International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

Hedge against total flow time uncertainty of the uniform parallel machine scheduling problem with interval data

Xiaoqing Xu, Jun Lin & Wentian Cui

To cite this article: Xiaoqing Xu, Jun Lin & Wentian Cui (2014) Hedge against total flow time uncertainty of the uniform parallel machine scheduling problem with interval data, International Journal of Production Research, 52:19, 5611-5625, DOI: 10.1080/00207543.2014.887865

To link to this article: https://doi.org/10.1080/00207543.2014.887865

Published online: 26 Feb 2014.

International Journal of Production Research, 2014
Vol. 52, No. 19, 5611–5625, http://dx.doi.org/10.1080/00207543.2014.887865

Taylor & Francis
Taylor & Francis Group

Hedge against total flow time uncertainty of the uniform parallel machine scheduling problem with interval data

Xiaoqing Xu, Jun Lin* and Wentian Cui

School of Management, Xi'an Jiaotong University, Xi'an, China

(*Received 9 August 2013; accepted 20 January 2014*)

We consider a total flow time minimisation problem of uniform parallel machine scheduling when job processing times are only known to be bounded within certain given intervals. A minmax regret model is proposed to identify a robust schedule that minimises the maximum deviation from the optimal total flow time over all possible realisations of the job processing times. To solve this problem, we first prove that the maximal regret for any schedule can be obtained in polynomial time. Then, we derive a mixed-integer linear programming (MILP) formulation of our problem by exploiting its structural properties. To reduce the computational time, we further transform our problem into a robust single-machine scheduling problem and derive another MILP formulation with fewer variables and constraints. Moreover, we prove that the optimal schedule under the midpoint scenario is a 2-approximation for our minmax regret problem. Finally, computational experiments are conducted to evaluate the performance of the proposed methods.

Keywords: scheduling; total flow time minimisation; uniform parallel machines; minmax regret; uncertain processing time

1. Introduction

In recent years, parallel machine scheduling problems have been widely studied for their theoretical and practical importance (Chekuri and Khanna 2004; Jia and Mason 2009; Ozlen and Azizoglu 2009; Chuang, Liao, and Chao 2010; Kang and Shin 2010; Ozlen and Webster 2010; Hsu, Cheng, and Yang 2011; Tan, Chen, and Zhang 2011; Rustogi and Strusevich 2012). Most studies have focused on problems with deterministic job processing times. However, because of the advent of new technologies, varying consumer tastes, and short product lifecycles, an increasing number of manufacturing systems have started operating in highly uncertain environments where traditional deterministic modelling approaches are inapplicable due to limited information on job processing times (Daniels and Kouvelis 1995; Yang and Yu 2002; Montemanni 2007; Allahverdi and Aydilek 2010). Industrial practitioners often view the failure to consider uncertainty in scheduling processes as a major contributor toward the gap between classical scheduling theories and practice.

Some stochastic approaches for handling scheduling problems under uncertainty are available but in order to use these approaches, certain information on probability distribution is required which can be inferred from a substantial amount of historical data (Weber 1982; Weber, Varaiya, and Walrand 1986; Hamada and Tamaki 1999; Cai, Wu, and Zhou 2009; Ranjbar, Davari, and Leus 2012). However, in highly uncertain environments, all that is available is often an educated guess of the lower and upper bounds of the job processing times because of the absence of historical data (Kasperski and Zieliński 2007, 2010; Allahverdi and Aydilek 2010). Moreover, risk-averse decision-makers are more interested in hedging against the worst-case performance than in optimising the expected performance, especially for one-time jobs (Daniels and Kouvelis 1995; Kouvelis and Yu 1997; Yang and Yu 2002; Lin and Ng 2011). In such cases, a natural criterion called the minmax regret or robust deviation (Kouvelis and Yu 1997; Aissi, Bazgan, and Vanderpooten 2009) can be used to find a solution (Kasperski and Zieliński 2007, 2010).

The minmax regret approach has been applied to numerous combinatorial optimisation problems with interval inputs, such as shortest path (Montemanni and Gambardella 2004, 2005b), spanning tree (Aissi, Bazgan, and Vanderpooten 2005a; Montemanni and Gambardella 2005a; Kasperski and Zieliński 2006; Montemanni 2006), assignment (Aissi, Bazgan, and Vanderpooten 2005b), knapsack (Deineko and Woeginger 2010) and production scheduling problems (Daniels and Kouvelis 1995; Kouvelis, Daniels, and Vairaktarakis 2000; Kasperski 2005; Averbakh 2006; Lebedev and Averbakh 2006; Montemanni 2007; Kasperski and Zieliński 2008; Lu, Lin, and Ying 2012; Xu et al. 2013). Daniels and Kouvelis (1995) have studied minmax regret single-machine scheduling problems with a total flow time criterion

*Corresponding author. Email: linjun@alumni.nus.edu.sg; ljun@mail.xjtu.edu.cn

while Kouvelis, Daniels, and Vairaktarakis (2000) have discussed minmax regret makespan minimisation in a two-machine flow shop. These problems are non-deterministic polynomial-time hard (NP-hard) in the interval scenario case. To solve such problems, heuristics and branch-and-bound algorithms have been developed (Daniels and Kouvelis 1995; Kouvelis, Daniels, and Vairaktarakis 2000). The minmax regret problem aimed at minimising the total flow time on a single machine as discussed by Daniels and Kouvelis (1995) has been further investigated by other researchers. Lebedev and Averbakh (2006) have addressed a case in which the processing time intervals between jobs have the same centre. They have shown that the problem can be solved in $O(n \log n)$ time if the number of jobs is even and is NP-hard if the number of jobs is odd. A mixed-integer linear programming (MILP) formulation of this problem has been proposed by Montemanni (2007) who has further investigated the method of transforming the preprocessing rules into valid inequalities. Kasperski and Zieliński (2008) have proved the fact that the optimal schedule under the midpoint scenario guarantees a 2-approximation of the optimal solution. Scenarios in which uncertain job processing times and sequence-dependent family set-up times are both explicitly represented by the interval data have been dealt with by Lu, Lin, and Ying (2012). The authors have reformulated this problem as a robust constrained shortest path problem that has been solved using a simulated annealing-based heuristic. Kasperski (2005) has studied a minmax regret scheduling problem with maximum lateness criterion and interval processing times of jobs on a single machine. He has also presented an $O(n^4)$ algorithm for constructing robust schedules with a minimum maximal regret. Averbakh (2006) has considered the minmax regret makespan-minimising permutation flow-shop problem with two jobs, m machines, and interval job processing times. An $O(m)$ algorithm has been developed to solve for the optimal schedule. Kasperski, Kurpisz, and Zieliński (2012) have discussed the minmax and minmax regret versions of the two-machine permutation flow-shop problem under processing time uncertainty which is specified as a discrete scenario set. All these studies have focused either on single-machine or on flow-shop scheduling problems. However, the minmax regret versions of parallel machine scheduling problems have been rarely investigated. To the best of our knowledge, only Xu et al. (2013) have addressed a makespan minimisation problem with interval job processing times on identical parallel machines.

In this paper, we investigate a total flow time minimisation problem of uniform parallel machine scheduling assuming that the processing times are uncertain but lie within the intervals. In Section 2, a minmax regret model is proposed to identify a robust schedule that minimises the maximum deviation from the optimal total flow time over all potential realisations of job processing times. In Section 3, to solve this problem, we first prove two properties of the worst-case (regret-maximising) scenarios and the maximum regret for a given schedule. Then, we present two MILP formulations, namely, MILP1 and MILP2, to identify the optimal robust solution. Moreover, we prove that the optimal schedule under the midpoint scenario is a 2-approximation for our minmax regret problem. In Section 4, computational experiments were conducted to investigate the validity and performance of the proposed methods. Finally, Section 5 summarises our findings and discusses the possible directions for future studies.

