Journal of Industrial and Management Optimization
Vol. 21, No. 8, August 2025, pp. 5574-5593
doi:10.3934/jimo.2025105

UNBOUNDED PARALLEL-BATCH SCHEDULING TO MINIMIZE THE TOTAL WEIGHTED TARDY SPAN

Shuaiyi Chen[✉1], Rubing Chen[✉*1], Jinjiang Yuan[✉1], C. T. Ng[✉2] and T. C. E. Cheng[✉2]

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

[2]Logistics Research Centre, Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hong Kong SAR, China

(Communicated by Bertrand M.T. Lin)

Abstract. For the unbounded parallel-batch scheduling on a single machine, we consider a new criterion, the total weighted tardy span of jobs, where the tardy span of a job is its completion time if it is tardy and is 0 if it is early. From the literature, the general problem is NP-hard and can be solved in pseudo-polynomial time. When the jobs are released at time 0, we show that the problem is NP-hard and present three pseudo-polynomial-time algorithms and computational experiments that show the performances of the three algorithms. Moreover, for the two cases, (i) the jobs are released at different times and the number of different processing times is a constant and (ii) the jobs are released at time 0 and have a common due date, we present polynomial-time algorithms.

1. **Introduction.** Lee et al.[19] first introduced the parallel-batch scheduling model, where a parallel-batch machine can handle up to b jobs simultaneously in a common batch. The parallel-batch scheduling model has many applications in real-world scenarios, for instance, semiconductor manufacturing (Arroyo and Leung [1]), dynamic mould manufacturing (Liu et al. [22]), porcelain production (Ou [25]), hospital sterilization services (Ozturk et al. [26]), multi-hybrid cell manufacturing system (Yilmaz et al. [29]), compact strip production (Zhang et al. [32]), blockchain system (Zhao et al. [31]), autoclave molding manufacturing (Zheng et al. [36]), and cloud manufacturing system (Zhang et al. [33]).

Due to factors such as cost considerations, frequency of use, and other constraints, batch processing is generally a bottleneck in the whole systems among these real-world scenarios. Consequently, effective utilization of batch processing is essential for both managers and researchers, and minimizing losses caused by tardy orders

2020 *Mathematics Subject Classification.* 90B35.

Key words and phrases. Unbounded parallel-batch scheduling, single machine, total weighted tardy span, NP-hard, algorithms.

The second author is supported by the China Postdoctoral Science Foundation under grant number 2023M743210, the National Natural Science Foundation of China under grant numbers 12271491, and the Foundation of Henan Educational Committee under grant number 25A110014. The third author is supported by the National Natural Science Foundation of China under grant numbers 12371318.

*Corresponding author: Rubing Chen.

or work is critical. Furthermore, the loss of each tardy order or work is usually assessed based on the relationship between its completion time and the promised completion time (due date). For instance, in the literature, the maximum tardiness, the total weighted tardiness, or the total weighted number of tardy jobs are classical criteria reflecting the losses and have been extensively studied. It is important to highlight the significance of evaluating the processing of tardy orders. The tardy span is an intuitive criterion, which is a new criterion introduced by Chen et al. [4]. The tardy span of an early order is 0, and the tardy span of a tardy order is its completion time. That is, the tardy span represents the waiting time of the order in the processing procedure.

In real-world business operations, the tardy span of orders serves as a critical performance metric with far-reaching implications. From the customer's perspective, it directly influences customer's satisfaction. The smaller the tardy span, the higher the customer satisfaction. When an order is delayed, it can disrupt the customer's production schedules, business operations, or personal plans. For instance, in a supply chain where a manufacturer relies on timely delivery of raw materials from suppliers, a tardy order for these materials can result in production halts and significant financial losses for the manufacturer. Prolonged completion times for tardy orders can undermine customer trust, potentially leading to customers switching to competitors over time. On the other hand, for businesses themselves, the tardy span is closely linked to cost management. Extended delays often incur additional expenses. By monitoring and reducing the tardy span of orders, businesses can enhance their cost control measures and improve overall profitability. Thus, in terms of market competitiveness, an organization's ability to efficiently manage tardy orders serves as a key differentiating factor, and companies that swiftly address issues related to delayed orders and minimize the tardy span criterion gain a distinct advantage.

The above discussion motivates us to study the unbounded parallel-batch scheduling for minimizing the total weighted tardy span on a single machine.

Problem formulation. A parallel-batch machine M can process up to b jobs simultaneously, where b is the batch capacity. If $b < +\infty$, then machine M is called bounded, and if $b = +\infty$, then machine M is called unbounded. In this paper, we only consider the case that M is an unbounded parallel-batch machine. Suppose that we have n jobs $J_1, J_2, \ldots, J_n$ to be processed on machine M without preemption. For each J_j, four integral parameters are given: a release time $r_j \geq 0$, a processing time $p_j > 0$, a weight $w_j \geq 0$, and a due date $d_j \geq 0$. Let $P = \sum_{j=1}^{n} p_j$.

A schedule π of the n jobs is denoted by $\pi = (B_1(\pi), B_2(\pi), \ldots, B_m(\pi))$, where $B_i(\pi)$ is the i-th batch in π. Given a schedule π, the following notations will be used in this paper:

- $S(B_i(\pi))$: the starting time of $B_i(\pi)$ in π.
- $p(B_i(\pi)) = \max\{p_j : J_j \in B_i(\pi)\}$: the processing time of $B_i(\pi)$ in π.
- $C(B_i(\pi))$: the completion time of $B_i(\pi)$ in π.
- $C_j(\pi) = C(B_i(\pi))$: the completion time of J_j in π if $J_j \in B_i(\pi)$.
- $U_j(\pi)$: the tardiness indicator of J_j in π, where $U_j(\pi) = 1$ if $C_j(\pi) > d_j$, and $U_j(\pi) = 0$ otherwise. Particularly, J_j is called tardy in π if $U_j(\pi) = 1$, and early in π if $U_j(\pi) = 0$.
- $C_j(\pi)U_j(\pi)$: the tardy span of J_j in π, where $C_j(\pi)U_j(\pi) = 0$ if J_j is early, and $C_j(\pi)U_j(\pi) = C_j(\pi)$ if J_j is tardy.
- $C_{\max}(\pi) = \max\{C_j(\pi) : 1 \leq j \leq n\}$: the makespan of π.

• $\sum w_jC_j(\pi)U_j(\pi) = \sum_{j=1}^{n} w_jC_j(\pi)U_j(\pi)$: the total weighted tardy span of π.

More scheduling criteria, such as $L_{\max}$, $\sum w_jC_j$, $\sum w_jT_j$, $\sum w_jU_j$, and $\sum f_j$, can be found in Brucker et al. [2]. In this paper, a scheduling problem on an unbounded parallel-batch machine for minimizing a scheduling criterion f is denoted by $1|r_j, \mathrm{PB}, b = +\infty|f$, where $\mathrm{PB}, b = +\infty$ indicates that the machine M is an unbounded parallel-batch machine.

Notice that Deng et al. [7] showed that $1|r_j, \mathrm{PB}, b = +\infty|\sum w_jC_j$ is NP-hard, and Liu et al. [23] presented an $O(n^4Q^3)$-time algorithm for $1|r_j, \mathrm{PB}, b = +\infty|\sum f_j$ problem, where f_j is an arbitrary regular criterion of J_j and $Q = \max\{r_j : 1 \leq j \leq n\} + P$. When $d_j = 0$ for each job J_j, $1|r_j, \mathrm{PB}, b = +\infty|\sum w_jC_jU_j$ problem is reduced to $1|r_j, \mathrm{PB}, b = +\infty|\sum w_jC_j$ problem. Moreover, $1|r_j, \mathrm{PB}, b = +\infty|\sum w_jC_jU_j$ problem is a special version of $1|r_j, \mathrm{PB}, b = +\infty|\sum f_j$ problem. Thus, the next conclusion holds.

Theorem 1.1. *$1|r_j, PB, b = +\infty|\sum w_jC_jU_j$ problem is NP-hard and solved in $O(n^4Q^3)$ time.*

From Theorem 1.1, the following cases of $1|r_j, \mathrm{PB}, b = +\infty|\sum w_jC_jU_j$ are studied in this paper:

$$1|\mathrm{PB}, b = +\infty|\sum w_jC_jU_j,$$

$$1|\mathrm{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j,$$

$$1|r_j, \mathrm{PB}, b = +\infty, \text{fix-p(k)}|\sum w_jC_jU_j,$$

where fix-p(k) means that the n jobs have k different processing times and k is a constant.

Literature review. In the literature, there are currently no studies that consider the total weighted tardy span. Next, we review some most relevant results, and please refer to Table 1, where for $1|r_j, \mathrm{PB}, b = +\infty, \text{family-jobs}|C_{\max}$ problem, jobs from different families cannot be processed in the same batch, and for $1|\mathrm{PB}, b = +\infty|^{\#}(\sum w_jC_j, C_{\max})$, $W = \sum_{j=1}^{n} w_j$, $L = C_{\max}(\sigma)$, and σ is an optimal schedule for $1|\mathrm{PB}, b = +\infty|\sum w_jC_j$.

More research on the unbounded paralle-batch scheduling model is well-studied, including the studies on drop-line jobs ([9]), deteriorating jobs ([20]), family jobs ([21]), and job rejection ([24],[34]), and MapReduce jobs ([28]). On the other hand, the combination of the bicriteria scheduling model and the unbounded parallel-batch scheduling model is also a popular topic in the literature. More research on this topic can be found in [3], [12], [13], [16], [17], [27], and [35].

Organization. In Section 2, we study $1|\mathrm{PB}, b = +\infty|\sum w_jC_jU_j$. For the problem, we present the NP-hardness proof and three pseudo-polynomial-time algorithms. In Section 3, we study $1|\mathrm{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j$ and $1|r_j, \mathrm{PB}, b = +\infty, \text{fix-p}|\sum w_jC_jU_j$. For the two problems, we present polynomial-time algorithms. Conclusions are presented in Section 4.

2. $1|\mathbf{PB}, b = +\infty|\sum w_jC_jU_j$ problem.

We now consider $1|\mathrm{PB}, b = +\infty|\sum w_jC_jU_j$. In Section 2.1, we show its NP-hardness, and in Sections 2.2 and 2.3, we present three pseudo-polynomial-time algorithms and computational experiments to compare the performance of the three algorithms.

TABLE 1. Some most relevant results in the literature

Problems	Time complexity	References
$1\|\text{PB}, b=+\infty\|\sum w_jC_j$	$O(n\log n)$	Brucker et al. [2]
$1\|\text{PB}, b=+\infty\|L_{\max}$	$O(n^2)$	Brucker et al. [2]
$1\|\text{PB}, b=+\infty\|\sum U_j$	$O(n^3)$	Brucker et al. [2]
$1\|\text{PB}, b=+\infty\|\sum f_j$	$O(n^2P)$	Brucker et al. [2]
$1\|\text{PB}, b=+\infty\|\sum w_jT_j$	NP-hard	Brucker et al. [2]
$1\|\text{PB}, b=+\infty\|\sum w_jU_j$	NP-hard	Brucker et al. [2]
$1\|\text{PB}, b=+\infty, \text{chains}\|f,\ f\in\{\sum C_j, C_{\max}\}$	strongly NP-hard	Cheng et al. [6]
$1\|\text{PB}, b=+\infty\|\sum T_j$	NP-hard.	Liu et al. [23]
$1\|r_j, \text{PB}, b=+\infty\|L_{\max}$	NP-hard	Cheng et al. [5]
$1\|r_j, \text{PB}, b=+\infty\|\sum w_jC_j$	NP-hard	Deng et al. [7]
$1\|r_j, \text{PB}, b=+\infty, \text{fix-p(k)}\|\sum w_jC_j$	polynomial	Deng et al. [7]
$1\|r_j, \text{PB}, b=+\infty\|C_{\max}$	$O(n^2)$	Lee et al. [18]
$1\|r_j, \text{PB}, b=+\infty\|\sum f_j$	$O(n^4Q^3)$	Liu et al. [23]
$1\|r_j, \text{PB}, b=+\infty, \text{family-jobs}\|C_{\max}$	strongly NP-hard	Yuan et al. [30]
$1\|\text{PB}, b=+\infty\|^{\#}(\sum w_jU_j, C_{\max})$	$O(n^2W^2P)$	Fan et al. [8]
$1\|\text{PB}, b=+\infty\|^{\#}(\sum w_jC_j, C_{\max})$	$O(n^2L^2)$	Fan et al. [8]
$1\|\text{PB}, b=+\infty\|^{\#}(C_{\max}, f_{\max})$	$O(n^3\log P)$ $O(n^4)$ $O(n^3)$	He et al. [14] Geng et al. [11] He et al. [15]

2.1. **NP-hardness verification.** We verify the NP-hardness of $1|\text{PB}, b=+\infty|\sum w_jC_jU_j$ by a polynomial transformation from the famous Partition problem. By [10], Partition problem is NP-complete. Recall that in an instance of partition problem, we are given $t+1$ positive integers $a_1, a_2, \ldots, a_t$ and A with $a_1+a_2+\cdots+a_t=2A$, and the decision is whether there is a partition (E_1, E_2) of $\{1,2,\ldots,t\}$ such that $\sum_{j\in E_1} a_j = \sum_{j\in E_2} a_j = A$? The vector $(a_1, a_2, \ldots, a_t; A)$ is called an instance of Partition problem.

