Robust scheduling for minimizing maximum lateness on a serial-batch processing machine

Robust scheduling for minimizing maximum lateness on a serial-batch processing machine

Wei Wu[a,*], Liang Tang[b], Andrea Pizzuti[c]

[a]Shizuoka University, 3-5-1, Johoku, Naka-ku, Hamamatsu, 432-8561, Shizuoka, Japan
[b]Dalian Maritime University, No.1 Linghai Road, Dalian, 116026, Liaoning, China
[c]Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, via Brecce Bianche, Ancona, I-60131, Italy

Abstract

We study a robust single-machine scheduling problem with uncertain processing times on a serial-batch processing machine to minimize maximum lateness. The problem can model many practical production and logistics applications which involve uncertain factors such as defect rates. A solution to a batch scheduling problem can be represented as a combination of a job-processing sequence and a partition of this sequence (batch sizing). To solve the problem, we prove that the job ordering rule for the earliest due date is optimal for any uncertainty set. For the batch sizing problem, we propose an exact algorithm based on dynamic programming with the same time complexity as solving the nominal problem.

Keywords: scheduling, batch scheduling, robust optimization, dynamic programming, heuristic algorithm

1. Introduction

In a serial-batch scheduling problem (SSP), the processing time of a batch is equal to the sum of the setup time s and the total processing time of all jobs in that batch, and the completion time of each job is equal to the completion time of the corresponding batch (Potts and Kovalyov, 2000). A solution to an SSP can be represented as a combination of a job-processing sequence and a partition of this sequence (batch sizing). A classical single-machine scheduling problem (without batch decision making) can be regarded as a special case of SSP in which $s = 0$. For the SSP to minimize the maximum lateness, Webster and Baker (1995) proposed a dynamic programming (DP) method with a time complexity of $O(n^2)$, where n is the number of jobs.

In many practical applications, robust services that take into account uncertain processing times are required due to factors such as defective products, unexpected machine interruptions, and a lack of labor. In this paper, we consider robust SSPs (RSSPs) under uncertain processing times modeled as a budgeted uncertainty set with budget parameter Γ ($\leq n$). This is because in many practical cases, the upper limit of the defect rate is given as part of the manufacturing requirements, and the on-site defect rate usually can be predicted from historical data.

A number of studies using robust optimization have been carried out in the fields of scheduling such as project scheduling (Balouka and Cohen, 2021; Bruni et al., 2017; Pass-Lanneau et al., 2023), airline crew scheduling (Antunes et al., 2019; Wen et al., 2020), and flow-shop scheduling (Feng et al., 2016; Goli et al., 2019). Because of numerous practical applications and the tractability of the theoretical analysis, many robust optimization studies have focused on single-machine scheduling. Bold and Goerigk (2022) considered recoverable robust single machine scheduling to minimize the total flow time with uncertain processing times, They provided positive complexity results for some special cases, and proposed a 2-approximation algorithm to the general case. Yue et al. (2018) investigated robust scheduling with uncertain release times for minimizing the maximum waiting time. Lu et al. (2015) considered run-based preventive maintenance decision in scheduling with failure uncertainty.

*Corresponding author. Tel: +81-53-478-1213; fax: +81-53-478-1213; address: 3-5-1, Johoku, Naka-ku, Hamamatsu 432-8561, Japan.
Email addresses: goi@shizuoka.ac.jp (Wei Wu), tangericliang@dlmu.edu.cn (Liang Tang), a.pizzuti@staff.univpm.it (Andrea Pizzuti)

For the serial-batch scheduling on a single machine, we report in Table 1 the theoretical results on complexity of SSPs and RSSPs under different criteria. For the SSP and RSSP with makespan criterion, any single-batch solution

Table 1: Theoretical results on complexity for SSPs and RSSPs under different criteria

	$C_{\max}$	$L_{\max}$	$\sum C_j$	$\sum w_jC_j$	$\sum T_j$	$\sum U_j$
SSP	$O(n)$	$O(n^2)^{\diamond}$	$O(n\log n)^{\dagger}$	$\mathcal{NP}$-hard$^{\flat}$	$\mathcal{NP}$-hard*	$O(n^3)^{\natural}$
RSSP	$O(n)$	$O(n^2)$	$O(n^4\Gamma^2)^{\ddagger}$	$\mathcal{NP}$-hard$^{\flat}$	$\mathcal{NP}$-hard*	$\mathcal{NP}$-hard$^{\sharp}$

$^{\diamond}$Webster and Baker (1995). †Coffman et al. (1990). ‡Wu et al. (2023). $^{\flat}$Albers and Brucker (1993).
*Du and Leung (1990). $^{\natural}$Brucker and Kovalyov (1996). $^{\sharp}$Bougeret et al. (2023).

