma^{(k)})$, $\sigma^{(k)}[J^A_{\sigma_A^{(k)}(i)}] \le \sigma^{(k+1)}[J^B_x] \le k^B_x$, and job J^B_x has no successor (under the constraint prec^B) between J^B_x and $J^A_{\sigma_A^{(k)}(i)}$ in schedule $\sigma^{(k)}$. Since $g^B_{\max}(\sigma^{(k)}) > g^B_{\max}(\sigma^{(k+1)}) \ge g^B_x(\sigma^{(k+1)})$, job J^B_x is available at position $\sigma^{(k)}[J^A_{\sigma_A^{(k)}(i)}]$ for completing at time $C^A_{\sigma_A^{(k)}(i)}(\sigma^{(k)})$ when $\sigma^{(k)}$ is generated by Algorithm I. But this contradicts the implementation of Step 3 of Algorithm I, in which the available B-jobs have the priority to be scheduled in any position. The claim follows.

Since $1 + 2 + \cdots + n_A \le N_A(\sigma) \le n_A n_B + 1 + 2 + \cdots + n_A$ for every feasible schedule σ, $N_A(\sigma)$ has at most $n_A n_B + 1$ possible values. Thus, from Claim 2, we have $K \le n_A n_B + 1$. The lemma follows. $\square$

Now our final result can be stated as follows.

Theorem 4.1. *Algorithm II solves problem $1|\beta^*|^\#(\sum f^A_i, g^B_{\max})$ correctly in $O(n_A n_B^3 + n_A^4 n_B) = O(n^5)$ time.*

Proof. Since each iteration of Algorithm II executes Algorithm I for solving a corresponding problem $1|\beta^*|\sum f^A_i : g^B_{\max} \le Q$, from Lemma 4.2, for each $k = 1, 2, \ldots, K$, the k-th iteration of Algorithm II generates an undominated schedule $\sigma^{(k)}$ and a corresponding undominated pair $(\sum f^A_i(\sigma^{(k)}), g^B_{\max}(\sigma^{(k)}))$. Since $g^B_{\max}$ is integer-valued, there is no undominated pair (f, g) such that $g^B_{\max}(\sigma^{(k)}) - 1 < g < g^B_{\max}(\sigma^{(k)})$, $k = 1, 2, \ldots, K-1$. Then Algorithm II generates all the nondominated pairs of problem $1|\beta^*|^\#(\sum f^A_i, g^B_{\max})$ and their corresponding nondominated schedules. It follows that Algorithm II solves problem $1|\beta^*|^\#(\sum f^A_i, g^B_{\max})$ correctly.

From Lemmas 4.1 and 4.3, each iteration of Algorithm II runs in $O(n_B^2 + n_A^3)$ time and there are totally $K \le n_A n_B + 1$ iterations. Then the time complexity of Algorithm II is given by $O(n_A n_B^3 + n_A^4 n_B) = O(n^5)$. The result follows. $\square$

5 Conclusions

In this paper, we studied the competing-agent Pareto-scheduling problem

$$1|k_j, \bar{d}_j, p_i^A = p^A, \text{prec}^B|^{\#}(\sum f_i^A, g_{\max}^B),$$

where "k_j" denotes the due-index constraint, "$\bar{d}_j$" denotes the deadline constraint, "$p_i^A = p^A$" denotes the common processing time p^A of the A-jobs, and "prec^B" denotes the precedence constraint of the B-jobs. We showed that the problem is polynomially solvable. From the known results in the literature, the research in this paper cannot be extended to a more general two-agent scheduling problem in single-machine setting.

For the further research, we suggest to consider the counterparts of the problem studied in this paper in the serial-batch scheduling environment and the parallel-batch scheduling environment. It is also interesting to present a polynomial algorithm for problem $1|p_i^A = p^A|\lambda_A \sum f_i^A + \lambda_B g_{\max}^B$ with time complexity better than $O(n^5)$, where λ_A and λ_B are two positive numbers.

Acknowledgments

The authors would like to thank the Associate Editor and three anonymous referees for their constructive comments and helpful suggestions. Gao was supported by the China Postdoctoral Science Foundation under grant number 2019M652555 and the National Natural Science Foundation of China under grant number 11901539. Yuan was supported by the National Natural Science Foundation of China under grant numbers 11671368 and 11771406.

References

[1] Agnetis, A., Billaut, J.C., Gawiejnowicz, S., Pacciarelli, D., Soukhal, A.: Multiagent Scheduling: Models and Algorithms. Springer, Berlin (2014)

[2] Agnetis, A., Mirchandani, P.B., Pacciarelli, D., Pacifici, A.: Scheduling problems with two competing agents. Oper. Res. **52**, 229-242 (2004)

[3] Chen, R.B., Yuan, J.J., Lu, L.F.: Single-machine scheduling with positional due indices and positional deadlines, Discrete Optim. DOI: 10.1016/j.disopt.2019.06.002 (2019)

[4] Gao, Y., Yuan, J.J.: Pareto minimizing total completion time and maximum cost with positional due indices. J. Oper. Res. Soc. China **3**, 381-387 (2015)

[5] Gao, Y., Yuan, J.J.: Bi-criteria Pareto-scheduling on a single machine with due indices and precedence constraints. Discrete Optim. **25**, 105-119 (2017)

[6] Geng, Z.C., Yuan, J.J.: Pareto optimization scheduling of family jobs on a p-batch machine to minimize makespan and maximum lateness. Theor. Comput. Sci. **570**, 22-29 (2015)

[7] He, C., Lin, H., Wang, X.M.: Single machine bicriteria scheduling with equal-length jobs to minimize total weighted completion time and maximum cost. 4OR-Q. J. Oper. Res. **12**, 87-93 (2014)

[8] Hoogeveen, H.: Multicriteria scheduling. Eur. J. Oper. Res. **167**, 592-623 (2005)

[9] Hoogeveen, J.A., van de Velde, S.L.: Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. Oper. Res. Lett. **17**, 205-208 (1995)

[10] Hoogeveen, J.A.: Single machine scheduling to minimize a function of two or three maximum cost criteria. J. Algorithms **21**, 415-433 (1996)

[11] Lawler, E.L.: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York (1976)

[12] Lawler, E.L.: A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. Ann. Discrete Math. **1**, 331-342 (1977)

[13] Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity results for scheduling chains on a single machine. Eur. J. Oper. Res. **4**, 270-275 (1980)

[14] Oron, D., Shabtay, D., Steiner, G.: Single machine scheduling with two competing agents and equal job processing times. Eur. J. Oper. Res. **244**, 86-99 (2015)

[15] T'kindt, V., Billaut, J.C.: Multicriteria scheduling: Theory, Models and Algorithms (2nd ed.). Springer, Berlin (2006)

[16] Wan, L., Yuan, J.J., Wei, L.J.: Pareto optimization scheduling with two competing agents to minimize the number of tardy jobs and the maximum cost. Appl. Math. Comput. **273**, 912-923 (2016)

[17] Zhang, Y., Yuan, J.J.: A note on a two-agent scheduling problem related to the total weighted late work. J. Comb. Optim. **37**, 989-999 (2019)

[18] Zhao, Q.L., Yuan, J.J.: Rescheduling to minimize the maximum lateness under the sequence disruptions of original jobs. Asia Pac. J. Oper. Res. 34(05):1750024, DOI: 10.1142/S0217595917500245 (2017)