Theorem 2.1. *$1|PB, b=+\infty|\sum w_jC_jU_j$ is NP-hard.*

Proof. Given an instance $(a_1, a_2, \ldots, a_t; A)$ of Partition problem, we use the following way to construct a scheduling instance $\mathcal{J}$ of $1|\text{PB}, b=+\infty|\sum w_jC_jU_j$:

- $\mathcal{J}$ has totally $2t$ jobs: $J_1, J_2, \ldots, J_{2t}$.
- For each J_j with $j=1,2,\ldots,t$, the parameters (p_j, w_j, d_j) are given by

$$p_j = 2jtA^2 + a_j,\ \ w_j = \frac{a_j}{A+(j+1)(j+2)tA^2},\ \ d_j = A + j(j+1)tA^2.$$

- For each J_{t+j} with $j=1,2,\ldots,t$, the parameters $(p_{t+j}, w_{t+j}, d_{t+j})$ are given by

$$p_{t+j} = 2jtA^2,\ \ w_{t+j} = A+1,\ \ d_{t+j} = A + j(j+1)tA^2.$$

- The decision is whether there is a schedule π of $\mathcal{J}$ with $\sum_{j=1}^{2t} w_jC_j(\pi)U_j(\pi) \le A$.

Clearly, $\mathcal{J}$ can be obtained from $(a_1, a_2, \ldots, a_t; A)$ in a polynomial time. To verify the above construction is a polynomial transformation, we need to show that $(a_1, a_2, \ldots, a_t; A)$ has a solution if and only if the required schedule π of $\mathcal{J}$ exists.

Given the assumption that $(a_1, a_2, \ldots, a_t; A)$ has a solution, $\{1, 2, \ldots, t\}$ can be partitioned into two subsets E_1 and E_2 such that $\sum_{j\in E_1} a_j = \sum_{j\in E_2} a_j = A$. By the symmetry of E_1 and E_2, we convent that $t \in E_1$. We use the following way to construct a schedule π of $\mathcal{J}$: π has a total of t batches $B_1(\pi), B_2(\pi), \ldots, B_t(\pi)$, that is, $\pi = (B_1(\pi), B_2(\pi), \ldots, B_t(\pi))$. In detail, for each index $j \in \{1, 2, \ldots, t\}$, we have $J_{t+j} \in B_j(\pi)$; moreover, $J_j \in B_j(\pi)$ if $j \in E_1$, and $J_j \in B_{j+1}(\pi)$ if $j \in E_2$. Formally, for each index $j \in \{1, 2, \ldots, t\}$, we define two job sets $\mathcal{J}'_j$ and $\mathcal{J}''_j$, where $\mathcal{J}'_j = \{J_j\}$ and $\mathcal{J}''_j = \emptyset$ if $j \in E_1$, and $\mathcal{J}'_j = \emptyset$ and $\mathcal{J}''_j = \{J_j\}$ if $j \in E_2$, implying that $\mathcal{J}'_j \cup \mathcal{J}''_j = \{J_j\}$. Then, $B_1(\pi) = \{J_{t+1}\} \cup \mathcal{J}'_1$, and for $j \in \{2, \ldots, t\}$, $B_j(\pi) = \{J_{t+j}\} \cup \mathcal{J}'_j \cup \mathcal{J}''_{j-1}$, implying that each batch of π has at most three jobs.

For each index $j \in \{1, 2, \ldots, t\}$, $p(B_j(\pi)) = 2jtA^2 + a_j$ if $j \in E_1$ and $p(B_j(\pi)) = 2jtA^2$ if $j \in E_2$, and so, $\sum_{i=1}^{j} 2itA^2 \le C(B_j(\pi)) \le \sum_{i=1}^{j} 2itA^2 + \sum_{i\in E_1} a_i = j(j+1)tA^2 + A$. For each job J_{t+j} with $j \in \{1, 2, \ldots, t\}$, $C_{t+j}(\pi) = C(B_j(\pi)) \le d_{t+j}$, implying that it is early in π. For each job J_j with $j \in E_1$, $C_j(\pi) = C(B_j(\pi)) \le d_j$, implying that it is early in π. For each job J_j with $j \in E_2$, $C_j(\pi) = C(B_{j+1}(\pi)) \ge \sum_{i=1}^{j+1} 2itA^2 > d_j$, implying that it is tardy in π. Thus, $\sum_{j=1}^{2t} w_j C_j(\pi) U_j(\pi) = \sum_{j\in E_2} w_j C(B_{j+1}(\pi)) \le \sum_{j\in E_2} w_j((j+1)(j+2)tA^2 + A) = \sum_{j\in E_2} a_j = A$. Therefore, π is a required schedule of $\mathcal{J}$.

Conversely, assume that there is a required schedule π of $\mathcal{J}$, that is $\sum_{j=1}^{2t} w_j C_j(\pi) U_j(\pi) \le A$. For our purpose, let π be an optimal schedule.

Claim 1. For $j \in \{1, 2, \ldots, t\}$, $J_{t+j} \in B_j(\pi)$ and $U_{t+j}(\pi) = 0$.

Since $w_{t+j} = A + 1$ for $j \in \{1, 2, \ldots, t\}$, we see that $U_{t+j}(\pi) = 0$. Since $d_{t+1} = A + 2tA^2$ and $p_j \ge 2tA^2$ for $j \in \{1, 2, \ldots, 2t\}$, we have $J_{t+1} \in B_1(\pi)$. Assume that $J_{t+j} \in B_j(\pi)$ for $j \in \{1, 2, \ldots, i-1\}$. Clearly, $2 \le i \le t$. Now we consider job J_{t+i}. If $J_{t+i} \in B_s(\pi)$ and $s \le i - 1$, then $C_{t+s}(\pi) = C(B_s(\pi)) \ge \sum_{j=1}^{s} p_{t+j} + 2tA^2 = \sum_{j=1}^{s} 2jtA^2 + 2tA^2 > d_{t+s}$, contradicting to $U_{t+s}(\pi) = 0$. If $J_{t+i} \in B_k(\pi)$ and $k > i$, then $C_{t+i}(\pi) = C(B_k(\pi)) \ge \sum_{j=1}^{i} p_{t+j} + 2tA^2 = \sum_{j=1}^{i} 2jtA^2 + 2tA^2 > d_{t+i}$, contradicting to $U_{t+i}(\pi) = 0$. Hence, $J_{t+i} \in B_i(\pi)$. Claim 1 holds.

Claim 2. For $j \in \{1, 2, \ldots, t\}$, $J_j \in B_k(\pi)$ for some $k \ge j$.

Suppose to the contrary that there exists some job $J_j \in B_k(\pi)$ with $j \in \{1, 2, \ldots, t\}$ and $1 \le k < j$. From Claim 1, $J_{t+k} \in B_j(\pi)$ and $U_{t+k}(\pi) = 0$. Then, from $p_j > p_{t+k}$, we have $C_{t+k}(\pi) = C(B_k(\pi)) \ge \sum_{i=1}^{k-1} p_{t+i} + p_j = \sum_{i=1}^{k-1} 2itA^2 + 2jtA^2 > d_{t+k}$, a contradiction to $U_{t+k}(\pi) = 0$. Claim 2 holds.

Claim 3. For $j \in \{1, 2, \ldots, t\}$, if $U_j(\pi) = 0$, then $J_j \in B_j(\pi)$.

For $j \in \{1, 2, \ldots, t\}$, suppose that $U_j(\pi) = 0$. From Claim 2, $J_j \in B_k(\pi)$ with $k \ge j$. If $k > j$, then from Claim 1 and $p_j < p_{t+k}$, we have $C_j(\pi) = C(B_k(\pi)) \ge \sum_{i=1}^{k} p_{t+i} > d_j$, contradicting to $U_j(\pi) = 0$. Claim 3 holds.

Claim 4. For $j \in \{1, 2, \ldots, t\}$, if $U_j(\pi) = 1$, then $J_j \in B_{j+1}(\pi)$.

For $j \in \{1, 2, \ldots, t\}$, suppose that $U_j(\pi) = 1$. From Claim 2, $J_j \in B_k(\pi)$ with $k \ge j$. From Claim 1, $J_{t+j} \in B_j(\pi)$ and $U_{t+j}(\pi) = 0$. If $k = j$, then $C_{t+j}(\pi) = C(B_j(\pi)) = C_j(\pi) > d_j = d_{t+j}$, contradicting to $U_{t+j}(\pi) = 0$. If $k > j + 1$, then let π' be the schedule obtained from π by shifting J_j to $B_{j+1}(\pi)$. Since $p_j < p_{t+j+1}$, we have $C_j(\pi) > C_j(\pi') = C(B_{j+1}(\pi')) \ge \sum_{i=1}^{j+1} p_{t+i} = \sum_{i=1}^{j+1} 2itA^2 > d_j$, and so, $U_j(\pi') = U_j(\pi) = 1$. Moreover, for each job J_i with $i \ne j$, $C_i(\pi') \le C_i(\pi)$, implying that $U_i(\pi') \le U_i(\pi)$. Hence, $\sum_{j=1}^{2t} w_j C_j(\pi') U_j(\pi') < \sum_{j=1}^{2t} w_j C_j(\pi) U_j(\pi) \le A$, contradicting to the optimality of π. Claim 4 holds.

Let $E_1 = \{j : U_j(\pi) = 0,\ j \in \{1,2,\ldots,t\}\}$ and $E_2 = \{j : U_j(\pi) = 1,\ j \in \{1,2,\ldots,t\}\}$. From Claims 3 and 4, $E_1 = \{j : J_j \in B_j(\pi),\ j \in \{1,2,\ldots,t\}\}$ and $E_2 = \{j : J_j \in B_{j+1}(\pi),\ j \in \{1,2,\ldots,t\}\}$. From Claims 1-4, (E_1, E_2) is a partition of $\{1,2,\ldots,t\}$. For $j \in \{1,2,\ldots,t\} \cap E_1$, $p(B_j(\pi)) = 2jtA^2 + a_j$. For $j \in \{1,2,\ldots,t\} \cap E_2$, $p(B_j(\pi)) = 2jtA^2$. From Claim 1, $U_{2t}(\pi) = 0$. Thus, $C_{2t}(\pi) = C(B_t(\pi)) = \sum_{j=1}^{t} p(B_j(\pi)) = t^2(t+1)A^2 + \sum_{j\in E_1} a_j \le d_{2t} = A + t^2(t+1)A^2$. This implies $\sum_{j\in E_1} a_j \le A$. For $j \in E_2$, $w_jC_j(\pi) \ge \frac{a_j \sum_{i=1}^{j+1} 2itA^2}{A+(j+1)(j+2)tA^2} = a_j - \frac{a_j}{1+(j+1)(j+2)tA}$. From Claims 1-4, $\sum_{j=1}^{2t} w_jC_j(\pi)U_j(\pi) = \sum_{j\in E_2} w_jC_j(\pi) \ge \sum_{j\in E_2}(a_j - \frac{a_j}{1+(j+1)(j+2)tA}) > \sum_{j\in E_2} a_j - 1$. Since $\sum_{j=1}^{2t} w_jC_j(\pi)U_j(\pi) \le A$, $\sum_{j\in E_2} a_j \le A$. Since $\sum_{j\in E_1} a_j + \sum_{j\in E_2} a_j = 2A$, $\sum_{j\in E_1} a_j = \sum_{j\in E_2} a_j = A$. Therefore, (E_1, E_2) is a solution of $(a_1, a_2, \ldots, a_t; A)$. The result follows. □

2.2. **Three pseudo-polynomial-time algorithms.** For a schedule $\pi = (B_1(\pi), B_2(\pi), \ldots, B_m(\pi))$, we call it an SPT-batch schedule if for $J_i \in B_x(\pi)$ and $J_j \in B_y(\pi)$, $p_i < p_j$ leads to $x \le y$. From Brucker et al. [2], there is an optimal schedule for $1|\mathrm{PB}, b = +\infty|f$ problem such that it is an SPT-batch schedule, where f is an arbitrary regular criterion. Thus, the following lemma holds for $1|\mathrm{PB}, b = +\infty|\sum w_jC_jU_j$ problem.