is optimal, and thus, an optimal solution can be obtained in linear time. Wu et al. (2023) investigated the RSSP to minimize the total flow time, and they proposed an exact algorithm with a time complexity of $O(n^4\Gamma^2)$, where the nominal problem can be solved in $O(n\log n)$ time (Coffman et al., 1990). In contrast, the RSSPs that minimize the total weighted flow time and the total tardiness belong to the $\mathcal{NP}$-hard class, because their nominal problems are known to be $\mathcal{NP}$-hard (see Albers and Brucker (1993) and Du and Leung (1990), respectively). The RSSP to minimize the number of late jobs is $\mathcal{NP}$-hard, because the robust single-machine scheduling problem with the same criterion (Bougeret et al., 2023) is a special case of it, whereas the nominal problem is polynomially solvable (Brucker and Kovalyov, 1996). Therefore, only the RSSP under the maximum lateness minimization criterion has not been clarified whether a polynomial-time algorithm for finding an optimal schedule is possible.

In this paper, we focus on the RSSP to minimize the maximum lateness. We first propose two efficient algorithms to evaluate the worst-case maximum lateness when a solution is provided. We then prove that, for any uncertain set of processing times, there always exists an optimal solution whose processing sequence of jobs is given by the earliest due date (EDD) rule. Based on this observation, we propose an approach based on dynamic programming where each subproblem is still a robust optimization problem. We prove that the proposed approach is an exact algorithm that runs in $O(n^2)$ time. It implies that even after considering the processing time uncertainty, we can obtain the exact solution of the robust optimization problem in the same time as solving the corresponding nominal problem.

2. Problem Description

Given a setup time s (> 0) and a set of jobs $J = \{1, 2, \ldots, n\}$, each job j with a processing time p_j and a due date d_j, we consider a batch scheduling problem to minimize the maximum lateness by searching for a job-processing sequence $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ and a partition of the sequence (batch-size sequence) $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots)$ to divide the sequence into batches. For a solution $(\boldsymbol{\varphi}, \boldsymbol{\pi})$ under processing time $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$, the completion time $C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})}$ of each job j is equal to the maximum completion time of jobs in the same batch. We use $L_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} = C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} - d_j$ to denote the lateness of job j, and the maximum lateness $L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}, \boldsymbol{\pi})$ of the solution $(\boldsymbol{\varphi}, \boldsymbol{\pi})$ is expressed as $L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}, \boldsymbol{\pi}) = \max_{j\in J} L_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})}$. As an example, Figure 1 shows the Gantt chart of a solution with $\boldsymbol{\pi} = (1, 3, 2)$ and $\boldsymbol{\varphi} = (2, 1)$ for an instance in which $n = 3$, $s = 2$, $\boldsymbol{p} = (p_1, p_2, p_3) = (4, 3, 2)$, and $\boldsymbol{d} = (d_1, d_2, d_3) = (6, 12, 7)$. In this example, the maximum lateness of the solution $(\boldsymbol{\varphi}, \boldsymbol{\pi})$ is 2 caused by job 1.

Using three-field notation, the single-machine s-batch scheduling problem (SSP) that minimizes the maximum lateness is presented as $1\,|\,\text{s-batch}\,|\,L_{\max}$. Let Π be the set of all job sequences and $\Phi_n = \{(\varphi_1, \varphi_2, \ldots, \varphi_u) \mid \forall k, \varphi_k \in \mathbb{Z}_{>0}; \sum_{k=1}^{u} \varphi_k = n\}$ be the set of all feasible partitions of the sequence, where $\mathbb{Z}_{>0}$ denotes the set of all positive integers. For the remainder of this paper, we refer to the SSP with maximum lateness criterion simply as SSP, that is,

$$\text{(SSP)} \quad \min_{\boldsymbol{\varphi}\in\Phi_n, \boldsymbol{\pi}\in\Pi} L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}, \boldsymbol{\pi}). \tag{1}$$

In many real-world applications, processing cannot be performed as scheduled due to processing delays caused by defective products, personnel shortages, or poor machine status. For a job j, given a standard processing time $\bar{p}_j$ and a maximum additional time $\hat{p}_j$ (≥ 0), the on-site processing time $p_j = \bar{p}_j + \hat{p}_j\gamma_j$ includes the standard processing time $\bar{p}_j$ and additional time $\hat{p}_j\gamma_j$, where $\gamma_j \in [0, 1]$. Given a budget parameter Γ, we consider the uncertain processing times represented by the following uncertainty set proposed by Bertsimas and Sim (2004):

$$P = \left\{(p_1, p_2, \ldots, p_n) \;\middle|\; \forall j \in J, p_j = \bar{p}_j + \gamma_j\hat{p}_j, 0 \leq \gamma_j \leq 1; \sum_{j\in J} \gamma_j \leq \Gamma\right\}. \tag{2}$$

