Robust Optimization on Unrelated Parallel Machine Scheduling With Setup Times

Weihao Wang, *Member, IEEE*, Chutong Gao, *Student Member, IEEE*, and Leyuan Shi, *Fellow, IEEE*

***Abstract*— The parallel machine scheduling problem has been a popular topic for many years due to its theoretical and practical importance. This paper addresses the robust makespan optimization problem on unrelated parallel machine scheduling with sequence-dependent setup times, where the processing times are uncertain, and the only knowledge is the time intervals they take values from. We propose a robust optimization model with the min-max regret criterion to formulate this problem. To solve this problem, we prove that the worst-case scenario with the maximum regret for a given solution belongs to a finite set of extreme scenarios. Based on this theoretical analysis, a procedure to obtain the maximum regret is proposed and an enhanced regret evaluation method (ERE) is designed to accelerate this process, which is of great significance to improve the efficiency of the algorithm. A multi-start decomposition-based heuristic algorithm (MDH) based on the analysis of properties is proposed to solve this problem. Computational experiments are conducted to justify the performance and robustness of these methods.**

***Note to Practitioners*—Various uncertainties may occur in the production process, which brings great challenges to production and operations management. A robust production schedule is of great significance for factories to make full use of production capacity and deal with production abnormalities. This study is motivated by an R&D and assembly task scheduling problem encountered in a high-end equipment manufacturing factory in which the processing time of each job is uncertain, and its distribution is also unknown due to limited information. In this study, with the consideration of sequence-dependent setup time and uncertain job-processing time, we view the labor groups with different skill levels as unrelated parallel machines and build a robust (min-max regret) scheduling model to formulate this problem so as to reduce the production makespan. An enhanced regret evaluation method is developed to improve the evaluation efficiency for a given solution, and a multi-start decomposition-based heuristic algorithm is proposed to solve this problem. This study can be applied in practice to release schedulers from burdensome work and provide high-quality robust schedules for this complicated production environment.**

***Index Terms*— Unrelated parallel machine, sequence-dependent setup time, min-max regret, uncertain processing time, robust scheduling.**

I. Introduction

With the popularity of mass customization, manufacturing companies are faced with more complex and dynamic customer demands and production environments. The manufacturing industry, like equipment manufacturing, semiconductor fabrication, and medical instrument production, etc., requires effective and efficient production schedules to cope with the various uncertainties and fulfill the intensive customer needs.

The problem we discuss in this article is derived from a factory producing high-end equipment in eastern China. The scheduler in this factory needs to schedule a series of R&D jobs to some different labor groups to minimize the maximum completion time of all jobs, i.e., makespan. Those groups process these jobs in parallel, but the skills and the kind of jobs they are adept at vary from each other, so the processing times of one job are unrelated in different groups. Thus these labor groups could be viewed as unrelated parallel machines. When a machine finishes one job, cleaning, adjustment, and reconfiguration are required to switch to another job. Thus machine and job sequence-dependent setup times need to be considered in this problem.

Assuming we know exactly the processing time of each job, this problem could be viewed as the Unrelated Parallel Machine Scheduling Problem with Sequence-dependent Setup Times (UPMSS) with the objective of makespan minimization, which is a complex combinatorial optimization problem and has been extensively studied. However, this assumption does not reflect the actual situation in the factory we investigate.

In this factory, due to the various uncertainties such as fails in product R&D, variations in machine conditions, and production environment, the processing time of each job on each machine is uncertain at the time when a scheduler dispatches the jobs. Besides, due to the lack of historical data and the multiple complex factors that lead to the uncertainty, the exact distributions of uncertain parameters are difficult to obtain, and the only knowledge is the estimated lower and upper bounds of the processing times (namely the processing time intervals) based on the historical experience of the scheduler. Under such circumstances, our aim is to find an optimal schedule, i.e., the job assignment and the job sequence on each machine, to hedge against the uncertainty of the processing times and minimize the makespan.

Manuscript received 20 August 2021; revised 27 December 2021; accepted 7 February 2022. Date of publication 16 March 2022; date of current version 6 January 2023. This article was recommended for publication by Associate Editor F. Chu and Editor X. Xie upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation of China under Grant 71690232. *(Corresponding author: Chutong Gao.)*

Weihao Wang is with the Department of Industrial Engineering and Management, Peking University, Beijing 100871, China (e-mail: wangweihao@pku.edu.cn).

Chutong Gao is with the Department of Mechanics and Engineering Science, Peking University, Beijing 100871, China (e-mail: oliverpku@pku.edu.cn).

Leyuan Shi is with the Department of Industrial and Systems Engineering, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: leyuan@engr.wisc.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2022.3151611.

Digital Object Identifier 10.1109/TASE.2022.3151611

Compared with the stochastic programming that models the uncertainties as random variables with proper probability distributions [1], robust optimization (RO) does not assume that probability distributions are known, instead it assumes that the uncertain data resides in an uncertainty set [2]. In this case, we use RO to hedge against the variations of the parameters. Two natural ways of defining the possible realizations of uncertain parameters (referred to as scenarios) are often used in robust optimization: discrete scenarios and interval scenarios [3]. Considering the time uncertainty in the factory we investigate, we adopt the continuous interval scenario in this case. That is to say, the processing time of each job on each machine can take an arbitrary value from a prespecified interval.

In robust optimization, two criteria, min-max and min-max regret, are often used to evaluate the robust solutions. For the problem we investigate, the min-max criterion optimizes the makespan under the sole worst-case scenario, which only consists of the upper bounds of the processing time intervals. This particular worst-case scenario is paid too much more importance. But in reality, it may be unlikely to occur. Therefore, using this criterion in our problem tends to be over-conservative. The min-max regret criterion, however, tries to find a robust solution that has the smallest maximum deviation of the makespan from the optimum across all possible scenarios. With min-max regret, the decision-maker aims to minimize the opportunity loss, i.e., the cost of a solution is compared ex-post to the cost of the best solution which could have been chosen [3], which leads to less conservative decision compared to the min-max criteria. However, its difficulty lies in the calculation of the optimal performance under each scenario, which by itself is already very complex, when evaluating each feasible solution.

To avoid the scheduler's decision from being too conservative and make use of the information about both lower bounds and upper bounds of processing times, we choose the min-max regret as the criterion. Thus this problem be summarized as the Robust Optimization on Unrelated Parallel Machine Scheduling with Sequence-dependent Setup Times and Min-max Makespan Regret Criterion (RUPMSS).

The main contributions of this study are concluded as follows.

1) As far as we know, we are the first to consider the uncertainty of the processing times and introduce the robustness with min-max regret criterion to the unrelated machine scheduling with sequence-dependent setup times. A robust scheduling model to minimize the regret of makespan is proposed to formulate this problem.
2) The critical properties of the worst-case scenario of this problem are derived so that only a finite set of scenarios need to be considered to evaluate the maximum regret for a given solution. Based on it, we give a procedure to obtain the worst-case scenario of a given solution and an Enhanced Regret Evaluation method (ERE) to accelerate this procedure.
3) A Multi-start Decomposition-based Heuristic Algorithm (MDH) is proposed based on the analysis of the decomposition property and the characteristics of the shift & interchange search methods. The performance of the algorithm outperforms the existing exact algorithms and heuristics for large cases.

The remainder of this article is organized as follows. Section II reviews the literature on the related problems. In Section III, we formulate RUMPSS as a mixed-integer programming model. Some properties for the worst-case scenarios are proposed, and a procedure of finding the worst-case scenario for a given solution is introduced in Section IV. In Section V, the ERE method is designed to calculate the maximum regret of a solution. The MDH algorithm with two local search methods is proposed in Section VI. Section VII is the numerical experiments, and Section VIII is the conclusion and some further directions.

II. Literature Review

Robust optimization has been a popular method to hedge against uncertainty in many combination optimization problems for decades, such as the assignment problem [4], [5], product packing problem [6], vehicle routing problem [7], [8], etc. More studies on robust optimization problem could refer to the survey papers by Kasperski and Zieliński [3], and by Aissi *et al.* [9].

The robust optimization has also been used to solve many scheduling problems to achieve certain objectives under various uncertainties, among which many studies have paid attention to the time uncertainty with interval parameters. For instance, some studies pay attention to the robust project scheduling problem. Artigues *et al.* [10] study a resource-constrained robust project scheduling problem with uncertain activity durations to minimize the maximum regret of makespan. They propose both an iterative exact algorithm called scenario relaxation method and a heuristic algorithm to solve this problem. Bruni *et al.* [11] consider an adjustable resource-constrained robust project scheduling problem with uncertain activity durations. The task sequencing needs to be decided without violating the task precedence relationship and resource constraints to minimize the worst-case makespan. An exact solution method similar to the Bender's decomposition method is proposed to solve this problem. Ng *et al.* [12] study a robust aircraft scheduling problem with arrival/departure delay using min-max regret criterion. The operation times take values from intervals and the objective is to make runway operations robust to mitigate the effects of delay propagation. They propose an artificial bee colony algorithm to obtain close-to-optimal results in regard to a one-hour flight traffic planning horizon. Kong *et al.* [13] use the robust optimization with min-max regret approach for integrated steel production and batch delivery scheduling with uncertain rolling times and deterioration effect, and develop an improved variable neighborhood search (IVNS) algorithm. They further propose an exact algorithm that combines CPLEX and two dynamic programming algorithms to obtain the maximum regret value of a given rolling sequence.

As for the robust production scheduling problem, Kasperski [14] discusses a robust single machine scheduling problem with the objective of lateness. The processing

times are specified as intervals and a polynomial algorithm is proposed to solve this problem. Kasperski and Zieliński [15] prove that solving the deterministic problem under the mid-point scenario provides a 2-approximation algorithm for interval data min-max regret sequencing problems with the total flow time criterion. Pereira [16] further studies another robust single machine scheduling problem with interval processing time and the objective of total weighted completion time. A branch-and-bound algorithm is proposed to solve this problem. Choi and Chung [17] consider the min-max regret version of a single-machine scheduling problem under processing time uncertainty to determine which jobs are processed by outsourcing. The objective is to minimize the sum of the processing cost in-house and the outsourcing cost. They study both the interval scenario set and discrete scenario set, and discuss the computational complexity for various cases. Drwal [18] studies a robust single machine scheduling problem with the objective to minimize the weighted number of late jobs. The author assumes the due dates of jobs are uncertain and belong to intervals. They develop a polynomial-time algorithm to solve the special case where all weights are equal to 1. And a MILP formulation is presented for the general case. The solution quality of two simple heuristic algorithms is assessed and a decomposition method is proposed to handle the large-scale problems. Drwal and Józefczyk [19] further study the above robust single machine scheduling problem with interval job processing time and the objective of the weighted number of late jobs, and propose a specialized branch-and-bound algorithm to solve it. Wang *et al.* [20] study the same problem and a 2-approximation algorithm and two heuristics are proposed.

As for the flow shop, Kouvelis *et al.* [21] show that a two-machine flow shop robust scheduling problem with uncertain processing time and min-max regret criterion is NP-hard. Gholami-Zanjani *et al.* [22] study a robust flow shop scheduling with interval processing times and sequence-dependent setup times with the objective of min-max weighted flow time. They analyze the robust counterpart of the model and apply it to a real case in a PCB assembly company. Liao and Fu [23] consider a general permutation flow shop scheduling problem with interval production time. A robust model is formulated with the bi-objectives of min-max regret of total completion time and min-max production tardiness. A directed graph tool is applied to identify the worst-case scenario and a genetic algorithm is implemented to solve this problem.