Lemma 2.2. *$1|PB, b = +\infty|\sum w_jC_jU_j$ problem has an optimal and SPT-batch schedule.*

From Lemma 2.2, we focus on the SPT-batch schedules. The n jobs are sorted by the SPT rule, that is $p_1 \le p_2 \le \cdots \le p_n$. Recall that $P = \sum_{j=1}^{n} p_j$. Lemma 2.2 enables us to design three algorithms for $1|\mathrm{PB}, b = +\infty|\sum w_jC_jU_j$.

First dynamic programming algorithm.

Given $j \in \{0,1,\ldots,n\}$, $k \in \{j, j+1, \ldots, n\}$, and $t \in \{0,1,\ldots,P\}$, let $F_j^{(1)}(t,k)$ be the minimum value of the total weighted tardy span of jobs $J_1, J_2, \ldots, J_j$ among all the SPT-batch schedules of jobs $J_1, J_2, \ldots, J_n$ satisfied the following conditions: (i) the maximum index of jobs in the batch contained J_j is k, and so, $J_j, J_{j+1}, \ldots, J_k$ are in the same batch, and (ii) t is the completion time of the batch containing J_j.

For an SPT-batch schedule assuming the value $F_j^{(1)}(t,k)$, there are two possibilities for J_j:

- J_j and J_{j-1} are in the same batch, and the processing time of the batch is p_k. If J_j is early, that is, $t \le d_j$, then $F_j^{(1)}(t,k) = F_{j-1}^{(1)}(t,k)$; if J_j is tardy, $F_j^{(1)}(t,k) = F_{j-1}^{(1)}(t,k) + w_jt$.
- J_j and J_{j-1} are in two different batches. Then, J_{j-1} is the job of the largest index in the previous batch, and the current batch consists of jobs $J_j, J_{j+1}, \ldots, J_k$ with processing time p_k. If J_j is early, that is, $t \le d_j$, then $F_j^{(1)}(t,k) = F_{j-1}^{(1)}(t - p_k, j-1)$; if J_j is tardy, $F_j^{(1)}(t,k) = F_{j-1}^{(1)}(t-p_k, j-1) + w_jt$.

The above discussion enables us to design the first dynamic programming algorithm.

Algorithm 2.1.

Preprocessing. The n jobs are sorted by the SPT rule, that is, $p_1 \le p_2 \le \cdots \le p_n$.

Initialization. $F_0^{(1)}(0,0) = 0$, and $F_0^{(1)}(t,k) = +\infty$ for $(t,k) \ne (0,0)$, $k \in \{0,1,\ldots,n\}$, and $t \in \{0,1,\ldots,P\}$.

Recursion. For $j = 1, 2, \ldots, n$, $k = j, j+1, \ldots, n$, and $t = 0, 1, \ldots, P$,

$$F_j^{(1)}(t,k) = \min\{X_1(j,t,k),\ X_2(j,t,k)\},$$

where

$$X_1(j,t,k) = \begin{cases} F_{j-1}^{(1)}(t,k), & \text{if } k = j \text{ and } t \le d_j, \\ F_{j-1}^{(1)}(t,k) + w_j t, & \text{if } k = j \text{ and } t > d_j, \end{cases}$$

$$X_2(j,t,k) = \begin{cases} F_{j-1}^{(1)}(t-p_k, j-1), & \text{if } k \ge j \text{ and } t \le d_j, \\ F_{j-1}^{(1)}(t-p_k, j-1) + w_j t, & \text{if } k \ge j \text{ and } t > d_j. \end{cases}$$

Optimal value. The optimal value is $\min\{F_n^{(1)}(t,n) : 0 \le t \le P\}$, and the corresponding SPT-batch schedule can be generated by backtracking.

Theorem 2.3. *Algorithm 2.1 solves* $1|PB, b = +\infty|\sum w_j C_j U_j$ *in* $O(n^2 P)$ *time.*

Proof. The correctness of Algorithm 2.1 is guaranteed by the above discussion. For the time complexity, the preprocessing procedure takes $O(n \log n)$ time to sort the n jobs. The initialization procedure takes $O(nP)$ time. There are $O(n^2 P)$ choices for (j,k,t), and for each choice of (j,k,t), the recursion procedure takes a constant time to calculate $F_j^{(1)}(t,k)$. Hence, the recursion procedure takes a total of $O(n^2 P)$ time. Optimal value and the corresponding SPT-batch schedule can be determined in $O(P)$ time. Therefore, the total running time of Algorithm 2.1 is $O(n^2 P)$. The result follows. □

In the following, we use an instance to show the implementation of Algorithm 2.1.

Example 2.1: Consider an instance of $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$ in which there are six jobs $J_1, J_2, \ldots, J_6$, and the corresponding parameters are given by $(p_1, p_2, p_3, p_4, p_5, p_6) = (1, 2, 3, 4, 5, 6)$, $(d_1, d_2, d_3, d_4, d_5, d_6) = (3, 4, 2, 9, 7, 10)$, $(w_1, w_2, w_3, w_4, w_5, w_6) = (7, 5, 1, 2, 1, 3)$. We apply Algorithm 2.1 to solve $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$ problem on this instance.

Preprocessing. The six jobs have been sorted by SPT rule.

Initialization. $F_0^{(1)}(0,0) = 0$, and $F_0^{(1)}(t,k) = +\infty$ for $(t,k) \ne (0,0)$, $k \in 0, 1, \ldots, 6$, and $t = 0, 1, \ldots, 21$.

Recursion. For all $j = 1, 2, \ldots, 6$, $k = j, j+1, \ldots, 6$, and $t = 0, 1, \ldots, 21$, the calculation of $F_j^{(1)}(t,k)$ is displayed in Table 2.

TABLE 2. The implementation of Algorithm 2.1

		t																					
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$j=1$	$k=1$	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=2$	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=3$	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=4$	∞	∞	∞	∞	28	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=5$	∞	∞	∞	∞	∞	35	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=6$	∞	∞	∞	∞	∞	∞	42	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=2$	$k=2$	∞	∞	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=3$	∞	∞	∞	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=4$	∞	∞	∞	∞	28	25	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=5$	∞	∞	∞	∞	∞	60	30	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=6$	∞	∞	∞	∞	∞	∞	72	35	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=3$	$k=3$	∞	∞	∞	3	4	5	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=4$	∞	∞	∞	∞	32	30	6	7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=5$	∞	∞	∞	∞	∞	65	36	7	8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=6$	∞	∞	∞	∞	∞	∞	78	42	8	9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=4$	$k=4$	∞	∞	∞	∞	32	30	6	3	4	5	26	∞	∞	∞								
	$k=5$	∞	∞	∞	∞	∞	65	36	7	3	4	25	28	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=6$	∞	∞	∞	∞	∞	∞	78	42	8	3	24	27	30	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=5$	$k=5$	∞	∞	∞	∞	∞	65	36	7	11	13	35	17	15	b27	19	41	∞	∞	∞	∞	∞	∞
	$k=6$	∞	∞	∞	∞	∞	∞	78	42	16	12	34	38	18	16	18	20	42	∞	∞	∞	∞	∞
$j=6$	$k=6$	∞	∞	∞	∞	∞	∞	78	42	16	12	34	71	54	46	53	58	83	68	69	74	79	104

Optimal value. The optimal value is $F_6^{(1)}(9,6)=12$, and the corresponding SPT-batch schedule is $\pi=(B_1(\pi),B_2(\pi))$, where $B_1(\pi)=\{J_1,J_2,J_3\}$ and $B_2(\pi)=\{J_4,J_5,J_6\}$.

Second dynamic programming algorithm.

For $i\in\{1,2,\ldots,j\}$, $j\in\{1,2,\ldots,n\}$, and $t\in\{0,1,\ldots,P\}$, let $W(i,j,t)$ be the value of $\sum_{k:i\le k\le j,d_k<t}w_kt$. Intuitively speaking, $W(i,j,t)$ is the total weighted tardy span of $J_i,J_{i+1},\ldots,J_j$ when $J_i,J_{i+1},\ldots,J_j$ are completed by time t. For $i=1,2,\ldots,j$, $W(i,j,t)$ can be calculated by the following procedure: (i) if $d_j<t$, $W(j,j,t)=w_jt$; if $d_j\ge t$, $W(j,j,t)=0$; (ii) for $i=1,2,\ldots,j-1$, if $d_j<t$, $W(i,j,t)=W(i+1,j,t)+w_it$; if $d_j\ge t$, $W(i,j,t)=W(i+1,j,t)$. The above procedure takes $O(n)$ time. Since there are $O(nP)$ choices for (j,t), it takes $O(n^2P)$ time for calculating all the values $W(i,j,t)$.

Given $j\in\{0,1,\ldots,n\}$ and $t\in\{0,1,\ldots,P\}$, let $F_j^{(2)}(t)$ be the minimum total weighted tardy span among all the SPT-batch schedules for jobs $J_1,J_2,\ldots,J_j$ such that J_j is the job with the largest index in the last batch and t is the completion time of the last batch. For an SPT-batch schedule assuming the value $F_j^{(2)}(t)$, suppose that the last batch consists of jobs $J_i,J_{i+1},\ldots,J_j$. Then, $F_j^{(2)}(t)=F_{i-1}^{(2)}(t-p_j)+W(i,j,t)$.

The above discussion enables us to design the second dynamic programming algorithm. It is observed that this algorithm is similar to Algorithm 2.1, but the difference is how to define the last batch of jobs $J_1,J_2,\ldots,J_j$.

Algorithm 2.2.

Preprocessing. The n jobs are sorted by the SPT rule, that is, $p_1\le p_2\le\cdots\le p_n$. For each $i\in\{1,2,\ldots,j\}$, $j\in\{1,2,\ldots,n\}$, and $t\in\{0,1,\ldots,P\}$, calculate all the values $W(i,j,t)$.

Initialization. $F_0^{(2)}(0)=0$, and $F_0^{(2)}(t)=+\infty$ for $t=1,2,\ldots,P$.

Recursion. For $j=1,2,\ldots,n$ and $t=0,1,2,\ldots,P$, calculate all the values $F_j^{(2)}(t)$ as follows:

$$F_j^{(2)}(t)=\min\{F_{i-1}^{(2)}(t-p_j)+W(i,j,t):1\le i\le j\}.$$

Optimal value. The optimal value is $\min\{F_n^{(2)}(t):0\le t\le P\}$, and the corresponding SPT-batch schedule is generated by backtracking.

Theorem 2.4. *Algorithm 2.2 solves* $1|PB,b=+\infty|\sum w_jC_jU_j$ *in* $O(n^2P)$ *time.*

Proof. The correctness of Algorithm 2.2 is guaranteed by the above discussion. For the time complexity, notice that the preprocessing procedure takes $O(n\log n)$ time to sort the n jobs, and $O(n^2P)$ time for calculating all the values $W(i,j,t)$. Hence, the preprocessing procedure takes a total of $O(n^2P)$ time. The initialization procedure takes $O(P)$ time. There are $O(nP)$ choices for (j,t), and for each choice of (j,t), the recursion procedure takes $O(n)$ time by using the preprocessing procedure. Hence, the recursion procedure takes a total of $O(n^2P)$ time. Optimal value and the corresponding SPT-batch schedule can be determined in $O(P)$ time. Therefore, the total running time of Algorithm 2.2 is $O(n^2P)$. The result follows. □

In the following, we use an instance to show the implementation of Algorithm 2.2.

Example 2.2. Consider the same instance in Example 2.1: the corresponding parameters of the six jobs $J_1, J_2, \ldots, J_6$ are given by $(p_1, p_2, p_3, p_4, p_5, p_6) = (1, 2, 3, 4, 5, 6)$, $(d_1, d_2, d_3, d_4, d_5, d_6) = (3, 4, 2, 9, 7, 10)$, $(w_1, w_2, w_3, w_4, w_5, w_6) = (7, 5, 1, 2, 1, 3)$. We apply Algorithm 2.2 to solve $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$ on this instance.

Preprocessing. The six jobs have been sorted by SPT rule. For each $i \in \{1, 2, \ldots, j\}$, $j \in \{1, 2, \ldots, 6\}$, and $t \in \{0, 1, \ldots, 21\}$, the calculation of $W(i, j, t)$ is displayed in Table 3.