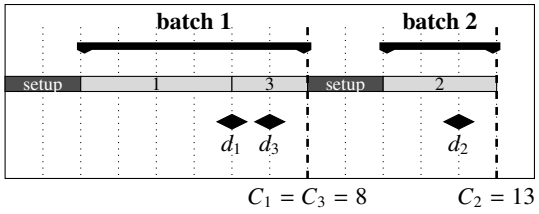

Figure 1: Gantt chart of a batch schedule with 3 jobs

The robust SSP (RSSP) under uncertain processing times can be expressed as

$$\text{(RSSP)} \quad \min_{\boldsymbol{\varphi}\in\Phi_n,\boldsymbol{\pi}\in\Pi} \max_{\boldsymbol{p}\in P} L_{\max}(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi}). \tag{3}$$

3. Worst-Case Analysis

Before designing an algorithm for solving the RSSP, we first introduce two algorithms to obtain the maximum lateness under a worst-case scenario when a solution is given. That is, given a solution $(\boldsymbol{\varphi},\boldsymbol{\pi})$, we consider the following worst-case evaluation problem (WEP):

$$\text{(WEP)} \quad \max_{\boldsymbol{p}\in P} L_{\max}(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi}) = \max_{\boldsymbol{p}\in P}\max_{j\in J}\left\{C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} - d_j\right\} = \max_{j\in J}\max_{\boldsymbol{p}\in P}\left\{C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} - d_j\right\} = \max_{j\in J}\left\{\max_{\boldsymbol{p}\in P} C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} - d_j\right\}. \tag{4}$$

Let $J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})} \subseteq J$ be a set containing all jobs from the first batch to the batch that includes job j. The worst-case completion time of job j (i.e., $\max_{p\in P} C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})}$) is composed of (i) the total setup time, (ii) the total standard processing time of all jobs in $J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$, and (iii) the total worst-case additional processing time of all jobs in $J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$, where the sum of (i) and (ii) can be expressed by $C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})}$. Thus, we can define (iii) as $\Delta_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})} = \max_{\boldsymbol{p}\in P} C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} - C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})}$. By using $\Delta_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$, the WEP (4) becomes

$$\max_{\boldsymbol{p}\in P} L_{\max}(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi}) = \max_{j\in J}\left\{C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})} + \Delta_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})} - d_j\right\}. \tag{5}$$

Because all values of $C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})}$ can be computed in linear time, efficient computation of the value of $\Delta_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$ is the key issue to solve the WEP. From the definition of P in (2), we have

$$\begin{aligned}
\Delta_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})} &= \max_{\boldsymbol{p}\in P} C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\boldsymbol{\pi})} - C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})} \\
&= \max\left\{\sum_{i\in J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}} \gamma_i\hat{p}_i \,\middle|\, \forall i\in J, 0\le\gamma_i\le 1; \sum_{i\in J}\gamma_i \le \Gamma\right\} = \max\left\{\sum_{i\in J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}} \gamma_i\hat{p}_i \,\middle|\, \forall i\in J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}, 0\le\gamma_i\le 1; \sum_{i\in J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}}\gamma_i \le \Gamma\right\} \quad (6)\\
&= \left[\text{The sum of the } \Gamma \text{ largest values of } \hat{p}_i, \text{ where } i\in J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}\right]. \quad (7)
\end{aligned}$$

For solution $(\boldsymbol{\varphi},\boldsymbol{\pi})$, we use $B_k^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$ to denote the set of jobs in the k-th batch. Then, for each job j in $B_k^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$, the set $J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$ can be represented by $J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})} = \bigcup_{k'=1}^{k} B_{k'}^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$.

The first algorithm is shown in Algorithm 1. For each k-th batch, in Steps 3–4, we use J' ($\subseteq J_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$, $j\in B_k^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$) to store Γ jobs with the largest additional processing time $\hat{p}$, and compute the largest lateness L caused by the jobs in the k-th batch in Step 5.

Step 4 is the bottleneck that selects Γ jobs from at most $\Gamma + |B_k^{(\boldsymbol{\varphi},\boldsymbol{\pi})}|$ jobs. Using a linear-time selection algorithm such as the one proposed by Blum et al. (1973), the overall time complexity of Algorithm 1 is $O(|\boldsymbol{\varphi}|\cdot\Gamma + n)$.

