mathematics

MDPI

Article

Scheduling of Jobs with Multiple Weights on a Single Machine for Minimizing the Total Weighted Number of Tardy Jobs

Shuen Guo [1,*], Hao Lang [2] and Hanxiang Zhang [2]

[1] School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, China
[2] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hong Kong SAR, China
* Correspondence: shuenoug520@163.com or gshuen@unc.edu

Abstract: We consider the scheduling of jobs with multiple weights on a single machine for minimizing the total weighted number of tardy jobs. In this setting, each job has m weights (or equivalently, the jobs have m weighting vectors), and thus we have m criteria, each of which is to minimize the total weighted number of tardy jobs under a corresponding weighting vector of the jobs. For this scheduling model, the feasibility problem aims to find a feasible schedule such that each criterion is upper bounded by its threshold value, and the Pareto scheduling problem aims to find all the Pareto-optimal points and for each one a corresponding Pareto-optimal schedule. Although the two problems have not been studied before, it is implied in the literature that both of them are unary NP-hard when m is an arbitrary number. We show in this paper that, in the case where m is a fixed number, the two problems are solvable in pseudo-polynomial time, the feasibility problem admits a dual-fully polynomial-time approximation scheme, and the Pareto-scheduling problem admits a fully polynomial-time approximation scheme.

Keywords: scheduling; pareto-optimal points; multi-weights; tardy jobs

MSC: 90B35; 90C27

Citation: Guo, S.; Lang, H.; Zhang, H. Scheduling of Jobs with Multiple Weights on a Single Machine for Minimizing the Total Weighted Number of Tardy Jobs. *Mathematics* **2023**, *11*, 1013. https://doi.org/10.3390/math11041013

Academic Editors: Adrian Deaconu, Petru Adrian Cotfas and Daniel Tudor Cotfas

Received: 7 January 2023
Revised: 9 February 2023
Accepted: 10 February 2023
Published: 16 February 2023

1. Introduction

The number of tardy jobs is a fundamental criterion related to the due date in scheduling problems. It is critical for examining the efficiency of production and evaluation management and has a wide range of applications in reality. The weighted number of tardy jobs is a familiar scheduling criterion in scheduling research. The objective function $\sum U_j$ was first studied by Moore [1] and then, $\sum w_j U_j$ was studied in many works, among which Lawler and Moore [2], Karp [3], and Sahni [4] are representative. Subsequently, more and more studies on these two objectives were presented, especially the primary–secondary scheduling problems and the multi-agent scheduling problems, which can better meet the needs of customers and managers. In the previous multi-criterion scheduling problems, each of the jobs to be scheduled generally had a single weight. However, there are cases in which the jobs have multi-weights.

Take the information transmission as an example. In computer systems, information transmission is an important and common business. In practice, an information service company would transmit n pieces of information (corresponding to n jobs $J_1, J_2, \ldots, J_n$) to m customers (where $m \geq 2$) every day. Each piece of information J_j has its own transmission time (corresponding to the processing time p_j of J_j), which may be different on different days. Based on the usual cooperation experience, all the m customers have an expected time for receiving each information J_j (corresponding to the due date d_j of J_j). The transmission time p_j of each piece of information J_j on each workday is known in advance. In addition, the importance of each piece of information to different customers is different. So each piece

of information J_j will have m weights $(w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)})$, called the weight vector of J_j, where $w_j^{(i)}$ is the weight of J_j with respect to the i-th customer and also can be understood as the degree of dissatisfaction of the i-th customer when J_j is tardy. The information service company definitely hopes all the customers can receive all the information on time, but sometimes this is impractical. Therefore, the decision maker needs to find a reasonable information transmission scheme to minimize all the customers' degrees of dissatisfaction, or equivalently, to minimize the sum of the weight vectors of the tardy jobs, which is called the objective vector. Considering the conflicts of interest and inconsistencies between different customers, finding all the non-dominated solutions is an ideal choice to the problem. This motivates us to study the problem in this paper.

For the sake of understanding the problem better, now let us consider a concrete instance with $n = 8$ and $m = 3$. The data are given in the following Table 1.

Table 1. Instance 1.

Job J_j	Processing Time p_j	Due Date d_j	Weights		
			Weight $w_j^{(1)}$	Weight $w_j^{(2)}$	Weight $w_j^{(3)}$
J_1	2	3	3	2	1
J_2	4	5	4	2	3
J_3	3	8	1	1	2
J_4	1	10	2	1	3
J_5	3	12	2	2	4
J_6	2	13	3	2	3
J_7	5	15	2	1	2
J_8	1	18	4	3	2

In this instance, there are many ways to schedule the jobs, and the objective weight vectors maybe different for different schedules. In the following Figures 1–3, we display three different schedules in which the early jobs are reasonably scheduled in the nondecreasing order of due dates, and the tardy jobs are scheduled after all the early jobs.

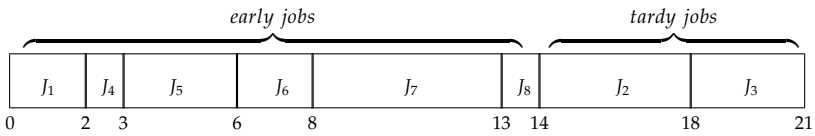

Figure 1. Schedule σ_1 corresponding to Instance I_1.

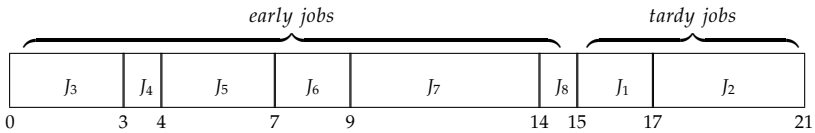

Figure 2. Schedule σ_2 corresponding to Instance I_1.

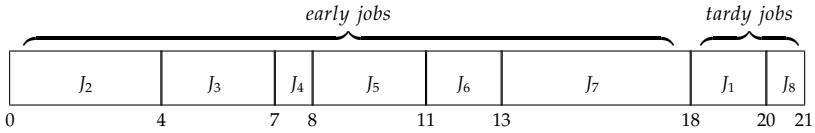

Figure 3. Schedule σ_3 corresponding to Instance I_1.

In schedule σ_1, jobs J_2 and J_3 are tardy. Then, the objective vector of σ_1 is

$$\mathbf{w}_2 + \mathbf{w}_3 = (w_2^{(1)}, w_2^{(2)}, w_2^{(3)}) + (w_3^{(1)}, w_3^{(2)}, w_3^{(3)}) = (4,2,3) + (1,1,2) = (5,3,5).$$

Similarly, the objective vectors of σ_2 and σ_3 are $(7,4,4)$ and $(7,5,3)$, respectively. From the above three objective vectors, the three schedules have different effects for the customers.

Problem formulation: Suppose that we have n jobs $J_1, J_2, \ldots, J_n$ to be processed on a single machine. Each job J_j has a processing time $p_j > 0$, a due date $d_j \geq 0$, and m weights $w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)}$, where $m \geq 2$. Given $i \in \{1,2,\ldots,m\}$, we call $\mathbf{w}^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_n^{(i)})$ the i-th *weighting vector* of the jobs. Then, each job J_j has weight $w_j^{(i)}$ under the i-th weighting vector. Given $j \in \{1,2,\ldots,n\}$, we call $\mathbf{w}_j = (w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)})$ the *weight vector* of job J_j. All the jobs are released at time 0 and preemption is not allowed. Then, a schedule σ can be denoted by a permutation $\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$, where $J_{\sigma(x)}$ is the x-th job in schedule σ. The notation and terminology will be used in this article.

- $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is the set of all jobs and $\mathcal{J}_x = \{J_1, J_2, \ldots, J_x\}$ is the set of the first x jobs of $\mathcal{J}$. Note that $\mathcal{J}_0 = \emptyset$ and $\mathcal{J}_n = \mathcal{J}$.
- $P_x = p_1 + p_2 + \cdots + p_x$ is the total processing time of the first x jobs of $\mathcal{J}$, $x = 1,2,\ldots,n$. P_n is also denoted by P.
- $W^{(i)} = w_1^{(i)} + w_2^{(i)} + \cdots w_n^{(i)}$ is the total i-th weight, $i = 1,2,\ldots,m$.
- $C_{\sigma(x)}(\sigma) = p_{\sigma(1)} + p_{\sigma(2)} + \cdots + p_{\sigma(x)}$ is the *completion time* of job $J_{\sigma(x)}$ in schedule σ, $1 \leq x \leq n$.
- J_j is called a *tardy job* in schedule σ if $C_j(\sigma) > d_j$ and an *early job* if $C_j(\sigma) \leq d_j$.
- The *tardy indicator number* of job J_j in schedule σ is given by $U_j(\sigma) = 0, 1$ according to J_j is early or tardy in σ.
- $\sum_{j=1}^{n} U_j(\sigma)$ is called the *number of tardy jobs* in schedule σ.
- $U_w^{(i)}(\sigma) = \sum_{j=1}^{n} w_j^{(i)} U_j(\sigma)$ is called the *weighted number of tardy jobs* with respect to the i-th weighting vector in schedule σ, $1 \leq i \leq m$.
- $\mathbf{U}_w(\sigma) = (U_w^{(1)}(\sigma), U_w^{(2)}(\sigma), \ldots, U_w^{(m)}(\sigma))$ is call the *objective vector* of schedule σ.

We summarize the above notation and terminology in the following Table 2:

Table 2. The notation and terminology.

Notation	Terminology
p_j	The processing time of job J_j, $1 \leq j \leq n$
d_j	The due date of job J_j, $1 \leq j \leq n$
$\mathbf{w}^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_n^{(i)})$	The i-th weighting vector, $1 \leq i \leq m$
$\mathbf{w}_j = (w_j^{(1)}, w_j^{(2)}, \ldots, w_j^{(m)})$	The weight vector of job J_j, $1 \leq j \leq n$
$\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$	The set of all jobs
$\mathcal{J}_x = \{J_1, J_2, \ldots, J_x\}$	The set of the first x jobs of $\mathcal{J}$
$P_x = p_1 + p_2 + \cdots + p_x$	The total processing time of the first x jobs of $\mathcal{J}$, $x = 1,2,\ldots,n$
$C_{\sigma(x)}(\sigma) = p_{\sigma(1)} + p_{\sigma(2)} + \cdots + p_{\sigma(x)}$	The completion time of job $J_{\sigma(x)}$ in schedule σ, $1 \leq x \leq n$
$U_j(\sigma)$	The tardy indicator number of job J_j in schedule σ
$\sum_{j=1}^{n} U_j(\sigma)$	The number of tardy jobs in schedule σ
$U_w^{(i)}(\sigma) = \sum_{j=1}^{n} w_j^{(i)} U_j(\sigma)$	The weighted number of tardy jobs with respect to $\mathbf{w}^{(i)}$ in schedule σ
$\mathbf{U}_w(\sigma) = (U_w^{(1)}(\sigma), U_w^{(2)}(\sigma), \ldots, U_w^{(m)}(\sigma))$	The objective vector of schedule σ

For convenience, several related concepts are defined below.