TABLE 3. The calculation of $W(i, j, t)$

		t																					
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$j=1$	$i=1$	0	0	0	0	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140	147
$j=2$	$i=1$	0	0	0	0	28	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240	252
	$i=2$	0	0	0	0	0	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	105
$j=3$	$i=1$	0	0	0	3	32	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260	273
	$i=2$	0	0	0	3	4	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120	126
	$i=3$	0	0	0	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$j=4$	$i=1$	0	0	0	3	32	65	78	91	104	117	150	165	180	195	210	225	240	255	270	285	300	315
	$i=2$	0	0	0	3	4	30	36	42	48	54	80	88	96	104	112	120	128	136	144	152	160	168
	$i=3$	0	0	0	3	4	5	6	7	8	9	30	33	36	39	42	45	48	51	54	57	60	63
	$i=4$	0	0	0	0	0	0	0	0	0	0	20	22	24	26	28	30	32	34	36	38	40	42
$j=5$	$i=1$	0	0	0	3	32	65	78	91	112	126	160	176	192	208	224	240	256	272	288	304	320	336
	$i=2$	0	0	0	3	4	30	36	42	56	63	90	99	108	117	126	135	144	153	162	171	180	189
	$i=3$	0	0	0	3	4	5	6	7	16	18	40	44	48	52	56	60	64	68	72	76	80	84
	$i=4$	0	0	0	0	0	0	0	0	8	9	30	33	36	39	42	45	48	51	54	57	60	63
	$i=5$	0	0	0	0	0	0	0	0	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$j=6$	$i=1$	0	0	0	3	32	65	78	91	112	126	160	209	228	247	266	285	304	323	342	361	380	399
	$i=2$	0	0	0	3	4	30	36	42	56	63	90	132	144	156	168	180	192	204	216	228	240	252
	$i=3$	0	0	0	3	4	5	6	7	16	18	40	77	84	91	98	105	112	119	126	133	140	147
	$i=4$	0	0	0	0	0	0	0	0	8	9	30	66	72	78	84	90	96	102	108	114	120	126
	$i=5$	0	0	0	0	0	0	0	0	8	9	10	44	48	52	56	60	64	68	72	76	80	84
	$i=6$	0	0	0	0	0	0	0	0	8	9	10	33	36	39	42	45	48	51	54	57	60	63

Initialization. $F_0^{(2)}(0) = 0$, and $F_0^{(2)}(t) = +\infty$ for $t = 0, 1, \ldots, 21$.

Recursion. For $j = 1, 2, \ldots, 6$, and $t = 0, 1, \ldots, 21$, the calculation of $F_j^{(2)}(t)$ is displayed in Table 4.

TABLE 4. The implementation of Algorithm 2.2

	t																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$j=1$	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=2$	∞	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=3$	∞	∞	3	4	5	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=4$	∞	∞	∞	32	30	6	3	4	5	26	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
$j=5$	∞	∞	∞	∞	65	36	7	11	13	35	17	15	17	19	41	∞	∞	∞	∞	∞	∞
$j=6$	∞	∞	∞	∞	∞	78	42	16	12	34	71	54	46	53	58	83	68	69	74	79	104

Optimal value. The optimal value is $F_6^{(2)}(9) = 12$, and the corresponding SPT-batch schedule is $\pi = (B_1(\pi), B_2(\pi))$, where $B_1(\pi) = \{J_1, J_2, J_3\}$ and $B_2(\pi) = \{J_4, J_5, J_6\}$.

Third dynamic programming algorithm.

For the third dynamic programming algorithm, unlike Algorithm 2.1 and Algorithm 2.2, it is based on solving the hierarchical scheduling problem $1|\text{PB}, b = +\infty|\text{Lex}(\sum w_j C_j U_j, C_{\max})$, where the hierarchical problem aims to find a feasible schedule that minimizes the secondary criterion $C_{\max}$ under the constraint that the primary criterion $\sum w_j C_j U_j$ is minimized. Clearly, each feasible schedule for $1|\text{PB}, b = +\infty|\text{Lex}(\sum w_j C_j U_j, C_{\max})$ is an optimal schedule for $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$ problem.

It is observed that $1|\text{PB}, b = +\infty|\text{Lex}(\sum w_j C_j U_j, C_{\max})$ has the same property in Lemma 2.2.

Lemma 2.5. *For $1|PB, b = +\infty|Lex(\sum w_j C_j U_j, C_{\max})$, there is an optimal and SPT-batch schedule.*

From Lemma 2.5, we focus on the SPT-batch schedules for $1|\text{PB}, b = +\infty|\text{Lex}(\sum w_jC_jU_j, C_{\max})$. Given $j \in \{0, 1, \ldots, n\}$, let Ω_j be the set of objective vectors of $1|\text{PB}, b = +\infty|\text{Lex}(\sum w_jC_jU_j, C_{\max})$ on the job set $\{J_1, J_2, \ldots, J_j\}$ such that there is an SPT-batch schedule assuming each objective vector. Clearly, $\Omega_0 = \{(0, 0)\}$.

Given $j \in \{0, 1, \ldots, n\}$, we consider a vector $(x, y) \in \Omega_j$ and an SPT-batch schedule π assuming (x, y). Assume that the last batch in π includes $J_k, J_{k+1}, \ldots, J_j$, $1 \le k \le j$. Let x' and y' be the total weighted tardy span and the makespan of $\{J_1, J_2, \ldots, J_{k-1}\}$ in π, respectively. Hence, $(x', y') \in \Omega_{k-1}$, π also assumes (x', y'), $y = y' + p_j$, and $x = x' + \sum_{i:k\le i\le j, d_i<y} w_i y = x' + W(k, j, y)$, where $W(k, j, y)$ is defined as the same as that in the second algorithm.

To save the running time, the third dynamic programming algorithm focuses on the nondominated vectors in each Ω_j. For any two vectors $(x_1, y_1) \in \Omega_j$ and $(x_2, y_2) \in \Omega_j$, if $(x_1, y_1) \le (x_2, y_2)$ and $(x_1, y_1) \ne (x_2, y_2)$, then (x_1, y_1) dominates (x_2, y_2). Hence, a vector $(x, y) \in \Omega_j$ is called the nondominated vector in Ω_j, if no vectors in Ω_j dominate (x, y).

Algorithm 2.3.

Preprocessing. The n jobs are sorted by the SPT rule, i.e., $p_1 \le p_2 \le \cdots \le p_n$. For each $i \in \{1, 2, \ldots, j\}$, $j \in \{1, 2, \ldots, n\}$ and $t \in \{0, 1, \ldots, P\}$, calculate all the values $W(i, j, t)$ are calculated as Algorithm 2.2.

Initialization. $\Omega_0 = \{(0, 0)\}$, and $\Omega_j = \emptyset$ for $j = 1, 2, \ldots, n$.

Recursion. For $j = 1, 2, \ldots, n$, update Ω_j by the following procedure:

(i) For each $k = 1, 2, \ldots, j$, if $\Omega_{k-1} \ne \emptyset$, then for each $(x', y') \in \Omega_{k-1}$, let $y = y' + p_j$, $x = x' + W(k, j, y)$, and add the vector (x, y) to Ω_j; otherwise, do nothing.

(ii) For any two vectors (x, y) and (x', y') in Ω_j with $x \le x'$ and $y \le y'$, eliminate (x', y').

Optimal value. Let $x^* = \min\{x : (x, y) \in \Omega_n\}$ and $y^* = \min\{y : (x^*, y) \in \Omega_n\}$. The optimal objective vector is given by $(x^*, y^*) \in \Omega_n$, and the corresponding SPT-batch schedule is generated by backtracking.

Theorem 2.6. *Algorithm 2.3 solves* $1|PB, b = +\infty|Lex(\sum w_jC_jU_j, C_{\max})$ *in* $O(n^2P)$ *time.*

Proof. The correctness of Algorithm 2.3 is guaranteed by the above discussion. For the time complexity, the preprocessing procedure takes $O(n^2P)$ time from Theorem 2.4. The initialization procedure takes $O(n)$ time. For the recursion procedure, notice that $y \le P$ for each vector $(x, y) \in \Omega_j$. Due to the elimination operation (ii) in the recursion procedure, each Ω_j has at most $P + 1$ different vectors. Moreover, for each vector in Ω_{k-1} with $1 \le k \le j$, at most one new vector is generated in Ω_j. Thus, the construction of Ω_j in the recursion procedure requires at most $O(nP)$ time, which is also the time needed for the elimination procedure. Moreover, j has $O(n)$ choices. Hence, the recursion procedure takes a total of $O(n^2P)$ time. Optimal value and the corresponding SPT-batch schedule can be determined in $O(P)$ time. Hence, the overall running time of Algorithm 2.3 is $O(n^2P)$. The result follows. □

As a byproduct of Theorem 2.6, we can obtain the following result.

Corollary 2.7. $1|PB, b = +\infty|\sum w_jC_jU_j$ *is solvable in* $O(n^2P)$ *time.*

In the following, we use an instance to show the implementation of Algorithm 2.3.

Example 2.3. Consider the same instance in Example 2.1: the corresponding parameters of the six jobs $J_1, J_2, \ldots, J_6$ are given by $(p_1, p_2, p_3, p_4, p_5, p_6) = (1, 2, 3, 4, 5, 6)$, $(d_1, d_2, d_3, d_4, d_5, d_6) = (3, 4, 2, 9, 7, 10)$, $(w_1, w_2, w_3, w_4, w_5, w_6) = (7, 5, 1, 2, 1, 3)$. We apply Algorithm 2.3 to solve $1|\text{PB}, b = +\infty|\text{Lex}(\sum w_j C_j U_j, C_{\max})$ on this instance.

Preprocessing. The six jobs have been sorted by SPT rule. For each $i \in \{1, 2, \ldots, j\}$, $j \in \{1, 2, \ldots, 6\}$, and $t \in \{0, 1, \ldots, 21\}$, the calculation of $W(i, j, t)$ is the same as Table 3.

Initialization. $\Omega_0 = \{(0, 0)\}$, and $\Omega_j = \emptyset$ for $j = 1, 2, \ldots, 6$.

Recursion. For $j = 1, 2, \ldots, 6$, the generation of Ω_j is displayed in Table 5, where the underlined states are retained in the eliminating process.

TABLE 5. The implementation of Algorithm 2.3

$j = 1$	Ω_0					
	(0, 1)					
	$\Omega_1 = \{(0, 1)\}$					
$j = 2$	Ω_0	Ω_1				
	(0, 2)	(0, 3)				
	$\Omega_2 = \{(0, 2)\}$					
$j = 3$	Ω_0	Ω_1	Ω_2			
	(3, 3)	(4, 4)	(5, 5)			
	$\Omega_3 = \{(3, 3)\}$					
$j = 4$	Ω_0	Ω_1	Ω_2	Ω_3		
	(32, 4)	**(30, 5)**	**(6,6)**	**(3, 7)**		
	$\Omega_4 = \{(32, 4), (30, 5), (6, 6), (3, 7)\}$					
$j = 5$	Ω_0	Ω_1	Ω_2	Ω_3	Ω_4	
	(65, 5)	**(36, 6)**	**(7,7)**	(11, 8)	(13, 9) (35, 10) (17, 11) (15, 12)	
	$\Omega_5 = \{(65, 5), (36, 6), (7, 7)\}$					
$j = 6$	Ω_0	Ω_1	Ω_2	Ω_3	Ω_4	Ω_5
	(78, 6)	**(42, 7)**	**(16,8)**	**(12, 9)**	(34, 10) (74, 11) (54, 12) (55, 13)	(71, 11) (72, 12) (46, 13)
	$\Omega_6 = \{(78, 6), (42, 7), (16, 8), (12, 9)\}$					

Optimal value. $x^* = \min\{x : (x, y) \in \Omega_6\} = 12$ and $y^* = \min\{y : (12, y) \in \Omega_6\} = 9$. Thus, the optimal objective vector is (12, 9), and the corresponding SPT-batch schedule is $\pi = (B_1(\pi), B_2(\pi))$, where $B_1(\pi) = \{J_1, J_2, J_3\}$ and $B_2(\pi) = \{J_4, J_5, J_6\}$.

By Corollary 2.7, the optimal value of $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$ problem is 12, and π is the corresponding optimal schedule.