Because parallel machines are widely present in various manufacturing systems, some work has also paid attention to the robust parallel machine scheduling problem with min-max regret criterion and interval parameters. Several studies focus on the objective of total flow time. Drwal and Rischke [24], Xu *et al.* [25] minimize the maximum regret of total flow time on identical and uniform parallel machines with interval job processing time, respectively. Siepak and Józefczyk [26], Conde [27] consider the robust scheduling problem above under unrelated parallel machine settings. Xu *et al.* [28] consider a robust identical parallel machine scheduling problem with interval job processing times to minimize the maximum regret of the makespan. They propose an iterative exact algorithm to solve the small-scale problem, and a hill-climbing and a simulated annealing algorithm to solve larger scale problems. Feng *et al.* [29] consider the robust scheduling problem of a two-stage hybrid flow shop setting where the first stage is a single machine and the second stage consists of identical parallel machines. They also select the min-max regret of makespan as the objective function and use the same method as Xu *et al.* [28] to solve this problem. Hu *et al.* [30] study a min-max regret robust identical machine scheduling problem with sequence-dependent setup times, where the processing times and arrival times of jobs both take values from their respective intervals. They consider the total completion time as the objective and propose a modified artificial bee colony algorithm for this problem. Wang and Cui [31] examine the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time. The objective of this problem is the total completion time of jobs processed in-house and the cost of outsourcing the rest. They transform the problem into an equivalent robust single machine scheduling problem and develop a 2-approximation algorithm.

The aforementioned literature on robust parallel machine scheduling rarely covers scheduling problems with unrelated parallel machines, sequence-dependent setup times, and the makespan objective together. As for the problem we investigate, a handful of studies have paid attention to the deterministic version of the problem, namely the UPMSS. The parallel machine scheduling problem is NP-hard even for a case of two identical parallel machines [32]. A series of models such as those in [33]–[35] are proposed to improve the scale of the instances that could be solved to optimality. Some exact algorithms for UPMSS could be found in the literature, for instance the Decomposition-based Method [36] and Mathematical-programming Base Algorithm [35]. Nevertheless, due to the NP-hard nature of this problem, various heuristic and meta-heuristic algorithms could be found in the literature. For instance, Lin and Ying [37] propose a Hybrid Artificial Bee Colony algorithm (HABC) that outperforms the best algorithms at that time such as ACO, TS, RSA, and Meta-RaPS. Wang *et al.* [38] propose a Hybrid Estimation of Distribution Algorithm with Iterated Greedy search method and compare it with two GA algorithms. Arnaout [39] further proposes a Worm Optimization algorithm for this problem. More related studies could refer to the survey paper [40].

To the best of our knowledge, the robust unrelated parallel machine scheduling with sequence-dependent setup times and the objective of minimizing makespan has not been studied by other researchers. So this study intends to fill this gap and solve the problem that could be encountered in many production areas.

III. Problem Formulation

We formulate our RUPMSS problem based on Avalos-Rosales *et al.* [34] 's model for the deterministic UPMSS. There is a set of jobs $N = \{1, \cdots, n\}$ to be processed on machines $M = \{1, \cdots, m\}$. p_{ij}^s, the processing time of job j on machine i under scenario $s \in S$ (the set of all the scenarios)

ranges from $[\underline{p}_{ij}, \overline{p}_{ij}]$. After processing job j on machine i, a fixed sequence-and-machine-dependent setup time s_{ijk} is needed if the next job on machine i is job k. Each job must be processed by exactly one machine and each machine may process at most one job per time. All jobs are released from the beginning. In this case, our aim is to minimize the maximum regret of a solution $\pi \in \Phi$, i.e., $R_{\max}(\pi) = \max_{s\in S}\{F(\pi, s) - F^s_*\}$, where $F(\pi, s)$ represents the makespan of solution π under scenario s, F^s_* is the optimal makespan under scenario s, and Φ is the set of all feasible schedules. The notations we use are summarized as follows:

NOTATIONS

N	set of jobs, $N = \{1, \ldots, n\}$.
N_0	set of jobs with a dummy job 0 added, $N_0 = \{0\} \cup N$.
M	set of machines, $M = \{1, \ldots, m\}$.
j, k	index of jobs, $j, k \in N_0$.
i	index of machines, $i \in M$.
S	set of scenarios, $S = \left\{s \mid p^s_{ij} \in [\underline{p}_{ij}, \overline{p}_{ij}], i \in M, j \in N\right\}$.
p^s_{ij}	the processing time of job j on machine i under scenario s, $p^s_{ij} \in [\underline{p}_{ij}, \overline{p}_{ij}]$. Note that $\underline{p}_{i0} = \overline{p}_{i0} = 0$ for any machine i. In other words, $p^s_{i0} = 0$ for any scenario $s \in S$.
s_{ijk}	the setup time of job k after job j on machine i. s_{i0k} is the setup time of job k if it is the first to be processed on machine i, and $s_{ij0} = 0$, which means job j is the last to be processed on machine i.
V	a large number.

VARIABLES

x_{ijk}	$x_{ijk} = 1$ if job k is the successor of job j on machine i; otherwise $x_{ijk} = 0$. Note that $x_{i0k}(x_{ik0}) = 1$ means that job k is the first (last) to be processed on machine i. $X = \{x_{ijk} \mid i \in M, j, k \in N_0, j \neq k\}$.
y_{ij}	$y_{ij} = 1$ if job j is assigned to machine i. Otherwise, $y_{ij} = 0$. $Y = \{y_{ij} \mid i \in M, j \in N\}$.
C^s_j	the completion time of job j under scenario s, $j \in N_0$.
$C^s_{\max}$	the makespan under scenario s.

We note that each feasible schedule $\pi \in \Phi$ could be described by $\{X, Y\}$, where Y specifies the assignment of the jobs and X further describes the sequence of the jobs on each machine. It's easy to see that $F(\pi, s) = C^s_{\max} = \max_{j\in N} C^s_j$, and $F^s_* = \min_{\pi\in\Phi} F(\pi, s)$.

Then, the model of RUPMSS could be formulated as follows:

$$\min_{\pi\in\Phi} \max_{s\in S}\{F(\pi, s) - F^s_*\} \tag{1}$$

$$\text{s.t.} \sum_{i\in M} y_{ij} = 1, \quad j \in N \tag{2}$$

$$y_{ij} = \sum_{k\in N_0, j\neq k} x_{ijk}, \quad i \in M,\ j \in N \tag{3}$$

$$y_{ik} = \sum_{j\in N_0, j\neq k} x_{ijk}, \quad i \in M,\ k \in N \tag{4}$$

$$\sum_{k\in N} x_{i0\ k} \leq 1, \quad i \in M \tag{5}$$

$$C^s_k - C^s_j + V\left(1 - x_{ijk}\right) \geq s_{ijk} + p^s_{ik},$$
$$j \in N_0,\ k \in N,\ j \neq k,\ i \in M,\ s \in S \tag{6}$$

$$C^s_{\max} \geq C^s_j, \quad j \in N,\ s \in S \tag{7}$$

$$\sum_{j\in N_0,\ k\in N,\ j\neq k} s_{ijk}x_{ijk} + \sum_{j\in N} p^s_{ij}y_{ij} \leq C^s_{\max},$$
$$i \in M,\ s \in S \tag{8}$$

$$C^s_0 = 0, \quad s \in S \tag{9}$$

$$C^s_j \geq 0, C^s_{\max} \geq 0, \quad j \in N,\ i \in M,\ s \in S \tag{10}$$

$$x_{ijk} \in \{0, 1\}, y_{ij} \geq 0, \quad j \in N,\ i \in M \tag{11}$$

Constraints (2) ensure that each job should be assigned to one machine exactly. Constraints (3) guarantee that each job has one and only one follow-up job on the machine it is assigned. The last job at the end of each machine would also be followed by a dummy job. Constraints (4) ensure that each job on each machine is followed by one and only one preceding job. For the first job on the machine, its preceding job is the dummy job. Constraints (5) state that at most one job is scheduled as the first job after the dummy job on each machine. The following constraints are all related to the scenarios the uncertain parameters may occur. Constraints (6) stipulate the relationship of the completion time of the jobs assigned to each machine. Specifically, under each scenario s, if job k is scheduled after job j on machine i, then the job k should be completed in at least $s_{ijk} + p^s_{ik}$ time units after the completion of job j. Constraints (7) stipulate that the makespan should be no less than the completion time of each job in each scenario, which are feasible cuts proved efficient by [34]. Constraints (8) are the definition of the makespan. In constraints (8), the left side of the inequality calculates the completion time of each machine as the sum of the sequence-dependent setup time and the processing time of all jobs assigned to it. Then these constraints state that in each scenario the overall makespan should be no less than the completion time of each machine. The completion times of dummy jobs under all scenarios are set to 0 as shown in constraints (9). Constraints (10) and (11) show the type and range of each variable. According to the problem structure and constraints (3) and (4), variable y_{ij} is the sum of binary variables x_{ijk} which is bounded by 1 in constraints (2). Therefore it could be relaxed to positive rather than binary variable in this model.

IV. THEORETICAL ANALYSIS

We could reformulate the nonlinear programming model above as below:

$$\min_{\pi,\ r}$$
$$F(\pi, s) - F^s_* \leq r, \quad s \in S \tag{12}$$
$$\text{Constraints (2)-(11)} \tag{13}$$

The model is difficult to solve, and the underlying difficulties lie in two aspects: (1) The number of the scenarios in the interval scenario setting is infinite, i.e., $|S| = +\infty$, and thus the number of the constraints and the number of decision variables (C_j^s) are also infinite in the formulation; (2) the calculation of F_*^s under every scenario requires solving a NP-hard deterministic UPMSS. Thus, we must examine the structure of the model to capture the characteristics of the worst-case scenarios. In the following analysis, we could limit the scenarios to some finite extreme cases. First, we give the definitions of the critical machine and the extreme scenario for a schedule.

Definition 1: We denote a machine i's completion time in schedule π under scenario s as $F_i(\pi, s)$. A machine f is critical in schedule π under scenario s if its completion time is the maximum among all machines, i.e.,$F_f(\pi, s) = \max_{i \in M} F_i(\pi, s)$, or equivalently, $F_f(\pi, s) = F(\pi, s)$.

Definition 2: Extreme scenario s^f, $f \in M$ for schedule π is defined as:

$$p_{ij}^{s^f} = \begin{cases} \overline{p}_{ij}, & \text{if } y_{fj} = 1 \\ \underline{p}_{ij}, & \text{if } y_{fj} = 0,\ i \in M,\ j \in N_0 \end{cases} \tag{14}$$

With these definitions, we propose Theorem 1 to obtain the worst-case scenario for a schedule.

Theorem 1: For a schedule π, let s^0 be a worst-case scenario for π, in which machine f is a critical machine. Then the extreme scenario s^f for schedule π satisfies:

(a) Machine f is also a critical machine in schedule π under scenario s^f;

(b) Scenario s^f is also a worst-case scenario for π.

Proof: Let $J(\pi, f, s^0)$ represent the set of jobs produced on the critical machine f for solution π under scenario s^0. Construct scenario s^f by the following operations:

$$\begin{cases} p_{ij}^{s^f} \leftarrow \overline{p}_{ij} & i = f,\ j \in J(\pi, f, s^0) \\ p_{ij}^{s^f} \leftarrow \underline{p}_{ij} & \text{otherwise.} \end{cases}$$

With solution π unchanged, we have $F_f(\pi, s^f) \geq F_f(\pi, s^0)$, and $F_i(\pi, s^f) \leq F_i(\pi, s^0), \forall i \neq f$. So machine f is still the critical machine under scenario s^f. (a) is proved.