Definition 1. *Given a vector* $\boldsymbol{Q} = (Q_1, Q_2, \ldots, Q_m)$*, the feasibility problem, denoted* $1||\boldsymbol{U}_w \leq \boldsymbol{Q}$*, asks for finding a* ***feasible schedule*** σ *such that* $\boldsymbol{U}_w(\sigma) \leq \boldsymbol{Q}$*. We call* $\boldsymbol{Q}$ *the* ***threshold vector*** *and call each* Q_i*,* $i = 1, 2, \ldots, m$*, the* ***threshold value*** *of the i-th criterion* $U_w^{(i)}$*.*

Definition 2. *A schedule* σ *of* $\mathcal{J}$ *is called* ***Pareto-optimal*** *if there is no schedule* π *of* $\mathcal{J}$ *such that* $\boldsymbol{U}_w(\pi) \leq \boldsymbol{U}_w(\sigma)$ *and* $\boldsymbol{U}_w(\pi) \neq \boldsymbol{U}_w(\sigma)$*. In this case, we also say that* $\boldsymbol{U}_w(\sigma)$ *is the* ***Pareto-optimal point*** *corresponding to the Pareto-optimal schedule* σ*. We use* $\mathcal{P}(\mathcal{J})$ *to denote the set of all Pareto-optimal points and call it the* ***Pareto frontier****.*

Definition 3. *The Pareto-scheduling problem for minimizing* $\boldsymbol{U}_w = (U_w^{(1)}, U_w^{(2)}, \ldots, U_w^{(m)})$*, denoted* $1||\mathcal{P}(\boldsymbol{U}_w)$*, i.e.,* $1||\mathcal{P}(U_w^{(1)}, U_w^{(2)}, \ldots, U_w^{(m)})$*, asks for determining the Pareto frontier* $\mathcal{P}(\mathcal{J})$ *and for each Pareto-optimal point in* $\mathcal{P}(\mathcal{J})$ *a corresponding Pareto-optimal schedule. We use* $1||\widetilde{\mathcal{P}}(\boldsymbol{U}_w)$ *to denote the* ***weakened version*** *which only asks for finding a set of objective vectors including all the Pareto-optimal points of* $1||\mathcal{P}(\boldsymbol{U}_w)$*.*

Definition 4. *A* ***dual-fully polynomial-time approximation scheme*** *(DFPTAS) for the feasibility problem* $1||\boldsymbol{U}_w \leq \boldsymbol{Q}$ *on instance* $\mathcal{J}$ *is a family of algorithm* $\{A_\varepsilon : \varepsilon > 0\}$ *such that, for each* $\varepsilon > 0$*, the algorithm* A_ε *runs in a polynomial time in* $\frac{1}{\varepsilon}$ *and the size of* $\mathcal{J}$*, and moreover,* A_ε *either finds a schedule* σ *of* $\mathcal{J}$ *such that* $\boldsymbol{U}_w(\sigma) \leq (1+\varepsilon)\boldsymbol{Q}$ *or correctly tells us that the problem* $1||\boldsymbol{U}_w \leq \boldsymbol{Q}$ *is infeasible.*

For the Pareto scheduling problem $1||\mathcal{P}(\boldsymbol{U}_w)$ *on instance* $\mathcal{J}$*, a family of algorithms* $\{A_\varepsilon : \varepsilon > 0\}$ *is called a* ***fully polynomial-time approximation scheme*** *(FPTAS) if for each positive number* ε*, the algorithm* A_ε *runs in a polynomial time in* $\frac{1}{\varepsilon}$ *and the size of* $\mathcal{J}$*, and moreover,* A_ε *generates a set* $\mathcal{P}_\varepsilon$ *of objective vectors such that, for each point (m-vector)* $\boldsymbol{u} \in \mathcal{P}(\mathcal{J})$*, there is a vector* $\boldsymbol{v} \in \mathcal{P}_\varepsilon$ *such that* $\boldsymbol{v} \leq (1+\varepsilon)\boldsymbol{u}$*.*

In this paper, we study the feasibility problem $1||\mathbf{U}_w \leq \mathbf{Q}$, the Pareto-scheduling problem $1||\mathcal{P}(\mathbf{U}_w)$, and the weakened Pareto-scheduling problem $1||\widetilde{\mathcal{P}}(\mathbf{U}_w)$.

Related known results: The weighted number of tardy jobs is a familiar scheduling criterion in scheduling research. In the early time, Moore [1] presented an $O(n \log n)$-time algorithm for the problem $1||\sum U_j$. For the problem $1|d_j = d|\sum w_j U_j$, Karp [3] showed its binary NP-hardness. For the problem $1||\sum w_j U_j$, Lawler and Moore [2] presented an $O(nP)$-time algorithm, and Sahni [4] presented an $O(nW)$-time algorithm, where $P = \sum_{j=1}^{n} p_j$ and $W = \sum_{j=1}^{n} w_j$.

Now, we consider the primary–secondary scheduling problem $1||\text{Lex}(\sum w_j^{(1)} U_j, \sum w_j^{(2)} U_j)$, which minimizes the primary criterion $\sum w_j^{(i)} U_j$ firstly and the secondary criterion $\sum w_j^{(2)} U_j$ next. This problem is binary NP-hard since the problem $1||\sum w_j U_j$ is binary NP-hard. Lee and Vairaktarakis [5] showed that the problem $1||\text{Lex}(\sum U_j, \sum w_j^{(2)} U_j)$ is binary NP-hard and is solvable in $O(n^2 P)$ time. According to Lee and Vairaktarakis [5], the exact complexity (unary NP-hard or solvable in pseudo-polynomial time) of the problem $1||\text{Lex}(\sum w_j^{(1)} U_j, \sum w_j^{(2)} U_j)$ is still open.

Our research is also related to the multi-agent scheduling. The notation and terminology in multi-agent scheduling can be found in Agnetis et al. [6]. In particular, "CO" indicates that we consider competing agents, and "ND" indicates that we consider non-disjoint agents. Cheng et al. [7] studied the feasibility problem $1|\text{CO}|\sum w_j^{(i)} U_j^{(i)} \leq Q_i|_{1 \leq i \leq m}$. They showed that the problem is unary NP-complete if m is an arbitrary number and is solvable in pseudo-polynomial time if m is a fixed number. This further implies that the Pareto-scheduling problem $1|\text{CO}|\mathcal{P}(\sum w_j^{(1)} U_j^{(1)}, \sum w_j^{(2)} U_j^{(2)}, \cdots, \sum w_j^{(m)} U_j^{(m)})$ is unary NP-hard. Note that the CO-agent model is a special version of the ND-agent model under the condition $d_j^{(k)} = d_j$ (which means that the due dates of the jobs are independent of the agents). Then, the problem $1|\text{ND}, d_j^{(k)} = d_j|\mathcal{P}(\sum w_j^{(1)} U_j^{(1)}, \sum w_j^{(2)} U_j^{(2)}, \cdots, \sum w_j^{(m)} U_j^{(m)})$ is

also unary NP-hard. For the constraint problem $1|\text{ND}, d_j^{(k)} = d_j|\sum w_j^{(1)}U_j^{(1)} : \sum w_j^{(i)}U_j^{(i)} \leq Q_i|_{2\leq i\leq m}$, Agnetis et al. [6] presented a pseudo-polynomial-time algorithm when m is a fixed number. Exact complexity of the constraint problem $1|\text{ND}|\sum w_j^{(1)}U_j^{(1)} : \sum w_j^{(2)}U_j^{(2)} \leq Q$ was posed as open in Agnetis et al. [6]. When the jobs have identical processing times, research for some double-weight bicriteria scheduling on uniform parallel machines can be found in Zou and Yuan [8].

Regarding multi-weight scheduling, Guo et al. [9] studied problems $1||\mathcal{P}(\sum w_j'U_j, \sum w_j''Y_j)$ and $1|d_j = d|\mathcal{P}(\sum w_j'U_j, \sum w_j''Y_j)$ and presented pseudo-polynomial algorithms and FPTASs for both of them. They also provided the computational complexity results of several special cases of the problem $1||\mathcal{P}(\sum w_j'U_j, \sum w_j''Y_j)$. Chen et al. [10] provided a pseudo-polynomial algorithm for the preemptive scheduling problems $1|pmtn|^{\#}(\sum w_j^{(1)}Y_j, \sum w_j^{(2)}U_j)$ and studied some subproblems. Guo et al. [11] provided a pseudo-polynomial algorithm and an FPTAS for the problem $1||\mathcal{P}(\mathbf{Y}_w)$.

Other recently related studies can be found in Zhang et al. [12], Masadeh et al. [13], Zhang et al. [14], Kim and Kim [15], Valouxis et al. [16], and Wang et al. [17] among many others.

Remark 1. *Since 0-weights are allowed, a scheduling problem under multiple weights with criteria* $\sum w_j^{(i)}U_j|_{i=1,2,\ldots,m}$ *is equivalent to its corresponding problem under ND-agents under the condition* $d_j^{(k)} = d_j$ *with criteria* $\sum w_j^{(i)}U_j^{(i)}|_{i=1,2,\ldots,m}$. *Then, our problem* $1||\mathcal{P}(U_w^{(1)}, U_w^{(2)}, \ldots, U_w^{(m)})$ *is equivalent to the ND-agent Pareto-scheduling problem* $1|ND, d_j^{(k)} = d_j|\mathcal{P}(\sum w_j^{(1)}U_j^{(1)}, \sum w_j^{(2)}U_j^{(2)}, \cdots, \sum w_j^{(m)}U_j^{(m)})$. *From the unary NP-hardness of the subproblem* $1|CO|\mathcal{P}(\sum w_j^{(1)}U_j^{(1)}, \sum w_j^{(2)}U_j^{(2)}, \cdots, \sum w_j^{(m)}U_j^{(m)})$, *where* m *is an arbitrary number, we conclude that our problem* $1||\mathcal{P}(U_w^{(1)}, U_w^{(2)}, \ldots, U_w^{(m)})$ *is unary NP-hard if* m *is an arbitrary number. Hence, it is reasonable to present algorithms for this problem when* m *is a fixed number.*

Yuan [18] showed that the problem $1|\overline{d}_j|U_j$ is unary NP-hard. This implies that the exact complexity of the open problem $1|\text{ND}|\sum w_j^{(1)}U_j^{(1)} : \sum w_j^{(2)}U_j^{(2)} \leq Q$ posed in Agnetis et al. [6] is unary NP-hard. This also means that our research in this paper cannot be extended to the ND-agent Pareto-scheduling problem $1|\text{ND}|\mathcal{P}(\sum w_j^{(1)}U_j^{(1)}, \sum w_j^{(2)}U_j^{(2)}, \cdots, \sum w_j^{(m)}U_j^{(m)})$, where m is a fixed number.

We notice that the open problem $1||\text{Lex}(\sum w_j^{(1)}U_j, \sum w_j^{(2)}U_j)$ posed in Lee and Vairaktarakis [5] is solvable in pseudo-polynomial time, since it can be reduced in pseudo-polynomial time to the constraint problem $1|\text{ND}, d_j^{(k)} = d_j|\sum w_j^{(2)}U_j^{(2)} : \sum w_j^{(1)}U_j^{(1)} \leq Q$ solvable in pseudo-polynomial time by Agnetis et al. [6], where Q is the optimal value of the problem $1||\sum w_j^{(1)}U_j$.

Finally, we illustrate the differences between our work and the known ones.

In the known two-agent or multi-agent scheduling problems, each job just has a contribution to one of the objectives. However, in this paper, each job must contribute to all of the objectives. Thus, our problems are indeed different from the multi-agent problems. Moreover, the problems in Guo et al. [9] and Chen et al. [10] are different from ours, first because of the objective functions and then because of the jobs with two weights in these problems but multi-weights in our problems. Although jobs in the problem studied in Guo et al. [11] have multi-weights, the objective function is different from that studied in this paper.

Our contributions: Let m be a fixed positive integer and assume that all the parameters $p_j, d_j, w_j^{(i)}, Q_i$ are integer-valued. Our contributions can be listed as follows.

- For the problem $1||\mathbf{U}_w \leq \mathbf{Q}$, we present a pseudo-polynomial $O(nQ_1Q_2\cdots Q_m)$-time algorithm and a DFPTAS of running time $O(n^{m+1}(\frac{1}{\varepsilon})^m)$.
- For the problem $1||\widetilde{\mathcal{P}}(\mathbf{U}_w)$, we present a pseudo-polynomial $O(nW^{(1)}W^{(2)}\cdots W^{(m)})$-time algorithm. By a further discussion, the problem $1||\mathcal{P}(\mathbf{U}_w)$ is also solvable in $O(n\prod_{i=1}^{m} W^{(i)})$ time.
- For the problem $1||\mathcal{P}(\mathbf{U}_w)$, we present an FPTAS of running time $O(\frac{n^{m+1}}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$.

Organization of the paper: This paper is organized as follows. In Section 2, we present some preliminaries. In Section 3, we study the feasiblility problem $1||\mathbf{U}_w \leq \mathbf{Q}$. In Section 4, we study the Pareto-scheduling problems $1||\mathcal{P}(\mathbf{U}_w)$ and $1||\widetilde{\mathcal{P}}(\mathbf{U}_w)$. In the last section, we conclude the paper and suggest future research topics.

2. Preliminaries

For convenience, we renumber the n jobs of $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ in the EDD (earliest due date first) order such that

$$d_1 \leq d_2 \leq \cdots \leq d_n. \tag{1}$$

We call $(J_1, J_2, \ldots, J_n)$ the *EDD-schedule* of $\mathcal{J}$. Moreover, we renumber the m weighting vectors $\mathbf{w}^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_n^{(i)})$, $i = 1, 2, \ldots, m$, in the non-decreasing order of their total weights such that

$$W^{(1)} \leq W^{(2)} \leq \cdots \leq W^{(m)}. \tag{2}$$

Then, we keep the indexing in (1) and (2) throughout this paper.

Some other important definitions are used in the rest of our paper.