2. Model construction

We consider a scheduling problem in which n independent and simultaneously available jobs ($J = \{1, 2, \ldots, n\}$) are to be processed on m parallel machines ($M = \{1, 2, \ldots, m\}$). Each machine $j \in M$ can be characterised by means of its processing speed q_j, where $1 = q_1 \leq q_2 \leq \ldots \leq q_m$. The processing time p_{ij} of the job $i \in J$ on machine $j \in M$ is equal to p_i/q_j, where p_i is the standard processing time for job i (i.e. $p_i = p_{i1}$). Each job can be processed by only one machine and each machine can process at the most one job at a time. The only information available on the processing time of each job $i \in J$ is a bounded interval $[\underline{p}_i, \bar{p}_i]$, where $0 \leqslant \underline{p}_i \leqslant \bar{p}_i < +\infty$. Let $\boldsymbol{s} = \{p_1^s, p_2^s, \ldots, p_n^s\}$ be a scenario of job processing times and $p_{ij}^s = p_i^s/q_j$ be the processing time of job $i \in J$ on machine $j \in M$ under scenario $\boldsymbol{s}$, where $p_i^s \in [\underline{p}_i, \bar{p}_i]$. Then, the processing time uncertainty can be modelled using the set of scenarios S which is the Cartesian product of the intervals.

Recall that if job i is processed on machine j and there are $(k-1)$ jobs following it on this machine j, then job i contributes $kp_{ij}^s = kp_i^s/q_j$ to the value of total flow time under scenario $\boldsymbol{s}$ (Conway, Maxwell, and Miller 1967; Pinedo 2008). We represent a schedule with a set of 0/1 variables $\boldsymbol{X}$ such that $x_{ijk} = 1$ if job i is the kth last job processed on machine j, otherwise, it is 0. Clearly, no more than n jobs can be assigned to a machine. Moreover, a feasible schedule should satisfy $\sum_{j=1}^{m} \sum_{k=1}^{n} x_{ijk} = 1$ ($i = 1, 2, \ldots, n$) and $\sum_{i=1}^{n} x_{ijk} \leqslant 1$ ($j = 1, 2, \ldots, m$ and $k = 1, 2, \ldots, n$). That is, each job should be processed exactly once and a maximum of one job can be assigned to each position of every machine. Let Φ be the set of feasible schedules. The total flow time for a schedule $\boldsymbol{X} \in \Phi$ under scenario $\boldsymbol{s}$ is defined as

$$F(\boldsymbol{X}, \boldsymbol{s}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{n} \frac{k}{q_j} p_i^s x_{ijk}. \tag{1}$$

The regret of a schedule $\boldsymbol{X}$ under scenario $\boldsymbol{s}$ is defined as

$$R(\boldsymbol{X},\boldsymbol{s}) = F(\boldsymbol{X},\boldsymbol{s}) - F^*_{\boldsymbol{s}}, \tag{2}$$

where $F^*_{\boldsymbol{s}}$ is the minimum total flow time under scenario $\boldsymbol{s}$. The problem of determining $F^*_{\boldsymbol{s}}$ for a specific scenario $\boldsymbol{s}$ is defined as the deterministic flow time minimisation uniform parallel machine scheduling problem ($Q||\Sigma C_i$):

Problem OPT-U($\boldsymbol{s}$): $F^*_{\boldsymbol{s}} = \min_{\boldsymbol{X}\in\Phi} F(\boldsymbol{X},\boldsymbol{s})$.

The OPT-U($\boldsymbol{s}$) problem can be solved in $O(n \log mn)$ time using the MFT algorithm (Horowitz and Sahni 1976).

The maximum regret for $\boldsymbol{X}$ is then defined as

$$R_{\max}(\boldsymbol{X}) = \max_{\boldsymbol{s}\in S} R(\boldsymbol{X},\boldsymbol{s}). \tag{3}$$

The scenario that maximises the regret across all the possible scenarios is called the worst-case scenario for $\boldsymbol{X}$. The minmax regret (robust) version of $Q||\Sigma C_i$ is

Problem ROB-U: $\min_{\boldsymbol{X}\in\Phi} R_{\max}(\boldsymbol{X})$.

The solution $\boldsymbol{X}^*$ corresponds to the optimal robust schedule.

The ROB-U problem is NP-hard because the minmax regret version of $1||\Sigma C_i$ which is NP-hard (Lebedev and Averbakh 2006) is a special case of this problem. In the next section, we shall solve this problem by developing two MILP formulations. Further, a 2-approximation algorithm is proposed.

3. Solution algorithms

The ROB-U problem can be reformulated as the following mathematical model (F1):

$$\min r \tag{4}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n} \frac{k}{q_j} p_i^{\boldsymbol{s}} x_{ijk} - F^*_{\boldsymbol{s}} \leqslant r, \quad \boldsymbol{s}\in S \tag{5}$$

$$\sum_{j=1}^{m}\sum_{k=1}^{n} x_{ijk} = 1, \quad i = 1,2,\ldots,n \tag{6}$$

$$\sum_{i=1}^{n} x_{ijk} \leqslant 1, \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n \tag{7}$$

$$x_{ijk} \in \{0,1\}, \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n. \tag{8}$$

Objective (4) and constraint (5) ensure that the solution is a robust schedule that minimises the maximum regret. Constraints (6)–(8) guarantee the feasibility of the robust schedule.

This MILP model cannot be solved directly because the number of scenarios is infinite in (5). Therefore, we first characterise the regret-maximising scenario for a given schedule. Based on these results, we can focus on a finite number of extreme scenarios (an extreme scenario is a scenario $\boldsymbol{s} = \{p_i^{\boldsymbol{s}}, i\in J\}$ such that $p_i^{\boldsymbol{s}} \in \{\underline{p}_i, \bar{p}_i\}$).

Let $\pi_i^{\boldsymbol{X}} = \sum_{j=1}^{m}\sum_{k=1}^{n} \frac{k}{q_j} x_{ijk}$ represent the position weight of job i for schedule $\boldsymbol{X}$. If job i is in the kth position from the last job on the machine j (i.e. $x_{ijk}=1$), then $\pi_i^{\boldsymbol{X}} = k/q_j$ because $\sum_{j=1}^{m}\sum_{k=1}^{n} x_{ijk} = 1$. Let $\boldsymbol{s}^{\boldsymbol{X}}$ be the worst-case scenario for $\boldsymbol{X}$. Further, we denote the optimal schedule under $\boldsymbol{s}^{\boldsymbol{X}}$ by $\boldsymbol{Y} = [y_{ihf}]_{n\times m\times n}$ such that $y_{ihf}=1$ if job i is in the fth position from the last job on machine h, otherwise, it is 0. Then, we have $\pi_i^{\boldsymbol{Y}} = \sum_{h=1}^{m}\sum_{f=1}^{n} \frac{f}{q_h} y_{ihf}$ and the maximum regret of schedule $\boldsymbol{X}$ can be expressed as

$$\begin{aligned} R_{\max}(\boldsymbol{X}) &= F(\boldsymbol{X},\boldsymbol{s}^{\boldsymbol{X}}) - F^*_{\boldsymbol{s}^{\boldsymbol{X}}} \\ &= F(\boldsymbol{X},\boldsymbol{s}^{\boldsymbol{X}}) - F(\boldsymbol{Y},\boldsymbol{s}^{\boldsymbol{X}}) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n} \frac{k}{q_j} x_{ijk} p_i^{\boldsymbol{s}^{\boldsymbol{X}}} - \sum_{i=1}^{n}\sum_{h=1}^{m}\sum_{f=1}^{n} \frac{f}{q_h} y_{ihf} p_i^{\boldsymbol{s}^{\boldsymbol{X}}} \\ &= \sum_{i=1}^{n} (\pi_i^{\boldsymbol{X}} - \pi_i^{\boldsymbol{Y}}) p_i^{\boldsymbol{s}^{\boldsymbol{X}}}. \end{aligned} \tag{9}$$

Property 1. For any schedule $\boldsymbol{X} \in \Phi$, there exists a worst-case scenario $\boldsymbol{s}^{\boldsymbol{X}}$ for $\boldsymbol{X}$, such that $p_i^{s^X} = \bar{p}_i$ when $\pi_i^{\boldsymbol{X}} > \pi_i^{\boldsymbol{Y}}$ and $p_i^{s^X} = \underline{p}_i$ when $\pi_i^{\boldsymbol{X}} \leqslant \pi_i^{\boldsymbol{Y}}$.