On machine f, we have

$$\begin{aligned} F(\pi, s^f) - F(\pi, s^0) &= \left(\max_{i \in M} F_i(\pi, s^f)\right) - \left(\max_{i' \in M} F_{i'}(\pi, s^0)\right) \\ &= F_f(\pi, s^f) - F_f(\pi, s^0) \\ &= \sum_{j \in J(\pi, f, s^0)} (\overline{p}_{fj} - p_{fj}^{s^0}). \end{aligned}$$

Denote the optimal solution under scenario s^0 as $\pi_*^{s^0}$. Let $F_*^{s^f}$ and $F_*^{s^0}$ be the optimal makespans under scenario s^f and s^0, respectively. Therefore we have the following inequalities:

$$\begin{aligned} F_*^{s^f} - F_*^{s^0} &\leq F\left(\pi_*^{s^0}, s^f\right) - F\left(\pi_*^{s^0}, s^0\right) \\ &\leq \sum_{j \in J\left(\pi, f, s^0\right)} (\bar{p}_{fj} - p_{fj}^{s^0}) \\ &= F\left(\pi, s^f\right) - F\left(\pi, s^0\right) \end{aligned} \tag{15}$$

The last inequality exists because for any schedule π, the increased makespan by switching from scenario s^0 to s^f would not be more than $\sum_{j \in J(\pi, f, s^0)} (\overline{p}_{fj} - p_{fj}^{s^0})$, which is the sum of all possible processing time increments when we construct scenario s^f from s^0. With (15) we could derive:

$$R_{\max}(\pi) = F\left(\pi, s^0\right) - F_*^{s^0} \leq F\left(\pi, s^f\right) - F_*^{s^f} \tag{16}$$

Since s^0 is a worst-case scenario for solution π, so is s^f. The proof of (b) is complete. ■

With Theorem 1, we can reduce the infinite set of scenarios S to a finite set of extreme scenarios to obtain the maximum regret of a solution. However, it is not known in advance which machine is the critical machine. To evaluate a solution π, we need to traverse m extreme scenarios $s^1, \cdots, s^m$ for π to calculate its maximum regret, i.e., $R_{\max}(\pi) = \max_{i \in M} \{F(\pi, s^i) - F_*^{s^i}\}$. Due to the existence of the nested NP-hard deterministic UPMSS problems for the value of $F_*^{s^i}$, the evaluation process is computationally expensive. Therefore, effective methods need to be proposed to accelerate the evaluation process.

V. Enhanced Regret Evaluation Method

In this section, we design a method consisting of several mechanisms to accelerate the maximum regret evaluation process of a certain solution so as to judge quickly whether this solution is likely to be better. First, we propose a lower bound of the maximum regret for a neighborhood solution to help judge whether this solution is potential to be better, or it can be discarded directly without accurate evaluation. If a solution needs to be evaluated accurately, we further propose two mechanisms which use machine completion time intervals and an upper bound of regret for each extreme scenario to reduce the number of machines to be traversed respectively, eliminate some extreme scenarios that cannot be the worst-case scenarios, and reduce the number of deterministic problems to be solved, therefore. In addition, we further propose that using a heuristic algorithm to solve deterministic UPMSS quickly and use the approximate maximum regret value to evaluate the solution. We call the method to evaluate a solution with these mechanisms as the Enhance Regret Evaluation Method (ERE).

A. Use a Lower Bound to Abandon Inferior Neighborhood Solutions

Because the neighborhood could be very large and most of the neighborhood solutions may not lead to the improvement, we first develop a lower bound of the maximum regret for a specific neighborhood solution to exclude those inferior solutions which would definitely not improve the solution. Assume π^* is the current best solution found in the solving process, and π' is one of its neighborhood solutions. We have

$$\begin{aligned} R_{\max}(\pi') &= \max_{i \in M} \{F(\pi', s^i) - F_*^{s^i}\} \\ &\geq \max_{i \in M} \{F(\pi', s^i) - F(\pi^*, s^i)\} \end{aligned}$$

Because $F(\pi^*, s^i) \geq F_*^{s^i}$, the inequality holds. Thus $R_{LB}(\pi', \pi^*) = \max_{i \in M} \{F(\pi', s^i) - F(\pi^*, s^i)\}$ is a lower bound for $R_{\max}(\pi')$. Because π^* is already known, $R_{LB}(\pi', \pi^*)$ is easy to calculate. If $R_{LB}(\pi', \pi^*) > R_{\max}(\pi^*)$, π' cannot be better than π^*. In this case, there is no need to traverse each extreme scenario of the solution π' and calculate its exact maximum regret.

B. Use Machine Completion Time Intervals to Eliminate Some Extreme Scenarios

For a solution that is not discarded by the above mechanism and whose accurate regret needs to be evaluated, we propose a mechanism which intends to reduce the number of machines to be traversed so as to reduce the deterministic problems to be solved and accelerate the evaluation process.

Theorem 2 provides the detailed elimination method. To introduce this theorem, first we need to calculate the completion time interval for each machine. Specifically for machine $i \in M$ in solution $\pi = \{X, Y\}$, the machine completion time lies in an interval $[\underline{F_i}(\pi), \overline{F_i}(\pi)]$ under the interval processing time. The minimum completion time of machine i over all scenarios is $\underline{F_i}(\pi) = \sum_{j \in N_0, k \in N, k \neq j} s_{ijk} x_{ijk} + \sum_{j \in N} \underline{p}_{ij} y_{ij}$ and its maximum completion time is $\overline{F_i}(\pi) = \sum_{j \in N_0, k \in N, k \neq j} s_{ijk} x_{ijk} + \sum_{j \in N} \overline{p}_{ij} y_{ij}$. With the completion time interval of each machine, we have the following theorem:

Theorem 2: For a given solution π and machine i, if its completion time's upper bound $\overline{F_i}(\pi)$ is lower than $\underline{F_{i'}}(\pi)$ of any other machine $i' \in M, i' \neq i$, s^i cannot be the worst-case scenario of solution π.

Proof: For solution π, if s^i is the worst-case scenario, machine i should be the critical machine. Thus the completion time on different machines should satisfy $F_i(\pi, s^i) \geq F_{i'}(\pi, s^i)$, for $\forall i' \neq i$. According to (2), the way we define s^i for solution π suggests that $F_{i'}(\pi, s^i) = \underline{F_{i'}}(\pi)$, and $F_i(\pi, s^i) = \overline{F_i}(\pi)$. Since machine i is the critical machine, which means $F_i(\pi, s^i) \geq F_{i'}(\pi, s^i)$, then we have $\overline{F_i}(\pi) \geq \underline{F_{i'}}(\pi)$. Thus we prove the theorem. ∎

According to Theorem 2, in the evaluation process, there is no need to calculate the regret of the extreme scenario with machine i as the critical machine once we find a machine i' that makes $\overline{F_i}(\pi) < \underline{F_{i'}}(\pi)$. Based on this theorem, we could eliminate some machines and the corresponding extreme scenarios, and reduce the number of deterministic UPMSS problem to be solved. We use S_ψ to denote the remained extreme scenarios for the subsequent evaluation of this solution.

C. Use an Upper Bound Regret of Each Extreme Scenario to Further Eliminate Some Extreme Scenarios

For those extreme scenarios in S_ψ of the solution π that have not been eliminated by the above mechanism, we use a lower bound of the deterministic UPMSS to obtain an upper bound for the regret of the corresponding scenario, which intends to further eliminate some extreme scenarios and reduce the number of the deterministic problems to be solved in the evaluation process.

Specifically, we calculate the regret $R_{\{1\}}(\pi)$ of one extreme scenario $s^{\{1\}} \in S_\psi$ first as the current maximum regret. For any other extreme scenario $s^{\{i\}} \in S_\psi$, we calculate the lower bound $LB^{s^{\{i\}}}$ of the deterministic problem under scenario $s^{\{i\}}$ and replace $F_*^{s^{\{i\}}}$ in $R_i(\pi) = F(\pi, s^{\{i\}}) - F_*^{s^{\{i\}}}$ with $LB^{s^{\{i\}}}$. Then we could obtain an upper bound regret of the extreme scenario $s^{\{i\}}$ as $F(\pi, s^{\{i\}}) - LB^{s^{\{i\}}}$. If it is less than the current maximum regret, this extreme scenario cannot be the worst-case scenario. So we don't need to calculate the $F_*^{s^{\{i\}}}$ of deterministic RUPMSS under extreme scenario $s^{\{i\}}$, which would further reduce the time to evaluate a certain solution.

We adopt a series of classical lower bounds for the deterministic UPMSS which has been used in many literature such as [41]. The lower bound LB^s for UPMSS under extreme scenario s is as follows:

1) $LB_1^s = \frac{1}{M} \sum_{k=1}^{N} \min_{i \in M, j \in N} (p_{ik}^s + s_{ijk})$;
2) $LB_2^s = \max_{k \in N} \{\min_{i \in M, j \in N} (p_{ik}^s + s_{ijk})\}$;
3) LB_3^s: the optimal objective value of the LP relaxation of UPMSS.

And $LB^s = \max\{LB_1^s, LB_2^s, LB_3^s\}$.

D. Solve the Deterministic Problem Using Heuristics

Despite the above mechanisms to accelerate the evaluation process, there remains a number of deterministic UPMSS problems to solve in each evaluation process. Due to the NP-hard nature of the deterministic problem, it is very time-consuming to obtain their exact optimal solutions. Thus, we substitute heuristics for MIP solvers like CPLEX to obtain high-quality approximate solutions for the deterministic problems and improve the time efficiency.

In this study, we choose the Hybrid Artificial Bee Colony algorithm (HABC) proposed by Lin and Ying [37], which yields superior performance for UPMSS compared to many other algorithms like ACO, TS, RSA, and Meta-RaPS, etc. The HABC algorithm is a population-based metaheuristic algorithm that mimics the behaviors of a bee colony. It has been successfully applied to a variety of real world-problems including UPMSS. For more details of HABC, readers could refer to Lin and Ying's work [37].

Based on the methods above, given the current best solution π^*, the procedure of the enhanced regret evaluation method for a given solution π is introduced as follows:

VI. Multi-Start Decomposition-Based Heuristic Algorithm

In this section, we propose a Multi-start Decomposition-based Heuristic (MDH) algorithm to solve this problem. There are two things to be decided in the RUPMSS: (1) the assignment of jobs to machines (corresponding to decision variables $y_{ij}, \forall i \in M, j \in N$) and (2) the sequencing of the jobs on each machine given the job assignment (corresponding to decision variables $x_{ijk}, \forall i \in M, j, k \in N$). Given multiple initial solutions, the MDH algorithm searches for a better solution with the local search methods in each initial solution's neighborhood. Note that for each solution π found in the search process, we use the ERE introduced in the previous

Enhanced Regret Evaluation Method (ERE)

INPUT: The solution π need to calculate the maximum regret, the current best solution π^*;

Step 1. Set possible extreme scenarios $S_\psi = \emptyset$, $R_{\max}(\pi) = 0$;

Step 2. Calculate the lower bound regret of solution π as $R_{LB}(\pi, \pi^*) = \max_{i \in M}\{F(\pi, s^i) - F(\pi^*, s^i)\}$. If $R_{LB}(\pi, \pi^*) > R_{\max}(\pi^*)$, abandon this solution π, stop.

Step 3. Calculate the machine completion time interval $[\underline{F_i}(\pi), \overline{F_i}(\pi)]$ for each machine $i \in M$. For each machine $i \in M$, if $\overline{F_i}(\pi) < \underline{F_{i'}}(\pi)$ of any other machine $i' \in M, i' \neq i$, move to the next machine; otherwise, add extreme scenario s^i into S_ψ. If all machines have been fathomed, go to Step 4.