Definition 5. *A schedule σ of $\mathcal{J}$ is called **efficient** if σ satisfies the following two conditions: (i) the early jobs in σ are consecutively scheduled from time 0 in the increasing order of their jobs indices, and (ii) the tardy jobs in σ are scheduled after all the early jobs (in an arbitrary order). By the job-shifting argument, the following lemma can be easily verified.*

Lemma 1. *The problems studied in this paper allow us to only consider the efficient schedules of $\mathcal{J}$.*

Definition 6. *For an efficient schedule σ of $\mathcal{J}$, the **early-makespan** of σ, denoted $\tau(\sigma)$, is defined to be the total processing time of the early jobs in σ. Then, $\tau(\sigma) = \sum\{p_j : U_j(\sigma) = 0\}$. The early-makespan criterion will play an important role in this paper.*

Because of the multi-criteria $\mathbf{U}_w = (U_w^{(1)}, U_w^{(2)}, \ldots, U_w^{(m)})$ studied in this paper, we need some knowledge about the operations on vector sets.

Let $k \geq 2$ be an integer. We use $\mathbf{0}$ to denote the all-zero k-vector, i.e., $\mathbf{0} = (0, 0, \ldots, 0)$.

Definition 7. *For two k-vectors $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_k)$, $\boldsymbol{u} \leq \boldsymbol{v}$ indicates that $u_i \leq v_i$ for $i = 1, 2, \ldots, k$, and $\boldsymbol{u} < \boldsymbol{v}$ indicates that $u_i < v_i$ for $i = 1, 2, \ldots, k$. The notation $\boldsymbol{u} \geq \boldsymbol{v}$ and $\boldsymbol{u} > \boldsymbol{v}$ can be understood similarly. If $\boldsymbol{u} \leq \boldsymbol{v}$, we say that $\boldsymbol{u}$ **dominates** $\boldsymbol{v}$, or equivalently, $\boldsymbol{v}$ **is dominated by** $\boldsymbol{u}$.*

Definition 8. *Let $\mathcal{V}$ be a finite k-vector set. A vector $\boldsymbol{v} \in \mathcal{V}$ is **non-dominated** if it is dominated by no vector of $\mathcal{V} \setminus \{\boldsymbol{v}\}$. We use $\mathcal{V}^*$ to denote the set consisting of all the non-dominated vectors of $\mathcal{V}$ and call $\mathcal{V}^*$ the non-dominated set of $\mathcal{V}$. Moreover, we define $\mathcal{V}^- = \{\boldsymbol{v}^- : \boldsymbol{v} \in \mathcal{V}\}$, where $\boldsymbol{v}^-$ is the $(k-1)$-vector obtained from $\boldsymbol{v}$ by deleting the last component of $\boldsymbol{v}$, i.e., $\boldsymbol{v}^- = (v_1, v_2, \ldots, v_{k-1})$ if $\boldsymbol{v} = (v_1, v_2, \ldots, v_k)$.*

Definition 9. *The **lexicographical order** on $\mathcal{V}$, denoted by $\prec$, is defined in the following way: for two vectors $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_k)$ of $\mathcal{V}$, $\boldsymbol{u}$ is a **predecessor** of $\boldsymbol{v}$, i.e., $\boldsymbol{u} \prec \boldsymbol{v}$, if and only if there is an index $i \in \{1, 2, \ldots, k\}$ such that $u_i < v_i$ and $u_x = v_x$ for $x = 1, 2, \ldots, i-1$. The **lexicographic sequence** of $\mathcal{V}$ is the sequence $(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_z)$ of all the*

vectors of $\mathcal{V}$ such that $\boldsymbol{u}_1 \prec \boldsymbol{u}_2 \prec \cdots \prec \boldsymbol{u}_z$, where $z = |\mathcal{V}|$. For convenience, we use $(\mathcal{V}, \prec)$ to denote the ordered set and also the lexicographic sequence of $\mathcal{V}$.

Clearly, under the lexicographic order $\prec$ on $\mathcal{V}$, every two vectors in $\mathcal{V}$ are comparable. Then, $\prec$ is a total order on $\mathcal{V}$. As a consequence, $(\mathcal{V}, \prec)$ exists and is unique.

Definition 10. *According to Zou and Yuan (2020), a subset $\widetilde{\mathcal{V}} \subseteq \mathcal{V}$ is called a **simplified version** of $\mathcal{V}$ if $\mathcal{V}^* \subseteq \widetilde{\mathcal{V}}$, and every two consecutive vectors in the lexicographic sequence $(\widetilde{\mathcal{V}}, \prec)$ are non-dominated by each other.*

Some properties about the non-dominated sets, lexicographic sequences, and simplified versions of vector sets are established in Zou and Yuan. Among these properties, the ones which can be used in this paper are described in the following lemma.

Lemma 2. *Let $k \geq 2$ be a fixed integer. Let $\mathcal{U}$ and $\mathcal{V}$ be two finite k-vector sets. Then, we have the following five statements:*

(i) $(\mathcal{U} \cup \mathcal{V})^* \subseteq \mathcal{U}^* \cup \mathcal{V}^*$ *and* $(\mathcal{U} \cup \mathcal{V})^* = (\mathcal{U}^* \cup \mathcal{V}^*)^*$.

(ii) *If $\mathcal{U} \subseteq \mathcal{V}$, then $(\mathcal{U}, \prec)$ is a subsequence of $(\mathcal{V}, \prec)$.*

(iii) *If $(\mathcal{U}, \prec) = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_z)$ and $\boldsymbol{u}_0$ is a k-vector, then $(\mathcal{U} + \boldsymbol{u}_0, \prec) = (\boldsymbol{u}_1 + \boldsymbol{u}_0, \boldsymbol{u}_2 + \boldsymbol{u}_0, \ldots, \boldsymbol{u}_z + \boldsymbol{u}_0)$, where $\mathcal{U} + \boldsymbol{u}_0 = \{\boldsymbol{u} + \boldsymbol{u}_0 : \boldsymbol{u} \in \mathcal{U}\}$.*

(iv) *Suppose that both $(\mathcal{U}, \prec)$ and $(\mathcal{V}, \prec)$ are given in hand. Then, $(\mathcal{U} \cup \mathcal{V}, \prec)$ can be obtained from $(\mathcal{U}, \prec)$ and $(\mathcal{V}, \prec)$ in $O(|\mathcal{U}| + |\mathcal{V}|)$ time (by using the similar operations as that for merging two sorted sequences of numbers).*

(v) *Suppose that $(\mathcal{V}, \prec)$ is given in hand. Then, a simplified version $\widetilde{\mathcal{V}}$ of $\mathcal{V}$, together with $(\widetilde{\mathcal{V}}, \prec)$, can be obtained from $(\mathcal{V}, \prec)$ in $O(|\mathcal{V}|)$ time (by iteratively checking two consecutive vectors and deleting the dominated ones).*

The following lemma reveals a new property of the lexicographic sequences of simplified versions.

Lemma 3. *Let $\mathcal{V}$ be a k-vector set where $k \geq 2$, and suppose that $\mathcal{V}$ is a simplified version of itself. Then, $|\mathcal{V}^-| = |\mathcal{V}|$ and $(\mathcal{V}^-, \prec)$ can be obtained from $(\mathcal{V}, \prec)$ by deleting the last component of each vector of $\mathcal{V}$ directly.*

Proof. Suppose that $(\mathcal{V}, \prec) = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_z)$, where $\mathbf{v}_x = (v_{x,1}, v_{x,2}, \ldots, v_{x,k})$ for $x = 1, 2, \ldots, z$. We only need to show that $\mathbf{v}_1^-, \mathbf{v}_2^-, \ldots, \mathbf{v}_z^-$ are z distinct $(k-1)$-vectors.

If possible, let x' and x'' be two indices with $1 \leq x' < x'' \leq z$ such that $\mathbf{v}_{x'}^- = \mathbf{v}_{x''}^-$. Since $\mathbf{v}_{x'} \prec \mathbf{v}_{x'+1} \prec \cdots \prec \mathbf{v}_{x''}$, from the meaning of "$\prec$", we have $\mathbf{v}_{x'}^- = \mathbf{v}_{x'+1}^- = \cdots = \mathbf{v}_{x''}^-$. Now, $(\mathbf{v}_{x'}^-, v_{x',k}) = \mathbf{v}_{x'} \prec \mathbf{v}_{x'+1} = (\mathbf{v}_{x'+1}^-, v_{x'+1,k}) = (\mathbf{v}_{x'}^-, v_{x'+1,k})$. From the meaning of "$\prec$" again, we have $v_{x',k} < v_{x'+1,k}$. This means that $\mathbf{v}_{x'}$ dominates $\mathbf{v}_{x'+1}$ in $\mathcal{V}$, contradicting the assumption that $\mathcal{V}$ is a simplified version. Consequently, $\mathbf{v}_1^-, \mathbf{v}_2^-, \ldots, \mathbf{v}_z^-$ are z distinct $(k-1)$-vectors.

Finally, it is routine to see that $\mathcal{V}^- = \{\mathbf{v}_1^-, \mathbf{v}_2^-, \ldots, \mathbf{v}_z^-\}$ and $(\mathcal{V}^-, \prec) = (\mathbf{v}_1^-, \mathbf{v}_2^-, \ldots, \mathbf{v}_z^-)$. The lemma follows. □

The following lemma, which was first established by Kung et al. [19], will also be used in our research.

Lemma 4. *Let $\mathcal{V}$ be a finite k-vector set, where $k \geq 2$, and let $K = |\mathcal{V}|$. Then $\mathcal{V}^*$ can be determined in $O(H_k(K))$ time, where $H_k(K) = K \log K$ if $k = 2, 3$, and $H_k(K) = K(\log K)^{k-2}$ if $k \geq 4$.*

3. The Feasibility Problem

In this section, we study the feasibility problem $1||\mathbf{U}_w \leq \mathbf{Q}$. In Section 3.1, we present a dynamic programming algorithm for the problem. In Section 3.2, we present a dual-fully polynomial-time approximation scheme (DFPTAS) for this problem. Recall that $\mathbf{U}_w = (U_w^{(1)}, U_w^{(2)}, \ldots, U_w^{(m)})$ and $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_m)$.

If $Q_i \leq 0$ for some $i \in \{1, 2, \ldots, m\}$, then the problem $1||\mathbf{U}_w \leq \mathbf{Q}$ is either infeasible or all the jobs must be early in a feasible schedule. Then, the EDD-schedule $\sigma = (J_1, J_2, \ldots, J_n)$ can be used to directly check the feasibility of the problem. Hence, we assume in this section that $\mathbf{Q} > \mathbf{0}$, i.e.,

$$Q_i > 0 \text{ for all } i = 1, 2, \ldots, m.$$

3.1. A Dynamic Programming Algorithm

For each $j \in \{1, 2, \ldots, n\}$ and each nonnegative m-vector $\mathbf{v} = (v_1, v_2, \ldots, v_m)$ with $\mathbf{v} \leq \mathbf{Q}$, we consider the feasibility problem $1||\mathbf{U}_w \leq \mathbf{v}$ on the instance $\mathcal{J}_j = \{J_1, J_2, \ldots, J_j\}$. Let $C(j : \mathbf{v})$ be the minimum value of early-makespan of a feasible efficient schedule for the problem, i.e., $C(j : \mathbf{v}) = \min_\sigma \tau(\sigma)$, where σ runs over all the feasible efficient schedules for the problem $1||\mathbf{U}_w \leq \mathbf{v}$ on $\mathcal{J}_j$. In the case where there is no feasible schedule for the problem, we just define $C(j : \mathbf{v}) = +\infty$. Thus, the problem $1||\mathbf{U}_w \leq \mathbf{v}$ on $\mathcal{J}_j$ is feasible if and only if $C(j : \mathbf{v}) < +\infty$.

In an efficient schedule σ of $\mathcal{J}_j$ which assumes the early-makespan $C(j : \mathbf{v})$, i.e., $\tau(\sigma) = C(j : \mathbf{v})$, the job J_j may have the following two possibilities in σ:

– J_j may be a tardy job in σ. In this case, let σ' be the schedule of $\mathcal{J}_{j-1}$ obtained from σ by deleting J_j. Then, $\mathbf{U}_w(\sigma') = \mathbf{U}_w(\sigma) - \mathbf{w}_j$. This means that σ' is an efficient schedule of $\mathcal{J}_{j-1}$ assuming the early-makespan $C(j-1 : \mathbf{v} - \mathbf{w}_j)$. Clearly, σ' and σ have the same early-makespan. Then, we have $C(j : \mathbf{v}) = C(j-1 : \mathbf{v} - \mathbf{w}_j)$.

– J_j may be an early job in σ. Since σ is an efficient schedule, J_j must be the last early job in σ. Let σ'' be the schedule of $\mathcal{J}_{j-1}$ obtained from σ by deleting J_j. Then, $\mathbf{U}_w(\sigma'') = \mathbf{U}_w(\sigma)$. This means that σ'' is an efficient schedule of $\mathcal{J}_{j-1}$ assuming the early-makespan $C(j-1 : \mathbf{v})$. Clearly, $\tau(\sigma'') = \tau(\sigma) - p_j$. Then, we have $C(j : \mathbf{v}) = C(j-1 : \mathbf{v}) + p_j$. It should be noted that J_j can be early only if $C(j-1 : \mathbf{v}) + p_j \leq d_j$.