Proof. Given any schedule $\boldsymbol{X}$, we can calculate $\pi_i^{\boldsymbol{X}}$, where $i \in J$, based on its definition. The maximum regret can then be rewritten as (9), which increases with $p_i^{s^X}$ when $\pi_i^{\boldsymbol{X}} - \pi_i^{\boldsymbol{Y}} > 0$ and is non-increasing with $p_i^{s^X}$ when $\pi_i^{\boldsymbol{X}} - \pi_i^{\boldsymbol{Y}} \leqslant 0$. Therefore, $\boldsymbol{s}^{\boldsymbol{X}}$ for $\boldsymbol{X}$ can be expressed a

$$p_i^{s^X} = \begin{cases} \bar{p}_i, & \text{if } \pi_i^{\boldsymbol{X}} > \pi_i^{\boldsymbol{Y}} \\ \underline{p}_i, & \text{if } \pi_i^{\boldsymbol{X}} \leqslant \pi_i^{\boldsymbol{Y}} \end{cases}, \ i \in J.$$

□

3.1 *MILP1: A mixed-integer linear program of the ROB-U problem*

Using property 1, we can obtain the maximum regret of $\boldsymbol{X}$ by solving the following $n \times mn$ assignment problem (F2):

$$\max \sum_{i=1}^{n}\sum_{h=1}^{m}\sum_{f=1}^{n}\left(\underline{p}_i \sum_{j=1}^{m}\sum_{k=1}^{\lfloor \frac{q_j f}{q_h} \rfloor}\left(\frac{k}{q_j}-\frac{f}{q_h}\right)x_{ijk} + \bar{p}_i \sum_{j=1}^{m}\sum_{k=\lceil \frac{q_j f}{q_h} \rceil}^{n}\left(\frac{k}{q_j}-\frac{f}{q_h}\right)x_{ijk}\right) y_{ihf} \tag{10}$$

$$\text{s.t.} \quad \sum_{h=1}^{m}\sum_{f=1}^{n} y_{ihf} = 1, \quad i = 1, 2, \ldots, n \tag{11}$$

$$\sum_{i=1}^{n} y_{ihf} \leqslant 1, \quad h = 1, 2, \ldots, m; \quad f = 1, 2, \ldots, n \tag{12}$$

$$y_{ihf} \in \{0, 1\}, \quad i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, m; \quad f = 1, 2, \ldots, n. \tag{13}$$

Note that for a given schedule $\boldsymbol{X}$, the value of the objective function (10) can be completely determined by $\boldsymbol{Y} = [y_{ihf}]_{n \times m \times n}$, since the coefficients of variable $\boldsymbol{Y}$ are constants. If the position weight of job i in schedule $\boldsymbol{X}$ is greater than that in $\boldsymbol{Y}$, then, $\bar{p}_i$ is the worst-case scenario and the coefficient of variable y_{ihf} is $\bar{p}_i(k/q_j - f/q_h)$. Otherwise, the coefficient of variable y_{ihf} is $\underline{p}_i(k/q_j - f/q_h)$. A bipartite graph with n job nodes and $m \times n$ position nodes can describe this assignment problem. A job node can be denoted by i ($i = 1, 2, \ldots, n$) and a position node by (h, f)($h = 1, 2, \ldots, m$, $f = 1, 2, \ldots, n$). The weight of the arc connecting i and (h, f) is the coefficient of the variable y_{ihf}. Constraints (11) and (12), together with the integrality constraint (13), guarantee that the schedule $\boldsymbol{Y}$ is feasible.

To the best of our knowledge, the assignment problem can be solved using the Hungarian method. Considering the fact that the complexity of this assignment algorithm is cubic on the number of objects to be assigned, we have the following property.

Property 2. Given any schedule $\boldsymbol{X} \in \Phi$, the maximum regret can be obtained by solving an $n \times mn$ assignment problem in $O((nm)^3)$ time.

Because F2 possesses the total unimodularity property (Pinedo 2008), we can replace the integrality constraint (13) in F2 with the following constraint:

$$y_{ihf} \geqslant 0, \quad i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, m; \quad f = 1, 2, \ldots, n. \tag{14}$$

The resulting problem (referred to as F2′) is a linear program.

To derive an MILP formulation of the ROB-U problem, we first introduce the following mathematical formulation (F3) using the aforementioned results:

$$\min\left(\max \sum_{i=1}^{n}\sum_{h=1}^{m}\sum_{f=1}^{n}\left(\underline{p}_i \sum_{j=1}^{m}\sum_{k=1}^{\lfloor \frac{q_j f}{q_h} \rfloor}\left(\frac{k}{q_j}-\frac{f}{q_h}\right)x_{ijk} + \bar{p}_i \sum_{j=1}^{m}\sum_{k=\lceil \frac{q_j f}{q_h} \rceil}^{n}\left(\frac{k}{q_j}-\frac{f}{q_h}\right)x_{ijk}\right) y_{ihf}\right) \tag{15}$$

$$\text{s.t.} \quad \sum_{j=1}^{m}\sum_{k=1}^{n} x_{ijk} = 1, \quad i = 1,2,\ldots,n \tag{16}$$

$$\sum_{i=1}^{n} x_{ijk} \leqslant 1, \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n \tag{17}$$

$$x_{ijk} \in \{0,1\}, \quad i = 1,2,...,n; \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n \tag{18}$$

$$\sum_{h=1}^{m}\sum_{f=1}^{n} y_{ihf} = 1, \quad i = 1,2,\ldots,n \tag{19}$$

$$\sum_{i=1}^{n} y_{ihf} \leqslant 1, \quad h = 1,2,\ldots,m; \quad f = 1,2,\ldots,n \tag{20}$$

$$y_{ihf} \geqslant 0, \quad i = 1,2,\ldots,n; \quad h = 1,2,\ldots,m; \quad f = 1,2,\ldots,n. \tag{21}$$

Note that the inner max problem in the objective function, along with the constraints (19)–(21), is the F2′ problem defined on the instance of $\boldsymbol{X}$ regulated by the outer min operator. By exploiting the duality property of the inner F2′ problem, we can transform F3 into a mixed-integer linear program.

Property 3. The ROB-U problem can be reformulated as the following MILP problem F4:

$$\min \sum_{i=1}^{n} \eta_i + \sum_{h=1}^{m}\sum_{f=1}^{n} \tau_{hf}$$

$$\text{s.t.} \quad \sum_{j=1}^{m}\sum_{k=1}^{n} x_{ijk} = 1, \quad i = 1,2,\ldots,n$$

$$\sum_{i=1}^{n} x_{ijk} \leqslant 1, \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n$$

$$\eta_i + \tau_{hf} \geqslant \underline{p}_i \sum_{j=1}^{m}\sum_{k=1}^{\lfloor \frac{q_j f}{q_h} \rfloor} \left(\frac{k}{q_j} - \frac{f}{q_h}\right) x_{ijk} + \bar{p}_i \sum_{j=1}^{m}\sum_{k=\lceil \frac{q_j f}{q_h} \rceil}^{n} \left(\frac{k}{q_j} - \frac{f}{q_h}\right) x_{ijk}$$

$$i = 1,2,\ldots,n; \quad h = 1,2,\ldots,m; \quad f = 1,2,\ldots,n$$

$$x_{ijk} \in \{0,1\}, \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n$$

$$\eta_i \in R, \quad i = 1,2,\ldots,n$$

$$\tau_{hf} \geqslant 0, \quad h = 1,2,...,m; \quad f = 1,2,\ldots,n.$$

Proof. Since F2′ is a linear programming problem for which strong duality holds, it can be transformed into the following dual problem:

$$\min \sum_{i=1}^{n} \eta_i + \sum_{h=1}^{m}\sum_{f=1}^{n} \tau_{hf} \tag{22}$$

$$\text{s.t.}\quad \eta_i + \tau_{hf} \geqslant \underline{p}_i \sum_{j=1}^{m} \sum_{k=1}^{\lfloor \frac{q_j f}{q_h} \rfloor} \left(\frac{k}{q_j} - \frac{f}{q_h}\right) x_{ijk} + \bar{p}_i \sum_{j=1}^{m} \sum_{k=\lceil \frac{q_j f}{q_h} \rceil}^{n} \left(\frac{k}{q_j} - \frac{f}{q_h}\right) x_{ijk},$$

$$i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, m; \quad f = 1, 2, \ldots, n \tag{23}$$

$$\eta_i \in R, \quad i = 1, 2, \ldots, n \tag{24}$$

$$\tau_{hf} \geqslant 0, \quad h = 1, 2, ..., m; \quad f = 1, 2, \ldots, n\,. \tag{25}$$

By replacing F2′ with its dual in F3, we obtain the MILP problem F4. □

F4 has $n^2 m + nm + n$ variables and constraints and can be solved using readily available optimisation software; thus, an easy-to-use exact algorithm for solving the robust uniform parallel machine scheduling problem with interval data has been derived.

