mathematics

MDPI

Article

Two-Agent Pareto-Scheduling of Minimizing Total Weighted Completion Time and Total Weighted Late Work

Yuan Zhang †, Zhichao Geng *,† and Jinjiang Yuan †

School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, China; zy2020@gs.zzu.edu.cn (Y.Z.); yuanjj@zzu.edu.cn (J.Y.)

* Correspondence: zcgeng@zzu.edu.cn

† These authors contributed equally to this work.

Received: 26 October 2020; Accepted: 16 November 2020; Published: 20 November 2020

Abstract: We investigate the Pareto-scheduling problem with two competing agents on a single machine to minimize the total weighted completion time of agent A's jobs and the total weighted late work of agent B's jobs, the B-jobs having a common due date. Since this problem is known to be NP-hard, we present two pseudo-polynomial-time exact algorithms to generate the Pareto frontier and an approximation algorithm to generate a $(1+\epsilon)$-approximate Pareto frontier. In addition, some numerical tests are undertaken to evaluate the effectiveness of our algorithms.

Keywords: scheduling; two agents; pareto frontier; approximation algorithms

1. Introduction

Problem description and motivation: Multi-agent scheduling has attracted an ever-increasing research interest due to its extensive applications (see the book of Agnetis et al. [1]). Among the common four problem-versions (including lexical-, positive-combination-, constrained-, and Pareto-scheduling, as shown in Li and Yuan [2]) for a given group of criteria for multiple agents, Pareto-scheduling has the most important practical value, since it reflects the effective tradeoff between the actual and (usually) conflicting requirements of different agents.

Our considered problem is formally stated as follows. Assume that two agents (A and B) compete to process their own sets of independent and non-preemptive jobs on a single machine. The set of the n_X jobs from agent $X \in \{A, B\}$ is $\mathcal{J}_X = \{J_1^X, J_2^X, \cdots, J_{n_X}^X\}$ with $\mathcal{J}_A \cap \mathcal{J}_B = \phi$. For convenience, we call a job from agent X an X-job. All jobs are available at time zero, and are scheduled consecutively without idle time due to the regularity of the objective functions as shown later. Each job J_j^X has a processing time p_j^X and a weight w_j^X. In addition, each B-job J_j^B has also a common due date d. We assume that all parameters p_j^X, w_j^X and d are known integers.

Let σ be a schedule. We use $C_j^X(\sigma)$ to denote the completion time of job J_j^X in σ. The objective function of agent A is the total weighted completion time, denoted by $\sum w_j^A C_j^A(\sigma)$, while the objective function

of agent B is the total weighted late work, denoted by $\sum w_j^B Y_j^B(\sigma)$. Here, the late work $Y_j^B(\sigma)$ of job J_j^B indicates the amount processed after the due date d, specifically,

$$Y_j^B(\sigma) = \begin{cases} 0, & \text{if } C_j(\sigma) \leq d, \\ C_j^B(\sigma) - d, & \text{if } d < C_j^B(\sigma) \leq d + p_j^B, \\ p_j^B, & \text{if } C_j^B(\sigma) > d + p_j^B. \end{cases} \quad (1)$$

Following Hariri et al. (1995) [3], job J_j^B is said to be *early*, *partially early*, and *late* in σ, if $Y_j^B(\sigma) = 0$, $0 < Y_j^B(\sigma) < p_j^B$, and $Y_j^B(\sigma) = p_j^B$, respectively.

Falling into the category of Pareto-scheduling, the problem studied in this paper aims at generating all Pareto-optimal points (PoPs) and the corresponding Pareto-optimal schedules (PoSs) (the definitions of PoPs and PoSs will be given in Section 2) of all jobs with regard to $\sum w_j^A C_j^A$ and $\sum w_j^B Y_j^B$). Using the notations in Agnetis et al. [1], our studied scheduling problem can be denoted by $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$. For this problem, we will devise some efficient approximate algorithms.

Our considered scheduling model arises from many practical scenarios. For example, in a factory, two'concurrent projects (A and B), each containing a certain amount of activities with distinct importance, have to share a limited resource. The former focuses on the mean completion time of its activities. In contrast, the latter requires its activities to be completed before the due date as much as possible, since, otherwise, the shortcomings of some key technical forces after the due date will occur and result in irretrievable loss. It is necessary to model the goal of project B as the weighted late work, that is, minimizing the parts left unprocessed before the due date. In addition, two projects naturally have to negotiate to seek a trade-off method of utilizing the common resource.

For another example, in a distribution center, two categories (A- and B-) goods are stored in a warehouse, in which the former comprises common goods and the latter comprises fresh goods with a shelf life. It is hoped that the shipping preparations for the A-goods will be completed as soon as possible. However, due to their limited shelf life, if they are transported after a certain time, the B-goods will not be fresh enough when they reach the customers. Therefore, it is reasonable to respectively model the goals of A-goods and B-goods by minimizing the total weighted completion time and the total weighted late work, and seek an efficient transportation method.

Related works and our contribution: Numerous works have addressed multi-agent scheduling problems in the literature. With the aim of this paper, we only summarize briefly some related results. Wan et al. [4] provided a strongly polynomial-time algorithm for the two-agent Pareto-scheduling problem on a single machine to minimize the number of the tardy A-jobs and the maximum cost of the B-jobs. Later, Wan et al. [5] investigated two Pareto-scheduling problems on a single machine with two competing agents and a linear-deterioration processing time: $1||^{\#}(E_{\max}^A, E_{\max}^B)$ and $1||^{\#}(\sum E_j^A, E_{\max}^B)$, where $\sum E_j^A$ is the total earliness of the A-jobs and $E_{\max}^X$ is the maximum earliness of the X-jobs. For these two problems, they respectively proposed a polynomial-time algorithm. Gao and Yuan [6] showed that the following two Pareto-scheduling problems with a positional due index and precedence constraints are both polynomially solvable: $1||^{\#}(\sum C_j^A, f_{\max}^B)$ and $1||^{\#}(f_{\max}^A, f_{\max}^B)$, where $f_{\max}^X$ indicates the maximum cost of the X-jobs. He et al. [7] extensively considered the versions of the problems in Gao and Yuan [6] with deteriorating or shortening processing times and without positional due indices and precedence constraints, and devised polynomial-time algorithms. Yuan et al. [8] showed the single-machine preemptive problem $1|r_j, pmtn|^{\#}(L_{\max}^a : L_{\max}^b)$ can be solved in a polynomial time, where $L_{\max}^X$ indicates the maximum lateness of the X-jobs. Wan [9] investigated the single-machine two-agent scheduling problem to minimize the maximum costs with position-dependent jobs, and developed a polynomial-time algorithm.

While most results on Pareto-scheduling concentrate on devising exact algorithms to obtain the Pareto frontier, there are also some methods (such as [10–14]) of developing approximate algorithms to generate the approximate Pareto frontier. Dabia et al. [10] adopted the trimming technique to derive the approximate Pareto frontier for some multi-objective scheduling problems. Yin et al. [15] considered two just-in-time (JIT) scheduling problems with two competing agents on unrelated parallel machines, in which the one agent's criterion is to maximize the weighted number of its JIT jobs, and another agent's criterion is either to maximize its maximum gains from its JIT jobs or to maximize the weighted number of its JIT jobs. They showed that the two problems are both unary NP-hard when the machine number is not fixed, and proposed either a polynomial-time algorithm or a fully polynomial-time approximation scheme (FPTAS) when the machine number is a constant. Yin et al. [16] also considered similar problems in the setting of a two-machine flow shop, and provided two pseudo-polynomial-time exact algorithms to find the Pareto frontier. Chen et al. [17] studied a multi-agent Pareto-scheduling problem in a no-wait flow shop setting, in which each agent's criterion is to maximize its own weighted number of JIT jobs. They showed that it is unary NP-hard when the number of agents is arbitrary, and presented pseudo-polynomial time algorithms and an $(1, 1-\epsilon, \ldots, 1-\epsilon)$-approximation algorithm when the number of agents is fixed.

From the perspective of methodology, as a type of optimization problem, the multi-agent scheduling problem's solution algorithms potentially allow for exploiting the optimal robot path planning by a gravitational search algorithm (Purcaru et al. [18]) and optimization based on phylogram analysis (Soares et al. [19]).

In the prophase work (Zhang and Yuan [20]), we proved that the constrained scheduling problem of minimizing the total late work of agent A's jobs with equal due dates subject to the makespan of agent B's jobs not exceeding a given upper bound, is NP-hard even if agent B has only one job. It implies the NP-hardness of our considered problem in this paper. Thus we limit the investigation to devising pseudo-polynomial-time exact algorithms and an approximation algorithm to generate the approximate Pareto frontier.

In addition, in our recent work (Zhang et al. [21]), we considered several three-agent scheduling problems under different constraints on a single machine, in which the three agents' criteria are to minimize the total weighted completion time, the weighted number of tardy jobs, and the total weighted late work. Among those problems, there are two questions related to this paper: $1|p_j^A \uparrow\downarrow w_j^A|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$, which is solved in $O(n_A n_B^2 U^A U^B)$, and $1|p_j^A \uparrow\downarrow w_j^A, d_j^B \uparrow\downarrow w_j^B|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$, which is solved in $O(n_A n_B U^A U^B)$. The notation $p_j^A \uparrow\downarrow w_j^A$ represents that the jobs of the first agent have inversely agreeable processing times and weights, i.e., the smaller the processing time for a job, the greater its weight, and the notation $d_j^B \uparrow\downarrow w_j^B$ represents that the jobs of agent B have inversely agreeable due dates and weights. U^A and U^B are the upper bounds on the criteria $\Sigma w_j^A C_j^A$ and $\Sigma w_j^B Y_j^B$, respectively. In contrast to Zhang et al. [21], in this article we remove the constraint $p_j^A \uparrow\downarrow w_j^A$ and turn to the optimization problem of B-jobs having a common due date.

The remainder of the paper is organized as follows. In Section 2, some preliminaries are provided. In Sections 3 and 4, we present two dynamic programming algorithms and an FPTAS. In Section 5, some numeral tests are undertaken to show the algorithms' efficiency. Section 6 concludes the paper and suggests the future research direction.