The intuition of the following Algorithm 1 can be described as follows. We always schedule the early jobs in their index order (EDD order) from time 0 and schedule the tardy jobs after the early jobs, so we mainly consider the processing of the early jobs. Our algorithm considers the jobs $J_1, J_2, \ldots, J_n$ one by one. Suppose that for some j with $1 \leq j \leq n$, the first $j-1$ jobs have been considered, and the last early job completes at time τ. Then, job J_j is considered with two possibilities: either $\tau + p_j \leq d_j$ and J_j is scheduled in the interval $[\tau, \tau + p_j]$ as an early job, or J_j is taken as a tardy job. The task of our algorithm is to record the effects of these operations and remove the infeasible objective vectors.

According to the above discussion, we obtain the following dynamic programming Algorithm DP1 for solving the problem $1||\mathbf{U}_w \leq \mathbf{Q}$.

In the above recursion function, the vector $\mathbf{v}$ has $O(Q_1 Q_2 \ldots Q_m)$ choices, and the iteration index j has $O(n)$ choices. Moreover, the calculation in each iteration needs a constant time since m is a fixed number. Then, we conclude the following result.

Theorem 1. *Algorithm DP1 solves the feasibility problem* $1||\boldsymbol{U}_w \leq \boldsymbol{Q}$ *in* $O(nQ_1Q_2 \ldots Q_m)$ *time.*

Algorithm 1: For solving the feasibility problem $1||\mathbf{U}_w \leq \mathbf{Q}$

Set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $d_1 \leq d_2 \leq \cdots \leq d_n$, set $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_m)$ with $\mathbf{Q} > \mathbf{0}$, and set

$$C(j:\mathbf{v}) = \begin{cases} 0, & \text{if } j = 0 \text{ and } \mathbf{v} \text{ is a nonnegative } m\text{-vector,} \\ +\infty, & \text{if } \mathbf{v} \text{ is not a nonnegative } m\text{-vector.} \end{cases}$$

for $j = 1, 2, \ldots, n$, **do**
  for $v = \boldsymbol{0}, \ldots, \boldsymbol{Q}$, **do**
    if $C(j-1:v) + p_j \leq d_j$, **then**
      $f(j:\mathbf{v}) = 0$
    end
    if $C(j-1:v) + p_j > d_j$, **then**
      $f(j:\mathbf{v}) = +\infty$
    end
    $$C(j:\mathbf{v}) = \min \begin{cases} C(j-1:\mathbf{v}-\mathbf{w}_j), \\ C(j-1:\mathbf{v}) + p_j + f(j:\mathbf{v}), \end{cases}$$
  end
end
if $C(n:\boldsymbol{Q}) = +\infty$, **then**
  the problem $1||\mathbf{U}_w \leq \mathbf{Q}$ is infeasible.
end
if $C(n:\boldsymbol{Q}) < +\infty$, **then**
  the problem $1||\mathbf{U}_w \leq \mathbf{Q}$ is feasible and a feasible efficient schedule for the problem can be obtained by backtracking.
end

3.2. A Numerical Example

Next, we establish a numerical example to explain how Algorithm DP1 works.

Instance I_2: The instance has three jobs as described in Table 3, and the threshold vector $\mathbf{Q} = (2, 3)$.

Table 3. Job Instance I_2 for Algorithm DP1.

Jobs	J_1	J_2	J_3
processing times p_j	3	2	4
due dates d_j	2	4	5
weights $w_j^{(1)}$	1	2	1
weights $w_j^{(2)}$	2	1	1

Then, we process Algorithm DP1 to solve the problem $1||\mathbf{U}_w \leq \mathbf{Q}$ on Instance I_2.

Initial condition:

$$C(0:\mathbf{v}) = \begin{cases} 0, & \text{if } \mathbf{v} \text{ is a nonnegative } m\text{-vector,} \\ +\infty, & \text{if } \mathbf{v} \text{ is not a nonnegative } m\text{-vector.} \end{cases}$$

The values of the recursion function (in Table 4):

Table 4. The values of the recursion function.

$\mathbf{v}$	$j=1$		$j=2$		$j=3$	
$\mathbf{v}=(0,0)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(0,1)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(0,2)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(0,3)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(1,0)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(1,1)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(1,2)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=0$	$f(2:\mathbf{v})=0$	$C(2:\mathbf{v})=2$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(1,3)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=0$	$f(2:\mathbf{v})=0$	$C(2:\mathbf{v})=2$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=2$
$\mathbf{v}=(2,0)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(2,1)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=+\infty$	$f(2:\mathbf{v})=+\infty$	$C(2:\mathbf{v})=+\infty$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(2,2)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=0$	$f(2:\mathbf{v})=0$	$C(2:\mathbf{v})=2$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=+\infty$
$\mathbf{v}=(2,3)$	$f(1:\mathbf{v})=+\infty$	$C(1:\mathbf{v})=0$	$f(2:\mathbf{v})=0$	$C(2:\mathbf{v})=2$	$f(3:\mathbf{v})=+\infty$	$C(3:\mathbf{v})=2$

Feasibility decision: Since $C(n:\mathbf{Q})=2<+\infty$, the problem $1||\mathbf{U}_w\leq\mathbf{Q}$ on Instance I_2 and both of schedule $\sigma_1=(J_2,J_1,J_3)$ and $\sigma_2=(J_2,J_3,J_1)$ are feasible.

3.3. A Dual-Fully Polynomial-Time Approximation Scheme

By rounding the multiple weights of jobs, Algorithm DP1 enables us to present a DFPTAS for the problem $1||\mathbf{U}_w\leq\mathbf{Q}$.

Let ε be a positive constant. From the job instance $\mathcal{J}=\{J_1,J_2,\ldots,J_n\}$ and the threshold vector $\mathbf{Q}>\mathbf{0}$, we obtain a *rounded instance* $\widetilde{\mathcal{J}}=\{\widetilde{J}_1,\widetilde{J}_2,\ldots,\widetilde{J}_n\}$ by setting, for $j=1,2,\ldots,n$,

$$\widetilde{p}_j=p_j \text{ and } \widetilde{d}_j=d_j \tag{3}$$

and, for $i=1,2,\ldots,m$,

$$\widetilde{w}_j^{(i)}=\begin{cases}\lceil\frac{2n}{\varepsilon}\rceil+n+1, & \text{if } w_j^{(i)}>Q_i,\\ \lceil\frac{2nw_j^{(i)}}{\varepsilon Q_i}\rceil, & \text{otherwise.}\end{cases} \tag{4}$$

Moreover, we define a new threshold vector $\widetilde{\mathbf{Q}}=(\widetilde{Q}_1,\widetilde{Q}_2,\ldots,\widetilde{Q}_m)$ by setting $\widetilde{Q}_i=\lceil\frac{2n}{\varepsilon}\rceil+n$ for $i=1,2,\ldots,m$, i.e.,

$$\widetilde{\mathbf{Q}}=(\lceil\frac{2n}{\varepsilon}\rceil+n,\lceil\frac{2n}{\varepsilon}\rceil+n,\ldots,\lceil\frac{2n}{\varepsilon}\rceil+n). \tag{5}$$

For a schedule σ of $\mathcal{J}$, we use $\widetilde{\sigma}$ to denote the schedule of $\widetilde{\mathcal{J}}$ which is obtained from σ by replacing each job J_j with the job $\widetilde{J}_j$. From (3), we know that σ is an efficient schedule of $\mathcal{J}$ if and only if $\widetilde{\sigma}$ is an efficient schedule of $\widetilde{\mathcal{J}}$.

The following two lemmas reveal some relations between the three feasibility problems $1||\mathbf{U}_w\leq\mathbf{Q}$ on $\mathcal{J}$, $1||\mathbf{U}_w\leq(1+\varepsilon)\mathbf{Q}$ on $\mathcal{J}$, and $1||\mathbf{U}_w\leq\widetilde{\mathbf{Q}}$ on $\widetilde{\mathcal{J}}$.

Lemma 5. *If $\widetilde{\sigma}$ is a feasible efficient schedule for the problem $1||\boldsymbol{U}_w\leq\widetilde{\boldsymbol{Q}}$ on $\widetilde{\mathcal{J}}$, then σ is a feasible efficient schedule for the problem $1||\boldsymbol{U}_w\leq(1+\varepsilon)\boldsymbol{Q}$ on $\mathcal{J}$.*

Proof. Since $\widetilde{\sigma}$ is a feasible efficient schedule for the problem $1||\mathbf{U}_w \leq \widetilde{\mathbf{Q}}$ on $\widetilde{\mathcal{J}}$, we have $\mathbf{U}_w(\widetilde{\sigma}) \leq \widetilde{\mathbf{Q}}$, i.e.,

$$U_w^{(i)}(\widetilde{\sigma}) = \sum_{j=1}^{n} \widetilde{w}_j^{(i)} U_j(\widetilde{\sigma}) \leq \widetilde{Q}_i = \lceil \frac{2n}{\varepsilon} \rceil + n, \ i = 1, 2, \ldots, m, \tag{6}$$

where the last equality in (6) follows from (5). Let $I = \{j \in \{1, 2, \ldots, n\} : U_j(\widetilde{\sigma}) = 1\}$. Then, $\sum_{j \in I} \widetilde{w}_j^{(i)} = U_w^{(i)}(\widetilde{\sigma})$. From (3) and from the relation of $\widetilde{\sigma}$ and σ, we have $I = \{j \in \{1, 2, \ldots, n\} : U_j(\sigma) = 1\}$ and $\sum_{j \in I} w_j^{(i)} = U_w^{(i)}(\sigma)$. From (4) and (6), for each $j \in I$ and each $i \in \{1, 2, \ldots, m\}$, we have $w_j^{(i)} \leq Q_i$ and $\widetilde{w}_j^{(i)} = \lceil \frac{2nw_j^{(i)}}{\varepsilon Q_i} \rceil$, and so,

$$w_j^{(i)} \leq \frac{\varepsilon Q_i}{2n} \widetilde{w}_j^{(i)}. \tag{7}$$

Consequently, from (6) and (7), for each $i \in \{1, 2, \ldots, m\}$, we have

$$\begin{aligned} U_w^{(i)}(\sigma) &= \textstyle\sum_{j=1}^{n} w_j^{(i)} U_j(\sigma) \\ &= \textstyle\sum_{j \in I} w_j^{(i)} \\ &\leq \frac{\varepsilon Q_i}{2n} \times \textstyle\sum_{j \in I} \widetilde{w}_j^{(i)} \\ &\leq \frac{\varepsilon Q_i}{2n} \times (\lceil \frac{2n}{\varepsilon} \rceil + n) \\ &\leq \frac{\varepsilon Q_i}{2n} \times (\frac{2n}{\varepsilon} + 1 + n) \\ &\leq (1 + \varepsilon) Q_i. \end{aligned}$$

It follows that σ is a feasible efficient schedule for the problem $1||\mathbf{U}_w \leq (1 + \varepsilon)\mathbf{Q}$ on $\mathcal{J}$. The lemma follows. □

Lemma 6. *If the problem $1||\boldsymbol{U}_w \leq \widetilde{\boldsymbol{Q}}$ on $\widetilde{\mathcal{J}}$ is infeasible, then the problem $1||\boldsymbol{U}_w \leq \boldsymbol{Q}$ on $\mathcal{J}$ is also infeasible.*

Proof. Suppose to the contrary that the problem $1||\mathbf{U}_w \leq \mathbf{Q}$ on $\mathcal{J}$ is feasible, and let σ be a feasible efficient schedule for this problem. As in Lemma 5, by setting $I = \{j \in \{1, 2, \ldots, n\} : U_j(\sigma) = 1\}$, we have $\sum_{j \in I} w_j^{(i)} = U_w^{(i)}(\sigma) \leq Q_i$ for $i \in \{1, 2, \ldots, m\}$. Moreover, for $j \in I$ and $i \in \{1, 2, \ldots, m\}$, we have $w_j^{(i)} \leq Q_i$ and $\widetilde{w}_j^{(i)} = \lceil \frac{2nw_j^{(i)}}{\varepsilon Q_i} \rceil \leq \frac{2nw_j^{(i)}}{\varepsilon Q_i} + 1$. Consequently, for each $i \in \{1, 2, \ldots, m\}$, we have

$$\begin{aligned} U_w^{(i)}(\widetilde{\sigma}) &= \textstyle\sum_{j=1}^{n} \widetilde{w}_j^{(i)} U_j(\widetilde{\sigma}) \\ &= \textstyle\sum_{j \in I} \widetilde{w}_j^{(i)} \\ &\leq \textstyle\sum_{j \in I} (\frac{2nw_j^{(i)}}{\varepsilon Q_i} + 1) \\ &\leq \frac{2n}{\varepsilon Q_i} \textstyle\sum_{j \in I} w_j^{(i)} + |I| \\ &\leq \frac{2n}{\varepsilon} + |I| \\ &\leq \lceil \frac{2n}{\varepsilon} \rceil + n \\ &= \widetilde{Q}_i. \end{aligned}$$

It follows that $\widetilde{\sigma}$ is a feasible efficient schedule for the problem $1||\mathbf{U}_w \leq \widetilde{\mathbf{Q}}$ on $\widetilde{\mathcal{J}}$. This contradicts the assumption that the problem $1||\mathbf{U}_w \leq \widetilde{\mathbf{Q}}$ on $\widetilde{\mathcal{J}}$ is infeasible. The lemma follows. □

Based on Lemmas 5 and 6, we present the following DFPTAS for solving the problem $1||\mathbf{U}_w \leq \mathbf{Q}$.