Step 4. Calculate the regret of first extreme scenario $s^{\{1\}}$ in S_ψ as $R_{\max}(\pi) = R_{\{1\}}(\pi) = F(\pi, s^{\{1\}}) - F_*^{s^{\{1\}}}$. For each extreme scenario $s^{\{i\}} \in S_\psi, s^{\{i\}} \neq s^{\{1\}}$, calculate the lower bound $LB^{s^{\{i\}}}$ of deterministic UPMSS under scenario $s^{\{i\}}$. If $F(\pi, s^{\{i\}}) - LB^{s^{\{i\}}} \leq R_{\{1\}}(\pi)$, skip this extreme scenario, go to next extreme scenario in S_ψ; otherwise calculate $F_*^{s^{\{i\}}}$. If $R_{\{i\}}(\pi) = F(\pi, s^{\{i\}}) - F_*^{s^{\{i\}}} > R_{\max}(\pi)$, update $R_{\max}(\pi) \leftarrow R_{\{i\}}(\pi)$. $F_*^{s^{\{1\}}}$ could be calculated by a heuristic algorithm. If all scenarios in S_ψ have been fathomed, go to Step 5.

Step 5. Terminate the algorithm and output the maximum regret $R_{\max}(\pi)$.

section to evaluate its quality quickly and calculate its maximum regret $R_{\max}(\pi)$. In the following, we investigate the job sequencing subproblem and the job assignment subproblem under RUPMSS, respectively. In addition, we introduce an upper bound of RUPMSS derived from one of the initial solutions of the MDH algorithm. Then we propose the MDH algorithm based on the findings.

A. Job Sequencing Problem

We first consider the sequence of jobs on each machine for a given job assignment. Specifically, a fixed job assignment means that for each machine, the jobs to be produced on it are known and unchanged. In this case, the production sequence of jobs on each machine needs to be decided to obtain a complete solution due to the existence of sequence-dependent setup times in RUPMSS, i.e., we need to decide X given a fixed $\hat{Y}$. In this subsection, we'd like to prove that for each fixed assignment, only one optimal job sequence needs to be considered to achieve the objective of min-max regret under this job assignment. Thus, we could decompose the problem into two stages.

To give a detailed interpretation of the property about job sequencing, we need to give some new definitions first.

Definition 3: For a given job assignment $\hat{Y}$, define schedule $\hat{\pi}^\dagger = \{\hat{X}^\dagger, \hat{Y}^\dagger\}$ as the optimal solution of the following MILP:

$$1\,OptSeq(\hat{Y}): \min \sum_{i \in M} \sum_{j \in N_0, k \in N, j \neq k} s_{ijk} x_{ijk} \tag{17}$$

$$y_{ij} = \hat{y}_{ij}, i \in M, j \in N \tag{18}$$

$$C_k - C_j + V(1 - x_{ijk}) \geq s_{ijk}, \quad j \in N_0, k \in N, j \neq k, i \in M \tag{19}$$

$$C_0 = 0 \tag{20}$$

$$x_{ijk} \in \{0, 1\}, C_j \geq 0, j, k \in N_0, i \in M$$

$$\text{Constraints (3)(4)(5)} \tag{21}$$

The objective (17) is the total setup time on all machines. Constraints (18) ensure that the optimal schedule shares the same job assignment as $\hat{Y}$. The other constraints describe a feasible schedule as before.

$OptSeq(\hat{Y})$ is meant to find the schedule $\hat{\pi}^\dagger$ that minimizes the total setup time under the fixed job assignment $\hat{Y}$. Since the setup times are deterministic and irrelevant to the realisation of scenarios, $OptSeq(\hat{Y})$ is also irrelevant to the scenarios. Then we give the following theorem:

Theorem 3: Denote the set of feasible schedules with the fixed job assignment $\hat{Y}$ as $\Phi(\hat{Y})$. $\arg\min_{\hat{\pi} \in \Phi(\hat{Y})} R_{\max}(\hat{\pi})$ is obtained by solving $OptSeq(\hat{Y})$. In other words, if $\hat{\pi}^\dagger$ is the optimal solution of $OptSeq(\hat{Y})$, then $R_{\max}(\hat{\pi}^\dagger) \leq R_{\max}(\hat{\pi}), \forall \hat{\pi} \in \Phi(\hat{Y})$.

Proof: For a fixed job assignment $\hat{Y}$, it is evident that minimizing the total setup time on all the machines is equivalent to minimizing the setup time on each machine separately. Let $s^{0\dagger}$ be the worst-case scenario of the optimal schedule $\hat{\pi}^\dagger$ of $OptSeq(\hat{Y})$. For any $\hat{\pi} \in \Phi(\hat{Y})$,

$$\begin{aligned} F_i(\hat{\pi}^\dagger, s^{0\dagger}) &= \sum_{j,k \in N_0} s_{ijk} \hat{x}_{ijk}^\dagger + \sum_{j \in N_0} p^{s^{0\dagger}} \hat{y}_{ij} \\ F_i(\hat{\pi}, s^{0\dagger}) &= \sum_{j,k \in N_0} s_{ijk} \hat{x}_{ijk} + \sum_{j \in N_0} p^{s^{0\dagger}} \hat{y}_{ij} \\ &\Longrightarrow F_i(\hat{\pi}^\dagger, s^{0\dagger}) \leq F_i(\hat{\pi}, s^{0\dagger}), \forall i \in M \\ &\Longrightarrow F(\hat{\pi}^\dagger, s^{0\dagger}) \leq F(\hat{\pi}, s^{0\dagger}) \end{aligned} \tag{22}$$

From (22), we could derive the following inequalities:

$$\begin{aligned} R_{\max}(\hat{\pi}^\dagger) &= F(\hat{\pi}^\dagger, s^{0\dagger}) - F_*^{s^{0\dagger}} \\ &\leq F(\hat{\pi}, s^{0\dagger}) - F_*^{s^{0\dagger}} = R(\hat{\pi}, s^{0\dagger}) \leq R_{\max}(\hat{\pi}) \end{aligned} \tag{23}$$

The proof is completed. ■

Theorem 3 gives us significant insight into the structure of RUPMSS: once we obtain a job assignment $\hat{Y}$, $OptSeq(\hat{Y})$ could yield a complete schedule $\hat{\pi}^\dagger$ with the smallest maximum regret in any schedules with job assignment $\hat{Y}$. So from now on, we only need to focus on finding the best $\hat{Y}$ that has the minimum $R_{\max}(\hat{\pi}^\dagger)$, where $\hat{\pi}^\dagger$ is decided by $\hat{Y}$.

B. Job Assignment Problem

Now we come to the decision of job assignment, that is, to assign jobs to the unrelated parallel machines. We propose a local search method with two operators to search for a better job assignment:

- **Shift**: Moving a job from one machine to another one;
- **Interchange**: Swapping two jobs on different machines.

These two operators are simple and flexible, and have been used in many problems with similar settings such as the workload balancing problem on the parallel machine [42],

[43], the task allocation problem [44], and the parallel machine scheduling problem [28], [34]. We use these two operators to search the neighborhood of the incumbent solution sequentially. If a new assignment with the corresponding optimal sequence introduced earlier has a lower maximum regret, we obtain a better solution and accept it as a new incumbent solution.

We further discuss under what condition the shift or interchange may find better solutions. The following theorems are derived to avoid ineffective search:

Theorem 4: If a given solution could be improved by adjusting the job assignment, one of the adjusted jobs must be on the critical machine of the incumbent solution under its worst-case scenario.

Proof: The proof is evident when there are only two machines. We need to consider the case where there are more than two machines.

Consider improving the solution π by altering the assignment of jobs on the solution with worst-case scenario s^0. We construct a new solution π' by altering the job assignment on its non-critical machines, e.g., by shift or interchange, and we use $s^{0'}$ to denote the worst-case scenario for solution π'. Assume by contradiction that the maximum regret of π' is less than that of π. Then we have

$$R(\pi, s^0) > R(\pi', s^{0'}) \geq R(\pi', s^0). \tag{24}$$

Given that the adjusted jobs are assigned on non-critical machines, the makespan on the critical machine remains unchanged. Thus the makespan in schedule π' cannot be lower than that of π under scenario s^0, i.e., $F(\pi', s^0) \geq F(\pi, s^0)$. Based on this result, we have:

$$R(\pi', s^0) = F(\pi', s^0) - F_*^{s^0} \geq F(\pi, s^0) - F_*^{s^0} = R(\pi, s^0),$$

which contradicts (24). Thus an adjustment for the job assignment without adjusting the jobs on the critical machine under the worst-case scenario would not improve the solution. The proof is completed. ■

We note that Theorem 4 could be true for any kind of adjustments besides shift and interchange. But in this problem we only adopt these two. Theorem 4 suggests that for the shift operator, the job we choose to shift must be on the critical machine of the incumbent solution under its worst-case scenario, and for the interchange operator, one of the swapped jobs must come from the critical machine.

1) Shift Procedure: For the shift local search, each of the n jobs can be shifted to the other $m-1$ machines, which can generate a total of $(m-1)n$ different assignments in the neighborhood of the initial solution π_0. But according to Theorem 4, only shifting the job from the critical machine under the worst-case scenario to other machines may improve the incumbent solution. Therefore only $(m-1)n_c$ assignment adjustments are attempted and evaluated, where n_c is the number of jobs on the critical machine. We denote the pth job on the critical machine as j_p. Once a new job assignment $\hat{Y}$ is constructed, the corresponding job sequencing problem $OptSeq(\hat{Y})$ is solved by the CPLEX solver to obtain a complete schedule. The procedure of shift local search is presented as follows:

Shift Procedure

INPUT: initial solution π_0, its critical machine i_c, maximum regret $R_{\max}(\pi_0)$, and the number of jobs on the critical machine n_c. Let the current solution $\pi \leftarrow \pi_0$.

Step 1. Set $p \leftarrow 0$.

Step 2. Set $p \leftarrow p+1$, $i \leftarrow 0$. If $p > n_c$ go to Step 6.

Step 3. Set $i \leftarrow i+1$. If $i > m$, go to Step 2. If $i = i_c$, repeat Step 3.

Step 4. Construct a new solution π' by shifting the job j_p to machine i to obtain a new job assignment Y', and solving $OptSeq(Y')$ (by the solver) to obtain π'. Evaluate solution π' and obtain its critical machine i_c', maximum regret $R_{\max}(\pi')$, and the number of jobs on critical machine n_c'.

Step 5. If $R_{\max}(\pi') < R_{\max}(\pi)$, set $\pi \leftarrow \pi'$, $R_{\max}(\pi) \leftarrow R_{\max}(\pi')$, $i_c \leftarrow i_c'$, $n_c \leftarrow n_c'$, go to Step 1. Otherwise, go to Step 3.

Step 6. Terminate the algorithm and output π, $R_{\max}(\pi)$, and i_c.

2) Interchange Procedure: For the interchange local search, we traverse each pair of jobs ($C_n^2 = n(n-1)/2$ in total) to see whether it satisfies Theorem 4. Only when two jobs on different machines and one of them is on the critical machine under the worst-case scenario, the interchange is tried to see whether it could improve the incumbent solution. According to Theorem 4, only $(n-n_c)n_c$ assignments in interchange neighborhood may bring improvement and are evaluated. As in the Shift procedure, once a new job assignment is constructed, the job sequencing problem is solved by the CPLEX solver.

The procedure of interchange local search is presented as follows:

Interchange Procedure

INPUT: initial solution π_0, its critical machine i_c, maximum regret $R_{\max}(\pi_0)$, and the number of jobs on the critical machine n_c. Let the current solution $\pi \leftarrow \pi_0$.