3.2 *MILP2: Transforming ROB-U into a robust single-machine problem*

In this section, we develop a method to transform ROB-U into a robust single-machine problem (ROB-S) by exploiting the properties of the optimal number of jobs assigned to each machine in order to reduce the computational time required to solve ROB-U. Let n_j denote the number of jobs assigned to machine j, $j \in M$. Following Eren and Guner (2009) and Hsu, Kuo, and Yang (2011), we define an allocation vector $V(n, m, q_j) = \{n_j, j \in M\}$ to denote the number of jobs allocated to each machine in schedule $\boldsymbol{X}$, where $\sum_{j=1}^{m} n_j = n$.

Let $V'_{\boldsymbol{s}}(n, m, q_j) = \{n'_j, j \in M\}$ denote the optimal allocation vector for $\boldsymbol{s}$, i.e. the allocation vector of an optimal schedule under scenario $\boldsymbol{s}$. If $V'_{\boldsymbol{s}^X}(n, m, q_h) = \{n'_h, h \in M\}$ is known in advance, F2′ can be reformulated as the following $n \times n$ assignment problem (F5) instead of an $n \times mn$ assignment problem:

$$\max \sum_{i=1}^{n} \sum_{h=1}^{m} \sum_{f=1}^{n'_h} \left(\underline{p}_i \sum_{j=1}^{m} \sum_{k=1}^{\lfloor \frac{q_j f}{q_h} \rfloor} \left(\frac{k}{q_j} - \frac{f}{q_h}\right) x_{ijk} + \bar{p}_i \sum_{j=1}^{m} \sum_{k=\lceil \frac{q_j f}{q_h} \rceil}^{n_j} \left(\frac{k}{q_j} - \frac{f}{q_h}\right) x_{ijk} \right) y_{ihf} \tag{26}$$

$$\text{s.t.} \quad \sum_{h=1}^{m} \sum_{f=1}^{n'_h} y_{ihf} = 1, \quad i = 1, 2, \ldots, n \tag{27}$$

$$\sum_{i=1}^{n} y_{ihf} = 1, \quad h = 1, 2, \ldots, m; \quad f = 1, 2, \ldots, n'_h \tag{28}$$

$$y_{ihf} \geqslant 0, \quad i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, m\,; \quad f = 1, 2, \ldots, n'_h\,. \tag{29}$$

Because $\sum_{h=1}^{m} n'_h = n$, the inequality constraint (12) in F2′ is replaced by the equality constraint (28), which ensures that exactly one job is assigned to each position in each machine. A standard bipartite graph with n job nodes and n position nodes can describe this assignment problem. A job node can be denoted by i ($i = 1, 2, \ldots, n$) and a position node by (h, f) ($h = 1, 2, \ldots, m$, $f = 1, 2, \ldots, n'_h$). The weight of the arc connecting i and (h, f) is the coefficient of variable y_{ihf}, which is constant for a given schedule $\boldsymbol{X}$.

The key question is how to determine the optimal allocation vector $V'_{\boldsymbol{s}^X}(n, m, q_h) = \{n'_h, h \in M\}$ for $\boldsymbol{s}^{\boldsymbol{X}}$. We will show below that for a given (n, m, q_h), the optimal allocation vectors for all the scenarios have a common feature and an identical optimal allocation vector independent of the scenario can be obtained using the following allocation algorithm.

Property 4. Given (n, m, q_h), any allocation vector $V(n, m, q_h) = \{n_h, h \in M\}$ that satisfies the following inequality is an optimal allocation vector for all the scenarios:

$$\frac{n_{h1} + 1}{q_{h1}} \geqslant \frac{n_{h2}}{q_{h2}} \quad (\forall h1, h2 \in M, n_{h2} > 0) \tag{30}$$

Proof. Suppose that an optimal allocation vector $V'_{\boldsymbol{s}}(n,m,q_h)=\{n'_h, h\in M\}$ under any scenario $\boldsymbol{s}$ differs from $V(n,\ m,\ q_h)=\{n_h,\ h\in M\}$. Consider any two different machines $h1$ and $h2$ such that $n'_{h1}-n_{h1}\geqslant 1$ and $n_{h2}-n'_{h2}\geqslant 1$. Given $(n_{h1}\ +\ 1)/q_{h1}\geqslant n_{h2}/q_{h2}$, we have

$$\frac{n'_{h1}}{q_{h1}}\geqslant\frac{n_{h1}+1}{q_{h1}}\geqslant\frac{n_{h2}}{q_{h2}}\geqslant\frac{n'_{h2}+1}{q_{h2}}.$$

Move job i in the first position of machine $h1$ in $V'_{\boldsymbol{s}}(n,m,q_h)=\{n'_h, h\in M\}$ to the first position of machine $h2$. Then, the total flow time changes by $[(n'_{h2}+1)/q_{h2}-n'_{h1}/q_{h1}]p^{\boldsymbol{s}}_i\leqslant 0$. Given that $V'_{\boldsymbol{s}}(n,m,q_h)$ is an optimal allocation vector, we have $[(n'_{h2}+1)/q_{h2}-n'_{h1}/q_{h1}]p^{\boldsymbol{s}}_i=0$. By repeating this process, $V'_{\boldsymbol{s}}(n,m,q_h)$ can be transformed into $V(n,\ m,\ q_h)=\{n_h,\ h\in M\}$, with the total flow time remaining unchanged. Thus, $V(n,\ m,\ q_h)=\{n_h,\ h\in M\}$ is also an optimal allocation vector for scenario $\boldsymbol{s}$. □

Based on property 4, we propose the following allocation algorithm to obtain an identical optimal allocation vector $V'(n,\ m,\ q_h)$ for all the scenarios. At each iteration, the algorithm assigns a job to machine l such that $(n_l\ +\ 1)/q_l=\min_{1\leqslant h\leqslant m}\{(n_h\ +\ 1)/q_h\}$. This process guarantees that the obtained optimal allocation vector satisfies (30).

The allocation algorithm

Input: m machines with speeds $1=q_1\leqslant q_2\leqslant\ldots\leqslant q_m$; the number of jobs n.
Output: The number of jobs on machine h, n_h $(1\leqslant h\leqslant m)$.
Step 1: Set $n_m=1$ and $n_h=0$ for $h\leftarrow 1,2,\ldots,m-1$. Let $i_h\leftarrow(n_h\ +\ 1)/q_h$. (Note that at first, the fastest machine m is assigned a job.)
Step 2: While $\sum_{h=1}^{m}n_h<n$
Let l be the largest index such that $i_l=\min_{1\leqslant h\leqslant m}\{i_h\}$;
$n_l\leftarrow n_l\ +\ 1$;
$i_l\leftarrow(n_l\ +\ 1)/q_l$;
end

Step 2 requires $(n\ -\ 1)$ iterations and each iteration requires no more than $(m\ -\ 1)$ comparisons to determine i_l. Horowitz and Sahni (1976) have mentioned that the number of comparisons can be reduced to $O(\log_2 m)$ by using a heap as in Kmith (1973). Therefore, given the values of n, m, and q_h, we can determine $V'(n,m,q_h)=V'_{\boldsymbol{s}^{\boldsymbol{X}}}(n,m,q_h)$ in $O(n\log_2 m)$ time using the allocation algorithm. The above results suggest that the computational complexity of obtaining the maximum regret of any schedule can be significantly reduced.

Property 5. Given $(n,\ m,\ q_h)$, the maximum regret of any schedule $\boldsymbol{X}\in\Phi$ can be obtained in $O(n^3)$ time.