2. Preliminaries

For self-consistency, in this section we describe some notions and properties related to Pareto-scheduling, and we present other useful notations in the description of the algorithms in the following sections.

Definition 1. *Consider two m-vectors* $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)$ *and* $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$.

(i) We say that $\boldsymbol{u}$ ***dominates*** $\boldsymbol{v}$*, denoted by* $\boldsymbol{u} \preceq \boldsymbol{v}$*, if* $u_i \leq v_i$ *for* $i = 1, 2, \ldots, m$*.*
(ii) We say that $\boldsymbol{u}$ ***strictly dominates*** $\boldsymbol{v}$*, denoted by* $\boldsymbol{u} \prec \boldsymbol{v}$*, if* $\boldsymbol{u} \preceq \boldsymbol{v}$ *and* $\boldsymbol{u} \neq \boldsymbol{v}$*.*
(iii) Given a constant $\epsilon > 0$*, we say that* $\boldsymbol{u}$ ***ϵ-dominates*** $\boldsymbol{v}$*, denoted by* $\boldsymbol{u} \preceq_\epsilon \boldsymbol{v}$*, if and only if* $u_i \leq (1+\epsilon)v_i$ *for* $i = 1, 2, \ldots, m$*.*

Definition 2. *Given two agents' criteria* $\gamma^A(\sigma)$ *and* $\gamma^B(\sigma)$*, a feasible schedule* σ *is called* ***Pareto-optimal*** *and the corresponding objective vector* $(\gamma^A(\sigma), \gamma^B(\sigma))$ *is called a* ***Pareto-optimal point****, if no other feasible schedule* π *satisfies* $(\gamma^A(\pi), \gamma^B(\pi)) \prec (\gamma^A(\sigma), \gamma^B(\sigma))$*. All the Pareto-optimal points form the* ***Pareto frontier****, denoted by* $\boldsymbol{P}$*.*

Let $\mathbf{R}$ be the set of the objective vectors of all feasible schedules, and $\mathbf{Q}$ be a subset of $\mathbf{R}$.

Definition 3. *A vector* $\boldsymbol{u} \in \boldsymbol{Q}$ *is called* ***non-dominated*** *in* $\boldsymbol{Q}$*, if there exists no other vector* $\boldsymbol{v} \in \boldsymbol{Q}$ *such that* $\boldsymbol{v} \prec \boldsymbol{u}$*.*

It is not difficult to see that, for the above definitions, the latter is an extension of the former, and especially when $\mathbf{Q}$ is exactly equal to $\mathbf{R}$, all the non-dominated vectors in $\mathbf{Q}$ compose the Pareto-optimal frontier. The following lemma establishes the relationship between sets $\mathbf{P}$ and a subset $\mathbf{Q} \subseteq \mathbf{R}$.

Lemma 1. *For any set* $\boldsymbol{Q}$ *with* $\boldsymbol{P} \subseteq \boldsymbol{Q} \subseteq \boldsymbol{R}$*, if* $\boldsymbol{O}$ *is the set including all the non-dominated vectors in* $\boldsymbol{Q}$*, then* $\boldsymbol{O} = \boldsymbol{P}$*.*

Proof. By Definition 2, for each Pareto-optimal point $\mathbf{u} \in \mathbf{P}$, there is no other vector $\mathbf{v} \in \mathbf{R}$ such that $\mathbf{v} \prec \mathbf{u}$, and naturally, such a fact also holds for the set $\mathbf{Q}$, since $\mathbf{Q} \subseteq \mathbf{R}$. Then, it follows that $\mathbf{P} \subseteq \mathbf{O}$ by the definition of the set $\mathbf{O}$. Next we show that $\mathbf{O} \subseteq \mathbf{P}$. If not, we pick up one vector $\mathbf{w}$ from $\mathbf{O} \setminus \mathbf{P}$. Again by Definition 2, there is some vector $\mathbf{w} \in \mathbf{P}$ such that $\mathbf{w} \prec \mathbf{u}$. Nevertheless, this is impossible, since $\mathbf{w} \in \mathbf{P} \subseteq \mathbf{Q}$ leads to no existence of such a vector $\mathbf{w}$ in $\mathbf{Q}$ by the assumption of $\mathbf{w}$ and Definition 3. Thus $\mathbf{O} = \mathbf{P}$. □

From Lemma 1, to generate the Pareto frontier $\mathbf{P}$, an alternative is to first determine a set $\mathbf{Q}$ with $\mathbf{P} \subseteq \mathbf{Q} \subseteq \mathbf{R}$, and then delete the dominated vectors in $\mathbf{Q}$. Throughout the reminder of this paper, such a subset $\mathbf{Q}$ is called an ***intermediate set***. Obviously, $\mathbf{R}$ is also an intermediate set.

Definition 4. *For a given constant* $\epsilon > 0$*, a* $(1+\epsilon)$***-approximate Pareto frontier****, denoted by* $\boldsymbol{P}_\epsilon$*, is a set of the objective vectors satisfying, for any* $(\gamma^A(\sigma), \gamma^B(\sigma)) \in \boldsymbol{P}$*, there exists at least one objective vector* $(\gamma^A(\sigma'), \gamma^B(\sigma')) \in \boldsymbol{P}_\epsilon$ *such that* $(\gamma^A(\sigma'), \gamma^B(\sigma')) \preceq_\epsilon (\gamma^A(\sigma), \gamma^B(\sigma))$*.*

Definition 5. *A family of algorithms* $\{\mathcal{A}_\epsilon : \epsilon > 0\}$ *is called a* **fully polynomial-time approximation scheme *(FPTAS)*** *if, for each* $\epsilon > 0$*,* $\mathcal{A}_\epsilon$ *generates a* $(1+\epsilon)$*-approximate Pareto frontier with a running time in the polynomial in the instance size and* $1/\epsilon$*.*

Besides those already mentioned in Section 1, the following notations will also be used later:

- $\mathcal{J}_j^X$: indicates the set of the first j jobs in $\mathcal{J}^X$, namely, $\mathcal{J}_j^X = \{J_1^X, J_2^X, \cdots, J_j^X\}$.
- $J_i^X \prec_\sigma J_j^{X'}$ indicates that job J_i^X immediately precedes $J_j^{X'}$ in schedule σ, where $X, X' \in \{A, B\}$.
- $s_j^X(\sigma)$ indicates the starting time of job J_j^X in σ.
- $P_{\text{sum}}^X = \sum_{j=1}^{n_X} p_j^X$ indicates the total processing time of all X-jobs.
- P_{sum} indicates the total processing time of all jobs, and $P_{\text{sum}} = P_{\text{sum}}^A + P_{\text{sum}}^B$.
- $W_{\text{sum}}^X = \sum_{j=1}^{n_X} w_j^X$ indicates the total weight of all X-jobs.

- W_{sum} indicates the total weight of all jobs, and $W_{\text{sum}} = W^A_{\text{sum}} + W^B_{\text{sum}}$.
- $p^X_{\max}$ indicates the maximum processing time of the X-jobs, namely, $p^X_{\max} = \max\{p^X_j : 1 \le j \le n_X\}$.
- $w^X_{\max}$ indicates the maximum weight of the X-jobs, namely, $w^X_{\max} = \max\{w^X_j : 1 \le j \le n_X\}$.

3. An Exact Algorithm

In this section a dynamic programming algorithm for problem $1|d^B_j = d|^{\#}(\sum w^A_j C^A_j, \sum w^B_j Y^B_j)$ is presented. For description convenience, for a given schedule σ, the job set $\mathcal{J}$ is divided into the following four subsets: $\mathcal{J}^{A_1}(\sigma) = \{J^A_j : C^A_j(\sigma) \le d\}$, $\mathcal{J}^{A_2}(\sigma) = \{J^A_j : C^A_j(\sigma) > d\}$, $\mathcal{J}^{B_1}(\sigma) = \{J^B_j : s^B_j(\sigma) < d\}$, and $\mathcal{J}^{B_2}(\sigma) = \{J^B_j : s^B_j(\sigma) \ge d\}$. Obviously, such a partition of the job set is well defined for a given schedule.

The following lemma establishes the structural properties of the Pareto-optimal schedule.

Lemma 2. *For each Pareto-optimal point (C, Y) of problem $1|d^B_j = d|^{\#}(\sum w^A_j C^A_j, \sum w^B_j Y^B_j)$, there is a Pareto-optimal schedule σ such that*
(i) $\mathcal{J}^{A_1}(\sigma) \prec_\sigma \mathcal{J}^{B_1}(\sigma) \prec_\sigma \mathcal{J}^{A_2}(\sigma) \prec_\sigma \mathcal{J}^{B_2}(\sigma)$.
(ii) the jobs in $\mathcal{J}^{B_1}(\sigma)$ are sequenced in the non-increasing order of their weights and the jobs in $\mathcal{J}^{B_2}(\sigma)$ are sequenced arbitrarily.
(iii) the jobs in $\mathcal{J}^{A_1}(\sigma)$ and $\mathcal{J}^{A_2}(\sigma)$ are sequenced according to the weighted shortest processing time (WSPT) rule.

Proof. In Lemma 2, statement (i) can easily be observed, since the jobs in $\mathcal{J}^{B_2}(\sigma)$ are late and this will not result in any increase in their total late work when moving them to the end of the schedule, and as many A-jobs as possible can be positioned before the B-jobs in $\mathcal{J}^{B_1}(\sigma)$, provided that the last job in $\mathcal{J}^{B_1}(\sigma)$ is not late. The left two statements in Lemma 2 can easily be proved by an interchange argument and the detail is omitted here. □

Lemma 2 allows us only to consider the feasible schedules simultaneously satisfying the conditions (i)-(iii). To this end, we re-number the n_A jobs in $\mathcal{J}^A$ in the WSPT order and the n_B B-jobs in the maximum weight first (MW) order so that

$$\frac{p^A_1}{w^A_1} \le \frac{p^A_2}{w^A_2} \le \cdots \le \frac{p^A_{n_A}}{w^A_{n_A}}. \tag{2}$$

$$w^B_1 \ge w^B_2 \ge \cdots \ge w^B_{n_B}. \tag{3}$$

Such a sorting takes $O(n \log n)$ time.