Step 1. Set $l \leftarrow 0$, $k \leftarrow 1$, $j \leftarrow 2$.

Step 2. Set $l \leftarrow l+1$. If $l > \frac{(n-1)n}{2}$ go to Step 7.

Step 3. If $j > n$, set $k \leftarrow k+1$. If $k > n$, set $k \leftarrow 1$. Then set $j \leftarrow k+1$.

Step 4. Judge whether both jobs j and k are on the same machine, or neither of them is on the critical machine i_c. If it is true, $j \leftarrow j+1$, go to Step 2.

Step 5. Interchange job j and k in solution π to obtain a new job assignment Y', and solving $OptSeq(Y')$ (by the solver) to obtain a new solution π'. Evaluate solution π' and obtain its critical machine i_c', maximum regret $R_{\max}(\pi')$.

Step 6. If $R_{\max}(\pi') < R_{\max}(\pi)$, set $\pi \leftarrow \pi'$, $R_{\max}(\pi) \leftarrow R_{\max}(\pi')$, $i_c \leftarrow i_c'$, and $l \leftarrow 0$. Go to Step 2.

Step 7. Terminate the algorithm and output π, $R_{\max}(\pi)$.

C. The Upper Bound From the Initial Solution Under Mid-Scenario

The efficiency of the local search algorithm depends on the choice of the initial solution. Here we introduce an

initial solution that could provide an upper bound for this problem. We generate an initial solution by solving the deterministic problem under the mid-scenario $s^{\frac{1}{2}} := \{p_{ij}^{s^{\frac{1}{2}}} = \frac{\overline{p}_{ij}+\underline{p}_{ij}}{2}, s_{ijk}, \forall i \in M, j, k \in N\}$. Theorem 5 below shows that this solution provides an upper bound for RUPMSS. Before describing the theorem, we first introduce some notations as below:

- α_{ij} the percentage deviation between the upper bound and lower bound of processing time of job j on machine i, i.e., $\alpha_{ij} = \frac{\overline{p}_{ij}-\underline{p}_{ij}}{\underline{p}_{ij}}$;
- α the maximum percentage deviation between the upper bound and lower bound for all processing times, i.e., $\alpha = \max_{i\in M, j\in N} \alpha_{ij}$;
- π_*^s the optimal solution for the deterministic UPMSS problem under scenario s.

Then we can derive that

$$\overline{p}_{ij} = (1 + \frac{\alpha_{ij}}{2+\alpha_{ij}})p_{ij}^{s^{\frac{1}{2}}}, \quad \underline{p}_{ij} = \frac{2}{2+\alpha_{ij}} p_{ij}^{s^{\frac{1}{2}}}.$$

Next we define some new scenarios for the proof of this theorem:

- $s^{\frac{1}{2}}$ $p_{ij}^{s^{\frac{1}{2}}} = \frac{\overline{p}_{ij}+\underline{p}_{ij}}{2}$; $s_{ijk}^{s^{\frac{1}{2}}} = s_{ijk}$;
- $\overline{s}$ $p_{ij}^{\overline{s}} = \overline{p}_{ij}$; $s_{ijk}^{\overline{s}} = s_{ijk}$;
- $\underline{s}$ $p_{ij}^{\underline{s}} = \underline{p}_{ij}$; $s_{ijk}^{\underline{s}} = s_{ijk}$;
- $\overline{s}^\alpha$ $p_{ij}^{\overline{s}^\alpha} = (1 + \frac{\alpha}{2+\alpha})p_{ij}^{s^{\frac{1}{2}}} \geq p_{ij}^{\overline{s}}$; $s_{ijk}^{\overline{s}^\alpha} = (1 + \frac{\alpha}{2+\alpha})s_{ijk} \geq s_{ijk}^{\overline{s}}$;
- $\underline{s}^\alpha$ $p_{ij}^{\underline{s}^\alpha} = (\frac{2}{2+\alpha})p_{ij}^{s^{\frac{1}{2}}} \leq p_{ij}^{\underline{s}}$; $s_{ijk}^{\underline{s}^\alpha} = (\frac{2}{2+\alpha})s_{ijk} \leq s_{ijk}^{\underline{s}}$;

Theorem 5: $\frac{2\alpha}{2+\alpha}F_*^{s^{\frac{1}{2}}}$ is an upper bound for the maximum regret of the mid-scenario initial solution, i.e., $R_{\max}(\pi_*^{s^{\frac{1}{2}}}) \leq \frac{2\alpha}{2+\alpha}F_*^{s^{\frac{1}{2}}}$.

Proof: Note that the processing time and setup time of scenarios $\overline{s}^\alpha$, $\underline{s}^\alpha$, and $s^{\frac{1}{2}}$ are proportional to each other. For any schedule π,

$$F(\pi, \overline{s}^\alpha) = (1 + \frac{\alpha}{2+\alpha})F(\pi, s^{\frac{1}{2}}); \tag{25}$$

$$F(\pi, \underline{s}^\alpha) = \frac{2}{2+\alpha}F(\pi, s^{\frac{1}{2}}); \tag{26}$$

So there exists an identical optimal solution π^* for deterministic problems under all the scenarios $\overline{s}^\alpha$, $\underline{s}^\alpha$, and $s^{\frac{1}{2}}$, i.e., there exists an optimal schedule $\pi^* = \pi_*^{s^{\frac{1}{2}}} = \pi_*^{\overline{s}^\alpha} = \pi_*^{\underline{s}^\alpha}$.

According to the relationship of processing time and setup time in scenarios $\overline{s}$, $\underline{s}$, $\overline{s}^\alpha$, $\underline{s}^\alpha$, we have

$$F(\pi, \overline{s}^\alpha) \geq F(\pi, \overline{s}); \tag{27}$$

$$F(\pi, \underline{s}^\alpha) \leq F(\pi, \underline{s}). \tag{28}$$

Suppose s^0 is a worst-case scenario for π^*. First we obtain the inequalities:

$$F_*^{\underline{s}} = F(\pi_*^{\underline{s}}, \underline{s}) \leq F(\pi_*^{s^0}, \underline{s}) \leq F(\pi_*^{s^0}, s^0) = F_*^{s^0} \tag{29}$$

Similarly,

$$F_*^{\underline{s}^\alpha} = F(\pi^*, \underline{s}^\alpha) \leq F(\pi_*^{\underline{s}}, \underline{s}^\alpha) \leq F(\pi_*^{\underline{s}}, \underline{s}) = F_*^{\underline{s}} \tag{30}$$

From (25)-(30) we could obtain

$$\begin{aligned} R_{\max}(\pi^*) &= F(\pi^*, s^0) - F_*^{s^0} \leq F(\pi^*, \overline{s}) - F_*^{\underline{s}} \\ &\leq F(\pi^*, \overline{s}^\alpha) - F_*^{\underline{s}^\alpha} = F(\pi^*, \overline{s}^\alpha) - F(\pi^*, \underline{s}^\alpha) \\ &= (1 + \frac{\alpha}{2+\alpha})F(\pi^*, s^{\frac{1}{2}}) - \frac{2}{2+\alpha}F(\pi^*, s^{\frac{1}{2}}) \\ &= \frac{2\alpha}{2+\alpha}F_*^{s^{\frac{1}{2}}} \end{aligned} \tag{31}$$

Equivalently, because $\pi^* = \pi_*^{s^{\frac{1}{2}}}$, we could obtain

$$R_{\max}(\pi_*^{s^{\frac{1}{2}}}) \leq \frac{2\alpha}{2+\alpha}F_*^{s^{\frac{1}{2}}} \tag{32}$$

■

Theorem 5 makes sure that our choice of initial solutions guarantees a performance no worse than the upper bound.

D. Multi-Start Decomposition Heuristic Algorithm

Heuristic algorithms that intend to find global optimal solutions usually require some type of diversification method to avoid being stuck in the local optima [45]. One method to achieve diversity is to re-conduct the search procedure from a different initial solution once a previous neighborhood has been explored. The multi-start method is a framework that executes multiple times of procedures from different initial points in the solution space [34], which has been used for many combinatorial optimization problems. Readers could refer to the survey paper [46], which exhibits its flexibility and capability.

To solve the RUPMSS, we propose a Multi-start Decomposition-based Heuristic (MDH) algorithm based on the theorems and the methods introduced above. In each iteration of the MDH method, an initial solution in the solution space is generated first. Then the shift local search and interchange local search are conducted respectively to search its neighborhood and seek improvement. After each iteration, a solution is obtained which is typically a local optimal solution, and the best solution obtained after all iterations is the output of the MDH algorithm.

In the MDH algorithm, we obtain the initial solutions by solving a series of deterministic UPMSS problems under certain scenarios. One of the initial solutions is the optimal solution under the mid-scenario, which could guarantee that the maximum regret of the final solution we find cannot be worse than the upper bound given by Theorem 5. Another two specified scenarios including the upper-bound scenario (all the processing times take the upper bound value of the interval) and the lower-bound scenario (all the processing time take the lower bound value of the interval) are also considered to obtain the initial solutions. Then other initial solutions are obtained by solving the deterministic UPMSS under several scenarios s in which the processing times are randomly chosen from their intervals. Those initial solutions identical to the previous ones are not considered for the subsequent exploration.

In the MDH algorithm, we use $Init$ to denote the total number of initial solutions we generate and π_{0_r} to denote the r-th initial solution we obtain. Similarly, π^* denote the current best solution and $R_{\max}(\pi^*)$ represents its maximum regret. The

algorithm will terminate when all the initial solutions have been fathomed or the running time reaches the time limit. The procedures of the MDH algorithm are presented as follows:

The Multi-Start Decomposition Heuristic Algorithm (MDH)

INPUT: the number of initial solutions $Init$, the time limit TL;

Step 1. Set $r \leftarrow 0$, $R_{\max}(\pi^*) \leftarrow BigNumber$;

Step 2. Set $r \leftarrow r+1$, If $r > Init$, go to Step 6; otherwise solve a deterministic UPMSS under a scenario to obtain the initial solution π_{0r}. If π_{0r} has occurred before, repeat Step 2.

Step 3. Conduct the shift local search for the solution π_{0r} to obtain a solution π_{0r}' and its maximum regret $R_{\max}(\pi_{0r}')$.

Step 4. Conduct the interchange local search for the solution π_{0r}' to obtain a solution π_{0r}'' and its maximum regret $R_{\max}(\pi_{0r}'')$.

Step 5. If $R_{\max}(\pi_{0r}'') < R_{\max}(\pi^*)$, update $\pi^* \leftarrow \pi_{0r}''$, $R_{\max}(\pi^*) \leftarrow R_{\max}(\pi_{0r}'')$, go to Step 2.

Step 6. Terminate the algorithm if the TL is reached. Output the solution π^* and its maximum regret $R_{\max}(\pi^*)$.

VII. Numerical Experiments

In this section, we conduct numerical experiments to test the effectiveness and efficiency of the method and algorithm we propose. The MIP solver we use is ILOG CPLEX 12.8. The algorithms are coded with C++ and we conducted the experiments on a 64-bit Windows 10 platform with Intel i7 3.4-GHz CPU and 16.0 GB RAM.