2.3. **Computational experiments.** For problem $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$, we now present computational experiments of applying the three pseudo-polynomial-time Algorithms 2.1, 2.2, and 2.3 to solve different instances. To evaluate the performance of the three algorithms, we code them in MATLAB and perform the experiments on a personal computer powered by an Intel(R) Core(TM)i5, 1.70GHz with 8GB RAM operating under Windows 11. We generated the instances with $n = 10, 50, 100, 150, 200$ jobs. For each job J_j $(j = 1, 2, \ldots, n)$, we sample the processing time p_j from the uniform integer distributions $U[1, p_{\max}]$ with $p_{\max} \in \{20, 40\}$, the due date d_j from the uniform integer distributions $U[\lceil 0.2 * \sum_{j=1}^{n} p_j \rceil, \lfloor 0.8 * \sum_{j=1}^{n} p_j \rfloor]$, and the weight w_i from the uniform integer distributions $U[1, 20]$. For each choice of n, we generated 20 random instances.

Table 6 shows the average and maximum running times (in seconds) of Algorithms 2.1, 2.2, and 2.3. Section 2.2 already suggest that the number of jobs in the instance class also has a major influence on the running time. For each algorithm,

the average running times and maximum running times increase with increasing n in all cases. Moreover, with the same selection of jobs' parameters, the running time of Algorithm 2.1 is significantly longer than that of Algorithm 2.2 and Algorithm 2.3, and the overall running time of Algorithm 2.2 and Algorithm 2.3 is not much different. This is mainly because the definition of the last batch is different. Algorithm 2.1 enumerates the processing time of the current last batch, and it is no less than the processing time of the target job J_j. But in Algorithm 2.2 and Algorithm 2.3, the processing time of the current last batch is the processing time of the target job J_j. Hence, Algorithm 2.1 need more variables and store more data than Algorithm 2.2 and Algorithm 2.3. The increase in variables and data requires more memory space and a larger cache during algorithm execution, resulting in longer running time of Algorithm 2.1.

TABLE 6. Running times (in seconds) of Algorithms 2.1, 2.2 and 2.3

		Algorithms 2.1		Algorithms 2.2		Algorithms 2.3	
$p_{\max}$	n	average-time	max-time	average-time	max-time	average-time	max-time
20	10	0.1103	0.3208	0.0193	0.0735	0.0205	0.1853
	50	15.9155	22.6538	0.4588	0.5911	1.0644	1.3543
	100	171.6305	203.6536	3.2308	3.4588	10.9800	12.7075
	150	561.2204	629.1357	11.4084	13.5433	43.8931	48.6346
	200	1.9834e+03	2.5626e+03	34.6359	37.7312	112.4627	114.7836
40	10	0.1593	0.2431	0.0221	0.0328	0.0092	0.0206
	50	32.7658	39.4390	0.9164	1.1038	1.1524	1.3126
	100	280.6231	320.3210	6.7334	7.9503	10.6998	11.5283
	150	1.2671e+03	1.3664e+03	34.4556	36.3379	40.3428	43.4845
	200	3.4637e+03	5.6947e+03	88.2433	97.5401	113.5564	116.5879

3. **Polynomial-time algorithms.** We now consider $1|r_j, \mathrm{PB}, b = +\infty$, fix-p$|\sum w_jC_jU_j$ problem and $1|\mathrm{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j$ problem. For each problem, we design a polynomial-time algorithm.

3.1. $1|r_j, \mathbf{PB}, b = +\infty, \mathbf{fix\text{-}p(k)}|\sum w_jC_jU_j$ **problem.** For $1|r_j, \mathrm{PB}, b = +\infty, \text{fix-p(k)}|\sum w_jC_jU_j$ problem, let $p^1, p^2, \ldots, p^k$ be the k different processing times such that $p^1 < p^2 < \cdots < p^k$. The n jobs are sorted in the ERD order, i.e., $r_1 \le r_2 \le \cdots \le r_n$. Particularly, set $r_0 := -\infty$ and $r_{n+1} := +\infty$. Let $\mathcal{S} = \{r_j + x_1p^1 + x_2p^2 + \cdots + x_kp^k : j = 1, 2, \ldots, n,\ 0 \le x_1, x_2, \ldots, x_k \le n,\ \sum_{i=1}^{k} x_i \le n\}$. Suppose that $|\mathcal{S}| = l$. Obviously, $l \le n\binom{n+k}{k} = O(n^{k+1})$. Let $\mathcal{S} = \{t_1, t_2, \ldots, t_l\}$ such that $t_1 < t_2 < \cdots < t_l$.

Lemma 3.1. *Let $\sigma = (B_1(\sigma), B_2(\sigma), \ldots, B_m(\sigma))$ be an optimal schedule for $1|r_j, PB, b = +\infty, fix\text{-}p(k)|\sum w_jC_jU_j$ problem. Then, $S(B_i(\sigma)) \in \mathcal{S}$ for $i \in \{1, 2, \ldots, m\}$.*

Proof. For $i \in \{1, 2, \ldots, m\}$, let $r(B_i(\sigma)) = \max\{r_j : J_j \in B_i(\sigma)\}$. Obviously, $S(B_1(\sigma)) = r(B_1(\sigma)) \in \mathcal{S}$, and $S(B_i(\sigma)) \ge r(B_i(\sigma))$ for $i \in \{2, 3, \ldots, m\}$. If $S(B_i(\sigma)) = r(B_i(\sigma))$ for each $i \in \{2, 3, \ldots, m\}$, then the lemma obviously holds. If there is some batch $B_i(\sigma)$ such that $S(B_i(\sigma)) > r(B_i(\sigma))$, $i \in \{2, 3, \ldots, m\}$, then let $v = \max\{j : S(B_j(\sigma)) = r(B_j(\sigma)),\ 1 \le j \le i-1\}$. Since $S(B_1(\sigma)) = r(B_1(\sigma))$, v is well-defined. Notice that the starting time of each batch is either the maximum release time of its jobs, or the completion time of the previous batch. Hence, $B_v(\sigma), B_{v+1}(\sigma), \ldots, B_i(\sigma)$ are processed consecutively without idle time in σ. It follows that $S(B_i(\sigma)) = r(B_v(\sigma)) + p(B_{v+1}) + p(B_{v+2}) + \cdots + p(B_{i-1})$. Obviously, $S(B_i(\sigma)) \in \mathcal{S}$. The result follows. □

Lemma 3.2. *For $1|r_j, PB, b = +\infty, \text{fix-p(k)}|\sum w_jC_jU_j$ problem has an optimal schedule $\sigma = (B_1(\sigma), B_2(\sigma), \ldots, B_m(\sigma))$ such that for each $x = 1, 2, \ldots, m$ and $i = 1, 2, \ldots, x$, if $J_j \in B_x(\sigma)$ and $r_j \le S(B_i(\sigma))$, then $p(B_i(\sigma)) < p_j$.*

Proof. Suppose to the contrary that there is some job $J_j \in B_x(\sigma)$ such that $p(B_i(\sigma)) \ge p_j$ and $r_j \le S(B_i(\sigma))$, $1 \le i < x \le m$. Let σ' be the schedule obtained from σ by shifting J_j to $B_i(\sigma)$. Since $p(B_i(\sigma)) \ge p_j$ and $r_j \le S(B_i(\sigma))$, σ' is a feasible schedule and $C_y(\sigma') \le C_y(\sigma)$ for each J_y. Hence, $\sum_{j=1}^{n} w_jC_j(\sigma')U_j(\sigma') \le \sum_{j=1}^{n} w_jC_j(\sigma)U_j(\sigma)$, implying that σ' is also an optimal schedule. Repeat the above procedure, there is a required optimal schedule for $1|r_j, \text{PB}, b = +\infty$, fix-p(k)$|\sum w_jC_jU_j$ problem. The result follows. □

For a time point $t_u \in \mathcal{S}$ and a job set $X = \{J_j : r_j \le t_u,\ 1 \le j \le n\}$, let $W(t_u, X) = \sum_{J_j \in X: d_j < t_u + p_{\max}(X)} w_j(t_u + p_{\max}(X))$, where $p_{\max}(X) = \max\{p_j : J_j \in X\}$.

For $h = 1, 2, \ldots, k$, let $\mathcal{J}^{(h)} = \{J_i : p_i = p^h,\ 1 \le i \le n\}$ be the set of jobs with processing time p^h. Moreover, for $j = 0, 1, \ldots, g$ and $g = 0, 1, \ldots, n+1$, let $\mathcal{J}^{(h)}_{j,g} = \{J_i \in \mathcal{J}^{(h)} : r_j < r_i \le r_g\}$ be a subset of $\mathcal{J}^{(h)}$ whose jobs are released after r_j and no later than r_g.

Given $u \in \{1, 2, \ldots, l\}$ and $j_1, j_2, \ldots, j_k \in \{0, 1, \ldots, n\}$, let $F(u, j_1, j_2, \ldots, j_k)$ be the minimum total weighted tardy span of $\mathcal{J}^{(1)}_{j_1,n} \cup \mathcal{J}^{(2)}_{j_2,n} \cup \cdots \cup \mathcal{J}^{(k)}_{j_k,n}$ such that the starting time of the first batch is not earlier than t_u. Moreover, if there is some $j_i \in \{j_1, j_2, \ldots, j_k\}$ such that $j_i = n$, then $\mathcal{J}^{(1)}_{j_1,n} \cup \mathcal{J}^{(2)}_{j_2,n} \cup \cdots \cup \mathcal{J}^{(k)}_{j_k,n}$ does not exist, and set $F(u, j_1, j_2, \ldots, j_k) := +\infty$.

For a schedule $\pi = (B_1(\pi), B_2(\pi), \ldots, B_m(\pi))$ assuming $F(u, j_1, j_2, \ldots, j_k)$, there are two possibilities for $B_1(\pi)$:

- $S(B_1(\pi)) > t_u$. From Lemma 3.1, $F(u, j_1, j_2, \ldots, j_k) = F(u+1, j_1, j_2, \ldots, j_k)$.
- $S(B_1(\pi)) = t_u$. Assume that $p(B_1(\pi)) = p^s$, $s \in \{1, 2, \ldots, k\}$. Let i_u be the index such that $r_{i_u} \le t_u < r_{i_u+1}$. From Lemma 3.2, $\mathcal{J}^{(1)}_{j_1,i_u} \cup \mathcal{J}^{(2)}_{j_2,i_u} \cup \cdots \cup \mathcal{J}^{(s)}_{j_s,i_u} \subseteq B_1(\pi)$, and the jobs of $\mathcal{J}^{(1)}_{i_u,n} \cup \mathcal{J}^{(2)}_{i_u,n} \cup \cdots \cup \mathcal{J}^{(s)}_{i_u,n} \cup \mathcal{J}^{(s+1)}_{j_{s+1},n} \cup \cdots \cup \mathcal{J}^{(k)}_{j_k,n}$ are in the remaining batches. Thus, we have $F(u, j_1, j_2, \ldots, j_k) = W(t_u, \mathcal{J}^{(1)}_{j_1,i_u} \cup \mathcal{J}^{(2)}_{j_2,i_u} \cup \cdots \cup \mathcal{J}^{(s)}_{j_s,i_u}) + F(u', i_u, i_u, \ldots, i_u, j_{s+1}, \ldots, j_k)$, where $t_{u'} = t_u + p^s$.

The above discussion enables us to design a dynamic programming algorithm for $1|r_j, \text{PB}, b = +\infty, \text{fix-p(k)}|\sum w_jC_jU_j$.

Algorithm 3.1.

Preprocessing. The n jobs are sorted by the ERD rule, that is, $r_1 \le r_2 \le \cdots \le r_n$. Determine the sets $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \ldots, \mathcal{J}^{(k)}$.

Initialization. For $j_1, j_2, \ldots, j_k \in \{0, 1, \ldots, n\}$, set $F(l, j_1, j_2, \ldots, j_k) := W(t_l, \mathcal{J}^{(1)}_{j_1,n} \cup \mathcal{J}^{(2)}_{j_2,n} \cup \cdots \cup \mathcal{J}^{(k)}_{j_k,n})$.

Recursion. For $u = l-1, l-2, \ldots, 1$, let i_u be the unique index such that $r_{i_u} \le t_u < r_{i_u+1}$. For $j_1, j_2, \ldots, j_k \in \{0, 1, \ldots, i_u\}$, calculate all the values $F(u, j_1, j_2, \ldots, j_k)$ as follows:

$$F(u, j_1, j_2, \ldots, j_k) = \begin{cases} F(u+1, j_1, j_2, \ldots, j_k), & S(B_1(\pi)) > t_u, \\ \min_{s:1 \le s \le k}(W(t_u, \mathcal{J}^{(1)}_{j_1,i_u} \cup \mathcal{J}^{(2)}_{j_2,i_u} \cup \cdots \cup \mathcal{J}^{(s)}_{j_s,i_u}) & \\ \quad + F(u', i_u, i_u, \ldots, i_u, j_{s+1}, \ldots, j_k)), & S(B_1(\pi)) = t_u, \end{cases}$$

where $t_{u'} = t_u + p^s$.