According to Lemma 1, the algorithm to be described adopts the strategy of first finding the intermediate set dynamically and then deleting the dominated points in it. It is necessary to mention that in the proposed algorithm we appropriately relax the conditions to find a modestly larger intermediate set. For briefly describing the dynamic programming algorithm, we introduce the following terminologies and notations.

- an $ABAB$**-schedule** is defined to be a schedule π for $\mathcal{I} \subseteq \mathcal{J}$ satisfying (i) $\pi = \pi_1 \prec \pi_2 \prec \pi_3 \prec \pi_4$, where among the four mutually disjointed subschedules $\pi_1, \pi_2, \pi_3, and \pi_4$, the A-jobs are included in π_1 and π_3, and the B-jobs are included in π_2 and π_4; (ii) no idle time exists between the jobs in each subschedule, but this is not necessarily so between two subschedules. Moreover, the idle time between π_3 and π_4 is supposed to be long enough; (iii) the jobs in each subschedule are sequenced in the increasing order of their indices.
- an $(\overrightarrow{x}, \overleftarrow{y})$**-schedule** is defined to be an $ABAB$-schedule π for $\mathcal{J}^A_x \cup (\mathcal{J}^B \setminus \mathcal{J}^B_{y-1})$ with no idle time existing between subschedules π_2 and π_3, where $x \in \{1, 2, \ldots, n_A\}$ and $y \in \{1, 2, \ldots, n_B\}$.

- a vector (t_1, t_2, t_3, C, Y) is introduced to denote a **state of** $(\overrightarrow{x}, \overleftarrow{y})$, in which t_1, t_2, t_3, C, and Y, respectively, stand for the end point of π_1, the start point of π_2, the end point of π_3, the total weighted completion time of the A-jobs of $\mathcal{J}_x^A$, and the total weighted late work of the B-jobs of $\mathcal{J}^B \setminus \{\mathcal{J}_{y-1}^B\}$. Note that a state of $(\overrightarrow{x}, \overleftarrow{y})$ at least corresponds to some $(\overrightarrow{x}, \overleftarrow{y})$-schedule.
- $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ denotes the set of all the states of $(\overrightarrow{x}, \overleftarrow{y})$.
- $\tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y})$ denotes the set obtained from $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ by deleting the vectors (t_1, t_2, t_3, C, Y), for which there is another vector $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y})$ with $t_1 \leq \overline{t_1}, t_2 \geq \overline{t_2}, t_3 \leq \overline{t_3}, C \leq \overline{C}$, and $Y \leq \overline{Y}$.
- Let $\mathbf{Q}_1 = \{(C, Y) : (t_1, t_2, t_3, C, Y) \in \tilde{\Gamma}(\overrightarrow{n_A}, \overleftarrow{1})\}$, and let $\tilde{\mathbf{Q}}_1$ be the set of the non-dominated vectors in $\mathbf{Q}_1$.

To solve problem $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$, we have to first compute $\tilde{\Gamma}(\overrightarrow{n_A}, \overleftarrow{1})$ and then obtain the Pareto-frontier $\tilde{\mathbf{Q}}_1$. This can be realized by dynamically computing the sets $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ for all the possible choices of the tuple (x, y). Note that each $(\overrightarrow{x}, \overleftarrow{y})$-schedule can be obtained either by adding job J_x^A to some $(\overrightarrow{x-1}, \overleftarrow{y})$-schedule, or by adding job J_y^B to some $(\overrightarrow{x}, \overleftarrow{y+1})$-schedule. Therefore, we can informally describe our dynamic programming algorithm as follows.

Initially, set $\Gamma(\overrightarrow{0}, \overleftarrow{n_B+1}) = \{(0, t_0, t_0, 0, 0) : d - p_{\max}^A + 1 \leq t_0 \leq d + p_{\max}^B - 1\}$ and $\Gamma(\overrightarrow{x}, \overleftarrow{y}) = \emptyset$ if $(x, y) \neq (0, n_B + 1)$. Then we recursively generate all the state sets $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ from the previously-generated sets $\Gamma(\overrightarrow{x-1}, \overleftarrow{y})$ and $\Gamma(\overrightarrow{x}, \overleftarrow{y+1})$. Specifically,

- For each state $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x-1}, \overleftarrow{y})$ with $\Gamma(\overrightarrow{x-1}, \overleftarrow{y}) \neq \phi$, add two states $(t_1', t_2', t_3', C', Y')$ and $(t_1'', t_2'', t_3'', C'', Y'')$ to the set $\Gamma(\overrightarrow{x}, \overleftarrow{y})$, with

$$(t_1', t_2', t_3', C', Y') = (t_1 + p_x^A, t_2, t_3, C + w_x^A(t_1 + p_x^A), Y),$$

and

$$(t_1'', t_2'', t_3'', C'', Y'') = (t_1, t_2, t_3 + p_x^A, C + w_x^A(t_3 + p_x^A), Y).$$

These two states respectively correspond to the newly obtained $(\overrightarrow{x}, \overleftarrow{y})$-schedules by scheduling job J_x^A immediately following the subschedule π_1 and immediately following the subschedule π_3, in some $(\overrightarrow{x-1}, \overleftarrow{y})$ schedule π that corresponds to the state (t_1, t_2, t_3, C, Y). Note that the first case occurs only when $t_1 + p_x^A \leq t_2$ is satisfied.

- For each state $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x}, \overleftarrow{y+1})$, also add two two states $(t_1', t_2', t_3', C', Y')$ and $(t_1'', t_2'', t_3'', C'', Y'')$ to the set $\Gamma(\overrightarrow{x}, \overleftarrow{y})$, with

$$(t_1', t_2', t_3', C', Y') = (t_1, t_2 - p_y^B, t_3, C, Y + w_y^B \max\{t_2 - d^B, 0\}),$$

and

$$(t_1'', t_2'', t_3'', C'', Y'') = (t_1, t_2, t_3, C, Y + w_y^B p_y^B).$$

These two states respectively correspond to the newly obtained $(\overrightarrow{x}, \overleftarrow{y})$-schedules by scheduling job J_y^B immediately preceding the subschedule π_2 and immediately following the subschedule π_4, in some $(\overrightarrow{x}, \overleftarrow{y+1})$ schedule π that corresponds to the state (t_1, t_2, t_3, C, Y). Note that the first case occurs only when $t_1 \leq t_2 - p_y^B < d^B$ is satisfied.

Note that, if in the above state-generation procedures we replace sets $\Gamma(\overrightarrow{x-1}, \overleftarrow{y})$ and $\Gamma(\overrightarrow{x}, \overleftarrow{y+1})$ with sets $\tilde{\Gamma}(\overrightarrow{x-1}, \overleftarrow{y})$ and $\tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y+1})$, then the resulting set of new states, denoted by $\Gamma'(\overrightarrow{x}, \overleftarrow{y})$, may be different from $\Gamma(\overrightarrow{x}, \overleftarrow{y})$. Recall that, when deleting those dominated vectors in the sets $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ and $\Gamma'(\overrightarrow{x}, \overleftarrow{y})$, the newly obtained sets are respectively denoted by $\tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y})$ and $\tilde{\Gamma}'(\overrightarrow{x}, \overleftarrow{y})$, which will be shown to be identical in the following lemma.

Lemma 3. $\tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y}) = \tilde{\Gamma}'(\overrightarrow{x}, \overleftarrow{y})$.

Proof. Since $\tilde{\Gamma}(\overrightarrow{x-1}, \overleftarrow{y}) \subseteq \Gamma(\overrightarrow{x-1}, \overleftarrow{y})$ and $\tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y+1}) \subseteq \Gamma(\overrightarrow{x}, \overleftarrow{y+1})$, it follows that $\Gamma'(\overrightarrow{x}, \overleftarrow{y}) \subseteq \Gamma(\overrightarrow{x}, \overleftarrow{y})$ by the generation procedure of the new states as described previously. If $\Gamma(\overrightarrow{x}, \overleftarrow{y}) = \Gamma'(\overrightarrow{x}, \overleftarrow{y})$, then naturally $\tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y}) = \tilde{\Gamma}'(\overrightarrow{x}, \overleftarrow{y})$. In the following, suppose that $\Gamma(\overrightarrow{x}, \overleftarrow{y}) \setminus \Gamma'(\overrightarrow{x}, \overleftarrow{y}) \neq \emptyset$. We next show that each state $(t_1', t_2', t_3', C', Y') \in \Gamma(\overrightarrow{x}, \overleftarrow{y}) \setminus \Gamma'(\overrightarrow{x}, \overleftarrow{y})$ is dominated by a state $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y}) \in \Gamma'(\overrightarrow{x}, \overleftarrow{y})$, namely, $\overline{t_1} \leq t_1', \overline{t_2} \geq t_2', \overline{t_3} \leq t_3', \overline{C} \leq C', \overline{Y} \leq Y'$.

Let π' be an $(\overrightarrow{x}, \overleftarrow{y})$-schedule corresponding to $(t_1', t_2', t_3', C', Y')$. According to the above discussion, there are four possibilities of deriving π' from some schedule π, which is assumed to correspond to the state (t_1, t_2, t_3, C, Y) in $\Gamma(\overrightarrow{x-1}, \overleftarrow{y})$ or $\Gamma(\overrightarrow{x}, \overleftarrow{y+1})$.