We generate a data set whose scale is in coordinate with the number of the tasks in a few days in the factory we investigate. The data scale is also in line with other similar research (Xu *et al.* [28], $m \leq 5, n \leq 50$; K.K.H. Ng *et al.* [12], $m \leq 3, n \leq 46$; Wang *et al.* [20], $m = 1, n \leq 20$). The numbers of jobs and machines are combinations of $n \in \{9, 12, 15, 20, 25, 30\}$, $m \in \{3, 5, 7\}$. The sequence-dependent setup time s_{ijk} is selected randomly from the integer-valued uniform distribution on the interval $[1, 10]$. We generate the intervals of the job-processing times by following the approach in [28], [47] for min-max regret problems. For each job, the integral lower and upper bounds of the processing time are chosen randomly from two uniform distributions on intervals $\underline{p}_{ij} \in [10, 50\beta]$, and $\overline{p}_{ij} \in [\underline{p}_{ij}, (1+\beta)\underline{p}_{ij}]$. β is a parameter with three values 0.5, 1, 1.5, which controls the gap between the upper and lower bounds of the interval, and thus the variability and randomness of processing times. Multiple values of β are set for sensitive analysis and could reflect the robustness of the algorithms to various data sets. 20 instances are generated for each combination "$n - m - \beta$," and a total of 1080 instances are generated for the experiments.

In the following experiments, the time limits for all the algorithms are set to 2 hours. The total number of initial solutions *Init* generated in the MDH algorithm is set to 5 to balance the effectiveness and efficiency.

We first conduct an experiment to verify the effectiveness of the mechanisms designed in the enhanced regret evaluation method. We run the MDH algorithm with only the initial solution under the mid-scenario of nine groups of small scale instances, and compare the total number of deterministic UPMSS problems solved and the running time of the algorithm with different regret evaluation mechanisms adopted. Specifically, the instances are chosen from $n \in [9, 15, 20]$, $m \in [3, 5, 7]$, and $\beta = 1$. The algorithms that adopt different regret evaluation mechanisms are represented with the following notations:

- *ORIG*: the basic evaluation method with no enhanced method (traverse and calculate the regret of every extreme scenario of a solution);
- *M3*: use the only method introduced in subsection V-C; namely, only use the upper bound regret of extreme scenarios to eliminate extreme scenarios that are impossible to be the worst-case scenario;
- *M23*: use methods introduced in subsection V-B and V-C; namely, use the upper bound regret of extreme scenarios and Theorem 2 to eliminate some extreme scenarios;
- *M123*: use methods introduced in subsection V-A, V-B, and V-C;
- *M1234*: use the complete procedure of the enhanced regret evaluation method.

The results are presented in Table I. Subcolumn "$\overline{time(s)}$" presents the average running time of the algorithm with the corresponding mechanisms. Subcolumn "$\overline{num}$" reports the average number of deterministic UPMSS problems that are solved for the evaluation in the solving process.

From the results, we could see that from *M1* to *M123*, each mechanism contributes to the reduction of the running time and the number of deterministic UPMSS problems solved significantly. As the data scale increases, the proportion of the running time reduced increases. This result also reflects that the evaluation process accounts for most of the running time of our MDH framework. As for *M123* and *M1234*, *M1234* doesn't perform better than *M123* until the data scale surpasses $m \cdot n^2 = 1200$. That is because the HABC algorithm adopted in subsection E.4 doesn't perform better than the CPLEX solver for small-scale instances. When the data scale increases, as we can see in the cases where $n = 20, m = 5$ and $n = 20, m = 7$, HABC massively reduces the algorithm running time, and contributes a lot to the efficiency of the regret evaluation. Overall, implementing heuristics (HABC) to replace the CPLEX solver in the evaluation suits the real needs in the industry, where the data scale is usually beyond the ability of the solvers.

To sum up, the ERE method contributes a lot to the efficiency of the MDH algorithm. With the implementation of the proposed mechanisms, the evaluation time and the overall running time of the MDH algorithm could be reduced significantly.

In the following experiment, we mainly compare the results of MDH with the Iterative Relaxation exact algorithm (IR) and the Simulated Annealing algorithm (SA) proposed by [28] and [29] for the similar problems. The implementation of IR algorithm for RUPMSS are introduced in Appendix . IR exact algorithm provides us a method to obtain the exact optimal

TABLE I
COMPARISON OF DIFFERENT EVALUATION MECHANISMS

		ORGI		*M3*		*M23*		*M123*		*M1234*	
m	n	$\overline{time(s)}$	$\overline{num}$	$\overline{time(s)}$	$\overline{num}$	$\overline{time(s)}$	$\overline{num}$	$\overline{time(s)}$	$\overline{num}$	$\overline{time(s)}$	$\overline{num}$
	9	9.78	84.90	7.22	56.30	5.94	44.30	3.16	12.25	4.97	12.05
3	15	56.23	206.70	37.76	133.90	35.61	120.80	18.05	52.30	30.71	45.70
	20	264.63	370.35	178.90	238.10	168.82	232.55	113.17	149.45	162.80	151.75
	9	21.66	114.50	14.51	68.80	8.20	35.20	3.55	6.85	7.28	7.45
5	15	325.33	289.50	180.27	158.00	114.99	97.80	32.93	22.65	26.57	14.10
	20	3613.21	507.00	2228.07	252.40	1735.06	194.00	518.24	63.55	125.94	42.50
	9	45.34	165.20	19.29	103.90	7.47	34.90	7.80	3.20	9.26	4.40
7	15	591.22	323.05	309.69	169.30	138.48	75.20	12.15	27.61	52.74	17.10
	20	8625.93	518.35	4600.44	259.45	4164.67	140.40	777.43	31.15	117.33	19.80

solutions for RUPMSS as a benchmark, at least for small-scale instances.

The notations used in the table to present the results are as follows:

- Gap_{mid}: gap between the maximum regret of the solution under mid-scenario and that of the solution by the IR algorithm;
- Gap_{IR}: gap between the maximum regret of the solution by the corresponding algorithm and that of the solution by the IR algorithm;
- *time*: running time of corresponding algorithm;
- *#Opt*: number of instances solved to optimality.

where Gap_{mid} is calculated as follows:

$$Gap_{mid} = \frac{obj(Mid) - obj(IR)}{obj(IR)} \times 100\%$$

and Gap_{IR} is calculated as follows:

$$Gap_{IR} = \frac{obj(Alg) - obj(IR)}{obj(IR)} \times 100\%$$

In these formulae, "obj" is the objective function value, i.e., the maximum regret in this case, of the corresponding algorithm. Specifically, $obj(Mid)$ is the maximum regret of the optimal solution under mid-scenario, which yields an upper bound for RUPMSS with Theorem 5. $obj(Alg)$ is the optimal maximum regret found by the corresponding algorithms (MDH and SA).

First, we pay attention to the numerical results of instances with $\beta = 1$. In other words, the upper bounds of the processing times are 2 times of the corresponding lower bounds for each instance. In Fig.1, we use scatter plots to compare the performances of IR, SA, and MDH. Each point in the plot represents one single instance, and its coordinate is the maximum regret/running time of the corresponding algorithms. The dotted line is $y = x$. The points on this line mean the algorithms corresponding to x-axis and y-axis yield the same performance for this instance. Those points below the $y = x$ line showcase the superior performance of the algorithm corresponding to the y-axis, and vice versa. Note that the logarithm scales on all the plots. To denote instances of different data scales, we define the "data scale" of one instance as $m \cdot n^2$, which is proportional to the number of the x_{ijk} variables in the optimization model. As the legend shows, the points representing small-scale instances (with small $m \cdot n^2$ values) have a brighter color, and larger-scale instances are colored darker.

Fig.1a and Fig.1b are the pairwise regret comparisons for MDH vs.IR and MDH vs.SA. The two graphs have similar distributions. We could see that for most instances, MDH outperforms both IR and SA, and more importantly, the larger the instance scale is, the closer the points tend to lean towards x-axis, which means MDH prevails more massively.

Fig.1c and Fig.1d are the pairwise time comparisons. For Fig.1c, we could see that IR performs better than MDH for the small-scale instances. As the data scale increases, the points flow across the dotted line $y = x$, which suggests that MDH prevails gradually. The vertical line at the right side indicates the instances where IR reaches the 2-hour time limit. Even for these instances, MDH can perform relatively well within the time limit. From Fig.1d, we could see that MDH and SA draw for small-scale instances. But when the data scale becomes larger, the mass of points are located below the dotted line and on the right side of the graph. Similar to Fig.1c, there are also dense points on a vertical line where SA reaches the time limit. For these instances, MDH yields superior performance compared to SA.

For detailed comparisons between the three algorithms, we showcase the results for instances with $\beta = 1, 0.5, 1.5$ in Table II. We start with $\beta = 1$ instances. From the subcolumn "*#Opt*" under IR column, we could see that for those instances with the scale smaller than or equal to "15-3" and "12-7," all the instances could be solved to optimality by IR within 2 hours. For these instances, the average gaps between IR algorithm and the initial solution under the subcolumn "$\overline{Gap_{mid}}$" are all less than 11%, which reflects that the quality of the mid-scenario initial solution we generate is fairly good. When the data scale surpasses $m = 15, n = 5$, some instances may not be solved to optimality by IR in 2 hours. When n reaches 25, for combinations "25-3" and "30-3," the values under subcolumn "$\overline{Gap_{mid}}$" are still negative, which means that for these problem scales, the IR algorithm could still find solutions better than the mid-scenario initial solutions. But when the number of machines m grows to 5, the values under subcolumn "$\overline{Gap_{mid}}$" are 0, that is to say, the solution returned by the IR algorithm has no improvement over the mid-scenario initial solution. As for the efficiency of the IR

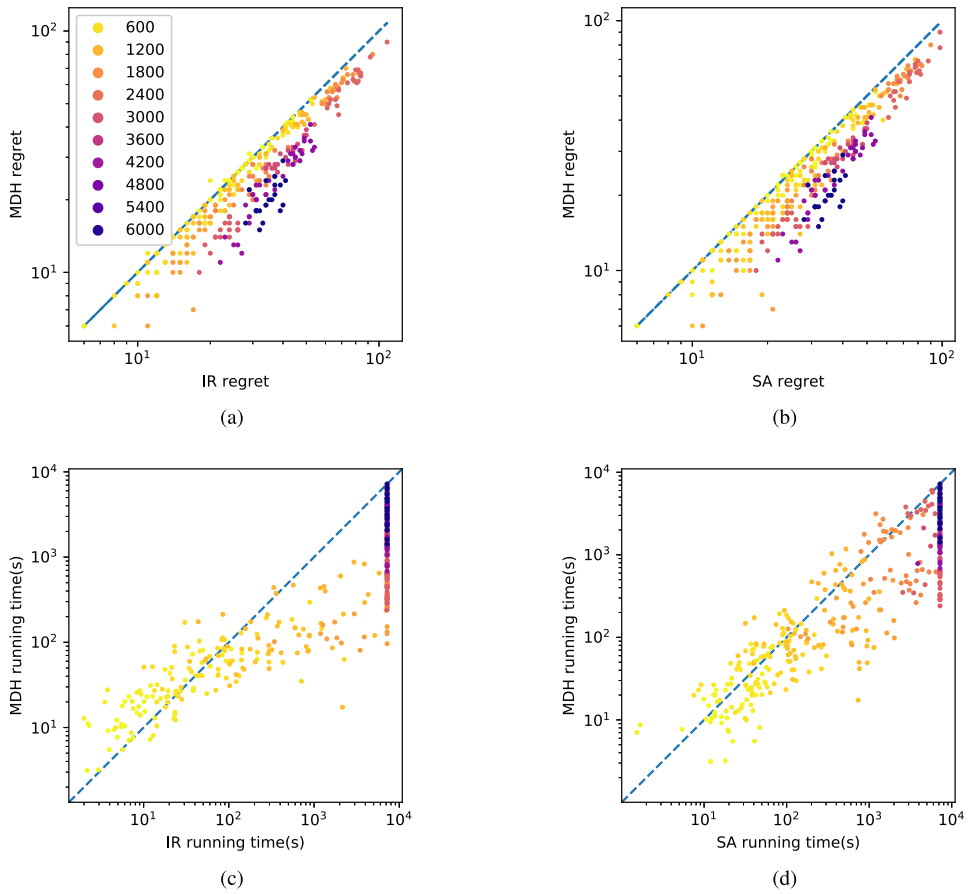

Fig. 1. Comparison between MDH, IR, and SA. (a) Regret comparison: MDH vs. IR. (b) Regret comparison: MDH vs. SA. (c) Time comparison: MDH vs. IR. (d) Time comparison: MDH vs. SA.

algorithm, as the scale of the problem becomes larger, the running time of the algorithm increases rapidly. When the data scale grows to $n = 20, m = 5$, IR reaches the 2-hour time limit.