Optimal value. The optimal value is $F(1,0,0,\ldots,0)$, and the corresponding schedule is generated by backtracking.

Theorem 3.3. *Algorithm 3.1 solves* $a1|r_j, PB, b=+\infty, \textit{fix-p(k)}|\sum w_jC_jU_j$ *in* $O(n^{2k+2})$ *time, where* k *is a constant.*

Proof. The correctness of Algorithm 3.1 is guaranteed by the above discussion. For the time complexity, the preprocessing procedure takes $O(n\log n)$ time to sort n jobs and $O(n\log n)$ time to determine the sets $\mathcal{J}^{(1)},\mathcal{J}^{(2)},\ldots,\mathcal{J}^{(k)}$ by sorting jobs in the SPT rule. In the initialization procedure, there are $O(n^k)$ choices for $(j_1,j_2,\ldots,j_k)$, and for each $(j_1,j_2,\ldots,j_k)$, determining the sets $\mathcal{J}^{(1)}_{j_1,n}\cup\mathcal{J}^{(2)}_{j_2,n}\cup\cdots\cup\mathcal{J}^{(k)}_{j_k,n}$ takes $O(\log n)$ time by binary search, and calculating $W(t_l,\mathcal{J}^{(1)}_{j_1,n}\cup\mathcal{J}^{(2)}_{j_2,n}\cup\cdots\cup\mathcal{J}^{(k)}_{j_k,n})$ takes $O(n)$ time. Hence, the initialization procedure takes $O(n^{k+1})$ time. In the recursion procedure, there are $O(n^{2k+1})$ choices for $(u,j_1,j_2,\ldots,j_k)$. For each $(u,j_1,j_2,\ldots,j_k)$, the index i_u is determined by binary search in $O(\log n)$ time. Moreover, there are k choices for s, and for each s, calculating $W(t_u,\mathcal{J}^{(1)}_{j_1,i_u}\cup\mathcal{J}^{(2)}_{j_2,i_u}\cup\cdots\cup\mathcal{J}^{(s)}_{j_s,i_u})+F(u',i_u,\ldots,i_u,j_{s+1},\ldots,j_k)$ takes $O(n)$ time, which is similar to the initialization procedure. Hence, the recursion procedure takes $O(n^{2k+2})$ time. Optimal value and the corresponding SPT-batch schedule can be determined in $O(n)$ time. Hence, the total running time of Algorithm 3.1 is $O(n^{2k+2})$ time. The result follows. □

In the following, we use an instance to show the implementation of Algorithm 3.1.

Example 3.1. Consider an instance of $1|r_j,\mathrm{PB},b=+\infty,\text{fix-p(k)}|\sum w_jC_jU_j$ problem in which there are five jobs $J_1,J_2,\ldots,J_5$ are given by $(r_1,r_2,r_3,r_4,r_5)=(0,0,1,1,7)$, $(p_1,p_2,p_3,p_4,p_5)=(2,4,2,4,2)$, $(d_1,d_2,d_3,d_4,d_5)=(3,1,3,8,11)$, and $(w_1,w_2,w_3,w_4,w_5)=(10,1,8,2,5)$. Clearly, $\mathcal{S}=\{0,1,\ldots,14,15,17,19,21\}=\{t_1,t_2,\ldots,t_{19}\}$, and $k=2$. We apply Algorithm 3.1 to solve $1|r_j,\mathrm{PB},b=+\infty,\text{fix-p(k)}|\sum w_jC_jU_j$ problem on this instance.

Preprocessing. The five jobs have been sorted by ERD rule. Moreover, $\mathcal{J}^{(1)}=\{J_1,J_3,J_5\}$ and $\mathcal{J}^{(2)}=\{J_2,J_4\}$.

Initialization. For $j_1,j_2\in\{0,1,\ldots,5\}$, the calculation of $F(21,j_1,j_2)$ is as follows:

- $j_2=0$ and $j_1\in\{0,1,\ldots,5\}$. $F(21,0,0)=WC(21,\mathcal{J}^{(1)}_{0,5}\cup\mathcal{J}^{(2)}_{0,5})=546$, $F(21,1,0)=WC(21,\mathcal{J}^{(1)}_{1,5}\cup\mathcal{J}^{(2)}_{0,5})=336$, $F(21,2,0)=WC(21,\mathcal{J}^{(1)}_{2,5}\cup\mathcal{J}^{(2)}_{0,5})=336$, $F(21,3,0)=WC(21,\mathcal{J}^{(1)}_{3,5}\cup\mathcal{J}^{(2)}_{0,5})=168$, $F(21,4,0)=WC(21,\mathcal{J}^{(1)}_{4,5}\cup\mathcal{J}^{(2)}_{0,5})=168$, and $F(21,5,0)=WC(21,\mathcal{J}^{(1)}_{5,5}\cup\mathcal{J}^{(2)}_{0,5})=63$.
- $j_2=1$ and $j_1\in\{0,1,\ldots,5\}$. $F(21,0,1)=WC(21,\mathcal{J}^{(1)}_{0,5}\cup\mathcal{J}^{(2)}_{1,5})=525$, $F(21,1,1)=WC(21,\mathcal{J}^{(1)}_{1,5}\cup\mathcal{J}^{(2)}_{1,5})=315$, $F(21,2,1)=WC(21,\mathcal{J}^{(1)}_{2,5}\cup\mathcal{J}^{(2)}_{1,5})=315$, $F(21,3,1)=WC(21,\mathcal{J}^{(1)}_{3,5}\cup\mathcal{J}^{(2)}_{1,5})=147$, $F(21,4,1)=WC(21,\mathcal{J}^{(1)}_{4,5}\cup\mathcal{J}^{(2)}_{1,5})=147$, and $F(21,5,1)=WC(21,\mathcal{J}^{(1)}_{5,5}\cup\mathcal{J}^{(2)}_{1,5})=42$.
- $j_2=2$ and $j_1\in\{0,1,\ldots,5\}$. $F(21,0,2)=WC(21,\mathcal{J}^{(1)}_{0,5}\cup\mathcal{J}^{(2)}_{2,5})=525$, $F(21,1,2)=WC(21,\mathcal{J}^{(1)}_{1,5}\cup\mathcal{J}^{(2)}_{2,5})=315$, $F(21,2,2)=WC(21,\mathcal{J}^{(1)}_{2,5}\cup\mathcal{J}^{(2)}_{2,5})=315$, $F(21,3,2)=WC(21,\mathcal{J}^{(1)}_{3,5}\cup\mathcal{J}^{(2)}_{2,5})=147$, $F(21,4,2)=WC(21,\mathcal{J}^{(1)}_{4,5}\cup\mathcal{J}^{(2)}_{2,5})=147$, and $F(21,5,2)=WC(21,\mathcal{J}^{(1)}_{5,5}\cup\mathcal{J}^{(2)}_{2,5})=42$.

• $j_2 = 3$ and $j_1 \in \{0, 1, \ldots, 5\}$. $F(21, 0, 3) = WC(21, \mathcal{J}_{0,5}^{(1)} \cup \mathcal{J}_{3,5}^{(2)}) = 483$, $F(21, 1, 3) = WC(21, \mathcal{J}_{1,5}^{(1)} \cup \mathcal{J}_{3,5}^{(2)}) = 273$, $F(21, 2, 3) = WC(21, \mathcal{J}_{2,5}^{(1)} \cup \mathcal{J}_{3,5}^{(2)}) = 273$, $F(21, 3, 3) = WC(21, \mathcal{J}_{3,5}^{(1)} \cup \mathcal{J}_{3,5}^{(2)}) = 105$, $F(21, 4, 3) = WC(21, \mathcal{J}_{4,5}^{(1)} \cup \mathcal{J}_{3,5}^{(2)}) = 105$, and $F(21, 5, 3) = WC(21, \mathcal{J}_{5,5}^{(1)} \cup \mathcal{J}_{3,5}^{(2)}) = 0$.

• $j_2 = 4$ and $j_1 \in \{0, 1, \ldots, 5\}$. $F(21, 0, 4) = WC(21, \mathcal{J}_{0,5}^{(1)} \cup \mathcal{J}_{4,5}^{(2)}) = 483$, $F(21, 1, 4) = WC(21, \mathcal{J}_{1,5}^{(1)} \cup \mathcal{J}_{4,5}^{(2)}) = 273$, $F(21, 2, 4) = WC(21, \mathcal{J}_{2,5}^{(1)} \cup \mathcal{J}_{4,5}^{(2)}) = 273$, $F(21, 3, 4) = WC(21, \mathcal{J}_{3,5}^{(1)} \cup \mathcal{J}_{4,5}^{(2)}) = 105$, $F(21, 4, 4) = WC(21, \mathcal{J}_{4,5}^{(1)} \cup \mathcal{J}_{4,5}^{(2)}) = 105$, and $F(21, 5, 4) = WC(21, \mathcal{J}_{5,5}^{(1)} \cup \mathcal{J}_{4,5}^{(2)}) = 0$.

• $j_2 = 5$ and $j_1 \in \{0, 1, \ldots, 5\}$. $F(21, 0, 5) = WC(21, \mathcal{J}_{0,5}^{(1)} \cup \mathcal{J}_{5,5}^{(2)}) = 483$, $F(21, 1, 5) = WC(21, \mathcal{J}_{1,5}^{(1)} \cup \mathcal{J}_{5,5}^{(2)}) = 273$, $F(21, 2, 5) = WC(21, \mathcal{J}_{2,5}^{(1)} \cup \mathcal{J}_{5,5}^{(2)}) = 273$, $F(21, 3, 5) = WC(21, \mathcal{J}_{3,5}^{(1)} \cup \mathcal{J}_{5,5}^{(2)}) = 105$, $F(21, 4, 5) = WC(21, \mathcal{J}_{4,5}^{(1)} \cup \mathcal{J}_{5,5}^{(2)}) = 105$, and $F(21, 5, 5) = WC(21, \mathcal{J}_{5,5}^{(1)} \cup \mathcal{J}_{5,5}^{(2)}) = 0$.

Recursion. In order to save space, we will not provide a detailed description of the recursive process for $u = 19, 17, 15, 14, \ldots, 3$. For $u = 2$, and $j_1, j_2 \in \{0, 1, \ldots, 5\}$, $t_2 = 1$, implies $i_2 = 4$, the calculation of $F(2, j_1, j_2)$ is as follows:

• $j_1 = 0$ and $j_2 \in \{0, 1, \ldots, 5\}$. $F(2, 0, 0) = \min\{F(3, 0, 0), WC(1, \mathcal{J}_{0,4}^{(1)} \cup \mathcal{J}_{0,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{0,4}^{(1)}) + F(3, 4, 0)\} = 7$, $F(2, 0, 1) = \min\{F(3, 0, 1), WC(1, \mathcal{J}_{0,4}^{(1)} \cup \mathcal{J}_{1,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{0,4}^{(1)}) + F(3, 4, 1)\} = 0$, $F(2, 0, 2) = \min\{F(3, 0, 2), WC(1, \mathcal{J}_{0,4}^{(1)} \cup \mathcal{J}_{2,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{0,4}^{(1)}) + F(3, 4, 2)\} = 0$, $F(2, 0, 3) = \min\{F(\ 3, 0, 3), WC(1, \mathcal{J}_{0,4}^{(1)} \cup \mathcal{J}_{3,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{0,4}^{(1)}) + F(3, 4, 3)\} = 0$, $F(2, 0, 4) = \min\{F(3, 0, 4), WC(1, \mathcal{J}_{0,4}^{(1)} \cup \mathcal{J}_{4,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{0,4}^{(1)}) + F(3, 4, 4)\} = 0$, $F(2, 0, 5) = \min\{F(3, 0, 5), WC(1, \mathcal{J}_{0,4}^{(1)} \cup \mathcal{J}_{5,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{0,4}^{(1)}) + F(3, 4, 5)\} = 0$,