Case 1. π' is obtained from π by scheduling job J_x^A directly after subschedule π_1. Then $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x-1}, \overleftarrow{y})$ with $t_1 + p_x^A \leq t_2$, and there is a state $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y}) \in \tilde{\Gamma}(\overrightarrow{x-1}, \overleftarrow{y})$ such that $\tilde{t_1} \leq t_1, \tilde{t_2} \geq t_2, \tilde{t_3} \leq t_3, \tilde{C} \leq C$, and $\tilde{Y} \leq Y$. Let $\tilde{\pi}$ be an $(\overrightarrow{x-1}, \overleftarrow{y})$-schedule corresponding to $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y})$, and let $\overline{\pi}$ be the $(\overrightarrow{x}, \overleftarrow{y})$-schedule obtained from $\tilde{\pi}$ by scheduling J_x^A directly after schedule $\tilde{\pi}_1$. Let $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y})$ be the state corresponding to $\overline{\pi}$. Note that the above operation to get $\overline{\pi}$ is feasible since $\tilde{t_1} + p_x^A \leq t_1 + p_x^A \leq t_2 \leq \tilde{t_2}$. Then we have $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y}) = (\tilde{t_1} + p_x^A, \tilde{t_2}, \tilde{t_3}, \tilde{C} + w_x^A(\tilde{t_1} + p_x^A), \tilde{Y})$. Combining with the fact that $(t_1', t_2', t_3', C', Y') = (t_1 + p_x^A, t_2, t_3, C + w_x^A(t_1 + p_x^A), Y)$, we have

$$\begin{cases} \overline{t_1} = \tilde{t_1} + p_x^A \leq t_1 + p_x^A = t_1', \\ \overline{t_2} = \tilde{t_2} \geq t_2 = t_2', \\ \overline{t_3} = \tilde{t_3} \leq t_3 = t_3', \\ \overline{C} = \tilde{C} + w_x^A(\tilde{t_1} + p_x^A) \leq C + w_x^A(t_1 + p_x^A) = C', \\ \overline{Y} = \tilde{Y} \leq Y = Y'. \end{cases} \tag{4}$$

Case 2. π' is obtained from π by scheduling J_x^A directly after schedule π_3. Then $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x-1}, \overleftarrow{y})$, and there is a state $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y}) \in \tilde{\Gamma}(\overrightarrow{x-1}, \overleftarrow{y})$ such that $\tilde{t_1} \leq t_1, \tilde{t_2} \geq t_2, \tilde{t_3} \leq t_3, \tilde{C} \leq C$, and $\tilde{Y} \leq Y$. Let $\tilde{\pi}$ be an $(\overrightarrow{x-1}, \overleftarrow{y})$-schedule corresponding to $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y})$, and let $\overline{\pi}$ be the $(\overrightarrow{x}, \overleftarrow{y})$-schedule obtained from $\tilde{\pi}$ by scheduling J_x^A directly after schedule $\tilde{\pi}_3$. Let $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y})$ be the state corresponding to $\overline{\pi}$. Then we have $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y}) = (\tilde{t_1}, \tilde{t_2}, \tilde{t_3} + p_x^A, \tilde{C} + w_x^A(\tilde{t_3} + p_x^A), \tilde{Y})$. Combining with the fact that $(t_1', t_2', t_3', C', Y') = (t_1, t_2, t_3 + p_x^A, C + w_x^A(t_3 + p_x^A), Y)$, we have

$$\begin{cases} \overline{t_1} = \tilde{t_1} \leq t_1 = t_1', \\ \overline{t_2} = \tilde{t_2} \geq t_2 = t_2', \\ \overline{t_3} = \tilde{t_3} + p_x^A \leq t_3 + p_x^A = t_3', \\ \overline{C} = \tilde{C} + w_x^A(\tilde{t_3} + p_x^A) \leq C + w_x^A(t_3 + p_x^A) = C', \\ \overline{Y} = \tilde{Y} \leq Y = Y'. \end{cases} \tag{5}$$

Case 3. π' is obtained from π by scheduling J_y^B directly before schedule π_2. Note that in this case, the condition $t_1 \leq t_2 - p_y^B < d^B$ must be satisfied. Then $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x}, \overleftarrow{y+1})$, and there is a state $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y}) \in \tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y+1})$ such that $\tilde{t_1} \leq t_1, \tilde{t_2} \geq t_2, \tilde{t_3} \leq t_3, \tilde{C} \leq C$, and $\tilde{Y} \leq Y$. Let $\tilde{\pi}$ be an $(\overrightarrow{x}, \overleftarrow{y+1})$-schedule corresponding to $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y})$, and let $\overline{\pi}$ be the $(\overrightarrow{x}, \overleftarrow{y})$-schedule obtained from $\tilde{\pi}$ by scheduling J_y^B directly before schedule π_2. Let $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y})$ be the state corresponding to $\overline{\pi}$. The above

operation to obtain $\overline{\pi}$ is feasible. In fact, $\tilde{t_1} \leq t_1 \leq t_2 - p_y^B$, which means there are enough spaces for J_y^B to be scheduled in. In the following we will illustrate that the condition $\tilde{t_2} - p_y^B < d$ is satisfied.

Claim 1. *If $\tilde{t_2} \neq t_2$, then $\tilde{t_2} \leq d$.*

Suppose to the contrary that $\tilde{t_2} > d$, then J_y^B is partially early or late in $\overline{\pi}$, implying that $J_{y+1}^B, J_{y+2}^B, \ldots, J_{n_B}^B$ are all late in $\tilde{\pi}$, i.e., there is no job in $\tilde{\pi}_2$, which further suggests that $\sum_{j=1}^{x} p_j^A = \tilde{t_1} + \tilde{t_3} - \tilde{t_2}$. What is more, since $\tilde{Y} \leq Y$, the jobs $J_{y+1}^B, J_{y+2}^B, \ldots, J_{n_B}^B$ are also late in π, which also indicates that $\tilde{Y} = Y$ and $\sum_{j=1}^{x} p_j^A = t_1 + t_3 - t_2$. From $\tilde{t_1} + \tilde{t_3} - \tilde{t_2} = t_1 + t_3 - t_2$, $\tilde{t_1} \leq t_1$, $\tilde{t_2} \geq t_2$, and $\tilde{t_3} \leq t_3$ we know that $\tilde{t_2} = t_2$ contradicts $\tilde{t_2} \neq t_2$. Thus, $\tilde{t_2} \leq d^B$. Claim 1 follows.

If $\tilde{t_2} - p_y^B \geq d^B$, then $\tilde{t_2} \neq t_2$. From Claim 1 we have $\tilde{t_2} \leq d < d + p_y^B$, i.e., $\tilde{t_2} - p_y^B < d$, which is a contradiction. Thus the condition $\tilde{t_2} - p_y^B < d^B$ is satisfied and the operation to get $\overline{\pi}$ is feasible. Then we have $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y}) = (\tilde{t_1}, \tilde{t_2} - p_y^B, \tilde{t_3}, \tilde{C}, \tilde{Y} + w_y^B \max\{\tilde{t_2} - d, 0\})$. Combining with the fact that $(t_1', t_2', t_3', C', Y') = (t_1, t_2 - p_y^B, t_3, C, Y + w_y^B \max\{t_2 - d, 0\})$, we have

$$\begin{cases} \overline{t_1} = \tilde{t_1} \leq t_1 = t_1', \\ \overline{t_2} = \tilde{t_2} - p_y^B \geq t_2 - p_y^B = t_2', \\ \overline{t_3} = \tilde{t_3} \leq t_3 = t_3', \\ \overline{C} = \tilde{C} \leq C = C'. \end{cases} \tag{6}$$

Next we prove that $\overline{Y} \leq Y'$. In fact, if $\tilde{t_2} = t_2$, then $\overline{Y} = \tilde{Y} + w_y^B \max\{\tilde{t_2} - d, 0\} \leq Y + w_y^B \max\{t_2 - d, 0\} = Y'$. If $\tilde{t_2} \neq t_2$, then from Claim 1 we know that $\tilde{t_2} \leq d$, and then $t_2 - d < 0$. Thus we have $\overline{Y} = \tilde{Y} \leq Y = Y'$.

Case 4. π' is obtained from π by scheduling J_y^B directly after schedule π_4. Then $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x}, \overleftarrow{y+1})$, and there is a state $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y}) \in \tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y+1})$ such that $\tilde{t_1} \leq t_1, \tilde{t_2} \geq t_2, \tilde{t_3} \leq t_3, \tilde{C} \leq C$, and $\tilde{Y} \leq Y$. Let $\tilde{\pi}$ be an $(\overrightarrow{x}, \overleftarrow{y+1})$-schedule corresponding to $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y})$, and let $\overline{\pi}$ be the $(\overrightarrow{x}, \overleftarrow{y})$-schedule obtained from $\tilde{\pi}$ by scheduling J_x^A directly after schedule $\tilde{\pi}_3$. Let $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y})$ be the state corresponding to $\overline{\pi}$. Then we have $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{C}, \overline{Y}) = (\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{C}, \tilde{Y} + w_y^B p_y^B)$. Combining with the fact that $(t_1', t_2', t_3', C', Y') = (t_1, t_2, t_3, C, Y + w_y^B p_y^B)$, we have

$$\begin{cases} \overline{t_1} = \tilde{t_1} \leq t_1 = t_1', \\ \overline{t_2} = \tilde{t_2} \geq t_2 = t_2', \\ \overline{t_3} = \tilde{t_3} \leq t_3 = t_3', \\ \overline{C} = \tilde{C} \leq C = C', \\ \overline{Y} = \tilde{Y} + w_y^B p_y^B \leq Y + w_y^B p_y^B = Y'. \end{cases} \tag{7}$$

The result follows. □

Theorem 1. *Algorithm 1 solves the Pareto-frontier scheduling problem $1|d_j^B = d|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$ in $O(n_A n_B d P_{sum} U^A U^B)$ time.*

