IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 13, 2019, accepted November 28, 2019, date of publication December 2, 2019, date of current version January 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2019.2957132

Single-Machine Rework Rescheduling to Minimize Total Waiting Time With Fixed Sequence of Jobs and Release Times

YANDONG GUO[1], MIN HUANG[2,3], QING WANG[2,3], AND V. JORGE LEON[4]

[1]College of Mathematics and Physics, Bohai University, Jinzhou 121013, China
[2]College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[3]State Key Laboratory of Synthetical Automation for Process Industries (Northeastern University), Shenyang 110819, China
[4]Department of Engineering Technology and Industrial Distribution and Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843-3367, USA

Corresponding author: Yandong Guo (gyd1030@163.com)

This work was supported in part by the Major International(Regional) Joint Research Project of NSFC under Grant 71620107003, in part by the National Science Foundation for Distinguished Young Scholars of China under Grant 71325002, in part by the National Natural Science Foundation of China under Grant 22078024 and Grant 21506014, in part by the Program for Liaoning Innovative Research Term in University under Grant LT2016007, in part by the Foundation for Innovative Research Groups of National Science Foundation of China under Grant 61621004, in part by the 111 Project (B16009), in part by the Fundamental Research Funds for State Key Laboratory of Synthetical Automation for Process Industries under Grant 2013ZCX11, in part by the Natural Science Foundation of Liaoning Province of China under Grant 201602003, and in part by the Educational Commission of Liaoning Province of China under Grant 2016006.

ABSTRACT Motivated by energy-saving efforts in the quartz manufacturing industry, this study investigates a single-machine rescheduling problem for a set of newly arrived rework jobs. The original jobs have release times, and the rework jobs need to be separated in a schedule already in progress. The objective is to achieve energy savings by minimizing the total waiting time. A mixed-integer linear programming model is formulated and its NP-hardness is proved. Properties of the optimal solution are derived and used to design the subsequent algorithms. A pseudo-polynomial optimal algorithm for a special case, a heuristic algorithm and a genetic algorithm with an adaptive local search mechanism for general problems are developed and tested. Numerical simulations show that the genetic algorithm yields high-quality solutions under various conditions. A case study suggests that significant energy savings can be achieved with the proposed rescheduling methodology.

INDEX TERMS Rescheduling, waiting time, mixed-integer linear programming, heuristics, rule-guided adaptive genetic algorithm, NP-hard.

I. INTRODUCTION

Optimal planning, management, and use of energy are critical because energy sources are scarce and energy generation processes can have significantly adverse impact on the environment. The manufacturing industry is one of the main consumers of energy in the world; hence, optimizing the use of energy in manufacturing is always concerned by enterprises. The industries of energy intensive processes are of particular interest from this perspective. Many of these industries are characterized by manufacturing processes subject to working conditions (e.g., temperature and pressure) that are very different from ambient conditions. Examples of these industries are steel mills, glass/quartz manufacturing, and hot metal forming and rolling.

Optimizing the use of energy in hot processes is of particular significance in the case of China given its size and growth in manufacturing, and its low energy utilization rate relative to other developed countries. In China, the energy costs of hot processing account for 25%–40% of total production costs. The annual electricity consumption is approximately 500 kW/h in hot processing, and the energy consumption ratio is 500–1000 kW/ton. The utilization of professionals for hot processing is approximately more than 80% in industrialized countries, while it is approximately 20% in China. Owing to the urgent need of reducing energy consumption and related adverse environmental impacts, it is important to improve the level of professionalism in production management.

This study is inspired by a process occurring in the quartz manufacturing industry. Quartz glass products are widely used in architecture, chemistry, medical treatment, electronics, aerospace, etc. Quartz products are assembled using three sequential manufacturing processes: preheating,

The associate editor coordinating the review of this manuscript and approving it for publication was Nicola Andriolli.