Proof. Given $(n,\ m,\ q_h)$, we can first derive an optimal allocation vector $V'_{\boldsymbol{s}^{\boldsymbol{X}}}(n,m,q_h)$ independent of the scenario $\boldsymbol{s}^{\boldsymbol{X}}$ using the given allocation algorithm. Then, the maximum regret can be obtained by solving an $n\times n$ assignment problem (F5). Recall that solving an assignment problem of size n requires an effort of $O(n^3)$ (using the well-known Hungarian method) and the computational complexity of the allocation algorithm is $O(n\log_2 m)$. Hence, the maximum regret of any schedule can be obtained in $O(n^3)$ time. □

Based on the aforementioned results, the ROB-U problem can be reformulated as the following mathematical model (F6) if the allocation vector $V(n,\ m,\ q_j)=\{n_j, j\in M\}$ of schedule $\boldsymbol{X}$ is given:

$$\min\left(\max\sum_{i=1}^{n}\sum_{h=1}^{m}\sum_{f=1}^{n'_h}\left(\underline{p}_i\sum_{j=1}^{m}\sum_{k=1}^{\lfloor\frac{q_jf}{q_h}\rfloor}\left(\frac{k}{q_j}-\frac{f}{q_h}\right)x_{ijk}+\bar{p}_i\sum_{j=1}^{m}\sum_{k=\lceil\frac{q_jf}{q_h}\rceil}^{n_j}\left(\frac{k}{q_j}-\frac{f}{q_h}\right)x_{ijk}\right)y_{ihf}\right)\tag{31}$$

$$\text{s.t.}\quad\sum_{j=1}^{m}\sum_{k=1}^{n_j}x_{ijk}=1,\quad i=1,2,\ldots,n\tag{32}$$

$$\sum_{i=1}^{n}x_{ijk}=1,\quad j=1,2,\ldots,m;\quad k=1,2,\ldots,n_j\tag{33}$$

$$x_{ijk} \in \{0,1\}, \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,m; \quad k = 1,2,\ldots,n_j \tag{34}$$

$$\sum_{h=1}^{m}\sum_{f=1}^{n'_h} y_{ihf} = 1, \quad i = 1,2,\ldots,n \tag{35}$$

$$\sum_{i=1}^{n} y_{ihf} = 1, \quad h = 1,2,\ldots,m; \quad f = 1,2,\ldots,n'_h \tag{36}$$

$$y_{ihf} \geqslant 0, \quad i = 1,2,\ldots,n; \quad h = 1,2,\ldots,m; \quad f = 1,2,\ldots,n'_h. \tag{37}$$

The values of n'_h ($h = 1, 2, \ldots, m$) corresponding to $\boldsymbol{Y}$, which are an optimal schedule under $\boldsymbol{s}^{\boldsymbol{X}}$, can be obtained using the allocation algorithm. Then, the key problem becomes the determination of the allocation vector $V^*(n,m,q_j) = \{n_j^*, j \in M\}$. of an optimal robust schedule $\boldsymbol{X}^*$.

Property 6. Given (n, m, q_j), the allocation vector $V^*(n,m,q_j) = \{n_j^*, j \in M\}$ of the optimal robust schedule $\boldsymbol{X}^*$ is the same as that of the optimal schedule for the deterministic $Q\|\Sigma C_i$ scheduling problem.

Proof. Let $V'(n,m,q_j) = \{n'_j, j \in M\}$ be the optimal allocation vector to the deterministic $Q\|\Sigma C_i$ scheduling problem. By contradiction, suppose there exists an optimal robust schedule $\boldsymbol{X}^*$ whose allocation vector $V^*(n,m,q_j) = \{n_j^*, j \in M\}$ is different from $V'(n,m,q_j) = \{n'_j, j \in M\}$. Then, we can identify two machines $j1$ and $j2$ such that $n_{j1}^* - n'_{j1} \geqslant 1$ and $n'_{j2} - n_{j2}^* \geqslant 1$. Let i be the first job on $j1$ in $\boldsymbol{X}^*$. Construct a new schedule $\boldsymbol{X}$ by moving the job i to the first position of machine $j2$. If $\boldsymbol{s}_0^*$ and $\boldsymbol{s}_0$ denote the worst-case scenarios for schedules $\boldsymbol{X}^*$ and $\boldsymbol{X}$, respectively, then

$$R(\boldsymbol{X}, \boldsymbol{s}_0) > R(\boldsymbol{X}^*, \boldsymbol{s}_0^*) \geqslant R(\boldsymbol{X}^*, \boldsymbol{s}_0). \tag{38}$$

Given that $(n'_{j1} + 1)/q_{j1} \geqslant n'_{j2}/q_{j2}$ (see (30)), we obtain

$$\frac{n_{j1}^*}{q_{j1}} \geqslant \frac{n'_{j1}+1}{q_{j1}} \geqslant \frac{n'_{j2}}{q_{j2}} \geqslant \frac{n_{j2}^*+1}{q_{j2}}.$$

Then, $F(\boldsymbol{X}, \boldsymbol{s}_0) - F(\boldsymbol{X}^*, \boldsymbol{s}_0) = [(n_{j2}^* + 1)/q_{j2} - n_{j1}^*/q_{j1}]p_i^{\boldsymbol{s}_0} \leqslant 0$. We therefore, derive

$$R(\boldsymbol{X}, \boldsymbol{s}_0) = F(\boldsymbol{X}, \boldsymbol{s}_0) - F_{\boldsymbol{s}_0}^* \leqslant F(\boldsymbol{X}^*, \boldsymbol{s}_0) - F_{\boldsymbol{s}_0}^* = R(\boldsymbol{X}^*, \boldsymbol{s}_0),$$

which contradicts (38).

Property 6 indicates that given the values of n, m and $q_j (j \in M)$, we can obtain the allocation vector of the optimal robust schedule $\boldsymbol{X}^*$ in F6 using the allocation algorithm.

To make the ROB-U problem more succinct, we describe the optimal robust schedule $\boldsymbol{X}^*$ using x_{ig} ($g = 1, 2, \ldots, n$) instead of x_{ijk} and introduce the concept of a position node list to establish the relationship between single machine and uniform parallel machine scheduling problems.

Le σ be the sequence of k/q_j in ascending order and g denote the position of (j, k) in σ, i.e. $\sigma(g) = k/q_j$. We can derive the position node list that provides the relationship between (j, k) and g. Then, the set of n position nodes (j, k) ($k = 1, 2, \ldots, n_j$; $j = 1, 2, \ldots, m$; $\sum_{j=1}^{m} n_j = n$) can be replaced by g ($g = 1, 2, \ldots, n$). Moreover, the following conditions hold: (1) if $x_{ijk} = 1$, then $x_{ig} = 1$; (2) $\sum_{j=1}^{m}\sum_{k=1}^{n_j} x_{ijk} = \sum_{g=1}^{n} x_{ig} = 1$ ($i = 1, 2, \ldots, n$). From property 6, we know that the allocation vector of $\boldsymbol{Y}$ is the same as that of $\boldsymbol{X}^*$, which implies that σ and the position node list of $\boldsymbol{X}^*$. and $\boldsymbol{Y}$ are identical. Let $\sigma(r) = f/q_h (r = 1, 2, \ldots, n)$. Then, y_{ihf} can be replaced by y_{ir}. Based on the above results, F6 can be reformulated as follows (F7):

$$\min\left(\max\sum_{i=1}^{n}\sum_{r=1}^{n}\left(\underline{p}_i\sum_{g=1}^{r}(\sigma(g)-\sigma(r))x_{ig} + \bar{p}_i\sum_{g=r}^{n}(\sigma(g)-\sigma(r))x_{ig}\right)y_{ir}\right) \tag{39}$$

$$\text{s.t.} \quad \sum_{g=1}^{n} x_{ig} = 1, \quad i = 1,2,\ldots,n \tag{40}$$

$$\sum_{i=1}^{n} x_{ig} = 1, \quad g = 1, 2, \ldots, n \tag{41}$$

$$x_{ig} \in \{0, 1\}, \quad i = 1, 2, \ldots, n; \quad g = 1, 2, \ldots, n \tag{42}$$

$$\sum_{r=1}^{n} y_{ir} = 1, \quad i = 1, 2, \ldots, n \tag{43}$$

$$\sum_{r=1}^{n} y_{ir} = 1, \quad r = 1, 2, \ldots, n \tag{44}$$

$$y_{ir} \geqslant 0, \quad i = 1, 2, \ldots, n; \quad r = 1, 2, \ldots, n \tag{45}$$

This formulation is the same as that of the robust single-machine scheduling problem discussed by Montemanni (2007), except that the position weights of $\underline{p}_i$ or $\bar{p}_i$ ($i = 1, 2, \ldots, n$) in the objective function are $\sigma(g)$ instead of g ($g = 1, 2, \ldots, n$).