Algorithm 1: For problem $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$

Set $\Gamma(\overrightarrow{0}, \overleftarrow{n_B+1}) = \{(0, t_0, t_0, 0, 0) : d - p_{\max}^A + 1 \le t_0 \le d + p_{\max}^B - 1\}$ and set $\Gamma(\overrightarrow{x}, \overleftarrow{y}) = \emptyset$ if $(x, y) \neq (0, n_B + 1)$.
for $x = 0, 1, \ldots, n_A$, $y = n_B + 1, n_B, \ldots, 1$, **do**
  for *each* $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x-1}, \overleftarrow{y})$, **do**
    if $0 < x \le n_A$, **then**
      $\Gamma(\overrightarrow{x}, \overleftarrow{y}) := \Gamma(\overrightarrow{x}, \overleftarrow{y}) \cup (t_1, t_2, t_3 + p_x^A, C + w_x^A(t_3 + p_x^A), Y)$
    end
    if $0 < x \le n_A$ *and* $t_1 + p_x^A \le t_2$, **then**
      $\Gamma(\overrightarrow{x}, \overleftarrow{y}) := \Gamma(\overrightarrow{x}, \overleftarrow{y}) \cup (t_1 + p_x^A, t_2, t_3, C + w_x^A(t_1 + p_x^A), Y)$
    end
  end
  for *each* $(t_1, t_2, t_3, C, Y) \in \Gamma(\overrightarrow{x}, \overleftarrow{y+1})$, **do**
    if $1 \le y < n_B + 1$, **then**
      $\Gamma(\overrightarrow{x}, \overleftarrow{y}) := \Gamma(\overrightarrow{x}, \overleftarrow{y}) \cup (t_1, t_2, t_3, C, Y + w_y^B p_y^B)$
    end
    if $1 \le y < n_B + 1$ *and* $t_1 \le t_2 - p_y^B < d$, **then**
      $\Gamma(\overrightarrow{x}, \overleftarrow{y}) := \Gamma(\overrightarrow{x}, \overleftarrow{y}) \cup (t_1, t_2 - p_y^B, t_3, C, Y + w_y^B \max\{t_2 - d, 0\})$
    end
  end
  For each newly generated $\Gamma(\overrightarrow{x}, \overleftarrow{y})$, set $\Gamma(\overrightarrow{x}, \overleftarrow{y}) := \tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y})$
end
Generate $\tilde{\mathbf{Q}}_1$ and, for each state $(C, Y) \in \tilde{\mathbf{Q}}_1$, derive the corresponding optimal schedule by backtracking.

Proof. The correctness of Algorithm 1 is guaranteed by Lemma 2, Lemma 1, and Lemma 3. Here we only analyze its time complexity. The initialization step takes $O(P_{\text{sum}} + n_A n_B)$ time, which is dominated by the final time complexity of Algorithm 1. In the implementation of Algorithm 1, we guarantee that $\Gamma(\overrightarrow{x}, \overleftarrow{y}) = \tilde{\Gamma}(\overrightarrow{x}, \overleftarrow{y})$. Note that $0 \le t_1 \le d^B$ and $d^B - p_{\max}^A + 1 \le t_3 \le P_{\text{sum}}$, then each state set $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ contains $O(d^B P_{\text{sum}} U^A U^B)$ states. Moreover, $\Gamma(\overrightarrow{x}, \overleftarrow{y})$ is obtained by performing at most two (constant) operations on the states in $\Gamma(\overrightarrow{x-1}, \overleftarrow{y}) \bigcup \Gamma(\overrightarrow{x}, \overleftarrow{y+1})$ for $x = 0, 1, \ldots, n_A$, $y = n_B + 1, n_B, \ldots, 1$. Note that the upper bounds of $\Sigma w_j^A C_j^A$ and $\Sigma w_j^B Y_j^B$ are given by $U^A = \sum_{j=1}^{n_A} w_j^A P_{\text{sum}}$ and $U^B = \sum_{j=1}^{n_B} w_j^B p_j^B$, respectively. Thus, the overall running time of Algorithm 1 is $O(n_A n_B d P_{\text{sum}} U^A U^B)$. □

4. An FPTAS

In this section, for problem $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$, we first give another dynamic programming algorithm, and then turn it into an FPTAS by the trimming technique. As for Algorithm 1, we first introduce the following terminologies and notations.

- An (x, y)**-schedule** is defined to be an $ABAB$-schedule π for $\mathcal{J}_x^A \cup \mathcal{J}_y^B$ with no idle time existing between subschedules π_1, π_2 and π_3, where $x \in \{1, 2, \ldots, n_A\}$ and $y \in \{1, 2, \ldots, n_B\}$.
- A vector $(t_1, t_2, t_3, W, k(\pi), C, Y)$ is introduced to denote a **state of** (x, y), in which t_1, t_2, t_3, W, k, C and Y, respectively, stand for the end point of π_1, the end point of π_2, the end point of π_3, the total weight of the jobs in π_3, the index of the last B-job in π_2, the total weighted completion time of the

A-jobs of $\mathcal{J}_x^A$, and the total weighted late work of the B-jobs of $\mathcal{J}_y^B$. Note that a state of (x, y) at least corresponds to some (x, y)-schedule.

- $\Gamma(x, y)$ denotes the set of all the states of (x, y).
- $\tilde{\Gamma}(x, y)$ denotes the set obtained from $\Gamma(x, y)$ by deleting the vectors $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{W}, k, \overline{C}, \overline{Y})$, for which there is another vector $(t_1, t_2, t_3, W, k, C, Y)$ with $t_1 \leq \overline{t_1}, t_2 \leq \overline{t_2}, t_3 \leq \overline{t_3}, W \leq \overline{W}, C \leq \overline{C}, Y \leq \overline{Y}$.
- Let $\mathbf{Q}_2 = \{(C, Y) : (t_1, t_2, t_3, W, k, C, Y) \in \tilde{\Gamma}(n_A, n_B)\}$, and let $\tilde{\mathbf{Q}}_2$ be the set of the non-dominated vectors in $\mathbf{Q}_2$.

Clearly, $\mathbf{Q}_2$ is an intermediate set. Similarly to the discussion for Algorithm 1, we can generate all the $\Gamma(x, y)$ for all the possible choices of the tuple (x, y) dynamically in the following way.

Initially, set $\Gamma(0, 0) = \{(0, 0, 0, 0, 0, 0, 0)\}$ and $\Gamma(x, y) = \emptyset$ if $(x, y) \neq (0, 0)$. Then we recursively generate all the state sets $\Gamma(x, y)$ from the previously generated sets $\Gamma(x-1, y)$ and $\Gamma(x, y-1)$. Specifically,

- For each state $(t_1, t_2, t_3, W, k, C, Y) \in \Gamma(x-1, y)$ with $\Gamma(x-1, y) \neq \emptyset$, add two states $(t_1', t_2', t_3', W', k', C', Y')$ and $(t_1'', t_2'', t_3'', W'', k'', C'', Y'')$ to the set $\Gamma(x, y)$, with

$$\begin{aligned}&(t_1', t_2', t_3', W', k', C', Y') = (t_1 + p_x^A, t_2 + p_x^A, t_3 + p_x^A, W, k, C + w_x^A(t_1 + p_x^A) + W p_x^A, Y \\ &+ w_k^B \max\{\min\{t_2 + p_x^A - d^B, p_x^A\}, 0\}), \text{and} (t_1'', t_2'', t_3'', W'', k'', C'', Y'') \\ &= (t_1, t_2, t_3 + p_x^A, W + w_x^A, k, C + w_x^A(t_3 + p_x^A), Y).\end{aligned}$$

These two states respectively correspond to the newly obtained (x, y)-schedules by scheduling job J_x^A immediately following the subschedule π_1 and immediately following the subschedule π_3, in some $(x-1, y)$ schedule π that corresponds to the state $(t_1, t_2, t_3, W, k, C, Y)$. Note that the first case occurs only when $t_1 + p_x^A \leq t_2$ is satisfied.

- For each state $(t_1, t_2, t_3, W, k, C, Y) \in \Gamma(x, y-1)$, also add two two states $(t_1', t_2', t_3', W', k', C', Y')$ and $(t_1'', t_2'', t_3'', W'', k'', C'', Y'')$ to the set $\Gamma(x, y)$, with

$$(t_1', t_2', t_3', W', k', C', Y') = (t_1, t_2 + p_y^B, t_3 + p_y^B, W, y, C + W p_y^B, Y + w_y^B \max\{t_2 + p_y^B - d^B, 0\}),$$

and

$$(t_1'', t_2'', t_3'', W'', k'', C'', Y'') = (t_1, t_2, t_3, W, k, C, Y + w_y^B p_y^B).$$

These two states respectively correspond to the newly obtained (x, y)-schedules by scheduling job J_y^B immediately after π_2 and immediately following the subschedule π_4, in some $(x, y-1)$ schedule π that corresponds to the state $(t_1, t_2, t_3, W, k, C, Y)$. Note that the first case occurs only when $t_2 < d^B$ is satisfied.

Note that, if in the above state-generation procedures we replace sets $\Gamma(x-1, y)$ and $\Gamma(x, y-1)$ with sets $\tilde{\Gamma}(x-1, y)$ and $\tilde{\Gamma}(x, y-1)$, then the resulting set of new states, denoted by $\Gamma'(x, y)$, may be different from $\Gamma(x, y)$. Recall that, when deleting those dominated vectors in the sets $\Gamma(x, y)$ and $\Gamma'(x, y)$, the newly obtained sets are respectively denoted by $\tilde{\Gamma}(x, y)$ and $\tilde{\Gamma}'(x, y)$, which will be shown to be identical in the following lemma, and its proof is similar to that of Lemma 3.

Lemma 4. $\tilde{\Gamma}(x,y) = \tilde{\Gamma}'(x,y)$.