We design another algorithm (Algorithm 2) by using a binary heap, which is a *min heap* (The value of the parent node is less than or equal to that of its child nodes), with the following operations:

Algorithm 1 Solve the WEP using a selection algorithm

Set $J' \leftarrow \emptyset$ and $L_{\max} \leftarrow -\infty$.
for $k = 1, 2, \ldots, |\boldsymbol{\varphi}|$ **do**
  $J'' \leftarrow J' \cup B_k^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$.
  $J' \leftarrow \{\text{top } \min\{\Gamma, |J''|\} \text{ jobs in } J'' \text{ in non-increasing order of } \hat{p}_j\}$.
  $L \leftarrow \max_{j \in B_k^{(\boldsymbol{\varphi},\boldsymbol{\pi})}} \left\{ C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})} - d_j \right\} + \sum_{j \in J'} \hat{p}_j$.
  $L_{\max} \leftarrow \max\{L_{\max}, L\}$.
end for
return $L_{\max}$.

- *push*: add a new node;
- *size*: obtain the number of nodes stored in the heap;
- *sum*: obtain the sum of node values stored in the heap;
- *pop_min*: remove the root node.

Algorithm 2 Solve the WEP using a binary heap

Initialize heap H, and set $L_{\max} \leftarrow -\infty$.
for $j = \pi_1, \pi_2, \ldots, \pi_n$ **do**
  $H.push(\hat{p}_j)$.
  if $H.size() > \Gamma$ **then**
    $H.pop\_min()$.
  end if
  $L_j \leftarrow C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi},\boldsymbol{\pi})} + H.sum() - d_j$.
  $L_{\max} \leftarrow \max\{L_{\max}, L_j\}$.
end for
return $L_{\max}$.

The idea is to store in H the value of $\hat{p}$ involved in $\Delta_j^{(\boldsymbol{\varphi},\boldsymbol{\pi})}$. Because the number of elements in H is bounded by Γ, the *push* and *pop_min* operations require $O(\log \Gamma)$ time, whereas the other two operations can be performed in constant time. The time complexity of Algorithm 2 is $O(n \log \Gamma)$.

4. Optimal Job Sequence

We use $\tilde{\boldsymbol{\pi}}$ to denote the EDD sequence in which jobs are sorted according to non-decreasing due date d_j. An optimal solutions to the SSP given by the EDD sequence $\tilde{\boldsymbol{\pi}}$ exists (see Webster and Baker (1995)). We observe that this property also holds for the RSSP.

Lemma 4.1. *For the RSSP, there exists an optimal solution with a job sequence given by the EDD sequencing rule.*

Proof. Consider an arbitrary optimal solution $(\boldsymbol{\varphi}^*, \boldsymbol{\pi}^*)$ of the RSSP. We compare job sequences $\boldsymbol{\pi}^*$ and $\tilde{\boldsymbol{\pi}}$ from tail to head, and let job $j^* = \pi_k^*$ of $\boldsymbol{\pi}^*$ and job $\tilde{j} = \tilde{\pi}_k$ of $\tilde{\boldsymbol{\pi}}$ (the kth jobs in both sequences) be the first occurrence of different jobs, that is, $\tilde{\pi}_j = \pi_j^*$ for each $j = k+1, \ldots, n$ and $\tilde{\pi}_k \neq \pi_k^*$ ($\tilde{j} \neq j^*$). From the definition of $\tilde{\boldsymbol{\pi}}$, we have $d_{\tilde{\pi}_k} = \max_{j=1}^{k} d_{\tilde{\pi}_j} = \max_{j=1}^{k} d_{\pi_j^*}$ and thus

$$d_{\tilde{j}} = d_{\tilde{\pi}_k} = \max_{j=1}^{k} d_{\pi_j^*} \geq d_{\pi_k^*} = d_{j^*}$$

holds. Based on the solution $(\boldsymbol{\varphi}^*, \boldsymbol{\pi}^*)$ in which $\tilde{j}$ necessarily precedes j^*, we consider a solution $(\bar{\boldsymbol{\varphi}}, \bar{\boldsymbol{\pi}})$ by shifting job $\tilde{j}$ to the same batch as job j^* and placing it behind job j^* in the job-processing sequence. Because we shifted job $\tilde{j}$ from the original batch to the later (or the same) batch, all jobs except $\tilde{j}$ got a non-increasing completion time under any scenario $\boldsymbol{p}$, that is,

$$C_j^{(\boldsymbol{p},\bar{\boldsymbol{\varphi}},\bar{\boldsymbol{\pi}})} \le C_j^{(\boldsymbol{p},\boldsymbol{\varphi}^*,\boldsymbol{\pi}^*)} \qquad \forall \boldsymbol{p} \in P, \forall j \in J, j \ne \tilde{j},$$

and thus

$$L_j^{(\boldsymbol{p},\bar{\boldsymbol{\varphi}},\bar{\boldsymbol{\pi}})} \le L_j^{(\boldsymbol{p},\boldsymbol{\varphi}^*,\boldsymbol{\pi}^*)} \qquad \forall \boldsymbol{p} \in P, \forall j \in J, j \ne \tilde{j}. \tag{8}$$