For the MDH algorithm, from the subcolumn "$\overline{Gap_{IR}}$" we could see that for those instances combinations that IR algorithm could yield optimal solutions, the gaps between MDH and IR are small. Besides, the number of instances that could be solved to optimality by MDH is close to that of IR. From the subcolumn "$\overline{time}$" for the MDH algorithm, we could see that except for the combinations with 9 jobs ($n = 9$), the running time of the MDH algorithm is less than that of the IR algorithm. For larger scale instances, the average gaps between MDH and IR are negative while the average running times of MDH are shorter, which indicates that MDH could find better solutions than IR in a shorter time. Due to that IR cannot find all the optimal solutions for large-scale instances, the number of optimal solutions by MDH cannot be counted and dash symbols "—" are presented in column "*#Opt*".

Compared to the SA algorithm, for all combinations, the MDH algorithm obtains lower gaps and more optimal solutions. As for the efficiency of the algorithms, the MDH algorithm consumes less time than the SA algorithm for nearly all combinations. So the MDH algorithm we propose performances better than the SA algorithm in terms of both efficiency and effectiveness.

For instances with $\beta = 0.5$ and $\beta = 1.5$, Table II demonstrate similar results to $\beta = 1$. From Table II we could see that MDH consistently outperforms SA and IR in terms of time efficiency and solution quality, regardless of the variations of data settings. We further conduct the sensibility analysis with these results. For those instances that IR could yield optimal solutions, it takes shorter time for IR when $\beta = 1.5$ than $\beta = 0.5$. With the β decreases, the running time and $\overline{Gap_{mid}(\%)}$ of IR becomes worse. When $\beta = 1.5$, SA reaches its time limit at "25-7" and obtains the best $\overline{Gap_{IR}(\%)}$ -0.94% at "25-5". While when $\beta = 0.5$, SA reaches its 7200-second time limit at the "20-5" data scale and the lowest $\overline{Gap_{IR}(\%)}$ is -0.66% at the "25-7" data scale. These results indicate that the overall computational complexity for instances with smaller β is higher due to the shorter time intervals, and SA's performance is unstable to different values of β. In contrast, MDH keeps a consistent performance for all the instances with $\beta = 0.5, 1, 1.5$. For instance, the maximum running times for MDH are all less than 6601.23s and do not reach the time limit. For all these three β values, MDH's running time becomes smaller than IR at "12-5," and begins to yield a negative $\overline{Gap_{IR}(\%)}$ since "20-5" data scale. It demonstrates that MDH is more robust

TABLE II
COMPARISON OF DIFFERENT ALGORITHMS

			IR			*SA*			*MDH*		
β	n	m	$\overline{Gap_{mid}}(\%)$	*#Opt*	$\overline{time}(s)$	$\overline{Gap_{IR}}(\%)$	*#Opt*	$\overline{time}(s)$	$\overline{Gap_{IR}}(\%)$	*#Opt*	$\overline{time}(s)$
0.5	9	3	8.24	20	7.36	1.51	17	32.37	2.79	17	14.24
		5	10.24	20	22.67	5.96	12	61.22	7.25	14	27.26
		7	23.50	20	30.35	14.75	16	91.26	1.25	19	47.08
	12	3	10.78	20	32.18	5.00	14	81.21	5.33	12	33.76
		5	8.59	19	1073.47	6.63	—	621.39	0.50	—	78.84
		7	8.00	20	355.69	7.54	16	670.01	6.79	14	103.41
	15	3	6.79	18	1629.90	2.22	—	367.99	3.09	—	152.15
		5	6.93	10	4896.68	6.47	—	4023.54	1.70	—	152.26
		7	8.67	3	6550.71	5.78	—	6075.32	1.97	—	408.07
	20	3	5.04	6	6021.24	3.38	—	3565.70	2.33	—	602.62
		5	0.48	0	7200.00	0.38	—	7200.00	-3.38	—	1179.8
		7	1.05	0	7200.00	1.05	—	7200.00	-6.82	—	814.26
	25	3	0.81	0	6984.37	0.00	—	7200.00	-0.16	—	1086.94
		5	0.00	0	7200.00	0.00	—	7200.00	-3.41	—	2197.68
		7	0.00	0	7200.00	-0.66	—	7200.00	-3.94	—	5176.76
	30	3	0.48	0	7200.00	0.47	—	7200.00	-2.38	—	2545.73
		5	0.00	0	7200.00	0.00	—	7200.00	-2.90	—	5445.41
		7	0.00	0	7200.00	0.00	—	7200.00	-2.01	—	6601.23
1	9	3	12.12	20	5.84	3.42	14	15.99	0.78	18	12.45
		5	13.17	20	14.85	7.35	10	26.30	1.45	18	16.83
		7	9.70	20	33.94	7.87	11	29.48	3.58	14	32.64
	12	3	6.86	20	47.79	3.61	12	47.70	0.70	18	46.39
		5	12.21	20	250.48	9.52	9	112.57	0.96	17	48.14
		7	10.49	20	280.80	8.14	10	96.19	3.54	14	92.55
	15	3	8.02	20	227.93	4.91	9	117.50	0.62	17	118.63
		5	20.39	19	1479.40	18.68	—	719.30	1.00	—	122.53
		7	9.35	11	4603.14	6.95	—	802.43	0.91	—	189.83
	20	3	8.45	14	3379.44	5.23	—	457.66	1.84	—	550.17
		5	6.74	0	7200.15	1.46	—	4130.64	-6.25	—	552.49
		7	1.63	0	7200.14	0.34	—	6747.23	-6.35	—	447.93
	25	3	4.32	0	7200.00	2.00	—	1877.93	-1.90	—	1894.12
		5	0.00	0	7200.00	-0.57	—	6995.62	-5.26	—	1461.08
		7	0.00	0	7200.00	0.00	—	7200.00	-0.94	—	856.57
	30	3	4.31	0	7200.00	0.70	—	5647.67	-3.92	—	4377.68
		5	0.00	0	7200.00	-0.21	—	7200.00	-6.80	—	3741.72
		7	0.00	0	7200.00	0.00	—	7200.00	-11.03	—	3938.84
1.5	9	3	5.54	20	6.69	1.60	16	11.43	1.03	18	10.27
		5	11.63	20	20.14	8.12	12	16.83	0.85	17	24.24
		7	9.40	20	34.20	5.20	13	25.98	0.82	19	37.64
	12	3	11.49	20	22.10	6.53	13	33.59	0.16	19	37.37
		5	10.97	20	144.61	8.24	8	67.70	1.41	15	66.84
		7	12.87	20	255.14	11.09	9	98.23	1.41	18	88.02
	15	3	9.79	20	289.83	6.95	9	76.83	0.44	17	168.74
		5	9.55	18	1845.09	5.65	—	378.27	2.22	—	161.06
		7	12.61	17	3129.19	10.27	—	363.32	0.60	—	219.36
	20	3	6.30	18	2223.92	3.39	—	575.28	1.51	—	686.97
		5	4.00	0	7200.00	2.70	—	2377.93	-4.60	—	841.26
		7	1.93	0	7200.00	-0.42	—	3602.71	-6.82	—	814.26
	25	3	6.38	0	7200.00	3.26	—	1393.98	-9.80	—	728.00
		5	0.40	0	7200.00	-0.94	—	6178.52	-6.06	—	1664.56
		7	0.00	0	7200.00	-0.50	—	7200.00	-10.90	—	1206.31
	30	3	2.67	0	7200.00	0.66	—	7200.00	-2.13	—	5386.16
		5	0.00	0	7200.00	0.00	—	7200.00	-11.74	—	4139.90
		7	0.00	0	7200.00	0.00	—	7200.00	-7.50	—	5139.99

than other algorithms and not so sensitive to the variations of data.

VIII. Conclusion

We study the min-max regret robust optimization on unrelated parallel machine scheduling with sequence-dependent setup times, which originates from a high-end equipment factory in eastern China. We first propose a robust optimization model and develop some critical properties regarding the worst-case scenario's characteristics. An ERE method is then proposed to accelerate the process of evaluating the maximum regret of a solution. A MDH algorithm is designed based on the findings on the decomposition property and the characteristics of the shift and interchange search methods. Numerical experiments show that the ERE method we propose significantly improves the efficiency of finding the worst-case scenario and the corresponding maximum regret. The MDH algorithm we proposed significantly outperforms other exact and heuristic algorithms designed for similar problems and is sufficiently robust to deal with various scenarios. In future work, we could further develop algorithms that could solve larger-scale instances. Besides, we can further explore more uncertainty conditions, e.g., the uncertainty of the setup times. The distributionally robust optimization (DRO) counterpart of this problem may also be considered. DRO could take more known statistical information beyond the intervals of the random parameters into account to further avoid the conservativeness of the robust approach.

Appendix
The Iterative Relaxation Exact Algorithm (IR)

A widely used approach for min-max robust optimization problems is the Iterative Relaxation algorithm (IR) (see Xu *et al.* [28] and Feng *et al.* [29]). The approach starts with a subset S^0 and adds one additional scenario from S to this subset iteratively. In every iteration, an upper bound and a lower bound are obtained respectively, and the gap becomes smaller after every iteration. The algorithm is terminated when the gap reaches 0 and an exact optimal solution is found.

We formulate the h-th relaxed model in the h-th iteration as follows, and define the current subset of scenarios as S^h:

$$\min_{\pi} \ r \tag{33}$$

$$\text{s.t. } C_k^s - C_j^s + V\left(1 - x_{ijk}\right) \geq s_{ijk} + p_{ik}^s, \quad j \in N_0,\ j \neq k,\ k \in N,\ i \in M,\ s \in S^h \tag{34}$$

$$\sum_{j \in N_0, k \in N, j \neq k} s_{ijk} x_{ijk} + \sum_{j \in N} p_{ij}^s y_{ij} \leq C_{\max}^s, \quad i \in M,\quad s \in S^h \tag{35}$$

$$C_{\max}^s \geq C_j^s, \quad j \in N,\ s \in S^h \tag{36}$$

$$C_0^s = 0, \quad s \in S^h \tag{37}$$

$$C_{\max}^s - F_s^* \leq r, \quad s \in S^h \tag{38}$$

$$x_{ijk} \in \{0,1\}, C_j^s \geq 0,\ C_{\max}^s \geq 0, \quad j,k \in N_0,\ i \in M,\ s \in S^h$$

$$y_{ij} \geq 0, \quad \forall i \in M,\ j \in N$$

$$\text{Constraints (2)-(5)} \tag{39}$$

Starting with $S^0 = \emptyset$, in the h-th iteration, we solve the h-th relaxed model above to obtain solution π^h. Its optimal objective value r^h gives a lower bound. The maximum regret of π, $R_{\max}(\pi^h)$ provides an upper bound. The worst-case scenario s^{h+1} for π^h is generated and $S^{h+1} = S^h \cup \{s^{h+1}\}$. The gap $R_{\max}(\pi^h) - r^h$ becomes smaller during the iteration process. When the gap becomes 0, the optimal solution is found.