Theorem 2. *Algorithm 2 solves* $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$ *in* $O(n_A n_B^2 d P_{sum} W_{sum}^A U^A U^B)$ *time.*

Algorithm 2: For solving $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$

Set $\Gamma(0,0) = \{(0,0,0,0,0,0,0)\}$ and set $\Gamma(x,y) = \emptyset$ if $(x,y) \neq (0,0)$.
for $x = 0,1,\ldots,n_A$, $y = 0,1,\ldots,n_B$, **do**
  for *each* $(t_1,t_2,t_3,W,k,C,Y) \in \Gamma(x-1,y)$, **do**
    if $0 < x \leq n_A$, **then**
      $\Gamma(x,y) := \Gamma(x,y) \cup (t_1,t_2,t_3+p_x^A, W+w_x^A, k, C+w_x^A(t_3+p_x^A), Y)$
      if $k = 0$ *or* $(k \neq 0$ *and* $t_2 + p_x^A - d^B < p_k^B)$, **then**
        $\Gamma(x,y) := \Gamma(x,y) \cup (t_1+p_x^A, t_2+p_x^A, t_3+p_x^A, W, k, C+w_x^A(t_1+p_x^A)+Wp_x^A, Y + w_k^B \max\{\min\{t_2+p_x^A-d, p_x^A\}, 0\})$
      end
    end
  end
  for *each* $(t_1,t_2,t_3,C,Y) \in \Gamma(x,y-1)$, **do**
    if $0 < y \leq n_B$, **then**
      $\Gamma(x,y) := \Gamma(x,y) \cup (t_1,t_2,t_3,W,k,C,Y+w_y^B p_y^B)$
      if $t_2 < d$, **then**
        $\Gamma(x,y) := \Gamma(x,y) \cup (t_1, t_2+p_y^B, t_3+p_y^B, W, y, C+Wp_y^B, Y+w_y^B \max\{t_2+p_y^B-d, 0\})$
      end
    end
  end
  The elimination step: for each newly generated $\Gamma(x,y)$, set $\Gamma(x,y) := \tilde{\Gamma}(x,y)$
end
Generate $\tilde{\mathbf{Q}}_2$ and, for each state $(C,Y) \in \tilde{\mathbf{Q}}_2$, derive the corresponding optimal schedule by backtracking.

Proof. The correctness of Algorithm 2 is guaranteed by the discussion above. Next we only analyze its time complexity. The initialization step takes $O(n_A n_B)$ time, which is dominated by the final time complexity of Algorithm 2. In the implementation of Algorithm 2, we guarantee that$\Gamma(x,y) = \tilde{\Gamma}(x,y)$. Note that $0 \leq t_1 \leq d^B$ and $d - p_{\max}^A + 1 \leq t_3 \leq P_{\text{sum}}$, $0 \leq k \leq n_B$ and $0 \leq W \leq W_{\text{sum}}^A$, then each state set $\Gamma(x,y)$ contains $O(n_B d P_{\text{sum}} W_{\text{sum}}^A U^A U^B)$ states. Moreover, $\Gamma(x,y)$ is obtained by performing at most two (constant) operations on the states in $\Gamma(x-1,y) \bigcup \Gamma(x,y-1)$ for $x = 0,1,\ldots,n_A$, $y = 0,1,\ldots,n_B$. Thus, the overall running time of Algorithm 2 is $O(n_A n_B^2 d^B P_{\text{sum}} W_{\text{sum}}^A U^A U^B)$. □

Next we turn Algorithm 2 into an FPTAS in the following way. Set $\Delta = 1 + \frac{\epsilon}{2n}$, $L^1 = \lceil log_\Delta d \rceil$, $L^3 = \lceil log_\Delta(P_{\text{sum}}) \rceil$, $L^W = \lceil log_\Delta(W_{\text{sum}}^A) \rceil$, $L^A = \lceil log_\Delta(U^A) \rceil$ and $L^B = \lceil log_\Delta(U^B) \rceil$. Set $I_i^1 = [\Delta^{(i-1)}, \Delta^i]$ for $i = 1,2,\ldots,L^1$, $I_i^3 = [\Delta^{(i-1)}, \Delta^i]$ for $i = 1,2,\ldots,L^3$, $I_i^W = [\Delta^{(i-1)}, \Delta^i]$ for $i = 1,2,\ldots,L^W$, $I_i^A = [\Delta^{(i-1)}, \Delta^i]$ for $i = 1,2,\ldots,L^A$ and $I_i^B = [\Delta^{(i-1)}, \Delta^i]$ for $i = 1,2,\ldots,L^B$. For $x = 0,1,\ldots,n_A$ and $y = 0,1,\ldots,n_B$, $\widehat{\Gamma}(x,y)$ is obtained from $\Gamma(x,y)$ by the following operation: for any two states (t_1,t_2,t_3,W,k,C,Y) and $(\overline{t_1},\overline{t_2},\overline{t_3},\overline{W},k,C,Y)$ in $\Gamma(x,y)$, if (t_1,t_3,W,C,Y) and $(\overline{t_1},\overline{t_3},\overline{W},\overline{C},\overline{Y})$ fall into the same box $I_u^1 \times I_v^3 \times I_w^W \times I_p^A \times I_q^B$ for $u = 1,2,\ldots,L^1$, $v = 1,2,\ldots,L^3$, $w = 1,2,\ldots,L^W$, $p = 1,2,\ldots,L^A$ and $q = 1,2,\ldots,L^B$ with

$t_2 \leq \overline{t_2}$, remaining the first one. Note that it takes $O(L^1L^3L^WL^AL^B)$ time to partition the boxes. Moreover, we define

$$\mathbf{Q}_3 = \{(C, Y) : (t_1, t_2, t_3, W, k, C, Y) \in \widehat{\Gamma}(n_A, n_B)\} \tag{8}$$

and let $\tilde{\mathbf{Q}}_3$ be the set of non-dominated vectors in $\mathbf{Q}_3$.

Theorem 3. *Algorithm 3 is an FPTAS for solving* $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$.

Algorithm 3: For solving $1|d_j^B = d|^{\#}(\sum w_j^A C_j^A, \sum w_j^B Y_j^B)$

1 Set $\Gamma(0,0) = \{(0,0,0,0,0,0,0)\}$ and set $\Gamma(x,y) = \emptyset$ if $(x,y) \neq (0,0)$.
2 **for** $x = 0, 1, \ldots, n_A$, $y = 0, 1, \ldots, n_B$, **do**
3 the same operations with Algorithm 2
4 **The elimination step:** for each newly generated $\Gamma(x,y)$, set $\Gamma(x,y) := \widehat{\Gamma}(x,y)$
5 **end**
6 Generate $\tilde{\mathbf{Q}}_3$ and, for each state $(C, Y) \in \tilde{\mathbf{Q}}_3$, derive the corresponding optimal schedule by backtracking.

Proof. By induction on $z = x + y$, We prove that, for any state $(t_1', t_2', t_3', W', k', C', Y') \in \Gamma(x,y)$, there is a state $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{W}, k', \overline{C}, \overline{Y}) \in \widehat{\Gamma}(x,y)$ such that $\overline{t_1} \leq \Delta^z t_1'$, $\overline{t_2} \leq t_2'$, $\overline{t_3} \leq \Delta^z t_3'$, $\overline{W} \leq \Delta^z W'$, $\overline{C} \leq \Delta^z C'$ and $\overline{Y} \leq \Delta^z Y'$.

This is obviously true for $z = 0$. Inductively suppose that it holds up to $z - 1$. Next we show that it also holds for z. Recall that each state $(t_1', t_2', t_3', W', k', C', Y') \in \Gamma(x,y)$ is derived from some state $(t_1, t_2, t_3, W, k, C, Y)$ in $\Gamma(x-1, y)$ or $\Gamma(x, y-1)$. Let π' be an (x,y)-schedule corresponding to $(t_1', t_2', t_3', W', k', C', Y')$, and let π be a schedule corresponding to the state $(t_1, t_2, t_3, W, k, C, Y)$. Using the induction hypothesis, there is a state $(\widehat{t_1}, \widehat{t_2}, \widehat{t_3}, \widehat{W}, k, \widehat{C}, \widehat{Y})$ in $\widehat{\Gamma}(x-1, y)$ or $\widehat{\Gamma}(x, y-1)$ such that $\widehat{t_1} \leq \Delta^{z-1} t_1$, $\widehat{t_2} \leq t_2$, $\widehat{t_3} \leq \Delta^{z-1} t_3$, $\widehat{W} \leq \Delta^{z-1} W$, $\widehat{C} \leq \Delta^{z-1} C$, and $\widehat{Y} \leq \Delta^{z-1} Y$. Let $\widehat{\pi}$ be an $(x-1, y)$-schedule or $(x, y-1)$-schedule corresponding to $(\widehat{t_1}, \widehat{t_2}, \widehat{t_3}, \widehat{W}, k, \widehat{C}, \widehat{Y})$, and if it is feasible, let $\tilde{\pi}$ be the (x,y)-schedule obtained from $\widehat{\pi}$ by performing the same operation that we perform on π to get π'. Let $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{W}, \tilde{k}, \tilde{C}, \tilde{Y})$ be the state corresponding to $\tilde{\pi}$. Furthermore, there is a state $(\overline{t_1}, \overline{t_2}, \overline{t_3}, \overline{W}, \tilde{k}, \overline{C}, \overline{Y}) \in \widehat{\Gamma}(x,y)$ in the same box with $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{W}, \tilde{k}, \tilde{C}, \tilde{Y})$ such that $\overline{t_1} \leq \Delta \tilde{t_1}$, $\overline{t_2} \leq \tilde{t_2}$, $\overline{t_3} \leq \Delta \tilde{t_3}$, $\overline{W} \leq \Delta \tilde{W}$, $\overline{C} \leq \Delta \tilde{C}$ and $\overline{Y} \leq \Delta \tilde{Y}$. There are four possible ways to get π' from π.