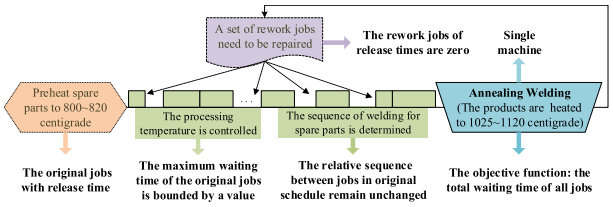

FIGURE 1. Rescheduling problem for quartz manufacturing.

welding, and annealing. Welding involves localized heating of the joint surfaces to temperatures near the melting point of 1600–1700°C and preheating of the whole component to at least 800–820°C is necessary to avoid cracking during the welding process. After welding, the product is annealed to eliminate any internal stresses caused by the welding process. Annealing involves heating the product to approximately 1025°C for a pre-specified amount of time. Parts are preheated and must wait in queue before welding; however, products flow directly without queuing from welding to annealing. An opportunity for energy savings occurs in the queue between preheating and welding. Preheated parts in the queue when they are cooling down and waiting to be welded. The longer the wait time, the more energy needed to heat the parts to annealing temperature. Hence, the energy will be saved by minimizing the total waiting time in this queue.

This study considers a rescheduling problem rather than a scheduling problem. Quartz products are particularly fragile and are also subject to high quality requirements. Parts with defects after annealing can be fixed by a rework pass through welding and annealing. Hence, a set of rework parts joins the sequence of original jobs to form the input queue of welding. The rescheduling problem is to decide how to schedule each rework job with the original job sequence such that the total waiting time is minimized. Fig. 1 illustrates the rescheduling problem under consideration. The rescheduling problem widely exists in discrete manufacturing industry with heat treatment process, e.g., semiconductor processing and ceramic products.

With respect to the concerned rescheduling problem in this paper, firstly, an optimization model of mixed-integer linear programming (MILP) is formulated and its NP completeness is proved. some useful properties of the optimal solution are deduced for the subsequently developed algorithms. And then a special case is discussed as NP-hard in the ordinary sense and solved by the designed pseudo-polynomial optimal algorithm. For general problems, a heuristic algorithm is developed, and its optimality is testified by five special problem cases. A genetic algorithm with adaptive local search scheme (GAA) for the general problem is designed. Numerical experiments show GAA obtains satisfactory solutions effectively. With respect to practical issues in the industry, a case study suggests that significant energy savings can be achieved with the proposed rescheduling methodology.

II. LITERATURE REVIEW

Early works on rescheduling is credited to Leon [1] and Wu *et al.* [2] Actual workshop production tends to be disturbed by unpredictable events. Vieira *et al.* [3] and Herrmann *et al.* [4] show the taxonomy of rescheduling literature.

This study investigates a single-machine rescheduling problem to incorporate rework jobs into a given existing schedule to minimize the total waiting time of all jobs. Unal *et al.* [5] investigated a similar problem considering sequence-dependent family setup times to minimize deviations of the jobs from their original due dates while minimizing the makespan of new jobs.

Hall *et al.* [6] solved single-machine rescheduling problems by polynomial time algorithms or discussing their NP-hardness. In each problem, the objective function contains a function of the completion time and a function of costs by disrupted. The degree of changes between the original and rescheduling schedules is defined in terms of sequence disruptions and time disruptions. Then they [7] studied a single-machine rescheduling problem with multiple disruptions and developing branch-and-bound algorithm. In a similar line of work, Yuan *et al.* [8] considered a single-machine problem to minimize the makespan subject to the constraint of disruption criterion bounded by a given value, release times, and the assumption that the original sequence is optimal. Hoogeveen *et al.* [9] solved a single-machine problem aiming at minimizing total setup time by polynomial time algorithms or NP-hardness proofs. They also studied a rescheduling problem to minimize the disruption cost and total setup times.