Following Montemanni (2007), the ROB-U problem is formulated as the following mixed-integer linear programme (F8):

$$\min\left(\sum_{i=1}^{n} \eta_i + \sum_{r=1}^{n} \tau_r\right)$$

$$\text{s.t.} \quad \sum_{g=1}^{n} x_{ig} = 1, \quad i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} x_{ig} = 1, \quad g = 1, 2, \ldots, n$$

$$x_{ig} \in \{0, 1\}, \quad i = 1, 2, \ldots, n; \quad g = 1, 2, \ldots, n$$

$$\eta_i + \tau_r \geqslant \underline{p}_i \sum_{g=1}^{r} (\sigma(g) - \sigma(r)) x_{ig} + \bar{p}_i \sum_{g=r}^{n} (\sigma(g) - \sigma(r)) x_{ig}, \quad i, r = 1, \ldots, n$$

$$\eta_i, \tau_r \in R, \quad i, r = 1, 2, \ldots, n\,.$$

The optimal robust schedule can be obtained by solving the MILP problem F8 which has $n^2 + 2n$ variables and constraints. The values of x_{ijk} can then be obtained using the position node list.

3.3 *The 2-approximation algorithm*

Only reasonably sized problems can be solved by the above MILP formulations since the ROB-U problem is NP-hard. Therefore, it is necessary to develop approximation algorithms for this problem. Kasperski and Zieliński (2008) have proved that the optimal schedule under the midpoint scenario $\bar{\boldsymbol{s}}$ (i.e. $p_i^{\bar{\boldsymbol{s}}} = (\underline{p}_i + \bar{p}_i)/2$) is a 2-approximation for the minmax regret version of $1||\Sigma C_i$ (i.e. ROB-S). In the following paragraph, we show that the minmax regret version of $Q||\Sigma C_i$ has the same property.

Property 7. Let $\bar{\boldsymbol{X}}$ be an optimal solution of the deterministic $Q||\Sigma C_i$ problem under the midpoint scenario $\bar{\boldsymbol{s}}$. Then, the following inequality holds:

$$R_{\max}(\bar{\boldsymbol{X}}) \leqslant 2R_{\max}(\boldsymbol{X}^*)$$

where $\boldsymbol{X}^*$ is an optimal robust schedule for the ROB-U problem.

Proof. The results in Section 3.2 have shown that the ROB-U problem can be transformed into the ROB-S problem using the position node list. Therefore, the 2-approximation of $\bar{\boldsymbol{X}}$ for the ROB-U problem also holds because the optimal schedule of the deterministic $1||\Sigma C_i$ problem under the midpoint scenario is a 2-approximation of the ROB-S problem (Kasperski and Zieliński 2008).

It is well known that the deterministic $Q||\Sigma C_i$ problem is polynomially solvable using the MFT algorithm (Horowitz and Sahni 1976). The following 2-approximation algorithm for the ROB-U problem is developed using the MFT algorithm and can be solved in polynomial time.

The algorithm for midpoint scenario (AM):

Input: m machines with speeds $1 = q_1 \le q_2 \le \ldots \le q_m$; the number of jobs n; the interval processing times $[\underline{p}_i, \bar{p}_i]$, $1 \le i \le n$.
Output: the optimal schedule $\bar{\boldsymbol{X}}$ for midpoint scenario $\bar{\boldsymbol{s}}$.
Step1: for $i \leftarrow 1, 2, \ldots, n$, let $p_i^{\bar{s}} = (\underline{p}_i + \bar{p}_i)/2$.
Step 2: Solve the deterministic $Q||\Sigma C_i$ problem under scenario $\bar{\boldsymbol{s}}$ using the MFT algorithm and output $\bar{\boldsymbol{X}}$.

Property 7 indicates that the solution $\bar{\boldsymbol{X}}$ obtained using AM is at the most two times worse than the optimal solution for the ROB-U problem.

4. Computational experiments

Extensive computational experiments were performed in order to assess the performance of the proposed algorithms (MILP1, MILP2 and AM). All the procedures were coded in MATLAB R2010 and performed using a system with an Intel® Core™ i5-2550 K processor running at 3.40 GHz with 1.95 GB RAM. Moreover, the MILP solver of ILOG CPLEX 12.0 was used to solve the F4 and F8 problems.

The first set of experiments involves test problems with $m = 3, 4, 5$ machines and $n = 10, 15, 20$ jobs which are taken from Chuang, Liao, and Chao (2010). To determine the processing rates of m machines (q_j, $j \in M$), we first generate m random values from a uniform distribution of integers from 1 to 10. Then, we divide these values by the smallest value and sort them in ascending order. We follow the approach of Daniels and Kouvelis (1995) to determine the intervals of the job processing times. For each job, the lower and upper bounds of the processing times are selected at random from two uniform distributions of integers between $\underline{p}_i [\in 10, 50\beta_1]$. and $\bar{p}_i \in [\underline{p}_i, \underline{p}_i(1 + \beta_2)]$. Further, β_1 and β_2 are two parameters that control the variability of the processing times across and within the jobs. Five values (0.2, 0.4, 0.6, 0.8 and 1.0) are used for both β_1 and β_2. For each combination of m, n, β_1 and β_2, 20 replications are generated, resulting in a

Table 1. Computational results for three-machine problems.

		MILP1[61]	MILP2	AM		Expected	
n	β_2	CPU	CPU	Ave	Max	Ave	Max
10	0.20	0.17	0.12	1.10	42.86	0.00	0.17
	0.40	0.17	0.12	1.23	16.67	0.01	0.13
	0.60	0.18	0.12	2.07	28.57	0.02	0.31
	0.80	0.21	0.12	1.67	18.82	0.03	0.70
	1.00	0.23	0.12	1.47	12.77	0.05	0.55
15	0.20	0.54	0.12	1.40	20.00	0.00	0.07
	0.40	1.70	0.13	1.78	20.00	0.01	0.08
	0.60	3.22	0.14	2.28	12.24	0.03	0.35
	0.80	3.25	0.14	2.68	13.58	0.06	0.45
	1.00	6.73	0.14	2.34	12.10	0.07	0.53
20	0.20	47.148[6]	0.13	1.33	11.36	0.00	0.04
	0.40	77.922[11]	0.15	2.09	16.13	0.02	0.15
	0.60	92.580[10]	0.17	2.18	14.81	0.03	0.15
	0.80	127.07[15]	0.25	3.07	15.60	0.06	0.36
	1.00	155.858[19]	0.33	2.46	9.93	0.07	0.54
Avg. [max]		34.47	0.15	1.94	42.86	0.03	0.70

Note: Superscripts denote the number of problems that remain unsolved by MILP1 in 10 min.

Table 2. Computational results for four-machine problems.

n	β_2	MILP1[66] CPU	MILP2 CPU	AM Ave	AM Max	Expected Ave	Expected Max
10	0.20	0.18	0.12	0.55	16.67	0.00	0.05
	0.40	0.19	0.12	1.02	14.71	0.01	0.21
	0.60	0.19	0.12	1.49	20.00	0.03	0.45
	0.80	0.20	0.12	2.51	48.00	0.03	0.75
	1.00	0.23	0.13	1.79	16.28	0.06	0.76
15	0.20	0.65	0.12	1.27	15.79	0.00	0.05
	0.40	1.95	0.13	1.71	19.05	0.01	0.23
	0.60	2.24	0.13	2.30	13.24	0.03	0.29
	0.80	3.38	0.14	2.01	13.01	0.04	0.41
	1.00	12.79	0.14	2.28	10.91	0.06	0.40
20	0.20	53.68[7]	0.13	1.01	17.65	0.00	0.03
	0.40	68.82[10]	0.14	2.49	16.07	0.01	0.10
	0.60	106.82[14]	0.19	2.54	16.53	0.03	0.20
	0.80	113.16[15]	0.25	2.25	10.81	0.06	0.60
	1.00	145.71[20]	0.34	2.79	11.40	0.07	0.48
Avg. [max]		34.01	0.16	1.87	48.00	0.03	0.76

Note: Superscripts denote the number of problems that remain unsolved by MILP1 in 10 min.

Table 3. Computational results for five-machine problems.

n	β_2	MILP1[53] CPU	MILP2 CPU	AM Ave	AM Max	Expected Ave	Expected Max
10	0.20	0.20	0.12	2.21	100.00	0.00	0.00
	0.40	0.21	0.12	1.34	25.00	0.02	0.39
	0.60	0.22	0.12	1.85	18.65	0.03	0.31
	0.80	0.22	0.12	2.18	34.48	0.04	0.40
	1.00	0.23	0.12	1.99	16.36	0.04	0.59
15	0.20	0.73	0.13	1.34	25.00	0.02	0.39
	0.40	0.59	0.13	1.85	18.65	0.03	0.31
	0.60	1.29	0.13	2.18	34.48	0.04	0.40
	0.80	1.47	0.14	1.99	16.36	0.04	0.59
	1.00	1.28	0.15	0.70	12.96	0.00	0.06
20	0.20	61.435[8]	0.14	1.68	20.69	0.00	0.08
	0.40	65.606[9]	0.15	1.98	16.09	0.01	0.10
	0.60	78.904[9]	0.18	2.29	18.18	0.02	0.14
	0.80	84.081[11]	0.29	1.46	12.88	0.04	0.27
	1.00	122.945[16]	0.38	2.10	12.75	0.07	0.54
Avg. [max]		27.96	0.16	1.81	100.00	0.03	0.59

Note: Superscripts denote the number of problems that remain unsolved by MILP1 in 10 min.

total of 4500 test problems. Each test problem is solved using MILP1, MILP2 and AM. The maximum allowed CPU time is set at 600 s.