Case 1. π' is obtained from π by scheduling J_x^A directly after schedule π_1. Note that in this case, the condition $t_2 + p_x^A - d^B < p_k^B$ must be satisfied. Then $(t_1, t_2, t_3, W, k, C, Y) \in \Gamma(x-1, y)$, and the operation to get $\tilde{\pi}$ is feasible since $\widehat{t_2} + p_x^A - d^B \leq t_2 + p_x^A - d^B < p_k^B$. Then we have $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{W}, \tilde{k}, \tilde{C}, \tilde{Y}) = (\widehat{t_1} + p_x^A, \widehat{t_2} + p_x^A, \widehat{t_3} + p_x^A, \widehat{W}, k, \widehat{C} + w_x^A(\widehat{t_1} + p_x^A) + \widehat{W} p_x^A, \widehat{Y} + w_k^B \max\{\min\{\widehat{t_2} + p_x^A - d^B, p_x^A\}, 0\})$. Combining with the fact that $(t_1', t_2', t_3', W', k', C', Y') = (t_1 + p_x^A, t_2 + p_x^A, t_3 + p_x^A, W, k, C + w_x^A(t_1 + p_x^A) + W p_x^A, Y + w_k^B \max\{\min\{t_2 + p_x^A - d^B, p_x^A\}, 0\})$, we have

$$\begin{cases} \overline{t_1} \leq \Delta \tilde{t_1} = \Delta(\widehat{t_1} + p_x^A) \leq \Delta^z (t_1 + p_x^A) = \Delta^z t_1', \\ \overline{t_2} \leq \tilde{t_2} = \widehat{t_2} + p_x^A \leq t_2 + p_x^A = t_2', \\ \overline{t_3} \leq \Delta \tilde{t_3} = \Delta(\widehat{t_3} + p_x^A) \leq \Delta^z (t_3 + p_x^A) = \Delta^z t_3', \\ \overline{W} \leq \Delta \tilde{W} = \Delta \widehat{W} \leq \Delta^z W = \Delta^z W', \\ \tilde{k} = k = k', \\ \overline{C} \leq \Delta \tilde{C} = \Delta(\widehat{C} + w_x^A(\widehat{t_1} + p_x^A) + \widehat{W} p_x^A) \leq \Delta^z (C + w_x^A(t_1 + p_x^A) + W p_x^A) = \Delta^z C', \\ \overline{Y} \leq \Delta \tilde{Y} = \Delta(\widehat{Y} + w_k^B \max\{\min\{\widehat{t_2} + p_x^A - d^B, p_x^A\}, 0\}) \leq \Delta^z (Y + w_k^B \max\{\min\{t_2 + p_x^A - d^B, p_x^A\}, 0\}) = \Delta^z Y. \end{cases} \tag{9}$$

Case 2. π' is obtained from π by scheduling J_x^A directly after schedule π_3. Then $(t_1, t_2, t_3, W, k, C, Y) \in \Gamma(x-1, y)$, and $\tilde{\pi}$ is clearly a feasible schedule. Then we have $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{W}, \tilde{k}, \tilde{C}, \tilde{Y}) = (\widehat{t_1}, \widehat{t_2}, \widehat{t_3} + p_x^A, \widehat{W} + w_x^A, k, \widehat{C} + w_x^A(\widehat{t_3} + p_x^A), \widehat{Y})$. Combining with the fact that $(t_1', t_2', t_3', W', k', C', Y') = (t_1, t_2, t_3 + p_x^A, W + w_x^A, k, C + w_x^A(t_3 + p_x^A), Y)$, we have

$$\begin{cases} \overline{t_1} \le \Delta\tilde{t_1} = \Delta\widehat{t_1} \le \Delta^z t_1 = \Delta^z t_1', \\ \overline{t_2} \le \tilde{t_2} = \widehat{t_2} \le t_2 = t_2', \\ \overline{t_3} \le \Delta\tilde{t_3} = \Delta(\widehat{t_3} + p_x^A) \le \Delta^z(t_3 + p_x^A) = \Delta^z t_3', \\ \overline{W} \le \Delta\tilde{W} = \Delta(\widehat{W} + w_x^A) \le \Delta^z(W + w_x^A) = \Delta^z W', \\ \tilde{k} = k = k', \\ \overline{C} \le \Delta\tilde{C} = \Delta(\widehat{C} + w_x^A(\widehat{t_3} + p_x^A)) \le \Delta^z(C + w_x^A(t_3 + p_x^A)) = \Delta^z C', \\ \overline{Y} \le \Delta\tilde{Y} = \Delta\widehat{Y} \le \Delta^z Y = \Delta^z Y'. \end{cases} \tag{10}$$

Case 3. π' is obtained from π by scheduling J_y^B directly after schedule π_2. Note that in this case, the condition $t_2 < d^B$ must be satisfied. Then $(t_1, t_2, t_3, W, k, C, Y) \in \Gamma(x, y-1)$, and the operation to get $\tilde{\pi}$ is feasible since $\widehat{t_2} \le t_2 < d^B$. Then we have $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{W}, \tilde{k}, \tilde{C}, \tilde{Y}) = (\widehat{t_1}, \widehat{t_2} + p_y^B, \widehat{t_3} + p_y^B, \widehat{W}, y, \widehat{C} + \widehat{W}p_y^B, \widehat{Y} + w_y^B \max\{\widehat{t_2} + p_y^B - d^B, 0\})$. Combining with the fact that $(t_1', t_2', t_3', W', k', C', Y') = (t_1, t_2 + p_y^B, t_3 + p_y^B, W, y, C + Wp_y^B, Y + w_y^B \max\{t_2 + p_y^B - d^B, 0\})$, we have

$$\begin{cases} \overline{t_1} \le \Delta\tilde{t_1} = \Delta\widehat{t_1} \le \Delta^z t_1 = \Delta^z t_1', \\ \overline{t_2} \le \tilde{t_2} = \widehat{t_2} + p_y^B \le t_2 + p_y^B = t_2', \\ \overline{t_3} \le \Delta\tilde{t_3} = \Delta(\widehat{t_3} + p_y^B) \le \Delta^z(t_3 + p_y^B) = \Delta^z t_3', \\ \overline{W} \le \Delta\tilde{W} = \Delta\widehat{W} \le \Delta^z W = \Delta^z W', \\ \tilde{k} = y = k', \\ \overline{C} \le \Delta\tilde{C} = \Delta(\widehat{C} + \widehat{W}p_y^B) \le \Delta^z(C + Wp_y^B) = \Delta^z C', \\ \overline{Y} \le \Delta\tilde{Y} = \Delta(\widehat{Y} + w_y^B \max\{\widehat{t_2} + p_y^B - d^B, 0\}) \le \Delta^z(Y + w_y^B \max\{t_2 + p_y^B - d^B, 0\}) = \Delta^z Y'. \end{cases} \tag{11}$$

Case 4. π' is obtained from π by scheduling J_y^B directly after schedule π_4. Then $(t_1, t_2, t_3, W, k, C, Y) \in \Gamma(x, y-1)$, and $\tilde{\pi}$ is clearly a feasible schedule. Then we have $(\tilde{t_1}, \tilde{t_2}, \tilde{t_3}, \tilde{W}, \tilde{k}, \tilde{C}, \tilde{Y}) = (\widehat{t_1}, \widehat{t_2}, \widehat{t_3}, \widehat{W}, k, \widehat{C}, \widehat{Y} + w_y^B p_y^B)$. Combining with the fact that $(t_1', t_2', t_3', W', k', C', Y') = (t_1, t_2, t_3, W, k, C, Y + w_y^B p_y^B)$, we have

$$\begin{cases} \overline{t_1} \le \Delta\tilde{t_1} = \Delta\widehat{t_1} \le \Delta^z t_1 = \Delta^z t_1', \\ \overline{t_2} \le \tilde{t_2} = \widehat{t_2} \le t_2 = t_2', \\ \overline{t_3} \le \Delta\tilde{t_3} = \Delta\widehat{t_3} \le \Delta^z t_3 = \Delta^z t_3', \\ \overline{W} \le \Delta\tilde{W} = \Delta\widehat{W} \le \Delta^z W = \Delta^z W', \\ \tilde{k} = k = k', \\ \overline{C} \le \Delta\tilde{C} = \Delta\widehat{C} \le \Delta^z C = \Delta^z C', \\ \overline{Y} \le \Delta\tilde{Y} = \Delta(\widehat{Y} + w_y^B p_y^B) \le \Delta^z(Y + w_y^B p_y^B) = \Delta^z Y'. \end{cases} \tag{12}$$

Thus, for each state (C', Y') in $\tilde{\mathbf{Q}}_2$, there is a state $(\overline{C}, \overline{Y})$ in $\mathbf{Q}_3$ such that $\overline{C} \leq \Delta^n C' \leq (1+\epsilon)C'$ and $\overline{Y} \leq \Delta^n Y' \leq (1+\epsilon)Y'$.

Next we analyze its time complexity. The initialization step takes $O(n_A n_B)$ time, which is dominated by the final time complexity of Algorithm 3. In the implementation of Algorithm 3, we guarantee that $\Gamma(x,y) = \widehat{\Gamma}(x,y)$. Note that there are $O(L^1 L^3 L^W L^A L^B)$ distinct boxes and $0 \leq k \leq n_B$, then there are at most $O(n_B L^1 L^3 L^W L^A L^B)$ different states $(t_1, t_2, t_3, W, k, C, Y)$ in $\Gamma(x,y)$. Moreover, $\Gamma(x,y)$ is obtained by performing at most two (constant) operations on the states in $\Gamma(x-1,y) \bigcup \Gamma(x,y-1)$ for $x = 0, 1, \ldots, n_A$, $y = 0, 1, \ldots, n_B$. Thus, the overall running time of Algorithm 3 is $O(n_A n_B^2 L^1 L^3 L^W L^A L^B)$. □

5. Numerical Results

In this section some numerical results are provided to show the efficiency of our proposed algorithms. For running our optimization algorithms, we need to input the following parameters relative with the job instances: the numbers of A-jobs and B-jobs, the processing times and weights of all the jobs, and the common due date of B-jobs. By running Algorithms 1 and 2, we get the Pareto frontier. To use Algorithm 3, we need to choose the value of ϵ (>0) to get a $(1+\epsilon)$-approximate Pareto frontier. Note that for the same instance, the Pareto frontiers obtained by Algorithms 1 and 2 are the same, except that the running time of Algorithm 1 is theoretically faster than that of Algorithm 2. The closer the $(1+\epsilon)$-approximate Pareto frontier obtained by Algorithm 3 is to the curve obtained by Algorithms 1 and 2, the closer it is to the optimal solution.