Yang [10] considered a single-machine problem by determining the order of rework jobs and their actual processing times to minimize the convex combination of the total costs of compression, disruption criterion, and classic scheduling criterion. Zhao *et al.* [11] considered the deteriorating jobs problem with processing times of the jobs depending on their starting time. Guo *et al.* [12] studied a single-machine rescheduling problem for original jobs with release times to minimize the total waiting time and proposed a heuristic algorithm. Liu *et al.* [13] studied a single machine problem to minimize cost considering unavailable time period for machine. Le *et al.* [14] considered a rescheduling problem to minimize the maximum lateness subject to an upper bound on the total sequence disruption. A two-stage heuristic algorithm and an exact branch-and-bound algorithm were developed to solve this problem. Zhao *et al.* [15] considered a single machine rescheduling problem to minimize the maximum lateness under the sequence disruptions of schedule. They showed that the three problems and an extension problem were equivalent and can be solved in polynomial time. Guo *et al.* [16] studied a single-machine rework rescheduling to minimize the maximum waiting-time. They proposed a heuristic algorithm, which assumes that the rework jobs have short processing times (i.e., less than the minimum regular processing time) in their problem. Wang *et al.* [22]–[24] considered a set of single machine rescheduling problems, considering machine breakdown and deterioration effect, preventive maintenance for the new jobs, with dynamic multiple objectives. Gao *et al.* [25] discussed the scheduling and rescheduling problem as a case

of reprocessing problems with increasing processing time and new job arrival. The rescheduling problems in flowshop and parallel-machine environment were studied respectively by Kunkun, Wang and Yin *et al.* [26]–[28]. Jiang *et al.* [29]–[31] considered some job shop scheduling problems about the energy saving issue in the manufacturing system.

III. MILP MODEL AND PROBLEM COMPLEXITY

The section presents a MILP model for a Single-machine Rescheduling problem with Release times for Original jobs (SRRO) and analyzes the problem complexity.

In the quartz manufacturing described above, the welding and annealing processes are considered as a single-machine process since the parts flow continues between the two processes. As previously mentioned, the rescheduling problem is to optimize the reheating energy for annealing by minimizing the total waiting time of jobs in the input queue of the welding process. Some important technical assumptions include:

a) Preheating leads to conflicting release times for all original jobs, while the release times of rework jobs are zero as they do not need re-preheating.
b) Science typecast products of quartz glass were combined with original jobs according to a certain logical sequence; a right-shift control policy is used when inserting rework jobs into the original sequence such that the original sequence is maintained.
c) The waiting times of the original jobs are subjected to a threshold, because the time in which any original job can wait in the queue is limited to the time it takes for its temperature dropping to the minimum preheating temperature of 800–820 °C.

A. PROBLEM FORMULATION

Let $J_o = \{1, \cdots, n_o\}$ be the set of original jobs with distinct release times, which is scheduled for processing non-preemptively on a single machine. The original schedule v of jobs in J_o is known, hence, the start time s_j of each original job denoted by $s_j(v)$, $j \in J_o$ is known. A set of new rework jobs $J_R = \{n_o + 1, \cdots, n_o + n_R\}$ is released at time 0 and needs to be scheduled by inserting them into the original schedule. Let $J = J_R \bigcup J_o$ and $n = |J| = n_o + n_R$. Each job j in J can be scheduled at its release time r_j and requires a processing time p_j, both r_j and p_j are nonnegative integers. The SRRO is to produce a feasible rescheduling denoted by σ, and let σ^* denote its optimal solution which minimizes the total waiting time $\sum w_j(\sigma)$, where $w_j(\sigma) = s_j(\sigma) - r_j$ for jobs in J. $s_j(\sigma)$, $j \in J$ are decision variables. Note that for technical consideration in manufacturing process, when the original jobs are rescheduled, it is required that the relative sequence between jobs in v should preserve and the maximum waiting time of the original jobs is bounded by a threshold value K. (Technical assumption(c) requires the waiting time of each original job is subjected to a threshold K_i, $i \in J_o$, without loss of generality, the boundary is applied as the common maximum waiting time for the convenience of modeling.)

Among all job permutations of σ or σ^*, the feasible rescheduling is defined as an active schedule which dominates all others, where each job is processed as early as possible once the processing order is determined. x_{ij} and s_i are decision variables, where x_{ij} equals to 1 implies that job i precedes job j and zero otherwise; s_i indicates the starting time of job i.