Algorithm DP1(ε): For the feasibility problem $1||\mathbf{U}_w \leq \mathbf{Q}$.
Input: A job instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $d_1 \leq d_2 \leq \cdots \leq d_n$, a positive threshold vector $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_m)$, and a constant $\varepsilon > 0$.
Step 1: Generate the rounded instance $\widetilde{\mathcal{J}} = \{\widetilde{J}_1, \widetilde{J}_2, \ldots, \widetilde{J}_n\}$ by using (3) and (4). Generate the rounded threshold vector $\widetilde{\mathbf{Q}}$ by using (5).
Step 2: Run Algorithm DP1 for the problem $1||\mathbf{U}_w \leq \widetilde{\mathbf{Q}}$ on instance $\widetilde{\mathcal{J}}$.
Feasibility decision: If DP1 tells us that $1||\mathbf{U}_w \leq \widetilde{\mathbf{Q}}$ on $\widetilde{\mathcal{J}}$ is infeasible, then the problem $1||\mathbf{U}_w \leq \mathbf{Q}$ on $\mathcal{J}$ is infeasible. If DP1 tells us that $1||\mathbf{U}_w \leq \widetilde{\mathbf{Q}}$ on $\widetilde{\mathcal{J}}$ is feasible and presents a feasible schedule $\widetilde{\sigma}$ for this problem, then the problem $1||\mathbf{U}_w \leq (1+\varepsilon)\mathbf{Q}$ on $\mathcal{J}$ is feasible and σ is a feasible schedule of the problem.

The correctness of Algorithm DP1(ε) is guaranteed by Theorem 1 and Lemmas 5 and 6. Note that $\widetilde{Q}_i = O(\frac{n}{\varepsilon})$ for each $i \in \{1, 2, \ldots, m\}$. From Theorem 1, the time complexity of Algorithm DP1(ε) is given by $O(n^{m+1}(\frac{1}{\varepsilon})^m)$. Consequently, we have the following result.

Theorem 2. *The feasibility problem $1||\boldsymbol{U}_w \leq \boldsymbol{Q}$ admits a dual-fully polynomial-time approximation scheme (DFPTAS) of running time $O(n^{m+1}(\frac{1}{\varepsilon})^m)$ for any given constant $\varepsilon > 0$.*

4. The Pareto-Scheduling Problem

In this section, we study the Pareto-scheduling problem $1||\mathcal{P}(\mathbf{U}_w)$. In Section 4.1, we present a pseudo-polynomial-time algorithm for problem $1||\widetilde{\mathcal{P}}(\mathbf{U})$ and $1||\mathcal{P}(\mathbf{U}_w)$ and provide the time complexity of the algorithm. In Section 4.2, we present a fully polynomial-time approximation scheme (DFPTAS) for the problem $1||\mathcal{P}(\mathbf{U}_w)$.

4.1. A Dynamic Programming Algorithm

We now consider the Pareto-scheduling problem $1||\mathcal{P}(\mathbf{U}_w)$ on the instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $d_1 \leq d_2 \leq \cdots \leq d_n$. Let $j \in \{1, 2, \ldots, n\}$ and let σ be an efficient schedule of the jobs of $\mathcal{J}_j = \{J_1, J_2, \ldots, J_j\}$. Recall that the early-makespan $\tau(\sigma)$ of σ is equal to the total processing time of the early jobs in σ. The $(m+1)$-vector

$$(\mathbf{U}_w(\sigma), \tau(\sigma)) = (U_w^{(1)}(\sigma), U_w^{(2)}(\sigma), \ldots, U_w^{(m)}(\sigma), \tau(\sigma))$$

is called a *state* of $\mathcal{J}_j$ corresponding to the schedule σ. In this case, we also say that σ is a $\mathbf{u}$-schedule, where $\mathbf{u} = (\mathbf{U}_w(\sigma), \tau(\sigma))$. Γ_j is used to denote the set of all states of $\mathcal{J}_j$ and is called the *state set* of $\mathcal{J}_j$. For convenience, we also define $\Gamma_0 = \{(0, 0, \ldots, 0)\} = \{\mathbf{0}\}$. For each state $\mathbf{u} \in \Gamma_0 \cup \Gamma_1 \cup \cdots \cup \Gamma_n$, we use $t(\mathbf{u})$ to denote the last component of $\mathbf{u}$. Hence, if $\mathbf{u} = (u_1, u_2, \ldots, u_m, \tau)$, then $\mathbf{u} = (\mathbf{u}^-, t(\mathbf{u}))$, where $\mathbf{u}^- = (u_1, u_2, \ldots, u_m)$ and $t(\mathbf{u}) = \tau$. Moreover, we define

$$\Gamma = \Gamma_n^- = \{\mathbf{u}^- : \mathbf{u} \in \Gamma_n\}.$$

Then, $\Gamma^* = \mathcal{P}(\mathcal{J})$ is the Pareto frontier, i.e., the set of the Pareto-optimal points, of the problem $1||\mathcal{P}(\mathbf{U}_w)$ on instance $\mathcal{J}$.

From Lemma 1 and from the meanings of Γ_j, $j = 0, 1, \ldots, n$, we have the following lemma, which provides a method to generate Γ_j from Γ_{j-1}, $j = 1, 2, \ldots, n$.

Lemma 7. *For $j = 1, 2, \ldots, n$, we have $\Gamma_j = \Gamma_j' \cup \Gamma_j''$, where*

$$\Gamma_j' = \{\boldsymbol{u} + (\boldsymbol{w}_j, 0) : \boldsymbol{u} \in \Gamma_{j-1}\}$$

and

$$\Gamma_j'' = \{\boldsymbol{u} + (\boldsymbol{0}, p_j) : \boldsymbol{u} \in \Gamma_{j-1} \text{ and } t(\boldsymbol{u}) + p_j \leq d_j\}.$$

Proof. Let $\Gamma_j^{(1)}$ be the set which consists of the states of Γ_j corresponding to the efficient schedules of $\mathcal{J}_j$ in which J_j is tardy. Let $\Gamma_j^{(2)}$ be the set which consists of the states of Γ_j corresponding to the efficient schedules of $\mathcal{J}_j$ in which J_j is early. Then, we have $\Gamma_j = \Gamma_j^{(1)} \cup \Gamma_j^{(2)}$. The remaining issue is to show that $\Gamma_j^{(1)} = \Gamma_j'$ and $\Gamma_j^{(2)} = \Gamma_j''$.

Given $\mathbf{u} \in \Gamma_{j-1}$, let σ be a $\mathbf{u}$-schedule of $\mathcal{J}_{j-1}$. Let σ' be the efficient schedule of $\mathcal{J}_j$ which is obtained from σ by adding J_j to be a tardy job. We have $\mathbf{U}_w(\sigma') = \mathbf{U}_w(\sigma) + \mathbf{w}_j$ and $\tau(\sigma') = \tau(\sigma)$. Then, the state corresponding to σ' is given by $\mathbf{u} + (\mathbf{w}_j, 0) \in \Gamma_j^{(1)}$. Consequently, we have

$$\Gamma_j' = \{\mathbf{u} + (\mathbf{w}_j, 0) : \mathbf{u} \in \Gamma_{j-1}\} \subseteq \Gamma_j^{(1)}. \tag{8}$$

Given $\mathbf{v} \in \Gamma_j^{(1)}$, let σ' be a $\mathbf{v}$-schedule of $\mathcal{J}_j$ in which J_j is tardy. Let σ be the efficient schedule of $\mathcal{J}_{j-1}$ which is obtained from σ' by deleting the tardy job J_j. Again, we have $\mathbf{U}_w(\sigma') = \mathbf{U}_w(\sigma) + \mathbf{w}_j$ and $\tau(\sigma') = \tau(\sigma)$. Then, the state corresponding to σ is given by $\mathbf{v} - (\mathbf{w}_j, 0) \in \Gamma_{j-1}$. Consequently, we have

$$\Gamma_j' = \{\mathbf{u} + (\mathbf{w}_j, 0) : \mathbf{u} \in \Gamma_{j-1}\} \supseteq \Gamma_j^{(1)}. \tag{9}$$

Given $\mathbf{u} \in \Gamma_{j-1}$ with $t(\mathbf{u}) + p_j \leq d_j$, let σ be a $\mathbf{u}$-schedule of $\mathcal{J}_{j-1}$. Let σ'' be the efficient schedule of $\mathcal{J}_j$ which is obtained from σ by adding J_j to be the last early job. The construction of σ'' does work since $\tau(\sigma'') = \tau(\sigma) + p_j = t(\mathbf{u}) + p_j \leq d_j$. We have $\mathbf{U}_w(\sigma'') = \mathbf{U}_w(\sigma)$. Then, the state corresponding to σ'' is given by $\mathbf{u} + (\mathbf{0}, p_i) \in \Gamma^{(2)}$. Consequently, we have

$$\Gamma_j'' = \{\mathbf{u} + (\mathbf{0}, p_j) : \mathbf{u} \in \Gamma_{j-1}, t(\mathbf{u}) + p_j \leq d_j\} \subseteq \Gamma_j^{(2)}. \tag{10}$$

Given $\mathbf{v} \in \Gamma_j^{(2)}$, let σ'' be a $\mathbf{v}$-schedule of $\mathcal{J}_j$ in which J_j is an early job. Then, J_j is the last early job in σ'', implying that $t(\mathbf{v}) = \tau(\sigma'') \leq d_j$. Let σ be the efficient schedule of $\mathcal{J}_{j-1}$ which is obtained from σ'' by deleting the last early job J_j. Again, we have $\mathbf{U}_w(\sigma'') = \mathbf{U}_w(\sigma)$ and $\tau(\sigma) + p_j = \tau(\sigma'') \leq d_j$. Then, the state corresponding to σ is given by $\mathbf{v} - (\mathbf{0}, p_j) \in \Gamma_{j-1}$. Consequently, we have

$$\Gamma_j'' = \{\mathbf{u} + (\mathbf{0}, p_j) : \mathbf{u} \in \Gamma_{j-1}, t(\mathbf{u}) + p_j \leq d_j\} \supseteq \Gamma_j^{(2)}. \tag{11}$$

From (8)–(11), we conclude that $\Gamma_j^{(1)} = \Gamma_j'$ and $\Gamma_j^{(2)} = \Gamma_j''$. The lemma follows. □

Let $T = \{0, 1, \ldots, P\}$ and $X_i = \{0, 1, \ldots, W^{(i)}\}$ for $i = 1, 2, \ldots, m$. Then, $|X_1 \times X_2 \times \cdots \times X_m \times T| = O(P \prod_{i=1}^m W^{(i)})$, and all the state sets $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$ are included in $X_1 \times X_2 \times \cdots \times X_m \times T$. A direct application of Lemma 7 may lead to a dynamic programming algorithm which generates $\Gamma_0, \Gamma_1, \ldots, \Gamma_n, \Gamma$ in $(nP \prod_{i=1}^m W^{(i)})$ time. Since a simplified version of Γ includes all the Pareto-optimal points, to save the time complexity, our algorithm will generate a sequence of state subsets $\widetilde{\Gamma}_0, \widetilde{\Gamma}_1, \ldots, \widetilde{\Gamma}_n, \widetilde{\Gamma}$, where $\widetilde{\Gamma}_i$ is a simplified version of Γ_i, $i = 0, 1, \ldots, n$, and $\widetilde{\Gamma}$ is a simplified version of Γ. With the help of lexicographic sequences, the time complexity can be reduced to $(n \prod_{i=1}^m W^{(i)})$.