As the iterations proceed, not only the number of the constraints becomes larger, but also the scale of the decision variables in the relaxed model is growing, which adds to the difficulty of iteration. So the time for this exact algorithm to find the optimal solution grows rapidly as the scale of the problem becomes larger, and it completely fails to solve large-scale cases.

References

[1] S. C. H. Leung, Y. Wu, and K. K. Lai, "A stochastic programming approach for multi-site aggregate production planning," *J. Oper. Res. Soc.*, vol. 57, no. 2, pp. 123–132, Feb. 2006.

[2] B. L. Gorissen, İ. Yanıkoğlu, and D. D. Hertog, "A practical guide to robust optimization," *Omega*, vol. 53, pp. 124–137, Jun. 2015.

[3] A. Kasperski and P. Zieliński, "Robust discrete optimization under discrete and interval uncertainty: A survey," in *Robustness Analysis in Decision Aiding, Optimization, and Analytics*. Cham, Switzerland: Springer, 2016, pp. 113–143.

[4] J. Pereira and I. Averbakh, "Exact and heuristic algorithms for the interval data robust assignment problem," *Comput. Oper. Res.*, vol. 38, no. 8, pp. 1153–1163, 2011.

[5] W. Wu, M. Iori, S. Martello, and M. Yagiura, "Exact and heuristic algorithms for the interval min-max regret generalized assignment problem," *Comput. Ind. Eng.*, vol. 125, pp. 98–110, Nov. 2018.

[6] W. Liu, T. Deng, and J. Li, "Product packing and stacking under uncertainty: A robust approach," *Eur. J. Oper. Res.*, vol. 277, no. 3, pp. 903–917, Sep. 2019.

[7] C. Lee, K. Lee, and S. Park, "Robust vehicle routing problem with deadlines and travel time/demand uncertainty," *J. Oper. Res. Soc.*, vol. 63, no. 9, pp. 1294–1306, 2012.

[8] D. Lu and F. Gzara, "The robust vehicle routing problem with time windows: Solution by branch and price and cut," *Eur. J. Oper. Res.*, vol. 275, no. 3, pp. 925–938, Jun. 2019.

[9] H. Aissi, C. Bazgan, and D. Vanderpooten, "Min–max and min–max regret versions of combinatorial optimization problems: A survey," *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 427–438, 2009.

[10] C. Artigues, R. Leus, and F. T. Nobibon, "Robust optimization for resource-constrained project scheduling with uncertain activity durations," *Flexible Services Manuf. J.*, vol. 25, nos. 1–2, pp. 175–205, Jun. 2013.

[11] M. E. Bruni, L. Di Puglia Pugliese, P. Beraldi, and F. Guerriero, "An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations," *Omega*, vol. 71, pp. 66–84, Sep. 2017.

[12] K. K. H. Ng, C. K. M. Lee, F. T. S. Chan, and Y. Qin, "Robust aircraft sequencing and scheduling problem with arrival/departure delay using the min-max regret approach," *Transp. Res. E, Logistics Transp. Rev.*, vol. 106, pp. 115–136, Oct. 2017.

[13] M. Kong, J. Pei, J. Xu, X. Liu, X. Yu, and P. M. Pardalos, "A robust optimization approach for integrated steel production and batch delivery scheduling with uncertain rolling times and deterioration effect," *Int. J. Prod. Res.*, vol. 58, no. 17, pp. 5132–5154, Sep. 2020.

[14] A. Kasperski, "Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion," *Oper. Res. Lett.*, vol. 33, no. 4, pp. 431–436, 2005.

[15] A. Kasperski and P. Zieliński, "A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion," *Oper. Res. Lett.*, vol. 36, no. 3, pp. 343–344, May 2008.

[16] J. Pereira, "The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective," *Comput. Oper. Res.*, vol. 66, pp. 141–152, Feb. 2016.

[17] B.-C. Choi and K. Chung, "Min–max regret version of a scheduling problem with outsourcing decisions under processing time uncertainty," *Eur. J. Oper. Res.*, vol. 252, no. 2, pp. 367–375, Jul. 2016.
[18] M. Drwal, "Robust scheduling to minimize the weighted number of late jobs with interval due-date uncertainty," *Comput. Oper. Res.*, vol. 91, pp. 13–20, Mar. 2018.
[19] M. Drwal and J. Józefczyk, "Robust min–max regret scheduling to minimize the weighted number of late jobs with interval processing times," *Ann. Oper. Res.*, vol. 284, no. 1, pp. 263–282, Jan. 2020.
[20] S. Wang, W. Cui, F. Chu, J. Yu, and J. N. D. Gupta, "Robust (min–max regret) single machine scheduling with interval processing times and total tardiness criterion," *Comput. Ind. Eng.*, vol. 149, Nov. 2020, Art. no. 106838.
[21] P. Kouvelis, R. L. Daniels, and G. Vairaktarakis, "Robust scheduling of a two-machine flow shop with uncertain processing times," *IIE Trans.*, vol. 32, no. 5, pp. 421–432, May 2000.
[22] S. M. Gholami-Zanjani, M. Hakimifar, N. Nazemi, and F. Jolai, "Robust and fuzzy optimisation models for a flow shop scheduling problem with sequence dependent setup times: A real case study on a PCB assembly company," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 6, pp. 552–563, Jun. 2017.
[23] W. Liao and Y. Fu, "Min–max regret criterion-based robust model for the permutation flow-shop scheduling problem," *Eng. Optim.*, vol. 52, no. 4, pp. 687–700, Apr. 2020.
[24] M. Drwal and R. Rischke, "Complexity of interval minmax regret scheduling on parallel identical machines with total completion time criterion," *Oper. Res. Lett.*, vol. 44, no. 3, pp. 354–358, May 2016.
[25] X. Xu, J. Lin, and W. Cui, "Hedge against total flow time uncertainty of the uniform parallel machine scheduling problem with interval data," *Int. J. Prod. Res.*, vol. 52, no. 19, pp. 5611–5625, Oct. 2014.
[26] M. Siepak and J. Józefczyk, "Solution algorithms for unrelated machines minmax regret scheduling problem with interval processing times and the total flow time criterion," *Ann. Oper. Res.*, vol. 222, no. 1, pp. 517–533, Nov. 2014.
[27] E. Conde, "A MIP formulation for the minmax regret total completion time in scheduling with unrelated parallel machines," *Optim. Lett.*, vol. 8, no. 4, pp. 1577–1589, Apr. 2014.
[28] X. Xu, W. Cui, J. Lin, and Y. Qian, "Robust makespan minimisation in identical parallel machine scheduling problem with interval data," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3532–3548, Jun. 2013.
[29] X. Feng, F. Zheng, and Y. Xu, "Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times," *Int. J. Prod. Res.*, vol. 54, no. 12, pp. 3706–3717, Jun. 2016.
[30] H. Hu, K. K. H. Ng, and Y. Qin, "Robust parallel machine scheduling problem with uncertainties and sequence-dependent setup time," *Sci. Program.*, vol. 2016, pp. 1–13, Nov. 2016.
[31] S. Wang and W. Cui, "Approximation algorithms for the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time," *Int. J. Prod. Res.*, vol. 59, no. 15, pp. 4579–4592, 2020.
[32] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.
[33] A. Guinet, "Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria," *Int. J. Prod. Res.*, vol. 31, no. 7, pp. 1579–1594, Jul. 1993.
[34] O. Avalos-Rosales, F. Angel-Bello, and A. Alvarez, "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times," *Int. J. Adv. Manuf. Technol.*, vol. 76, nos. 9–12, pp. 1705–1718, Feb. 2015.
[35] L. Fanjul-Peyro, R. Ruiz, and F. Perea, "Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times," *Comput. Oper. Res.*, vol. 101, pp. 173–182, Jan. 2019.
[36] T. T. Tran, A. Araujo, and J. C. Beck, "Decomposition methods for the parallel machine scheduling problem with setups," *INFORMS J. Comput.*, vol. 28, no. 1, pp. 83–95, Feb. 2016.
[37] S.-W. Lin and K.-C. Ying, "ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times," *Comput. Oper. Res.*, vol. 51, pp. 172–181, Nov. 2014.
[38] L. Wang, S. Wang, and X. Zheng, "A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times," *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 3, pp. 235–246, Jul. 2016.
[39] J.-P. Arnaout, "A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times," *Ann. Oper. Res.*, vol. 285, nos. 1–2, pp. 273–293, Feb. 2020.
[40] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 345–378, 2015.
[41] J. P. Arnaout, G. Rabad, and R. Musa, "A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times," *J. Int. Manuf.*, vol. 21, no. 6, pp. 693–701, Dec. 2010.
[42] A. Cossari, J. C. Ho, G. Paletta, and A. J. Ruiz-Torres, "Minimizing workload balancing criteria on identical parallel machines," *J. Ind. Prod. Eng.*, vol. 30, no. 3, pp. 160–172, Apr. 2013.
[43] A. Cossari, J. C. Ho, G. Paletta, and A. J. Ruiz-Torres, "A new heuristic for workload balancing on identical parallel machines and a statistical perspective on the workload balancing criteria," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1382–1393, Jul. 2012.
[44] W.-H. Chen and C.-S. Lin, "A hybrid heuristic to solve a task allocation problem," *Comput. Oper. Res.*, vol. 27, no. 3, pp. 287–303, Mar. 2000.
[45] R. Martí, "Multi-start methods," in *Handbook of Metaheuristics*. Boston, MA, USA: Springer, 2003, pp. 355–368.
[46] R. Martí, M. G. Resende, and C. C. Ribeiro, "Multi-start methods for combinatorial optimization," *Eur. J. Oper. Res.*, vol. 226, no. 1, pp. 1–8, 2013.
[47] R. L. Daniels and P. Kouvelis, "Robust scheduling to Hedge against processing time uncertainty in single-stage production," *Manage. Sci.*, vol. 41, no. 2, pp. 363–376, Feb. 1995.

Weihao Wang (Member, IEEE) received the B.S. degree in telecommunications engineering from Tianjin University, Tianjin, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Industrial Engineering and Management, Peking University, Beijing, China.

His research interests include systems optimization, operations management, intelligent manufacturing, and supply chain management.

Chutong Gao (Student Member, IEEE) is currently pursuing the B.Eng. degree in engineering mechanics with the Department of Mechanics and Engineering Science, Peking University, Beijing, China.

His research interests include operations management, machine learning, and supply chain management.

Leyuan Shi (Fellow, IEEE) received the B.S. degree in mathematics from Nanjing Normal University, Nanjing, China, in 1982, the M.S. degree in applied mathematics from Tsinghua University, Beijing, China, in 1985, and the M.S. degree in engineering and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, USA, in 1990 and 1992, respectively.

She is currently a Professor with the Department of Industrial and Systems Engineering, University of Wisconsin–Madison, Madison, WI, USA. She has authored or coauthored two books, including *Nested Partitions Method, Theory and Applications* (Springer, 2009) and *Modeling, Control and Optimization of Complex Systems* (Springer, 2003). Her research is devoted to the theory and development of large-scale optimization algorithms, discrete-event simulation methodology, and modeling and analysis of discrete dynamic systems, with applications to complex systems, such as supply chain networks, manufacturing systems, communication networks, and financial engineering.

Dr. Shi is a member of INFORMS. She is currently an Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING and an Associate Editor of *Discrete Event Dynamic Systems*.