Based on the mixed integer programming denotation proposed by Nemhauser *et al.* [17], the mathematical model of SRRO is formulated as follows:

$$(SRRO): \quad \min \sum_{i=1}^{n} w_i(\sigma) \tag{1}$$

$$s.t.\ x_{ij} + x_{ji} = 1 \quad i, j \in J, i < j \tag{2}$$

$$x_{ij} + x_{jk} + x_{ki} \le 2 \quad i, j, k \in J,\ i \ne j \ne k \tag{3}$$

$$s_j(\sigma) \ge s_i(v)x_{ij} + \sum_{k<i, k\ne j} p_k(x_{ik} + x_{kj} - 1) + \sum_{k\ge i, k\ne j} p_k x_{kj} \quad i, j \in J \tag{4}$$

$$s_i(v) = 0 \quad i \in J_R \tag{5}$$

$$x_{jj} = 1 \quad j \in J \tag{6}$$

$$s_j(\sigma) \ge s_j(v) \quad j \in J_o \tag{7}$$

$$x_{ij} = 1 \quad i, j \in J_o, i < j \tag{8}$$

$$s_j(\sigma) - r_j \le K \quad j \in J_o \tag{9}$$

$$x_{ij} \in \{0, 1\} \quad i, j \in J \tag{10}$$

where (1) is the objective function of the total waiting time of all jobs in rescheduling. (2) is a set of conflict constraints to ensure either job j being processed before job i or job i is processed before job j. (3) is the transitivity constraints that ensure a linear order between the three jobs, such that, if job i is processed before job j and job j is processed before job k, and the k is processed after j, i.e., the linear order is fixed as i, j and k. In the original schedule, (4) to (7) arrange the jobs in non-increasing order of the starting times. The original jobs with release time required by assumption (a) are constrained with the linear ordering variables. (8) ensures the original sequence being maintained in rescheduling as the requirement of assumption (b), whereas (9) implies the waiting times of the original jobs are subjected to a threshold as the requirement of assumption (c). Finally, (10) denotes the integrality constraints.

B. COMPLEXITY ANALYSIS

In this section, we provide the proof that SRRO is NP-Complete.

Theorem 1: SRRO is NP-Complete.

Proof: The proof of NP-Complete is by reduction from the classic NP-Complete problem: Even-Odd Partition Problem [18], i,e., given a set $\{a_1, \cdots, a_{2t}\}$ of positive integers, where $a_i > a_{ij}$ for $1 \le i \le 2t - 1$ and $\sum_{i=1}^{2t} a_i = 2A$, whether there exists a partition of the index set $S = \{1, \cdots, 2t\}$ into disjunct subsets S_1 and S_2, such that $\sum_{i\in s_1} a_i = \sum_{i\in s_2} a_i = A$, and that each contains exactly one of the indices $2i - 1$ and $2i$ for $i = 1, \cdots, t$.

If there exists a solution to a given case of the Even-Odd Partition Problem, then we let $\delta_1, \cdots, \delta_t$ denote such a partition, where $\delta_i = 0$ if $2i-1 \in S_1$ and $\delta_i = 1$ if $2i \in S_i$ for $i = 1, \cdots, t$, i.e, $\sum_{i=1}^{t} a_{2i-1}(1-\delta_i) + \sum_{i=1}^{t} a_{2i}\delta_i = \sum_{i=1}^{t} a_{2i-1}\delta_i + \sum_{i=1}^{t} a_{2i}(1-\delta_i)$

Consider the following case of SRRO:

$$n_o = 8t, \quad n_R = 2t,$$

For original jobs:

$$p_i = \begin{cases} 2B^{\lceil i/2 \rceil} + 2a_i & i = 1, \cdots, 2t \\ 2B^t & i = 2t+1, \cdots, 8t \end{cases}$$

$$r_i = \begin{cases} \sum_{j=0}^{i-1}(2B^{\lceil j/2 \rceil} + 2a_j) + \sum_{j=8t}^{i+8t-1} \\ (B^{\lceil j-8t/2 \rceil} + a_{j-8t}) \quad i = 1, \cdots, 2t \\ (i-1)B^t + \sum_{j=1}^{t}(24(t-j)+15)B^j + \sum_{j=1}^{2t} \\ (12t-6j)a_j + 9A \quad i = 2t+1, \cdots, 8t \end{cases}$$