In solution $(\bar{\boldsymbol{\varphi}}, \bar{\boldsymbol{\pi}})$, jobs j^* and $\tilde{j}$ $(j^* \ne \tilde{j})$ are assigned in the same batch, so it holds that

$$C_{\tilde{j}}^{(\boldsymbol{p},\bar{\boldsymbol{\varphi}},\bar{\boldsymbol{\pi}})} = C_{j^*}^{(\boldsymbol{p},\bar{\boldsymbol{\varphi}},\bar{\boldsymbol{\pi}})} \le C_{j^*}^{(\boldsymbol{p},\boldsymbol{\varphi}^*,\boldsymbol{\pi}^*)} \qquad \forall \boldsymbol{p} \in P.$$

From $d_{\tilde{j}} \ge d_{j^*}$, we have

$$L_{\tilde{j}}^{(\boldsymbol{p},\bar{\boldsymbol{\varphi}},\bar{\boldsymbol{\pi}})} \le L_{j^*}^{(\boldsymbol{p},\boldsymbol{\varphi}^*,\boldsymbol{\pi}^*)} \qquad \forall \boldsymbol{p} \in P. \tag{9}$$

Combining (8) and (9), we have

$$L_{\max}(\boldsymbol{p}, \bar{\boldsymbol{\varphi}}, \bar{\boldsymbol{\pi}}) = \max_{j \in J} L_j^{(\boldsymbol{p},\bar{\boldsymbol{\varphi}},\bar{\boldsymbol{\pi}})} \le \max_{j \in J} L_j^{(\boldsymbol{p},\boldsymbol{\varphi}^*,\boldsymbol{\pi}^*)} = L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}^*, \boldsymbol{\pi}^*) \qquad \forall \boldsymbol{p} \in P,$$

which implies $\max_{\boldsymbol{p} \in P} L_{\max}(\boldsymbol{p}, \bar{\boldsymbol{\varphi}}, \bar{\boldsymbol{\pi}}) \le \max_{\boldsymbol{p} \in P} L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}^*, \boldsymbol{\pi}^*)$. Thus, solution $(\bar{\boldsymbol{\varphi}}, \bar{\boldsymbol{\pi}})$ is also optimal. Iterating this shift operation leads to an optimal solution with jobs processed in the EDD sequence. □

Note that the proof of Lemma 4.1 does not depend on any property of the uncertainty set P, so Lemma 4.1 also holds for any uncertainty set other than (2).

Because there exists an optimal solution for the RSSP whose job-processing sequence is given by the EDD sequence $\tilde{\boldsymbol{\pi}}$, for the convenience of description, we assume that jobs are sorted and numbered in the EDD order (i.e., $d_1 \le d_2 \le \ldots \le d_n$). For a batch-size sequence $\boldsymbol{\varphi}$ under scenario $\boldsymbol{p}$, notations $C_j^{(\boldsymbol{p},\boldsymbol{\varphi},\tilde{\boldsymbol{\pi}})}$, $\Delta_j^{(\boldsymbol{\varphi},\tilde{\boldsymbol{\pi}})}$, $B_k^{(\boldsymbol{\varphi},\tilde{\boldsymbol{\pi}})}$, and $L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}, \tilde{\boldsymbol{\pi}})$ can be simplified as $C_j^{(\boldsymbol{p},\boldsymbol{\varphi})}$, $\Delta_j^{(\boldsymbol{\varphi})}$, $B_k^{(\boldsymbol{\varphi})}$, and $L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi})$, respectively. Consequently, by using the observation in (5), the RSSP in (3) can be reduced to a robust batch-sizing problem:

$$\text{(RSSP)} \quad \min_{\boldsymbol{\varphi} \in \Phi_n} \max_{\boldsymbol{p} \in P} L_{\max}(\boldsymbol{p}, \boldsymbol{\varphi}) = \min_{\boldsymbol{\varphi} \in \Phi_n} \max_{j \in J} \left\{ C_j^{(\bar{\boldsymbol{p}},\boldsymbol{\varphi})} + \Delta_j^{(\boldsymbol{\varphi})} - d_j \right\}. \tag{10}$$