The intuition of the following Algorithm 2 can be described as follows. We always schedule the early jobs in their index order (EDD order) from time 0 and schedule the tardy jobs after the early jobs, so we mainly consider the processing of the early jobs. Our algorithm considers the jobs $J_1, J_2, \ldots, J_n$ one by one. Suppose that for some j with $1 \leq j \leq n$, the first $j-1$ jobs have been considered, and the last early job completes at time τ. Then, job J_j is considered by two possibilities: either $\tau + p_j \leq d_j$ and J_j is scheduled in

the interval $[\tau, \tau + p_j]$ as an early job, or J_j is taken as a tardy job. The task of our algorithm is to record the effects of these operations and remove the dominated objective vectors.

According to the above discussion, we obtain the following dynamic programming Algorithm DP2 to generate the simplified versions $\widetilde{\Gamma}_0, \widetilde{\Gamma}_1, \ldots, \widetilde{\Gamma}_n, \widetilde{\Gamma}$.

Algorithm 2: For generating the simplified versions $\widetilde{\Gamma}_0, \widetilde{\Gamma}_1, \ldots, \widetilde{\Gamma}_n, \widetilde{\Gamma}$

Set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $d_1 \leq d_2 \leq \cdots \leq d_n$, set $\widetilde{\Gamma}_0 := \{(0, 0, \ldots, 0)\} = \{\mathbf{0}\}$, and set $(\widetilde{\Gamma}_0, \prec) := (\mathbf{0})$ be the lexicographic sequence of $\widetilde{\Gamma}_0$

for $j = 1, 2, \ldots, n$, **do**

set $\mathbf{u}_j = (\mathbf{w}_j, 0)$ and $\mathbf{v}_j = (\mathbf{0}, p_j)$.

Generate the two sets

$$\widetilde{\Gamma}'_j = \{\mathbf{u} + \mathbf{u}_j : \mathbf{u} \in \widetilde{\Gamma}_{j-1}\}$$

and

$$\widetilde{\Gamma}''_j = \{\mathbf{u} + \mathbf{v}_j : \mathbf{u} \in \widetilde{\Gamma}_{j-1} \text{ and } t(\mathbf{u}) + p_j \leq d_j\}.$$

Generate the two lexicographic sequences $(\widetilde{\Gamma}'_j, \prec)$ and $(\widetilde{\Gamma}''_j, \prec)$ from $(\widetilde{\Gamma}_{j-1}, \prec)$.

Generate $\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j$ and its lexicographic sequence $(\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j, \prec)$ from $(\widetilde{\Gamma}'_j, \prec)$ and $(\widetilde{\Gamma}''_j, \prec)$.

Generate a simplified version $\widetilde{\Gamma}_j$ of $\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j$, together with its lexicographic sequence $(\widetilde{\Gamma}_j, \prec)$, from $(\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j, \prec)$.

end

Determine $\widetilde{\Gamma}^-_n$ and generate $(\widetilde{\Gamma}^-_n, \prec)$ from $(\widetilde{\Gamma}_n, \prec)$. Then, generate a simplified version $\widetilde{\Gamma}$ of $\widetilde{\Gamma}^-_n$ from $(\widetilde{\Gamma}^-_n, \prec)$.

Output the sets $\widetilde{\Gamma}_0, \widetilde{\Gamma}_1, \ldots, \widetilde{\Gamma}_n, \widetilde{\Gamma}$. The efficient schedules corresponding to the states in $\widetilde{\Gamma}$ can be generated by backtracking.

Lemma 8. *Algorithm DP2 runs in $O(n \prod_{i=1}^{m} W^{(i)})$ time.*

Proof. Algorithm DP2 has totally n iterations for running Step 2. We consider the implementation of Step 2 at iteration j, where $j \in \{1, 2, \ldots, n\}$.

Step 2.1 runs in $O(|\widetilde{\Gamma}_{i-1}|)$ time clearly. In Step 2.2, $\mathbf{u}' + \mathbf{u}_j \prec \mathbf{u}'' + \mathbf{u}_j$ in $(\widetilde{\Gamma}'_j, \prec)$ if and only if $\mathbf{u}' \prec \mathbf{u}''$ in $(\widetilde{\Gamma}_{j-1}, \prec)$, and $\mathbf{v}' + \mathbf{v}_j \prec \mathbf{v}'' + \mathbf{v}_j$ in $(\widetilde{\Gamma}''_j, \prec)$ if and only if $\mathbf{v}' \prec \mathbf{v}''$ in $(\widetilde{\Gamma}_{j-1}, \prec)$. Thus, from Lemma 2(ii,iii), Step 2.2 runs in $O(|\widetilde{\Gamma}_{j-1}|)$ time. From Lemma 2(iv), Step 2.3 runs in $O(|\widetilde{\Gamma}'_j| + |\widetilde{\Gamma}''_j|)$ time, which is also $O(|\widetilde{\Gamma}_{j-1}|)$ time since $|\widetilde{\Gamma}'_j| = |\widetilde{\Gamma}_{j-1}|$ and $|\widetilde{\Gamma}''_j| \leq |\widetilde{\Gamma}_{j-1}|$. From Lemma 2(v), Step 2.4 runs in $O(|\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j|) = O(|\widetilde{\Gamma}_{j-1}|)$ time. Consequently, at iteration j, Step 2 runs in $O(|\widetilde{\Gamma}_{j-1}|)$ time. Since $\widetilde{\Gamma}_{j-1} \subseteq X_1 \times X_2 \times \cdots \times X_m \times T$ is a simplified version of itself, from Lemma 3, we have $|\widetilde{\Gamma}_{j-1}| = |\widetilde{\Gamma}^-_{j-1}| \leq \prod_{i=1}^{m} |X_i| = O(\prod_{i=1}^{m} W^{(i)})$, $j = 1, 2, \ldots, n$. In fact, by the same reason, we also have $|\widetilde{\Gamma}_n| = |\widetilde{\Gamma}^-_n| = O(\prod_{i=1}^{m} W^{(i)})$.

Summing up the running times $O(|\widetilde{\Gamma}_{j-1}|)$ for $j = 1, 2, \ldots, n$, we conclude that the running time of Algorithm DP2 for Step 2 at all the n iterations is $O(n \prod_{i=1}^{m} W^{(i)})$.

Finally, from Lemma 3, Step 4 takes $O(|\widetilde{\Gamma}_n|) = O(\prod_{i=1}^{m} W^{(i)})$ time to generate $\widetilde{\Gamma}^-_n$ and $(\widetilde{\Gamma}^-_n, \prec)$ from $(\widetilde{\Gamma}_n, \prec)$, and from Lemma 2(v), Step 4 takes $O(|\widetilde{\Gamma}^-_n|) = O(\prod_{i=1}^{m} W^{(i)})$ time to generate $\widetilde{\Gamma}$ from $(\widetilde{\Gamma}^-_n, \prec)$, which are dominated by the time complexity of Step 2. Consequently, the time complexity of Algorithm DP2 is given by $O(n \prod_{i=1}^{m} W^{(i)})$. The lemma follows. □

We show the correctness of Algorithm DP2 in the following lemma.

Lemma 9. *Let $\widetilde{\Gamma}_0, \widetilde{\Gamma}_1, \ldots, \widetilde{\Gamma}_n, \widetilde{\Gamma}$ be the sets returned by Algorithm DP2. Then, we have the following three statements.*

(i) *For each $j = 0, 1, \ldots, n$, $\widetilde{\Gamma}_j$ is a simplified version of Γ_j.*

(ii) *$\widetilde{\Gamma}$ is a simplified version of Γ.*

(iii) $|\widetilde{\Gamma}| = O(\prod_{i=1}^{m-1} W^{(i)})$.

Proof. We prove statement (i) by induction on j. From Algorithm DP2, we have $\widetilde{\Gamma}_0 = \{\mathbf{0}\} = \Gamma_0$. Hence, statement (i) holds for $j = 0$.

Inductively, we assume that $j \geq 1$ and statement (i) is correct up to index $j-1$, i.e., $\widetilde{\Gamma}_{j-1}$ is a simplified version of Γ_{j-1}. Then, Algorithm DP2 generates the two sets $\widetilde{\Gamma}'_j = \{\mathbf{u} + \mathbf{u}_j : \mathbf{u} \in \widetilde{\Gamma}_{j-1}\}$ and $\widetilde{\Gamma}''_j = \{\mathbf{u} + \mathbf{v}_j : \mathbf{u} \in \widetilde{\Gamma}_{j-1} \text{ and } t(\mathbf{u}) + p_j \leq d_j\}$. From Lemma 7, by setting $\Gamma'_j = \{\mathbf{u} + \mathbf{u}_j : \mathbf{u} \in \Gamma_{j-1}\}$ and $\Gamma''_j = \{\mathbf{u} + \mathbf{v}_j : \mathbf{u} \in \Gamma_{j-1} \text{ and } t(\mathbf{u}) + p_j \leq d_j\}$, we have $\Gamma_j = \Gamma'_j \cup \Gamma''_j$. Since $\widetilde{\Gamma}_{j-1}$ is a simplified version of Γ_{j-1}, we have $\Gamma^*_{j-1} \subseteq \widetilde{\Gamma}_{j-1}$. Then, the construction of $\widetilde{\Gamma}'_j$ and $\widetilde{\Gamma}''_j$ implies that $\Gamma'^*_j \subseteq \widetilde{\Gamma}'_j$ and $\Gamma''^*_j \subseteq \widetilde{\Gamma}''_j$. Since $\widetilde{\Gamma}_j$ is a simplified version of $\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j$, we further have $(\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j)^* \subseteq \widetilde{\Gamma}_j$. From Lemma 2(i), we have $\Gamma^*_j = (\Gamma'^*_j \cup \Gamma''^*_j)^* \subseteq (\widetilde{\Gamma}'_j \cup \widetilde{\Gamma}''_j)^* \subseteq \widetilde{\Gamma}_j$. Consequently, $\widetilde{\Gamma}_j$ is a simplified version of $\Gamma'_j \cup \Gamma''_j = \Gamma_j$.

From the above discussion, statement (i) follows from the principle of induction.

We next prove statement (ii). From statement (i), $\widetilde{\Gamma}_n$ is a simplified version of Γ_n. Then, we have $\Gamma^*_n \subseteq \widetilde{\Gamma}_n \subseteq \Gamma_n$ and $\Gamma^*_n = \widetilde{\Gamma}^*_n$. From the fact that $\Gamma = \Gamma^-_n$, we have that $\Gamma^* \subseteq (\Gamma^*_n)^- \subseteq \widetilde{\Gamma}^-_n \subseteq \Gamma^-_n = \Gamma$, implying that $(\widetilde{\Gamma}^-_n)^* = \Gamma^*$. Since $\widetilde{\Gamma}$ is a simplified version of $\widetilde{\Gamma}^-_n$, we have $\Gamma^* = (\widetilde{\Gamma}^-_n)^* \subseteq \widetilde{\Gamma} \subseteq \widetilde{\Gamma}^-_n \subseteq \Gamma^-_n = \Gamma$. It follows that $\widetilde{\Gamma}$ is a simplified version of Γ. Thus, statement (ii) holds.

Statement (iii) follows from Lemma 3 by noting that $\widetilde{\Gamma}$ is a simplified version of itself, and $\widetilde{\Gamma} \subseteq X_1 \times X_2 \times \cdots \times X_m$. This completes the proof. □

From Lemmas 8 and 9(ii), we have the following result.

Theorem 3. *The problem $1||\widetilde{\mathcal{P}}(\boldsymbol{U})$ is solvable in $O(n \prod_{i=1}^{m} W^{(i)})$ time.*

Let $K = \prod_{i=1}^{m-1} W^{(i)}$. From Lemma 9(iii), we have $|\widetilde{\Gamma}| = O(K)$. Note that $\widetilde{\Gamma}^* = \Gamma^* = \mathcal{P}(\mathcal{J})$. From Lemma 4, an additional $O(H_m(K))$-time is needed to obtain $\mathcal{P}(\mathcal{J})$ from $\widetilde{\Gamma}$, where $H_m(K) = K \log K$ if $m = 2, 3$, and $H_m(K) = K(\log K)^{m-2}$ if $m \geq 4$. In any case, we have $H_m(K) \leq K(\log K)^m$. Since $W^{(1)} \leq W^{(2)} \leq \cdots \leq W^{(m)}$, we have $\log K = \log \prod_{i=1}^{m-1} W^{(i)} \leq (m-1) \log W^{(m)}$. Thus, by using the fact that m is a fixed number, we can easily verify that

$$(\log K)^m = O((\log W^{(m)})^m) = O(W^{(m)}).$$

It follows that $H_m(K) \leq K(\log K)^m = O(\prod_{i=1}^{m} W^{(i)})$, which is dominated by the time complexity $O(n \prod_{i=1}^{m} W^{(i)})$ used for generating $\widetilde{\Gamma}$. Thus, we have the following corollary.