For rework jobs:

$p_i = B^{\lceil i-8t/2 \rceil} + a_{i-8t}$, for $i = 8t+1, \cdots, 10t$;

$B = tA^3$; $K = 0$;

$g(v) = \sum_{i=1}^{8t} w_i(v)$, for $i = 1, \cdots, 8t$, $v = \{j_1, j_2, \cdots, j_{8t}\}$;

$D = \sum_{j=1}^{t}(60t-24j+11)B^j + (16t^2-3t)B^t + \sum_{j=1}^{2t}(30t-9j)a_j + 5A$; $\sum_{j=1}^{10t} w_j(\sigma) \le D$.

It is proved that there exists a feasible rescheduling for this case of SRRO with $\sum_{j=1}^{10t} w_j(\sigma) \le D$ if and only if there exists a solution to the Even-Odd Partition Problem.

(Sufficiency)

$$r_1 = 0; \quad p_1 = 2B + 2a_1, \; r_2 = 2B + 2a_1 + B + a_1;$$

$$p_2 = 2B + 2a_2, \cdots, \quad r_{2t} = \sum_{j=0}^{2t-1}(2B^{\lceil j/2 \rceil} + 2a_j) + \sum_{j=8t}^{10t-1}(B^{\lceil (j-8t)/2 \rceil} + a_{j-8t}); \quad p_{2t} = 2B^t + 2a_{2t},$$

$$r_{2t+1} = \sum_{j=1}^{t}(24(t-j)+15)B^j + \sum_{j=1}^{2t}(12t-6j)a_j + 9A;$$

$$p_{2t+1} = 2B^t, \cdots, \quad r_{8t} = (8t-1)B^t + \sum_{j=1}^{t}(24(t-j)+15) \times B^j + \sum_{j=1}^{2t}(12t-6j)a_j + 9A; \quad p_{8t} = 2B^t,$$

According to a known optimal rescheduling, all original jobs are processed at their release times immediately. Hence, $2t$ idle times of the machine come into being, namely,

$$[2B + 2a_1, 2B + 2a_1 + B + a_1], \ldots, [2B^t + 2a_{2t}, \sum_{j=1}^{t}(24(t-j)+15)B^j + \sum_{j=1}^{2t}(12t-6j)a_j + 9A].$$

The processing times of rework jobs with identical release time of zero are $p_{8t+1} = B + a_1, p_{8t+2} = B + a_2, \cdots, p_{10t} = B^t + a_{2t}$ respectively. In the formed rescheduling, the starting time of any original job remains unchanged and each original job is scheduled in the odd position, while $2t$ rework jobs are processing in idle times of each original job and scheduled in the even position. The total waiting time of jobs in the first four positions is

$$\begin{aligned} &(2B + 2a_1) + (3B + 3a_1) + (5B + 3a_1 + 2a_2) \\ &\quad + (6B + 3a_1 + 3a_2) \\ &= 16B + 11a_1 + 5a_2 \\ &= 16B + 21a_1/2 + 11a_2/2 + (a_1 - a_2)/2 \end{aligned} \tag{11}$$

Similarly, the total waiting times for all jobs $\sum_{i=1}^{10t} w_i(\sigma)$ is

$$\sum_{j=1}^{t}(60t-24j+11)B^j + (16t^2-3t)B^t + \sum_{j=1}^{2t}(30t-9j+5/2)a_j + \sum_{i\in S_1} a_i/2 - \sum_{i\in S_2} a_i/2 \tag{12}$$

Since there exists a solution to the Even-Odd Partition Problem, $\sum_{i\in S_1} a_i = \sum_{i\in S_2} a_i = A$, *i.e.*, $\sum_{j=1}^{10t} w_j(\sigma) = D$.