Tables 1–3 list the mean CPU time (in s) used by MILP1 and MILP2 and the average and maximum percentage deviations of the maximum regret of the optimal midpoint scenario schedule from the minmax regret, respectively. Moreover, for evaluating the expected flow time of the robust schedule, it is assumed that the job processing times are both independent and uniformly distributed. Then, the optimal midpoint scenario schedule obtained using AM is also the schedule that minimises the expected flow time. Therefore, the last two columns of Tables 1–3 list the percentage deviations of the expected flow time of the robust schedule (expected) from the minimum expected flow time corresponding to the optimal midpoint scenario schedule. The results are summarised in terms of β_2 and n to show the effects of the processing time variability and problem size on the computational performance.

Table 4. Computational results for large-size problems.

n	β_2	MILP2[100] CPU	AM Ave	AM Max
25	0.20	0.172	1.53	13.33
	0.40	0.182	2.50	8.06
	0.60	0.563	2.10	14.06
	0.80	2.433	3.34	16.95
	1.00	8.001	2.52	8.48
30	0.20	0.177	1.88	14.12
	0.40	0.386	2.21	9.16
	0.60	4.781	2.90	10.04
	0.80	15.458	2.57	7.81
	1.00	86.7542	3.01	8.60
35	0.20	0.227	2.06	9.09
	0.40	2.268	2.78	11.69
	0.60	36.109	3.08	12.90
	0.80	127.9535	2.43	9.15
	1.00	312.35922	1.49	6.27
40	0.20	0.285	2.09	9.37
	0.40	7.636	2.34	8.72
	0.60	218.87013	1.91	7.31
	0.80	392.37128	1.78	8.44
	1.00	397.57230	0.98	4.96
Avg. [max]		80.728	2.27	16.95

Note: Superscripts denote the number of problems that remain unsolved by MILP2 in 10 min.

Tables 1–3 show that the CPU time required by MILP1 increases rapidly with the number of jobs (n) as well as with the variability of the processing time (β_2). When n is 20, 180 problems remain unsolved (61, 66 and 53 for $m = 3$, $m = 4$ and $m = 5$, respectively) using MILP1 for 10 min. As compared to MILP1, MILP2 can solve all the problems and requires far lesser CPU time. Moreover, the CPU time required by MILP2 increases slowly with the problem size and the variability of processing time. This indicates that MILP2 can be used to derive the exact solution for large-size problems. The tables also show that the average percentage deviations of the maximum regret of the optimal midpoint scenario schedule from the minmax regret are 1.94, 1.87, and 1.81 for three-, four-, and five-machine problems, respectively, which are much smaller than the given upper bound of 200%. For the 4500 test problems, the average percentage deviation of the expected flow time of the robust optimal schedule is only 0.03 and the maximum percentage deviation is 0.76. The low percentage deviations indicate that by using the minmax regret approach, we can hedge against the total flow time uncertainty while maintaining the expected flow time to be close to the optimal value.

To examine the size of the problems that can be solved by MILP2, a second set of experiments is designed. Since the computational times required by MILP2 marginally change with the number of machines (m), we generate the problem instances with $n = 25$, 30, 35 and 40 jobs and $m = 5$ machines. The values of the parameters β_1 and β_2 are still set as 0.2, 0.4, 0.6, 0.8 and 1.0. For each combination of n, β_1 and β_2, 10 replications are run, resulting in 1000 test problems. The results are summarised in Table 4 which lists the mean CPU times required by MILP2 and the average and maximum percentage deviations of AM. From the table, it is observed that the CPU time required by MILP2 increases rapidly with n and β_2 for medium-size problems. Almost all the problems with $n = 25$ and 30 jobs can be solved by MILP2 in 10 min. More than one-tenth (27 out of 250) of the problems with $n = 35$ jobs remain unsolved and about one-third (71 out of 250) of the problems with $n = 40$ jobs remain unsolved. Since MILP2 is identical to the MILP formulation proposed by Montemanni (2007), the problem size solvable using these two formulations should be the same. Accordingly, MILP2 can handle problems with up to 45 jobs within a reasonable computational time. Moreover, the results in Table 4 indicate that the solution quality obtained using AM does not deteriorate with an increase in the problem size; the average percentage deviation is 2.27 and the maximum percentage deviation is 16.95.

5. Conclusion

In recent times, parallel machine scheduling problems have been extensively studied because of their wide range of potential applications. In most studies, job processing times or their distributions are assumed to be precisely known. In practice, however, job processing times are unknown and their exact probability distributions are often unavailable because of the absence of historical data, especially for one-time jobs.

This paper describes a minmax regret (robust) version of the $Q||\Sigma C_i$ scheduling problem (ROB-U) with interval job processing times. To solve this NP-hard problem, we first propose properties 1 and 2, which show that the worst-case scenario is an extreme scenario and that the maximum regret for any schedule can be obtained by solving an $n \times mn$ assignment problem in $O((mn)^3)$ time. Then, in Property 3, we show that the ROB-U problem can be formulated as a MILP formulation (MILP1) with $n^2 m + nm + n$ variables and constraints. To reduce the computational time required, we reformulate our problem by exploiting the property of the allocation vector, which defines the number of jobs allocated to each machine. Property 4 indicates that the allocation vector of the optimal schedule under any scenario is the same; then, the allocation algorithm is proposed to determine the optimal allocation vector. Based on the above results, property 5 shows that the maximum regret for any schedule can be obtained by solving an $n \times n$ assignment problem in $O(n^3)$ time. Property 6 proves that the optimal allocation vectors of the robust and deterministic uniform parallel machine scheduling problems are the same. Based on this, the robust uniform parallel machine scheduling problem is transformed into a robust single-machine scheduling problem. The relationship between the single-machine and uniform parallel machine scheduling problems is established using a position node list. Finally, a MILP formulation (MILP2) with $n^2 + 2n$ variables and constraints is proposed for solving the minmax regret uniform parallel machine scheduling problem. Moreover, we prove that the optimal solution of the deterministic $Q||\Sigma C_i$ problem under the midpoint scenario is at the most two times worse than the optimal robust solution of the ROB-U problem.

Our computational results demonstrate that MILP2 is much more efficient than MILP1 and can be used to solve moderate-size problems in reasonable times. The average percentage deviation of the maximum regret of the optimal midpoint scenario schedule from the minmax regret is less than 3% which is much smaller than the given upper bound (200%). Therefore, the 2-approximation algorithm can be used either to generate reasonable initial solutions for other algorithms or to directly solve large-scale problems since it is a polynomial-time algorithm. Moreover, our computational results indicate that by using the minmax regret approach, we can hedge against the total flow time uncertainty while maintaining the expected flow time to be close to the optimal value.

This paper offers several effective approaches for solving scheduling problems in highly uncertain manufacturing environments. In future, an extension of this work may be possible by investigating other parallel machine scheduling problems with interval data.

Funding

The authors are grateful for financial support from the National Natural Science Foundation of China [grant number 71072128], [grant number 71371149], [grant number 71001084].

References

Aissi, H., C. Bazgan, and D. Vanderpooten. 2005a. "Approximation Complexity of Min–Max (Regret) Versions of Shortest Path, Spanning Tree, and Knapsack." *Algorithms – Esa* 2005 (3669): 862–873.

Aissi, H., C. Bazgan, and D. Vanderpooten. 2005b. "Complexity of the Min–Max and Min–Max Regret Assignment Problems." *Operations Research Letters* 33 (6): 634–640.

Aissi, H., C. Bazgan, and D. Vanderpooten. 2009. "Min–Max and Min–Max Regret Versions of Combinatorial Optimization Problems: A Survey." *European Journal of Operational Research* 197 (2): 427–438.