Corollary 1. *The Pareto-scheduling problem $1||\mathcal{P}(\boldsymbol{U}_w)$ is solvable in $O(n \prod_{i=1}^{m} W^{(i)})$ time.*

4.2. A Numerical Example

Next, we establish a numerical example to explain how Algorithm DP2 works.

Instance I_3: The instance has three jobs as described in the following Table 5.

Table 5. Job Instance I_3 for Algorithm DP2.

Jobs	J_1	J_2	J_3
processing times p_j	4	3	4
due dates d_j	3	5	7
weights $w_j^{(1)}$	5	2	3
weights $w_j^{(2)}$	2	3	4
weights $w_j^{(3)}$	3	3	2

Then, we process Algorithm DP2 to solve the problem $1||\mathcal{P}(\mathbf{U}_w)$ on instance I_3.
Initialition: Set $\widetilde{\Gamma}_0 = \{(0,0,\ldots,0)\} = \{\mathbf{0}\}$ and $(\widetilde{\Gamma}_0, \prec) = (\mathbf{0})$.
Generating $\widetilde{\Gamma}_j$ from $\widetilde{\Gamma}_{j-1}$:
When $j = 1$, $\mathbf{u}_1 = (5,2,3,0)$ and $\mathbf{v}_1 = (0,0,0,4)$.

$$\widetilde{\Gamma}_1' = \{(5,2,3,0)\} \text{ and } \widetilde{\Gamma}_1'' = \emptyset.$$

Then, $(\widetilde{\Gamma}_1, \prec) = \{(5,2,3,0)\}$
When $j = 2$, $\mathbf{u}_2 = (2,3,3,0)$ and $\mathbf{v}_2 = (0,0,0,3)$.

$$\widetilde{\Gamma}_2' = \{(7,5,6,0)\} \text{ and } \widetilde{\Gamma}_2'' = \{(5,2,3,3)\}.$$

Then, $(\widetilde{\Gamma}_2, \prec) = \{(5,2,3,3),(7,5,6,0)\}$
When $j = 3$, $\mathbf{u}_3 = (3,4,2,0)$ and $\mathbf{v}_3 = (0,0,0,4)$.

$$\widetilde{\Gamma}_3' = \{(8,6,5,3),(10,9,8,0)\} \text{ and } \widetilde{\Gamma}_3'' = \{(5,2,3,7),(7,5,6,4)\}.$$

Then, $(\widetilde{\Gamma}_3, \prec) = \{(5,2,3,7),(7,5,6,4),(8,6,5,3),(10,9,8,0)\}$
Output: $\widetilde{\Gamma}_n^- = \{(5,2,3),(7,5,6),(8,6,5),(10,9,8)\}$ and the simplified version $\widetilde{\Gamma} = \{(5,2,3)\}$. The efficient schedule corresponding to the state in $\widetilde{\Gamma}$ is $\sigma = \{J_2, J_3, J_1\}$.

4.3. A Fully Polynomial-Time Approximation Scheme

Based on Algorithm DP2, by using the trimming technique, we now present an FPTAS for the Pareto-scheduling problem $1||\mathcal{P}(\mathbf{U}_w)$. Recall that $X_i = \{0,1,\ldots,W^{(i)}\}$, $i = 1,2,\ldots,m$. The intuition of our FPTAS can be described as follows.

Given a constant $\varepsilon > 0$, we partition each interval $[0, W^{(i)}]$ into $O(\frac{n}{\varepsilon}\log \mathrm{W}^{(i)})$ subintervals, $i = 1,2,\ldots,m$. Then, the space $[0,W^{(1)}] \times [0,W^{(2)}] \times \cdots \times [0,W^{(m)}]$ (corresponding to $X_1 \times X_2 \times \cdots \times X_m$) is naturally partitioned into $O(\frac{n^m}{\varepsilon^m}\prod_{i=1}^m \log \mathrm{W}^{(i)})$ small boxes (of dimension m). In our algorithm, we generate a sequence of sets $\overline{\Gamma}_0, \overline{\Gamma}_1, \ldots, \overline{\Gamma}_n$ such that, for each $j \in \{1,2,\ldots,n\}$, we have $\overline{\Gamma}_j \subseteq \Gamma_j$, and for each small box, at most one vector $\mathbf{u} \in \overline{\Gamma}_j$ can make $\mathbf{u}^-$ in this box. The sequence $\overline{\Gamma}_0, \overline{\Gamma}_1, \ldots, \overline{\Gamma}_n$ is generated iteratively by n iterations. Initially, we set $\overline{\Gamma}_0 = \{\mathbf{0}\}$. At iteration $j \in \{1,2,\ldots,n\}$, we generated $\overline{\Gamma}_j$ from $\overline{\Gamma}_{j-1}$ in the following way: First generate two subsets $\overline{\Gamma}_j'$ and $\overline{\Gamma}_j''$ of Γ_j by the similar way as Algorithm DP2, and then $\overline{\Gamma}_j$ is obtained from $\overline{\Gamma}_j' \cup \overline{\Gamma}_j''$ by eliminating all the vectors $\mathbf{v}$ if there is another vector $\mathbf{u} \in \overline{\Gamma}_j' \cup \overline{\Gamma}_j''$ such that $\mathbf{u}^-$ and $\mathbf{v}^-$ are in the same box and $t(\mathbf{u}) \leq t(\mathbf{v})$. Finally, $\overline{\Gamma}_n^-$ will act as the $(1+\varepsilon)$-approximation of the Pareto frontier $\mathcal{P}(\mathcal{J})$. Our algorithm borrows the approach used in Li and Yuan [20].

Algorithm DP2(ε): An FPTAS for the problem $1||\mathcal{P}(\mathbf{U}_w)$.
Input: A job instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $d_1 \leq d_2 \leq \cdots \leq d_n$ and a constant $\varepsilon > 0$.
Step 0: For $i = 1,2,\ldots,m$, partition the interval $[0, \mathrm{W}^{(i)}]$ into $\xi_i + 1$ subintervals $I_0^{(i)}, I_1^{(i)}, \ldots, I_{\xi_i}^{(i)}$ such that $I_0^{(i)} = [0,1)$, $I_r^{(i)} = [(1+\varepsilon)^{\frac{r-1}{n}}, (1+\varepsilon)^{\frac{r}{n}})$ for $1 \leq r \leq \xi_i - 1$, and $I_{\xi_i}^{(i)} =$

$[(1+\varepsilon)^{\frac{\xi_i-1}{n}}, \mathrm{W}^{(i)}]$, where $\xi_i = n\lceil \log_{1+\varepsilon} \mathrm{W}^{(i)} \rceil = O(\frac{n}{\varepsilon} \log \mathrm{W}^{(i)})$. This results in $\prod_{i=1}^{m} \xi_i$ small m-dimensional boxes

$$I_{r_1}^{(1)} \times I_{r_2}^{(2)} \times \cdots \times I_{r_m}^{(m)},\ r_i \in \{0, 1, \ldots, \xi_i\} \text{ for } i = 1, 2, \ldots, m.$$

Step 1: Set $\overline{\Gamma}_0 := \{\mathbf{0}\}$, and set $j := 1$. Go to Step 2.
Step 2: Let $\mathbf{u}_j = (\mathbf{w}_j, 0)$ and $\mathbf{v}_j = (\mathbf{0}, p_j)$. Do the following.
(2.1) Generate the two sets

$$\overline{\Gamma}'_j = \{\mathbf{u} + \mathbf{u}_j : \mathbf{u} \in \overline{\Gamma}_{j-1}\}$$

and

$$\overline{\Gamma}''_j = \{\mathbf{u} + \mathbf{v}_j : \mathbf{u} \in \overline{\Gamma}_{j-1} \text{ and } t(\mathbf{u}) + p_j \le d_j\}.$$

Then, determine $\overline{\Gamma}'_j \cup \overline{\Gamma}''_j$.

(2.2) For any two states $\mathbf{u}$ and $\mathbf{v}$ of $\overline{\Gamma}'_j \cup \overline{\Gamma}''_j$ such that $\mathbf{u}^-$ and $\mathbf{v}^-$ fall within the same box $I_{r_1}^{(1)} \times I_{r_2}^{(2)} \times \cdots \times I_{r_m}^{(m)}$ with $t(\mathbf{u}) \le t(\mathbf{v})$, eliminate the state $\mathbf{v}$ from $\overline{\Gamma}'_j \cup \overline{\Gamma}''_j$.

(2.3) Set $\overline{\Gamma}_j$ to be the set of all the states of $\overline{\Gamma}'_j \cup \overline{\Gamma}''_j$ which are not eliminated in Step 2.2.
Step 3: If $j = n$, go to Step 4. Otherwise, if $j < n$, then set $j := j + 1$ and go back to Step 2.
Step 4: Set $\overline{\Gamma} := \overline{\Gamma}_n^-$. For each point $(u_1, u_2, \ldots, u_m) \in \overline{\Gamma}$, obtain the corresponding schedule by backtracking.

The small boxes $I_{r_1}^{(1)} \times I_{r_2}^{(2)} \times \cdots \times I_{r_m}^{(m)}$ with $r_i \in \{0, 1, \ldots, \xi_i\}$ for $i = 1, 2, \ldots, m$, play an important role. The following property of these boxes can be easily observed from their construction in Step 0 of Algorithm DP2(ε).

Lemma 10. *Let $I_{r_1}^{(1)} \times I_{r_2}^{(2)} \times \cdots \times I_{r_m}^{(m)}$ be a small box constructed in Step 0 of Algorithm DP2(ε). For every two states $\boldsymbol{u}$ and $\boldsymbol{v}$ with $\boldsymbol{u}^-$ and $\boldsymbol{v}^-$ falling within this small box, we have $\boldsymbol{u}^- \le (1+\varepsilon)^{\frac{1}{n}} \boldsymbol{v}^-$.*

Given the instance $\mathcal{J}$ and the constant $\varepsilon > 0$, let $\widetilde{\Gamma}_0, \widetilde{\Gamma}_1, \ldots, \widetilde{\Gamma}_n$ be the state sets generated by Algorithm DP2, and let $\overline{\Gamma}_0, \overline{\Gamma}_1, \ldots, \overline{\Gamma}_n$ be the state sets generated by Algorithm DP2(ε). Then, we have the following lemma.

Lemma 11. *For each state $\boldsymbol{u} \in \widetilde{\Gamma}_j$, there exists a state $\overline{\boldsymbol{u}} \in \overline{\Gamma}_j$ such that $t(\overline{\boldsymbol{u}}) \le t(\boldsymbol{u})$ and $\overline{\boldsymbol{u}}^- \le (1+\varepsilon)^{\frac{j}{n}} \boldsymbol{u}^-$, i.e., $\overline{u}_i \le (1+\varepsilon)^{\frac{j}{n}} u_i$ for $i = 1, 2, \ldots, m$.*

Proof. We prove the lemma by induction over $j = 1, 2, \ldots, n$. Note that there are two states $(0, w_1^{(1)}, w_1^{(2)}, \ldots, w_1^{(m)})$ and $(p_1, 0, 0, \ldots, 0)$ generated in $\widetilde{\Gamma}_1$ and $\overline{\Gamma}_1$ in the first iteration, and no one is eliminated in Algorithm DP2(ε). So, we have $\widetilde{\Gamma}_1 = \overline{\Gamma}_1$. Then, the result holds for $j = 1$.

Inductively, suppose in the following that $j \in \{2, 3, \ldots, n\}$, and the lemma holds for the indices up to $j - 1$. Let $\mathbf{u} \in \widetilde{\Gamma}_j$. Then, either $\mathbf{u} \in \widetilde{\Gamma}'_j$ or $\mathbf{u} \in \widetilde{\Gamma}''_j$.