(Necessity) Consider a feasible rescheduling with $k = 0$ and $\sum_{j=1}^{10t} w_j(\sigma) = D$. In the rescheduling, each original job is scheduled in an odd position and its starting time remains unchanged, which inserts all rework jobs into machine idle times or after the last scheduled original job. There exists $2t$ idle times on the machine with respect to the given instance. If a rework job is arranged behind the last scheduled original job, then a computation similar to (10) and (11) shows that $\sum_{j=1}^{10t} w_j(\sigma)$ must be larger than D. Thus, the rework jobs are processing in all idle times of the machine or at the even positions of rescheduling with no idle time of machine. According to the computation,

$$\begin{aligned} \sum_{j=1}^{10t} w_j(\sigma) &= \sum_{j=1}^{t}(60t-24j+11)B^j \\ &\quad + (16t^2-3t)B^t + \sum_{j=1}^{2t}(30t-9j)a_j \\ &\quad + 5/2\sum_{j=1}^{2t} a_j + \sum_{i\in S_1} a_i/2 \\ &\quad - \sum_{i\in S_2} a_i/2 \\ &= D \\ &= \sum_{j=1}^{t}(60t-24j+11)B^j \\ &\quad + (16t^2-3t)B^t + \sum_{j=1}^{2t}(30t-9j)a_j + 5A \end{aligned}$$

$\sum_{i\in S_1}(a_i) = \sum_{i\in S_2}(a_i) = A$ is valid; thus, the Even-Odd Partition Problem has a solution.

IV. PROPERTIES OF THE OPTIMAL SOLUTION

There are six properties of the optimal solution proposed and proved in this section for the SRRO. First, a classical scheduling rule is discussed, which is better to analyze the model.

SPT Rule Known as the Shortest Processing Time, the jobs are sequenced in non-decreasing order of their processing times. Smith [19] shows that the SPT rule provides an optimal schedule for the classical problem $1\|\sum C_i$.

In a schedule ω (perhaps includes rework jobs), the maximum allowable delay time of job $[i]$ is defined as $\overline{\Delta}_{[i]}(\omega) =$

$min\{\Delta_{[i]}, I_{i+1} + \overline{\Delta}_{[i+1]}(\omega)\}$ where $\Delta_{[j]} = K - \{s_{[j]}(v) - r_{[j]}\}, [j] \in J_o$ and for $\Delta_{[j]} = \infty, [j] \in J_R$, $I_{[i+1]}$ indicates the machine idle time before the job $[i+1]$. Then a property is derived as follows.

Property 1: A schedule ω' is feasible if and only if $p_i \leq I_{[i]} + \overline{\Delta}_{[i]}(\omega)$ for some $j \in J_R$ and $[i] \in J$, which inserts job j into ω immediately before the job $[i]$.

Proof (Sufficiency): In an active schedule ω, $p_i \leq I_{[i]} + \overline{\Delta}_{[i]}(\omega)$ for some $j \in J_R$ and $[i] \in J$ implies that for the job $[i]$ and all jobs after it, the sum of the maximum allowable delay time and the machine idle time ahead of the job is not less than p_j, for all $j \in J_R$; thus, the job j can be inserted into ω immediately before the job i, and the obtained schedule ω' is a feasible solution.

(Necessity) A schedule ω' is feasible, which inserts the job j into the schedule immediately before the job $[i]$ is feasible and, as there is no constraint for waiting time of rework jobs, the constraint of $w_i(\omega') \leq K$ is only required for all $l \in J_o$. That is, in schedule ω for every original job after the job $[i]$, the sum of maximal delayable time and machine idle time ahead of the original jobs must be not less than p_i, for all $j \in J_R$; thus, $p_i \leq I_{[i]} + \overline{\Delta}_{[i]}(\omega)$ for some $j \in J_R$ and $[i] \in J$.

Property 2: In an optimal rescheduling of SRRO, each subsequence of consecutive rework jobs must be arranged in non-descending order of processing times.