Allahverdi, A., and H. Aydilek. 2010. "Heuristics for the Two-machine Flowshop Scheduling Problem to Minimise Makespan with Bounded Processing times." *International Journal of Production Research* 48 (21): 6367–6385.

Averbakh, I. 2006. "The Minmax Regret Permutation Flow-shop Problem with Two Jobs." *European Journal of Operational Research* 169 (3): 761–766.

Cai, X. Q., X. Y. Wu, and X. Zhou. 2009. "Stochastic Scheduling on Parallel Machines to Minimize Discounted Holding Costs." *Journal of Scheduling* 12 (4): 375–388.

Chekuri, C., and S. Khanna. 2004. "On Multidimensional Packing Problems." *SIAM Journal on Computing* 33 (4): 837–851.

Chuang, M. C., C. J. Liao, and C. W. Chao. 2010. "Parallel Machine Scheduling with Preference of Machines." *International Journal of Production Research* 48 (14): 4139–4152.

Conway, R. W., W. L. Maxwell, and L. W. Miller. 1967. *Theory of Scheduling*. Reading, MA: Addison Wesley.
Daniels, R. L., and P. Kouvelis. 1995. "Robust Scheduling to Hedge against Processing Time Uncertainty in Single-stage Production." *Management Science* 41 (2): 363–376.
Deineko, V. G., and G. J. Woeginger. 2010. "Pinpointing the Complexity of the Interval Min–Max Regret Knapsack Problem." *Discrete Optimization* 7 (4): 191–196.
Eren, T., and E. Guner. 2009. "A Bicriteria Parallel Machine Scheduling with a Learning Effect." *International Journal of Advanced Manufacturing Technology* 40 (11–12): 1202–1205.
Hamada, T., and M. Tamaki. 1999. "Some Results on a Bayesian Sequential Scheduling on Two Identical Parallel Processors." *Journal of the Operations Research Society of Japan* 42 (3): 316–329.
Horowitz, E., and S. Sahni. 1976. "Exact and Approximate Algorithms for Scheduling Nonidentical Processors." *Journal of the Acm* 23 (2): 317–327.
Hsu, C. J., T. C. E. Cheng, and D. L. Yang. 2011. "Unrelated Parallel-machine Scheduling with Rate-modifying Activities to Minimize the Total Completion Time." *Information Sciences* 181 (20): 4799–4803.
Hsu, C. J., W. H. Kuo, and D. L. Yang. 2011. "Unrelated Parallel Machine Scheduling with Past-sequence-dependent Setup Time and Learning Effects." *Applied Mathematical Modelling* 35 (3): 1492–1496.
Jia, J., and S. J. Mason. 2009. "Semiconductor Manufacturing Scheduling of Jobs Containing Multiple Orders on Identical Parallel Machines." *International Journal of Production Research* 47 (10): 2565–2585.
Kang, Y. H., and H. J. Shin. 2010. "An Adaptive Scheduling Algorithm for a Parallel Machine Problem with Rework Processes." *International Journal of Production Research* 48 (1): 95–115.
Kasperski, A. 2005. "Minimizing Maximal Regret in the Single Machine Sequencing Problem with Maximum Lateness Criterion." *Operations Research Letters* 33 (4): 431–436.
Kasperski, A., and P. Zieliński. 2006. "An Approximation Algorithm for Interval Data Minmax Regret Combinatorial Optimization Problems." *Information Processing Letters* 97 (5): 177–180.
Kasperski, A., and P. Zieliński. 2007. "On Combinatorial Optimization Problems on Matroids with Uncertain Weights." *European Journal of Operational Research* 177 (2): 851–864.
Kasperski, A., and P. Zieliński. 2008. "A 2-Approximation Algorithm for Interval Data Minmax Regret Sequencing Problems with the Total Flow Time Criterion." *Operations Research Letters* 36 (3): 343–344.
Kasperski, A., and P. Zieliński. 2010. "Minmax Regret Approach and Optimality Evaluation in Combinatorial Optimization Problems with Interval and Fuzzy Weights." *European Journal of Operational Research* 200 (3): 680–687.
Kasperski, A., A. Kurpisz, and P. Zieliński. 2012. "Approximating a Two-machine flow Shop Scheduling under Discrete Scenario Uncertainty." *European Journal of Operational Research* 217 (1): 36–43.
Kmith, D. E. 1973. *The Art of Computer Programming, Vol. 3: Sorting and Searching*. Reading, MA: Addison Wesley.
Kouvelis, P., and G. Yu. 1997. *Robust Discrete Optimization and Its Applications* Boston, MA: Kluwer Academic.
Kouvelis, P., R. L. Daniels, and G. Vairaktarakis. 2000. "Robust Scheduling of a Two-machine Flow Shop with Uncertain Processing times." *IIE Transactions* 32 (5): 421–432.
Lebedev, V., and I. Averbakh. 2006. "Complexity of Minimizing the Total Flow Time with Interval Data and Minmax Regret Criterion." *Discrete Applied Mathematics* 154 (15): 2167–2177.
Lin, J., and T. S. Ng. 2011. "Robust Multi-market Newsvendor Models with Interval Demand Data." *European Journal of Operational Research* 212 (2): 361–373.
Lu, C. C., S. W. Lin, and K. C. Ying. 2012. "Robust Scheduling on a Single Machine to Minimize Total Flow Time." *Computers & Operations Research* 39: 1682–1691.
Montemanni, R. 2006. "A Benders Decomposition Approach for the Robust Spanning Tree Problem with Interval Data." *European Journal of Operational Research* 174 (3): 1479–1490.
Montemanni, R. 2007. "A Mixed Integer Programming Formulation for the Total Flow Time Single Machine Robust Scheduling Problem with Interval Data." *Journal of Mathematical Modelling and Algorithms* 6 (2): 287–296.
Montemanni, R., and L. M. Gambardella. 2004. "An Exact Algorithm for the Robust Shortest Path Problem with Interval Data." *Computers & Operations Research* 31 (10): 1667–1680.
Montemanni, R., and L. M. Gambardella. 2005a. "A Branch and Bound Algorithm for the Robust Spanning Tree Problem with Interval Data." *European Journal of Operational Research* 161 (3): 771–779.
Montemanni, R., and L. M. Gambardella. 2005b. "The Robust Shortest Path Problem with Interval Data via Benders Decomposition." *4OR: A Quarterly Journal of Operations Research* 3 (4): 315–328.
Ozlen, M., and M. Azizoglu. 2009. "Generating All Efficient Solutions of a Rescheduling Problem on Unrelated Parallel Machines." *International Journal of Production Research* 47 (19): 5245–5270.
Ozlen, M., and S. Webster. 2010. "Minimising Total Flow-time on Two Parallel Machines with Planned Downtimes and Resumable Jobs." *International Journal of Production Research* 48 (1): 201–226.
Pinedo, M. 2008. *Scheduling: Theory, Algorithms, and Systems*. 3rd ed. New York: Springer.
Ranjbar, M., M. Davari, and R. Leus. 2012. "Two Branch-and-bound Algorithms for the Robust Parallel Machine Scheduling Problem." *Computers & Operations Research* 39 (7): 1652–1660.

Rustogi, K., and V. A. Strusevich. 2012. "Simple Matching vs Linear Assignment in Scheduling Models with Positional Effects: A Critical Review." *European Journal of Operational Research* 222 (3): 393–407.

Tan, Z. Y., Y. Chen, and A. Zhang. 2011. "Parallel Machines Scheduling with Machine Maintenance for Minsum Criteria." *European Journal of Operational Research* 212 (2): 287–292.

Weber, R. R. 1982. "Scheduling Jobs with Stochastic Processing Requirements on Parallel Machines to Minimize Makespan or Flowtime." *Journal of Applied Probability* 19 (1): 167–182.

Weber, R. R., P. Varaiya, and J. Walrand. 1986. "Scheduling Jobs with Stochastically Ordered Processing times on Parallel Machines to Minimize Expected Flowtime." *Journal of Applied Probability* 23 (3): 841–847.

Xu, X., W. Cui, J. Lin, and Y. Qian. 2013. "Robust Makespan Minimisation in Identical Parallel Machine Scheduling Problem with Interval Data." *International Journal of Production Research* 51 (12): 3532–3548.

Yang, J., and G. Yu. 2002. "On the Robust Single Machine Scheduling Problem." *Journal of Combinatorial Optimization* 6 (1): 17–33.