5. Exact Algorithm

In this section, we newly define Δ_j as

$$\Delta_j = \max \left\{ \sum_{i=1}^{j} \gamma_i \hat{p}_i \,\middle|\, \gamma_i \in [0,1], \sum_{i=1}^{j} \gamma_i \le \Gamma \right\} \qquad \forall j \in J, \tag{11}$$

which is a value independent of solution $\boldsymbol{\varphi}$. For a solution $\boldsymbol{\varphi}$ containing a batch composed of tasks $j, j+1, \ldots, j'-1$, comparing (6) and (11), it is clear that for each $i = j, j+1, \ldots, j'-1$, equation $\Delta_i^{(\boldsymbol{\varphi})} = \Delta_{j'-1}$ holds. With the EDD property, we can further obtain

$$\max_{i=j}^{j'-1} \left\{ \Delta_i^{(\boldsymbol{\varphi})} - d_i \right\} = \Delta_{j'-1} - d_j. \tag{12}$$

Using this property, we propose a dynamic programming algorithm for the RSSP as follows. let $[\boldsymbol{x}, \boldsymbol{y}]$ denote the concatenation of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, and we define $f(j)$ as the optimal value for the following subproblem in which $j \in J$ is the first job in some batch:

$$f(j) = \min_{\substack{\boldsymbol{\varphi}' \in \Phi_{j-1}, \\ \boldsymbol{\varphi}'' \in \Phi_{n-j+1}}} \max_{i=j}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',\boldsymbol{\varphi}''])} - C_{j-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',\boldsymbol{\varphi}''])} + \Delta_i^{([\boldsymbol{\varphi}',\boldsymbol{\varphi}''])} - d_i \right\}. \tag{13}$$

Letting $C_0^{(\cdot,\cdot)} = 0$, we know from (10) that $f(1)$ is the optimal value of the RSSP, where $\boldsymbol{\varphi}'$ is empty. To analyze the expression in (13), we consider splitting $\boldsymbol{\varphi}''$ into two subsequences: a single-batch sequence containing jobs from j to $j'-1$, and a sequence $\boldsymbol{\varphi}'''$ containing the rest of the jobs. Note that $\boldsymbol{\varphi}'''$ can be empty. Thus, (13) becomes

$$
\begin{aligned}
f(j) &= \min_{j'=j+1}^{n+1} \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max_{i=j}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\} \\
&= \min_{j'=j+1}^{n+1} \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max \left\{ \max_{i=j}^{j'-1} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\}, \right. \\
&\qquad\qquad \left. \max_{i=j'}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\} \right\} \\
&= \min_{j'=j+1}^{n+1} \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max \left\{ \max_{i=j}^{j'-1} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\}, \right. \\
&\qquad\qquad \left. \max_{i=j'}^{n} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j'-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\} \right\} \right\}.
\end{aligned}
$$

Because the value of $s + \sum_{i'=j}^{j'-1} \bar{p}_{i'}$ only depends on j and j', $f(j)$ can be further expressed as

$$
\begin{aligned}
f(j) = \min_{j'=j+1}^{n+1} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max \left\{ \max_{i=j}^{j'-1} \left\{ \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\}, \right. \right. \\
\left. \left. \max_{i=j'}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j'-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\} \right\} \right\}.
\end{aligned}
$$

Using the result in (12), we have

$$
f(j) = \min_{j'=j+1}^{n+1} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max \left\{ \Delta_{j'-1} - d_j, \max_{i=j'}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j'-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\} \right\} \right\}.
$$

This time, the value of $\Delta_{j'-1} - d_j$ does not depend on either $\boldsymbol{\varphi}'$ or $\boldsymbol{\varphi}'''$, and it holds that

$$
f(j) = \min_{j'=j+1}^{n+1} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + \max \left\{ \Delta_{j'-1} - d_j, \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max_{i=j'}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j'-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - d_i \right\} \right\} \right\}.
$$

For $i = j', j'+1, \ldots, n$, the values of $C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])} - C_{j'-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])}$ and $\Delta_i^{([\boldsymbol{\varphi}',j'-j,\boldsymbol{\varphi}'''])}$ only depends on $\boldsymbol{\varphi}'''$, so we have

$$
f(j) = \min_{j'=j+1}^{n+1} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + \max \left\{ \Delta_{j'-1} - d_j, \min_{\substack{\boldsymbol{\varphi}'\in\Phi_{j'-1},\\ \boldsymbol{\varphi}'''\in\Phi_{n-j'+1}}} \max_{i=j'}^{n} \left\{ C_i^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',\boldsymbol{\varphi}'''])} - C_{j'-1}^{(\bar{\boldsymbol{p}},[\boldsymbol{\varphi}',\boldsymbol{\varphi}'''])} + \Delta_i^{([\boldsymbol{\varphi}',\boldsymbol{\varphi}'''])} - d_i \right\} \right\} \right\}.
$$