If $\mathbf{u} \in \widetilde{\Gamma}'_j$, then there is a state $\mathbf{x} \in \widetilde{\Gamma}_{j-1}$ such that $\mathbf{u} = \mathbf{x} + \mathbf{u}_j$, where $\mathbf{u}_j = (\mathbf{w}_j, 0)$. By the induction hypothesis, there is a state $\overline{\mathbf{x}} \in \overline{\Gamma}_{j-1}$ such that $t(\overline{\mathbf{x}}) \le t(\mathbf{x})$ and $\overline{\mathbf{x}}^- \le (1+\varepsilon)^{\frac{j-1}{n}} \mathbf{x}^-$. Note that $\overline{\mathbf{x}} + \mathbf{u}_j \in \overline{\Gamma}'_j$. From Algorithm DP2($\varepsilon$), there is a state $\overline{\mathbf{u}} \in \overline{\Gamma}_j$ such that $t(\overline{\mathbf{u}}) \le t(\overline{\mathbf{x}} + \mathbf{u}_j)$ and such that $(\overline{\mathbf{x}} + \mathbf{u}_j)^-$ and $\overline{\mathbf{u}}^-$ fall within a common small box $I_{r_1}^{(1)} \times I_{r_2}^{(2)} \times \cdots \times I_{r_m}^{(m)}$. From Lemma 10, we have $\overline{\mathbf{u}}^- \le (1+\varepsilon)^{\frac{1}{n}} (\overline{\mathbf{x}} + \mathbf{u}_j)^-$. By collecting the above information together, we have $\overline{\mathbf{u}} \in \overline{\Gamma}_j$, $t(\overline{\mathbf{u}}) \le t(\overline{\mathbf{x}} + \mathbf{u}_j) \le t(\mathbf{x} + \mathbf{u}_j) = t(\mathbf{u})$, and $\overline{\mathbf{u}}^- \le (1+\varepsilon)^{\frac{1}{n}} (\overline{\mathbf{x}} + \mathbf{u}_j)^- \le (1+\varepsilon)^{\frac{1}{n}} ((1+\varepsilon)^{\frac{j-1}{n}} \mathbf{x} + \mathbf{u}_j)^- \le (1+\varepsilon)^{\frac{j}{n}} (\mathbf{x} + \mathbf{u}_j)^- = (1+\varepsilon)^{\frac{j}{n}} \mathbf{u}^-$, as required.

If $\mathbf{u} \in \widetilde{\Gamma}''_j$, then there is a state $\mathbf{x} \in \widetilde{\Gamma}_{j-1}$ such that $t(\mathbf{x}) + p_j \leq d_j$ and $\mathbf{u} = \mathbf{x} + \mathbf{v}_j$, where $\mathbf{v}_j = (\mathbf{0}, p_j)$. By the induction hypothesis, there is a state $\overline{\mathbf{x}} \in \overline{\Gamma}_{j-1}$ such that $t(\overline{\mathbf{x}}) \leq t(\mathbf{x})$ and $\overline{\mathbf{x}}^- \leq (1+\varepsilon)^{\frac{j-1}{n}}\mathbf{x}^-$. Since $t(\overline{\mathbf{x}}) + p_j \leq t(\mathbf{x}) + p_j \leq d_j$, we have $\overline{\mathbf{x}} + \mathbf{v}_j \in \overline{\Gamma}''_j$. From Algorithm DP2($\varepsilon$), there is a state $\overline{\mathbf{u}} \in \overline{\Gamma}_j$ such that $t(\overline{\mathbf{u}}) \leq t(\overline{\mathbf{x}} + \mathbf{v}_j)$ and such that $(\overline{\mathbf{x}} + \mathbf{v}_j)^- = \overline{\mathbf{x}}^-$ and $\overline{\mathbf{u}}^-$ fall within a common small box $I^{(1)}_{r_1} \times I^{(2)}_{r_2} \times \cdots \times I^{(m)}_{r_m}$. From Lemma 10, we have $\overline{\mathbf{u}}^- \leq (1+\varepsilon)^{\frac{1}{n}}\overline{\mathbf{x}}^-$. By collecting the above information together, we have $\overline{\mathbf{u}} \in \overline{\Gamma}_j$, $t(\overline{\mathbf{u}}) \leq t(\overline{\mathbf{x}} + \mathbf{v}_j) \leq t(\mathbf{x} + \mathbf{v}_j) = t(\mathbf{u})$, and $\overline{\mathbf{u}}^- \leq (1+\varepsilon)^{\frac{1}{n}}\overline{\mathbf{x}}^- \leq (1+\varepsilon)^{\frac{1}{n}} \cdot (1+\varepsilon)^{\frac{j-1}{n}}\mathbf{x}^- \leq (1+\varepsilon)^{\frac{j}{n}}\mathbf{x}^- = (1+\varepsilon)^{\frac{j}{n}}\mathbf{u}^-$, as required.

Thus, the lemma holds for the index j. This completes the proof. □

Theorem 4. *The Pareto-scheduling problem* $1||\mathcal{P}(\boldsymbol{U}_w)$ *admits a fully polynomial-time approximation scheme (FPTAS) of running time* $O(\frac{n^{m+1}}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$ *for any given constant* $\varepsilon > 0$.

Proof. Recall that $\Gamma = \Gamma_n^-$, $\widetilde{\Gamma}_n$ is a simplified version of Γ_n, and $\mathcal{P}(\mathcal{J})$ is the Pareto frontier of the problem $1||\mathcal{P}(\mathbf{U}_w)$ on instance $\mathcal{J}$. Then, $\Gamma^* = \mathcal{P}(\mathcal{J})$. From Lemma 9(ii), $\widetilde{\Gamma}$, which is a simplified version of $\widetilde{\Gamma}_n^-$ returned by Algorithm DP2, is also a simplified version of Γ. This means that

$$\mathcal{P}(\mathcal{J}) = \Gamma^* \subseteq \widetilde{\Gamma} \subseteq \widetilde{\Gamma}_n^-. \tag{12}$$

We first show that the point set $\overline{\Gamma}$, which is returned by Algorithm DP2(ε), is a $(1+\varepsilon)$-approximation of $\mathcal{P}(\mathcal{J})$. For each point $\mathbf{v} \in \mathcal{P}(\mathcal{J})$, from (12), we have $(v_1, v_2, \ldots, v_m) \in \widetilde{\Gamma}_n^-$. Thus, there is a non-negative number t such that $(\mathbf{v}, t) \in \widetilde{\Gamma}_n$. From Lemma 11, there is a state $\mathbf{u} \in \overline{\Gamma}_n$ such that $\mathbf{u}^- \leq (1+\varepsilon)^{\frac{n}{n}}(\mathbf{v}, t)^- = (1+\varepsilon)\mathbf{v}$. Since $\overline{\Gamma} = \overline{\Gamma}_n^-$, we have $\mathbf{u}^- \in \overline{\Gamma}$. Thus, each point in $\mathcal{P}(\mathcal{J})$ has a $(1+\varepsilon)$-approximation in $\overline{\Gamma}$. Consequently, $\overline{\Gamma}$ is a $(1+\varepsilon)$-approximation of $\mathcal{P}(\mathcal{J})$.

Next, we discuss the time complexity of Algorithm DP2(ε). In Step 0, it takes $O(\prod_{i=1}^{m} n\lceil\log_{1+\varepsilon} W^{(i)}\rceil) = O(\frac{n^m}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$ time to partition the intervals and form the small boxes. Step 1 needs $O(1)$ time for initialization. Note that at the end of each iteration (for performing Step 2), at most one state is left in each small box. Then, there are at most $O(\frac{n^m}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$ distinct states in $\overline{\Gamma}_j$ for $j = 1, 2, \ldots, n$. Moreover, out of each state in $\overline{\Gamma}_{j-1}$, at most one new state is generated in $\overline{\Gamma}_j$. Therefore, the construction of $\overline{\Gamma}'_j$, $\overline{\Gamma}''_j$, and $\overline{\Gamma}'_j \cup \overline{\Gamma}''_j$ in Steps 2.1 and 2.2 requires at most $O(\frac{n^m}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$ time, which is also the time needed for the elimination procedure in Step 2.3. Step 4 requires at most $O(\frac{n^m}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$ time and Step 2 is repeated n times. Then, the overall running time of Algorithm DP2(ε) is given by $O(\frac{n^{m+1}}{\varepsilon^m}\prod_{i=1}^{m}\log W^{(i)})$. □

5. Conclusions

In this paper, we study the feasibility problem $1||\mathbf{U}_w \leq \mathbf{Q}$, the Pareto-scheduling problem $1||\mathcal{P}(\mathbf{U}_w)$, and the weakened version $1||\widetilde{\mathcal{P}}(\mathbf{U}_w)$, where m is a fixed number. We present a pseudo-polynomial algorithm and a DFPTAS through rounding technique for the problem $1||\mathbf{U}_w \leq \mathbf{Q}$. Then, we provide a pseudo-polynomial algorithm for the problems $1||\mathcal{P}(\mathbf{U}_w)$ and $1||\widetilde{\mathcal{P}}(\mathbf{U}_w)$ together with an FPTAS for the problem $1||\mathcal{P}(\mathbf{U}_w)$.

Scheduling of jobs with multiple weights can be applied in various production lines or machine environments. In a further study, we will suggest extending the problems studied in our paper and Guo et al. [11] to other scheduling models, such as parallel-machine scheduling, serial-batch scheduling, parallel-batch scheduling, and so on.

Author Contributions: Conceptualization, S.G. ; methodology, S.G., H.L. and H.Z.; validation, S.G., H.L. and H.Z.; writing—original draft preparation, S.G.; software, S.G., H.L. and H.Z.; visualization, S.G., H.L. and H.Z.; writing—review and editing, S.G., H.L. and H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Moore, J.M. An *n* job, one machine sequencing algorithm for minimizing the number of late jobs. *Manag. Sci.* **1968**, *15*, 102–109. [CrossRef]
2. Lawler, E.L.; Moore, J.M. A functional equation and its applications to resource allocation and sequencing problems. *Manag. Sci.* **1969**, *16*, 77–84. [CrossRef]
3. Karp, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*; IBM Thomas J. Watson Research Center: Yorktown Heights, NY, USA; Plenum, NY, USA, 1972; pp. 85–103.
4. Sahni, S. Algorithms for scheduling independent tasks. *J. Assoc. Comput. Mach.* **1976**, *23*, 116–127. [CrossRef]
5. Lee, C.Y.; Vairaktarakis, G. Complexity of single machine hierarchical scheduling: A survey. In *Complexity in Numerical Optimization*; Pardalos, P.M., Ed.; World Scientific: River Edge, NJ, USA, 1993; pp. 269–298.
6. Agnetis, A.; Billaut, J.C.; Gawiejnowicz, S.; Pacciarelli, D.; Soukhal, A. *Multiagent Scheduling: Models and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2014.
7. Cheng, T.C.E.; Ng, C.T.; Yuan, J.J. Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theor. Comput. Sci.* **2006**, *362*, 273–281. [CrossRef]
8. Zou, J.; Yuan, J.J. Single-machine scheduling with maintenance activities and rejection. *Discret. Optim.* **2020**, *38*, 100609. [CrossRef]
9. Guo, S.E.; Lu, L.F.; Yuan, J.J.; Ng, C.; Cheng, T.C.E. Pareto-scheduling with double-weighted jobs to minimize the weighted number of tardy jobs and total weighted late work. *Nav. Res. Logist.* **2022**, *69*, 816–837. [CrossRef]
10. Chen, R.B.; He, R.Y.; Yuan, J.J. Preemptive scheduling to minimize total weighted late work and weighted number of tardy jobs. *Comput. Ind. Eng.* **2022**, *167*, 107969. [CrossRef]
11. Guo, S.E.; Geng, Z.C.; Yuan, J.J. Single-machine Pareto-scheduling with multiple weighting vectors for minimizing the total weighted late works. *J. Ind. Manag. Optim.* **2023**, *19*, 456–471. [CrossRef]
12. Zhang, Y.; Geng, Z.C.; Yuan, J.J. Two-Agent Pareto-Scheduling of Minimizing Total Weighted Completion Time and Total Weighted Late Work. *Mathematics* **2020**, *8*, 2070. [CrossRef]
13. Masadeh, R.; Alsharman, N.; Sharieh, A.; Mahafzah, B.A.; Abdulrahman, A. Task scheduling on cloud computing based on sea lion optimization algorithm. *Int. J. Web Inf. Syst.* **2021**, *17*, 99–116. [CrossRef]
14. 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.* **2021**, *68*, 378–393. [CrossRef]
15. Kim, Y.J.; Kim, B.S. Population-Based Meta-Heuristic Algorithms for Integrated Batch Manufacturing and Delivery Scheduling Problem. *Mathematics* **2022**, *10*, 4127. [CrossRef]
16. Valouxis, C.; Gogos, C.; Dimitsas, A.; Potikas, P.; Vittas, A. A Hybrid Exact–Local Search Approach for One-Machine Scheduling with Time-Dependent Capacity. *Algorithms* **2022**, *15*, 450. [CrossRef]
17. Wang, Y.C.; Wang, S.H.; Wang, J.B. Resource Allocation Scheduling with Position-Dependent Weights and Generalized Earliness–Tardiness Cost. *Mathematics* **2023**, *11*, 222. [CrossRef]
18. Yuan, J.J. Unary NP-hardness of minimizing the number of tardy jobs with deadlines. *J. Sched.* **2017**, *20*, 211–218. [CrossRef]
19. Kung, H.T.; Luccio, F.; Preparata, F.P. On finding the maxima of a set of vectors. *J. Assoc. Comput. Mach.* **1975**, *22*, 469–476. [CrossRef]
20. 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]