• $j_1 = 1$ and $j_2 \in \{0, 1, \ldots, 5\}$. $F(2, 1, 0) = \min\{F(3, 1, 0), WC(1, \mathcal{J}_{1,4}^{(1)} \cup \mathcal{J}_{0,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{1,4}^{(1)}) + F(3, 4, 0)\} = 7$, $F(2, 1, 1) = \min\{F(3, 1, 1), WC(1, \mathcal{J}_{1,4}^{(1)} \cup \mathcal{J}_{1,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{1,4}^{(1)}) + F(3, 4, 1)\} = 0$, $F(2, 1, 2) = \min\{F(3, 1, 2), WC(1, \mathcal{J}_{1,4}^{(1)} \cup \mathcal{J}_{2,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{1,4}^{(1)}) + F(3, 4, 2)\} = 0$, $F(2, 1, 3) = \min\{F(3, 1, 3), WC(1, \mathcal{J}_{1,4}^{(1)} \cup \mathcal{J}_{3,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{1,4}^{(1)}) + F(3, 4, 3)\} = 44$, $F(2, 1, 4) = \min\{F(3, 1, 4), WC(1, \mathcal{J}_{1,4}^{(1)} \cup \mathcal{J}_{4,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{1,4}^{(1)}) + F(3, 4, 4)\} = 0$, $F(2, 1, 5) = \min\{F(3, 1, 5), WC(1, \mathcal{J}_{1,4}^{(1)} \cup \mathcal{J}_{5,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{1,4}^{(1)}) + F(3, 4, 5)\} = 0$,

• $j_1 = 2$ and $j_2 \in \{0, 1, \ldots, 5\}$. $F(2, 2, 0) = \min\{F(3, 2, 0), WC(1, \mathcal{J}_{2,4}^{(1)} \cup \mathcal{J}_{0,4}^{(2)}) + F(5, 4, 4), WC(\ 1, \mathcal{J}_{2,4}^{(1)}) + F(3, 4, 0)\} = 7$, $F(2, 2, 1) = \min\{F(3, 2, 1), WC(1, \mathcal{J}_{2,4}^{(1)} \cup \mathcal{J}_{1,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{2,4}^{(1)}) + F(3, 4, 1)\} = 0$, $F(2, 2, 2) = \min\{F(3, 2, 2), WC(1, \mathcal{J}_{2,4}^{(1)} \cup \mathcal{J}_{2,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{2,4}^{(1)}) + F(3, 4, 2)\} = 0$, $F(2, 2, 3) = \min\{F(3, 2, 3), WC(1, \mathcal{J}_{2,4}^{(1)} \cup \mathcal{J}_{3,4}^{(2)}) + F(5, 4, 4), WC(1, \mathcal{J}_{2,4}^{(1)}) + F(3, 4, 3)\} = 0$, $F(2, 2, 4) = \min\{F(3, 2, 4), WC(1, \mathcal{J}_{2,4}^{(1)} \cup \mathcal{J}_{4,4}^{(2)}) + F(5, 4, 4),$

$WC(1,\mathcal{J}_{2,4}^{(1)})+F(3,4,4)\}=0$, $F(2,2,5)=\min\{F(3,2,5),WC(1,\mathcal{J}_{2,4}^{(1)}\cup\mathcal{J}_{5,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{2,4}^{(1)})+F(3,4,5)\}=0$,

• $j_1=3$ and $j_2\in\{0,1,\ldots,5\}$. $F(2,3,0)=\min\{F(3,3,0),WC(1,\mathcal{J}_{3,4}^{(1)}\cup\mathcal{J}_{0,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{3,4}^{(1)})+F(3,4,0)\}=5$, $F(2,3,1)=\min\{F(3,3,1),WC(1,\mathcal{J}_{3,4}^{(1)}\cup\mathcal{J}_{1,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{3,4}^{(1)})+F(3,4,1)\}=0$, $F(2,3,2)=\min\{F(3,3,2),WC(1,\mathcal{J}_{3,4}^{(1)}\cup\mathcal{J}_{2,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{3,4}^{(1)})+F(3,4,2)\}=0$, $F(2,3,3)=\min\{F(3,3,3),WC(1,\mathcal{J}_{34}^{(1)}\cup\mathcal{J}_{3,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{3,4}^{(1)})+F(3,4,3)\}=0$, $F(2,3,4)=\min\{F(3,3,4),WC(1,\mathcal{J}_{3,4}^{(1)}\cup\mathcal{J}_{4,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{3,4}^{(1)})+F(3,4,4)\}=0$, $F(2,3,5)=\min\{F(3,3,5),WC(1,\mathcal{J}_{3,4}^{(1)}\cup\mathcal{J}_{5,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{3,4}^{(1)})+F(3,4,5)\}=0$,

• $j_1=4$ and $j_2\in\{0,1,\ldots,5\}$. $F(2,4,0)=\min\{F(3,4,0),WC(1,\mathcal{J}_{4,4}^{(1)}\cup\mathcal{J}_{0,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{4,4}^{(1)})+F(3,4,0)\}=5$, $F(2,4,1)=\min\{F(3,4,1),WC(1,\mathcal{J}_{4,4}^{(1)}\cup\mathcal{J}_{1,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{4,4}^{(1)})+F(3,4,1)\}=0$, $F(2,4,2)=\min\{F(3,4,2),WC(1,\mathcal{J}_{4,4}^{(1)}\cup\mathcal{J}_{2,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{4,4}^{(1)})+F(3,4,2)\}=0$, $F(2,4,3)=\min\{F(3,4,3),WC(1,\mathcal{J}_{4,4}^{(1)}\cup\mathcal{J}_{3,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{4,4}^{(1)})+F(3,4,3)\}=0$, $F(2,4,4)=\min\{F(3,4,4),WC(1,\mathcal{J}_{4,4}^{(1)}\cup\mathcal{J}_{4,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{4,4}^{(1)})+F(3,4,4)\}=0$, $F(2,4,5)=\min\{F(3,4,5),WC(1,\mathcal{J}_{4,4}^{(1)}\cup\mathcal{J}_{5,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{4,4}^{(1)})+F(3,4,5)\}=0$,

• $j_1=5$ and $j_2\in\{0,1,\ldots,5\}$. $F(2,5,0)=\min\{F(3,5,0),WC(1,\mathcal{J}_{5,4}^{(1)}\cup\mathcal{J}_{0,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{5,4}^{(1)})+F(3,4,0)\}=5$, $F(2,5,1)=\min\{F(3,5,1),WC(1,\mathcal{J}_{5,4}^{(1)}\cup\mathcal{J}_{1,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{5,4}^{(1)})+F(3,4,1)\}=0$, $F(2,5,2)=\min\{F(3,5,2),WC(1,\mathcal{J}_{5,4}^{(1)}\cup\mathcal{J}_{2,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{5,4}^{(1)})+F(3,4,2)\}=0$, $F(2,5,3)=\min\{F(3,5,3),WC(1,\mathcal{J}_{5,4}^{(1)}\cup\mathcal{J}_{3,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{5,4}^{(1)})+F(3,4,3)\}=0$, $F(2,5,4)=\min\{F(3,5,4),WC(1,\mathcal{J}_{5,4}^{(1)}\cup\mathcal{J}_{4,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{5,4}^{(1)})+F(3,4,4)\}=0$, $F(2,5,5)=\min\{F(3,5,5),WC(1,\mathcal{J}_{5,4}^{(1)}\cup\mathcal{J}_{5,4}^{(2)})+F(5,4,4),WC(1,\mathcal{J}_{5,4}^{(1)})+F(3,4,5)\}=0$.

The optimal value is $F(1,0,0)=\min\{F(2,0,0),WC(0,\mathcal{J}_{0,3}^{(1)}\cup\mathcal{J}_{0,3}^{(2)})+F(4,3,3),WC(0,\mathcal{J}_{0,3}^{(1)})+F(2,3,0)\}=7$ and the corresponding schedule is $\pi=(B_1(\pi),B_2(\pi),B_3(\pi))$, where $B_1(\pi)=\{J_1,J_3\}$, $B_2(\pi)=\{J_2\,J_4\}$, and $B_3(\pi)=\{J_5\}$.

3.2. $1|\mathbf{PB},b=+\infty,d_j=d|\sum w_jC_jU_j$**.** For $1|\mathrm{PB},b=+\infty,d_j=d|\sum w_jC_jU_j$ problem, the jobs of the same processing times can be assigned to a common batch in an optimal schedule. Moreover, the jobs of the same processing time can be regarded as a new job, and the weight of the new job is defined as the sum of the weights of these equal-length jobs. Since all the jobs have a common due date, this operation does not affect the essence of the problem. Therefore, we assume that the processing times of the n jobs are different.

Sort the n jobs by the SPT order, that is, $p_1<p_2<\cdots<p_n$. If $p_n\le d$, then the schedule in which all the jobs are scheduled in one batch and started at time 0 is optimal for $1|\mathrm{PB},b=+\infty,d_j=d|\sum w_jC_jU_j$ problem, and the optimal value is 0. Then, we assume that $p_n>d$.

The following lemma establishes an important property.

Lemma 3.4. *$1|PB, b = +\infty, d_j = d|\sum w_jC_jU_j$ has an optimal SPT-batch schedule $\sigma = (B_1(\sigma), B_2(\sigma), \ldots, B_m(\sigma))$ such that each job of $B_i(\sigma)$ is tardy in σ, $i = 2, 3, \ldots, m$.*

Proof. Let $\sigma = (B_1(\sigma), B_2(\sigma), \ldots, B_m(\sigma))$ be an optimal SPT-batch schedule such that the number of batches is as small as possible. If some batch $B_i(\sigma)$ includes some early job(s), $2 \le i \le m$, then from the fact that $d_j = d$ for each job J_j, all the jobs of $B_1(\sigma) \cup B_2(\sigma) \cup \cdots \cup B_i(\sigma)$ are early in σ. Note that $b = +\infty$. Let $\sigma' = (B_1(\sigma) \cup B_2(\sigma) \cup \cdots \cup B_i(\sigma), B_{i+1}(\sigma), \ldots, B_m(\sigma))$. Obviously, σ' is also an SPT-batch schedule, the jobs of $B_1(\sigma') = B_1(\sigma) \cup B_2(\sigma) \cup \cdots \cup B_i(\sigma)$ are early in σ', and $\sum_{j=1}^{n} w_jC_j(\sigma')U_j(\sigma') \le \sum_{j=1}^{n} w_jC_j(\sigma)U_j(\sigma)$. This implies that σ' is also an optimal schedule. But the number of batches of σ' is less than that of σ, a contradiction to the choice of σ. The result follows. □

From Lemma 2.2, we focus on SPT-batch schedules for $1|\text{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j$. To solve this problem, we will apply the $O(n \log n)$-time algorithm for $1|\text{PB}, b = +\infty|\sum w_jC_j$ problem in Brucker et al. [2] as a subroutine. By using a special shortest-path network as an auxiliary tool, the principle of the $O(n \log n)$-time algorithm without introducing the detail can be stated as follows: for $j \in \{1, 2, \ldots, n\}$, let $OPT(j)$ be the minimum total weighted completion time among all the SPT-batch schedules for jobs $J_j, J_{j+1}, \ldots, J_n$. Especially, set $OPT(n+1) := 0$. For each $j = n, n-1, \ldots, 1$, $OPT(j)$ can be calculated by the following recursion:

$$OPT(j) = \min\{W(j, n)p_k + OPT(k+1) : j \le k \le n\},$$

where $W(j, n) = w_j + w_{j+1} + \cdots + w_n$. The optimal value is $OPT(1)$.

From Brucker et al. [2], the following lemma holds.

Lemma 3.5. *All the values $OPT(1), OPT(2), \ldots, OPT(n)$ can be generated in $O(n \log n)$ time.*

For $j \in \{1, 2, \ldots, n\}$, let $\mathcal{P}(j)$ be the variation of $1|\text{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j$ such that in any SPT-batch schedule,
(i) jobs $J_1, J_2, \ldots, J_j$ are in the first batch,
(ii) jobs $J_{j+1}, \ldots, J_n$ are in the other batches,
(iii) for jobs $J_1, \ldots, J_j$, calculate their total weighted tardy span, and
(iii) for jobs $J_{j+1}, \ldots, J_n$, calculate their total weighted completion time.

By applying the $O(n \log n)$-time algorithm for $1|\text{PB}, b = +\infty|\sum w_jC_j$ in Brucker et al. [2] as a subroutine, the main idea for solving $\mathcal{P}(j)$ is described as follows: given $j = 1, 2, \ldots, n$, let F_j be the minimum objective value for $\mathcal{P}(j)$ among all the SPT-batch schedules. Recall that $W(j, n) = w_j + w_{j+1} + \cdots + w_n$. Set $W(n+1, n) := 0$ and $p_{n+1} := 0$. Hence, $F_j = OPT(j+1) + W(j+1, n)p_j$ if $p_j \le d$, and $F_j = OPT(j+1) + W(1, n)p_j$ if $p_j > d$. Finally, $1|\text{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j$ can be solved by enumerating j, solving $\mathcal{P}(j)$ for each j, and picking the best one.

The above discussion enables us to design an algorithm for $1|\text{PB}, b = +\infty, d_j = d|\sum w_jC_jU_j$.

Algorithm 3.2.

Preprocessing. The n jobs are sorted by the SPT rule, that is, $p_1 < p_2 < \cdots < p_n$.