From the definition in (13), we finally obtain the following recursive formula:

$$
f(j) = \min_{j'=j+1}^{n+1} \left\{ s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + \max \left\{ \Delta_{j'-1} - d_j, f(j') \right\} \right\}. \tag{14}
$$

Formula (14) provides us with another understanding of the subproblem. Let us consider $d_j - \Delta_{j'-1}$ $(j < j')$ as a *modified* due date that involves the additional processing times from job 1 to job $j'-1$, where job j and job $j'-1$

are the first and the last jobs in the same batch, respectively. Then, $f(j)$ can be seen as the optimal value of the subproblem with standard processing times and modified due dates where the start time is 0, and the job set is limited to $\{j, j+1, \ldots, n\}$. In the subproblem, if the first batch consists of jobs $j, j+1, \ldots, j'-1$, then the maximum lateness of the jobs in the first batch is

$$s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} - (d_j - \Delta_{j'-1}), \tag{15}$$

and the maximum lateness of other jobs $j', j'+1, \ldots, n$ can be represented by

$$s + \sum_{i'=j}^{j'-1} \bar{p}_{i'} + f(j'). \tag{16}$$

The maximum lateness for all jobs is the larger value between (15) and (16). Thus, $f(j)$ can be obtained by minimizing the maximum lateness considering all the cases for the first batch, that is, $j' = j+1, \ldots, n+1$.

By setting the boundary condition to $f(n+1) = -\infty$, the optimal value $f(1)$ can be obtained by backward computation for $j = n, n-1, \ldots, 1$ according to (14). By preprocessing all values of $\sum_{i=j}^{j'-1} \bar{p}_i$ and $\Delta_{j'-1}$, surprisingly, we achieved the same time and space complexity $O(n^2)$ as solving the nominal problem SSP.

To better illustrate the DP method, consider the following example with four jobs.

$$n = 4; \quad \Gamma = 2; \quad s = 1; \qquad (d_1, d_2, d_3, d_4) = (1, 6, 7, 9);$$
$$(\bar{p}_1, \bar{p}_2, \bar{p}_3, \bar{p}_4) = (1, 2, 2, 2); \qquad (\hat{p}_1, \hat{p}_2, \hat{p}_3, \hat{p}_4) = (1, 2, 2, 1).$$

According to (11), we can obtain

$$\Delta_1 = \hat{p}_1 = 1, \qquad \Delta_2 = \hat{p}_1 + \hat{p}_2 = 3, \qquad \Delta_3 = \hat{p}_2 + \hat{p}_3 = 4, \qquad \Delta_4 = \hat{p}_2 + \hat{p}_3 = 4.$$

From the boundary condition $f(5) = -\infty$, we apply the recursive formulation (14) to compute each $f(j)$ as follows:

$$\begin{aligned}
f(4) &= s + \bar{p}_4 + \max\{\Delta_4 - d_4, f(5)\} = -2 && \leftarrow \text{optimal batch-size solution: } (1)\\
f(3) &= \min\{s + \bar{p}_3 + \max\{\Delta_3 - d_3, f(4)\} = 1, && \leftarrow \text{optimal batch-size solution: } (1,1)\\
&\qquad s + \sum_{j'=3}^{4} \bar{p}_{j'} + \max\{\Delta_4 - d_3, f(5)\} = 2\}\\
&= 1\\
f(2) &= \min\{s + \bar{p}_2 + \max\{\Delta_2 - d_2, f(3)\} = 4,\\
&\qquad s + \sum_{j'=2}^{3} \bar{p}_{j'} + \max\{\Delta_3 - d_2, f(4)\} = 3, && \leftarrow \text{optimal batch-size solution: } (2,1)\\
&\qquad s + \sum_{j'=2}^{4} \bar{p}_{j'} + \max\{\Delta_4 - d_2, f(5)\} = 5\}\\
&= 3\\
f(1) &= \min\{s + \bar{p}_1 + \max\{\Delta_1 - d_1, f(2)\} = 5, && \leftarrow \text{optimal batch-size solution: } (1,2,1)\\
&\qquad s + \sum_{j'=1}^{2} \bar{p}_{j'} + \max\{\Delta_2 - d_1, f(3)\} = 6,\\
&\qquad s + \sum_{j'=1}^{3} \bar{p}_{j'} + \max\{\Delta_3 - d_1, f(4)\} = 9,
\end{aligned}$$

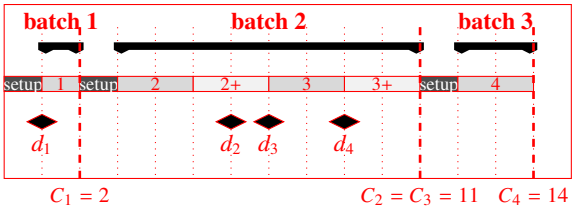

Figure 2: The optimal solution obtained by the DP method

$$s + \sum_{j'=1}^{4} \bar{p}_{j'} + \max\{\Delta_4 - d_1, f(5)\} = 11\}$$

$$= 5$$

From the backward computation, we obtain the optimal value of 5 for this example, and the corresponding optimal solution of batch sizing is $(1, 2, 1)$. A visualization of the optimal solution is shown in Figure 2. In this solution, the worst-case scenario, where additional time occurs in jobs 2 and 3 (marked as '2+' and '3+'), results in the maximum lateness of 5 (caused by jobs 2 and 4).