We randomly generate some job instances, in which the numbers of the jobs are set to be $n = 4$ ($n_A = n_B = 2$), $n = 6$ ($n_A = n_B = 3$), and $n = 10$ ($n_A = n_B = 5$). The processing times and the weights of the jobs are randomly generated between 1 and 2. The common due date of B-jobs is set to be 5. What is more, we set $\epsilon = 1$. We ran our algorithms on these instances in a Matlab R2016b environment on an Intel(R) Core(TM) CPU, 2.50 GHz, 4 GB of RAM computer. In fact, when the number of the jobs is small, the Pareto frontier or the approximate Pareto frontier can be found relatively quickly, but when the number of the jobs increases, the running time will increase hugely. The following three Figures 1–3 present the Pareto frontier and $(1+\epsilon)$-approximate Pareto frontier generated by Algorithms 1–3. As can be seen from the three figures, the results obtained by Algorithms 1 and 2 are exactly the same. The results of Algorithm 3 are consistent with those of Algorithms 1 and 2, which may be due to the coincidence caused by the small size of the instance we chose and the few choices in the sizes of the jobs. In fact, considering that the problem we studied is NP-hard, our algorithm can only reach pseudo-polynomial-time theoretically. Therefore, our algorithm is theoretically more suitable for small-scale instances, where the sizes of the jobs are relatively uniform, which fits with the nature of such problems in real life, such as in logistics distribution centers where we use boxes of fixed sizes.

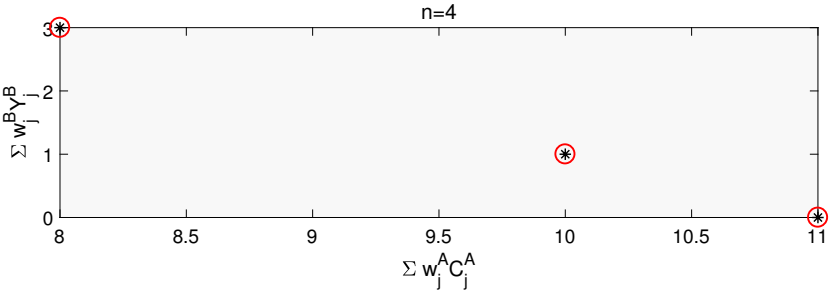

Figure 1. The black stars are the points generated by Algorithms 1 and 2, the red circles are points generated by Algorithm 3.

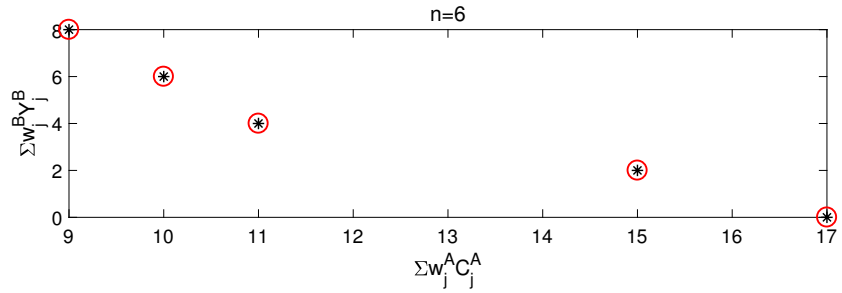

Figure 2. The black stars are the points generated by Algorithms 1 and 2, the red circles are points generated by Algorithm 3.

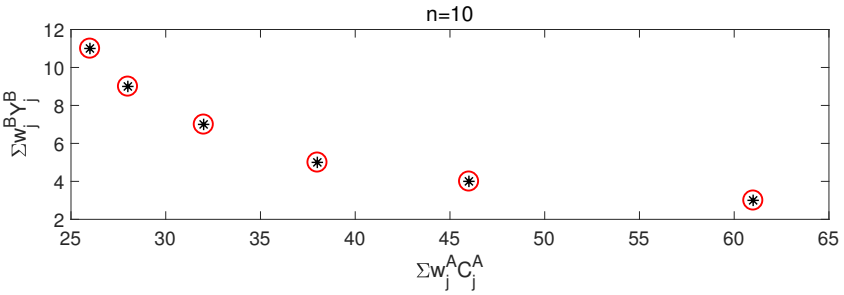

Figure 3. The black stars are the points generated by Algorithms 1 and 2, the red circles are points generated by Algorithm 3.

6. Conclusions

In this paper we investigated a Pareto-optimal problem of scheduling two agents' jobs on a single machine to minimize one agent's total weighted completion time and the other's total weighted late work. For this problem, we devised two dynamic programming algorithms to obtain the Pareto frontier, and an FPTAS to generate an approximate Pareto frontier. Some numerical results were also provided. Compared with the two problems $1|p_j^A \uparrow\downarrow w_j^A|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$ and $1|p_j^A \uparrow\downarrow w_j^A, d_j^B \uparrow\downarrow w_j^B|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$ studied in Zhang et al. [20], the constraint $p_j^A \uparrow\downarrow w_j^A$ was removed from the problem considered in this paper and we turned to the optimization problem under the condition that B-jobs had a common due date. Table 1 lists the computational complexity of the above three problems. As we can see from Table 1, the condition $p_j^A \uparrow\downarrow w_j^A$ seems to have a greater impact on the complexity result of the problem. In future research, we can try to devise more efficient approximation algorithms for our considered problem with a constant performance-ratio, and we can also study two-agent problems with other combinations of objective functions.

Table 1. Complexity of three problems.

Problems	Complexity	Reference
$1\|p_j^A \uparrow\downarrow w_j^A\|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$	$O(n_A n_B^2 U^A U^B)$	Zhang et al. [20]
$1\|p_j^A \uparrow\downarrow w_j^A, d_j^B \uparrow\downarrow w_j^B\|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$	$O(n_A n_B U^A U^B)$	Zhang et al. [20]
$1\|d_j^B = d\|^{\#}(\Sigma w_j^A C_j^A, \Sigma w_j^B Y_j^B)$	$O(n_A n_B d P_{\text{sum}} U^A U^B)$	Theorem 1

Author Contributions: Supervision, J.Y.; writing–original draft, Y.Z.; writing–review and editing, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the NSFC under grant numbers 12071442, 11671368 and 11771406.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three letter acronym
LD	linear dichroism

References

1. Agnetis, A.; Billaut, J.C.; Gawiejnowicz, S.; Pacciarelli, D.; Soukhal, A. *Multiagent Scheduling: Models and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2014.
2. Li, S.S.; Yuan, J.J. Single-machine scheduling with multi-agents to minimize total weighted late work. *J. Sched.* **2020**, *23*, 497–512. [CrossRef]
3. Hariri, A.M.A.; Potts, C.N.; Van Wassenhove, L.N. Single machine scheduling to minimize total weighted late work. *ORSA J. Comput.* **1995**, *7*, 232–242. [CrossRef]
4. 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.* **2016**, *273*, 912–923. [CrossRef]
5. Wan, L.; Wei, L.J.; Xiong, N.X.; Yuan, J.J.; Xiong, J.C. Pareto optimization for the two-agent scheduling problems with linear non-increasing deterioration based on Internet of Things. *Future Gene. Comp. Syst.* **2017**, *76*, 293–300. [CrossRef]
6. Gao, Y.; Yuan, J.J. Bi-criteria Pareto-scheduling on a single machine with due indices and precedence constraints. *Discret. Optim.* **2017**, *25*, 105–119. [CrossRef]
7. He, C.; Leung, J. Two-agent scheduling of time-dependent jobs. *J. Comb. Optim.* **2017**, *34*, 362–377. [CrossRef]
8. Yuan, J.J.; Ng, C.T.; Cheng, T.C.E. Two-agent single-machine scheduling with release dates and preemption to minimize the maximum lateness. *J. Sched.* **2015**, *18*, 147–153. [CrossRef]
9. Wan, L. Two-Agent Scheduling to Minimize the Maximum Cost with Position-Dependent Jobs. *Discrete Dyn. Nat. Soc.* **2015**. [CrossRef]
10. Dabia, S.; Talbi, E.G.; Van Woensel, T.; De Kok, T. Approximating multi-objective scheduling problems. *Comput. Oper. Res.* **2013**, *40*, 1165–1175. [CrossRef]
11. Lee, K.; Choi, B.C.; Leung, J.Y.T.; Pinedo, M.L. Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. *Inf. Process. Lett.* **2009**, *109*, 913–917. [CrossRef]
12. Legriel, J.; Guernic, C.L.; Cotton, S.; Maler, O. Approximating the pareto front of multi-criteria optimization problems. In Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems held at the 13th Joint European Conferences on Theory and Practice of Software, Paphos, Cyprus, 20–28 March 2010.
13. Marinescu, R. Efficient approximation algorithms for multi-objective constraint optimization. In Proceedings of the 2nd International Conference on Algorithmic Decision Theory, Piscataway, NJ, USA, 26–28 October 2011.
14. Vassilvitskii, S.; Yannakakis, M. Efficiently computing succinct trade-off curves. *Theor. Comput. Sci.* **2005**, *348*, 334–356. [CrossRef]
15. Yin, Y.Q.; Cheng, S.R.; Cheng, T.C.E.; Wang, D.J.; Wu, C.C. Just-in-time scheduling with two competing agents on unrelated parallel machines. *Omega-Int. J. Manag. Sci.* **2016**, *63*, 41–47. [CrossRef]
16. Yin, Y.Q.; Cheng, T.; Wang, D.J.; Wu, C.C. Two-agent flowshop scheduling to maximize the weighted number of just-in-time jobs. *J. Sched.* **2017**, *20*, 313–335. [CrossRef]

17. Chen, R.X.; Li, S.S.; Li, W.J. Multi-agent scheduling in a no-wait flow shop system to maximize the weighted number of just-in-time jobs. *Eng. Optim.* **2019**, *51*, 217–230. [CrossRef]
18. Purcaru, C.; Precup, R.E.; Iercan, D.; Fedorovici, L.O.; David, R.C.; Dragan, F. Optimal robot path planning using gravitational search algorithm. *Int. J. Artif. Intell.* **2013**, *10*, 1–20.
19. Soares, A.; Râbelo, R.; Delbem, A. Optimization based on phylogram analysis. *Expert Syst. Appl.* **2017**, *78*, 32–50. [CrossRef]
20. Zhang, Y.; Yuan, J.J. A note on a two-agent scheduling problem related to the total weighted late work. *J. Comb. Optim.* **2019**, *37*, 989–999. [CrossRef]
21. Zhang, Y.; Yuan, J.J.; Ng, C.T.; Cheng, T.C.E. Pareto-optimization of three-agent scheduling to minimize the total weighted completion time, weighted number of tardy jobs, and total weighted late work. *Nav. Res. Logist.* **2020**, in press.