Step 1. For $j = 1, 2, \ldots, n$, calculate all the values $W(1, j)$. For $j = 1, 2, \ldots, n-1$, apply the $O(n \log n)$-time algorithm in Brucker et al. [2] to calculate all the values $OPT(j+1)$. Set $OPT(n+1) := 0$.

Step 2. For $j = 1, 2, \ldots, n$, calculate F_j as follows:

$$F_j = \begin{cases} OPT(j+1) + W(j+1, n)p_j, & \text{if } p_j \le d, \\ OPT(j+1) + W(1, n)p_j, & \text{if } p_j > d. \end{cases}$$

Step 3. The optimal value is $\min\{F_j : j = 1, 2, \ldots, n\}$ and the corresponding SPT-batch is generated by backtracking.

Theorem 3.6. *Algorithm 3.2 solves* $1|PB, b = +\infty, d_j = d|\sum w_j C_j U_j$ *in* $O(n \log n)$ *time.*

Proof. The correctness of Algorithm 3.2 is guaranteed by the above discussion. For the time complexity, the preprocessing procedure takes $O(n \log n)$ time to sort jobs. In Step 1, it takes $O(n)$ time to calculate all the values $W(1, j)$. From Lemma 3.5, all the values $OPT(j+1), OPT(j+2), \ldots, OPT(n)$ can be obtained in $O(n \log n)$ time. In Step 2, there are $O(n)$ choices for j, and for each j, it takes a constant time for calculating F_j by Step 1. Hence, Step 2 takes $O(n)$ time. Optimal value and the corresponding SPT-batch schedule can be determined in $O(n)$ time. Hence, the total running time of Algorithm 3.2 is $O(n \log n)$. The result follows. □

In the following, we use an instance to show the implementation of Algorithm 3.2.

Example 3.2. Consider an instance of $1|\text{PB}, b = +\infty, d_j = d|\sum w_j C_j U_j$: there are six jobs $J_1, J_2, \ldots, J_6$, and the corresponding parameters are given by $(p_1, p_2, p_3, p_4, p_5, p_6) = (1, 2, 3, 4, 5, 6)$, $(d_1, d_2, d_3, d_4, d_5, d_6) = (4, 4, 4, 4, 4, 4)$, $(w_1, w_2, w_3, w_4, w_5, w_6) = (7, 5, 1, 2, 1, 3)$. We apply Algorithm 3.2 to solve $1|\text{PB}, b = +\infty, d_j = d|\sum w_j C_j U_j$ on this instance.

Preprocessing. The six jobs have been sorted by SPT rule.

Step 1. $W(1,1) = 7$, $W(1,2) = 12$, $W(1,3) = 13$, $W(1,4) = 15$, $W(1,5) = 16$, $W(1,6) = 19$. $OPT(2) = 66$, $OPT(3) = 42$, $OPT(4) = 36$, $OPT(5) = 24$, $OPT(6) = 18$. Set $OPT(7) := 0$.

Step 2. For $j = 1$, $p_1 = 1 \le d = 4$. Then, $F_1 = OPT(2) + W(2,6)p_1 = 78$. For $j = 2$, $p_2 = 2 \le d = 4$. Then, $F_2 = OPT(3) + W(3,6)p_2 = 56$. For $j = 3$, $p_3 = 3 \le d = 4$. Then, $F_3 = OPT(4) + W(4,6)p_3 = 54$. For $j = 4$, $p_4 = 4 \le d = 4$. Then, $F_4 = OPT(5) + W(5,6)p_4 = 40$. For $j = 5$, $p_5 = 5 > d = 4$. Then, $F_5 = OPT(6) + W(1,6)p_5 = 113$. For $j = 6$, $p_6 = 6 > d = 4$. Then, $F_6 = OPT(7) + W(1,6)p_6 = 114$.

Step 3. The optimal value is $F_4 = 40$, and the corresponding SPT-batch is $\pi = (B_1(\pi), B_2(\pi))$, where $B_1(\pi) = \{J_1, J_2, J_3, J_4\}$ and $B_2(\pi) = \{J_5, J_6\}$.

4. **Conclusions.** In this paper, we study the unbounded parallel-batch scheduling on a single machine to minimize the total weighted tardy span. For $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$, we show its NP-hardness and present three $O(n^2 P)$-time algorithms. In addition, we present computational experiments to compare the performance of these algorithms. For $1|\text{PB}, b = +\infty, r_j, \text{fix-p(k)}|\sum w_j C_j U_j$, we present an $O(n^{2k+2})$-time algorithm, where k, that is, the number of different job processing times, is a constant. For $1|\text{PB}, b = +\infty, d_j = d|\sum w_j C_j U_j$, we present an $O(n \log n)$-time algorithm.

For further research, we suggest the following topics:

(i) Design effective approximation algorithms for $1|\text{PB}, b = +\infty|\sum w_j C_j U_j$.

(ii) Study the complexity of $1|\text{PB}, b = +\infty|\sum C_j U_j$.

Acknowledgments. We sincerely thank the Assistant Editor and three anonymous referees for their constructive comments and helpful suggestions on an early version of our paper. This research was supported in part by the China Postdoctoral Science Foundation under grant number 2023M743210, the National Natural Science Foundation of China under grant numbers 12271491 and 12371318, and the Foundation of Henan Educational Committee under grant number 25A110014.

REFERENCES

[1] J. E. C. Arroyo and J. Y.-T. Leung, An effective iterated greedy algorithm for scheduling unrelated parallel batch machines with non-identical capacities and unequal ready times, *Computers & Industrial Engineering*, **105** (2017), 84-100.

[2] P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn and S. L. Van De Velde, Scheduling a batching machine, *Journal of Scheduling*, **1** (1998), 31-54.

[3] M. Cabo, J. L. González-Velarde, E. Possani and Y. Á. R. Solís, Bi-objective scheduling on a restricted batching machine, *Computers & Operations Research*, **100** (2018), 201-210.

[4] R. B. Chen, J. J. Yuan, Q. L. Zhao and Z. H. Zhou, Single-machine scheduling for minimizing total weighted completion time of tardy jobs, In submission, (2024).

[5] T. C. E. Cheng, Z. H. Liu and W. Yu, Scheduling jobs with release dates and deadlines on a batch processing machine, *IIE Transactions*, **33** (2001), 685-690.

[6] T. C. E. Cheng, C. T. Ng, J. J. Yuan and Z. H. Liu, Single machine parallel batch scheduling subject to precedence constraints, *Naval Research Logistics*, **51** (2004), 949-958.

[7] X. T. Deng, H. D. Feng, P. X. Zhang, Y. Z. Zhang and H. Zhu, Minimizing mean completion time in a batch processing system, *Algorithmica*, **38** (2004), 513-528.

[8] B. Q. Fan, J. J. Yuan and S. S. Li, Bi-criteria scheduling on a single parallel-batch machine, *Applied Mathematical Modelling*, **36** (2012), 1338-1346.

[9] Y. Gao, J. J. Yuan and Z. G. Wei, Unbounded parallel-batch scheduling with drop-line tasks, *Journal of Scheduling*, **22** (2019), 449-463.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Vol. 174. San Francisco: Freeman, 1979.

[11] Z. C. Geng and J. J. Yuan, A note on unbounded parallel-batch scheduling, *Information Processing Letters*, **115** (2015), 969-974.

[12] Z. C. Geng and J. J. Yuan, Scheduling family jobs on an unbounded parallel-batch machine to minimize makespan and maximum flow time, *Journal of Industrial and Management Optimization*, **14** (2018), 1479-1500.

[13] C. He and L. Li, Hierarchical optimization on an unbounded parallel-batching machine, *RAIRO-Operations Research*, **52** (2018), 55-60.

[14] C. He, H. Lin, J. J. Yuan and Y. D. Mu, Batching machine scheduling with bicriteria: Maximum cost and makespan, *Asia-Pacific Journal of Operational Research*, **31** (2014), 1450025, 10 pp.

[15] C. He, J. Wu, J. L. Xu and J. L. Wang, An improved algorithm on unbounded parallel-batching scheduling to minimize maximum cost and makespan, *RAIRO-Operations Research*, **57** (2023), 731-741.

[16] Z. H. Jia, Y. Wang, C. Wu, Y. Yang, X. Y. Zhang and H. P. Chen, Multi-objective energy-aware batch scheduling using ant colony optimization algorithm, *Computers & Industrial Engineering*, **131** (2019), 41-56.

[17] M. Y. Kovalyov and D. Šešok, Two-agent scheduling with deteriorating jobs on a single parallel-batching machine: Refining computational complexity, *Journal of Scheduling*, **22** (2019), 603-606.

[18] C.-Y. Lee, Minimizing makespan on a single batch processing machine with dynamic job arrivals, *International Journal of Production Research*, **37** (1999), 219-236.

[19] C.-Y. Lee, R. Uzsoy and L. A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations, *Operations Research*, **40** (1992), 764-775.

[20] D. W. Li and X. W. Lu, Parallel-batch scheduling with deterioration and rejection on a single machine, *Applied Mathematics-A Journal of Chinese Universities*, **35** (2020), 141-156.

[21] C. H. Li, F. Wang, J. N. D. Gupta and T. P. Chung, Scheduling identical parallel batch processing machines involving incompatible families with different job sizes and capacity constraints, *Computers & Industrial Engineering*, **169** (2022), 108115.

[22] J. J. Liu, Z. T. Li, Q. X. Chen and N. Mao, Controlling delivery and energy performance of parallel batch processors in dynamic mould manufacturing, *Computers & Operations Research*, **66** (2016), 116-129.
[23] Z. H Liu, J. J. Yuan and T. C. E. Cheng, On scheduling an unbounded batch machine, *Operations Research Letters*, **31** (2003), 42-48.
[24] L. F. Lu, L. Q. Zhang and J. J. Yuan, The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan, *Theoretical Computer Science*, **396** (2008), 283-289.
[25] J. W. Ou, Near-linear-time approximation algorithms for scheduling a batch-processing machine with setups and job rejection, *Journal of Scheduling*, **23** (2020), 525-538.
[26] O. Ozturk, M. L. Espinouse, M. D. Mascolo and A. Gouin, Makespan minimisation on parallel batch processing machines with non-identical job sizes and release dates, *International Journal of Production Research*, **50** (2012), 6022-6035.
[27] B. Shahidi-Zadeh, R. Tavakkoli-Moghaddam, A. Taheri-Moghadam and I. Rastgar, Solving a bi-objective unrelated parallel batch processing machines scheduling problem: A comparison study, *Computers & Operations Research*, **88** (2017), 71-90.
[28] Z. J. Wang, F. F. Zheng, Y. F. Xu, M. Liu and L. H. Sun, Total weighted tardiness for scheduling MapReduce jobs on parallel batch machines, *Journal of Industrial and Management Optimization*, **19** (2023), 5953-5968.
[29] O. F. Yilmaz and M. B. Durmusoglu, A performance comparison and evaluation of metaheuristics for a batch scheduling problem in a multi-hybrid cell manufacturing system with skilled workforce assignment, *Journal of Industrial and Management Optimization*, **14** (2018), 1219-1249.
[30] J. J. Yuan, Z. H. Liu, C. T. Ng and T. E. Cheng, The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan, *Theoretical Computer Science*, **320** (2004), 199-212.
[31] W. J. Zhao, S. F. Jin and W. Y. Yue, A stochastic model and social optimization of a blockchain system based on a general limited batch service queue, *Journal of Industrial and Management Optimization*, **17** (2021), 1845-1861.
[32] B. Zhang, Q. K. Pan, L. Gao and X. L. Zhang, A hybrid variable neighborhood search algorithm for the hot rolling batch scheduling problem in compact strip production, *Computers & Industrial Engineering*, **116** (2018), 22-36.
[33] H. Zhang, K. Li, C. B. Chu and Z. H. Jia, Parallel batch processing machines scheduling in cloud manufacturing for minimizing total service completion time, *Computers & Operations Research*, **146** (2022), 105899.
[34] L. Q. Zhang, L. F. Lu and C. T. Ng, The unbounded parallel-batch scheduling with rejection, *Journal of the Operational Research Society*, **63** (2012), 293-298.
[35] X. Zheng and Z. Chen, An improved deep Q-learning algorithm for a trade-off between energy consumption and productivity in batch scheduling, *Computers & Industrial Engineering*, **188** (2024), 109925.
[36] S. X. Zheng, N. M. Xie and Q. Wu, Single batch machine scheduling with dual setup times for autoclave molding manufacturing, *Computers & Operations Research*, **133** (2021), 105381.

Received January 2025; revised March 2025; early access June 2025.