Note that by slightly extending the definition of Δ_j in (11), the proposed method can also be applied to cases where Γ is a positive non-integer.

6. Conclusion

We studied the robust serial-batch scheduling problem (RSSP) to minimize the maximum lateness, and we proposed an exact polynomial-time algorithm with the same time complexity as solving the nominal problem. In designing the algorithm, we showed that the job sequence by the earliest due date rule is optimal not only for the budgeted uncertainty set, but also for any other uncertainty set. We hope that such a property can contribute to the study of RSSPs under other uncertainty sets. Finally, the problem considering parallel-batch would be another interesting research direction.

Acknowledgements

This work was supported by JSPS KAKENHI [Grant No. 21K14367], and the National Natural Science Foundation of China (72372015). We are grateful to the anonymous reviewers for their helpful comments on the first version of the manuscript.

References

Albers, S., Brucker, P., 1993. The complexity of one-machine batching problems. Discrete Applied Mathematics 47, 87–107.
Antunes, D., Vaze, V., Antunes, A.P., 2019. A robust pairing model for airline crew scheduling. Transportation science 53, 1751–1771.
Balouka, N., Cohen, I., 2021. A robust optimization approach for the multi-mode resource-constrained project scheduling problem. European Journal of Operational Research 291, 457–470.
Bertsimas, D., Sim, M., 2004. The price of robustness. Operations Research 52, 35–53.
Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., Tarjan, R.E., et al., 1973. Time bounds for selection. Journal of Computer and System Sciences 7, 448–461.
Bold, M., Goerigk, M., 2022. Investigating the recoverable robust single machine scheduling problem under interval uncertainty. Discrete Applied Mathematics 313, 99–114.
Bougeret, M., Pessoa, A., Poss, M., 2023. Single machine robust scheduling with budgeted uncertainty. Operations Research Letters 51, 137–141.
Brucker, P., Kovalyov, M.Y., 1996. Single machine batch scheduling to minimize the weighted number of late jobs. Mathematical Methods of Operations Research 43, 1–8.
Bruni, M.E., Pugliese, L.D.P., Beraldi, P., Guerriero, F., 2017. An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. Omega 71, 66–84.
Coffman, E.G., Yannakakis, M., Magazine, M.J., Santos, C., 1990. Batch sizing and job sequencing on a single machine. Annals of Operations Research 26, 135–147.

Du, J., Leung, J.Y.T., 1990. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15, 483–495.
Feng, X., Zheng, F., Xu, Y., 2016. Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times. International Journal of Production Research 54, 3706–3717.
Goli, A., Babaee Tirkolaee, E., Soltani, M., 2019. A robust just-in-time flow shop scheduling problem with outsourcing option on subcontractors. Production & Manufacturing Research 7, 294–315.
Lu, Z., Cui, W., Han, X., 2015. Integrated production and preventive maintenance scheduling for a single machine with failure uncertainty. Computers & Industrial Engineering 80, 236–244.
Pass-Lanneau, A., Bendotti, P., Brunod-Indrigo, L., 2023. Exact and heuristic methods for anchor-robust and adjustable-robust rcpsp. Annals of Operations Research , 1–34.
Potts, C.N., Kovalyov, M.Y., 2000. Scheduling with batching: A review. European Journal of Operational Research 120, 228–249.
Webster, S., Baker, K.R., 1995. Scheduling groups of jobs on a single machine. Operations Research 43, 692–703.
Wen, X., Ma, H.L., Chung, S.H., Khan, W.A., 2020. Robust airline crew scheduling with flight flying time variability. Transportation Research Part E: Logistics and Transportation Review 144, 102132.
Wu, W., Hayashi, T., Haruyasu, K., Tang, L., 2023. Exact algorithms based on a constrained shortest path model for robust serial-batch and parallel-batch scheduling problems. European Journal of Operational Research 307, 82–102.
Yue, F., Song, S., Zhang, Y., Gupta, J.N., Chiong, R., 2018. Robust single machine scheduling with uncertain release times for minimising the maximum waiting time. International Journal of Production Research 56, 5576–5592.