Proof: Proof by contradiction. In an optimal rescheduling σ^*, a new feasible rescheduling σ' can be obtained by assuming that rework jobs $[a+1]$ and $[a]$ satisfying $p_{[a+1]} < p_{[a]}$ exist in a consecutive subsequence of rework jobs while maintaining the original sequence of jobs by interchanging $[a+1]$ and $[a]$. The waiting time of rework job $[a+1]$ decreases by $p_{[a]}$, whereas the waiting time of rework job $[a]$ is increased by $p_{[a+1]}$. Thus, the objective value of σ' is $\sum_{j=1}^{n} w_j(\sigma') = \sum_{j=1}^{n} w_j(\sigma^*) + p_{[a+1]} - p_{[a]}$. $\sum_{j=1}^{n} w_j(\sigma') \leq \sum_{j=1}^{n} w_j(\sigma^*)$ because $p_{[a+1]} < p_{[a]}$, i.e., $p_{[a+1]} - p_{[a]} < 0$, which contradicts σ^* with being an optimal rescheduling according to the supposition.

Property 3: In any optimal rescheduling, every machine idle time must immediately be followed by an original job, and every rework job followed an idle time must have a processing time larger than that idle time,i.e.,$0 \leq r_{[i]} - C_{[i-1]} < p_j, i+1 \leq j \leq n, [j] \in J_R$.

Proof: There cannot be any idle time immediately preceding a rework job, since rework jobs have a release time of 0, and shifting the job left to use that idle time would reduce the total waiting time. In addition, it is reasonable to suppose that there is a rework job with processing time less than the idle time before it in an optimal rescheduling σ^*; then a new feasible rescheduling σ' would be obtained by inserting this rework job in this idle time. In the rescheduling σ', the waiting time of this rework job is reduced and the waiting times of other jobs are unchanged; therefore, $\sum_{j=1}^{n} w_j(\sigma') < \sum_{j=1}^{n} w_j(\sigma^*)$, which contradicts with the optimality of σ^*.

Property 4: For SRRO with no idle time in the original schedule, there exists an optimal rescheduling in which the jobs of J_R are sequenced in SPT order, and there is no idle time between jobs.

Proof: The policy will always maintain the sequence for original jobs in this study, *i.e.*, the jobs of J_o are sequenced as them in v. We only analyze the order for the jobs in J_R. Supposed that there is an optimal scheduling σ^* in which the jobs of J_R are ordered in non-SPT order. Let $j \in J_R$ be the job with the smallest processing time that is the scheduled for later in J_R in σ^* than in SPT order of J_R, and let $j \in J_R$ be the last job of J_R that precedes the job i in σ^* for $p_j > p_i$. A new schedule σ' can be obtained from σ^* by interchanging i and j, so that $w_j(\sigma') = w_i(\sigma^*) - p_j + p_i \leq w_i(\sigma^*)$, and any job between j and i is delayed by $p_j - p_i$ units of time in earlier σ' than in σ^*. A straightforward computation shows that the total waiting time does not increase due to the interchange. If the position of $j \in J_R$ in σ' is not less than that in SPT order of J_R (as is the case with $i \in J_R$), then the waiting times of original jobs between j and i are not more than their positions in σ^*. Thus, σ' is feasible and better than σ^*. A finite number of repetitions of this argument may ensure that an optimal schedule solution exists where the jobs of J_R are ordered in SPT. The same sequence of jobs of J_o in v, and there is no idle time in the optimal schedule, feasibility is maintained and the total waiting time is reduced through removing the idle times.

In most practical production, usually the maximum processing time for rework jobs $p^R_{\max}$ is not more than the minimum processing time of original $p^o_{\min}$ of jobs, i.e., $p^R_{\max} \leq p^o_{\min}$. As a special case of SRRO with $p^R_{\max} \leq p^o_{\min}$, the following two properties are proposed.

Property 5: In each optimal rescheduling, if an original job is scheduled followed by a rework job, the processing time of this rework job must be more than the sum of the maximum delayable time and the idle time ahead of the original job.

Proof: By contradiction. Assume that in an optimal rescheduling σ^*, an original job b is processed and is followed by a rework job d, where $p_d \leq \overline{\Delta}_b(\sigma^*) + I_b$. Another feasible rescheduling σ' satisfying $w_b(\sigma') \leq K$ is obtained by interchanging the positions of b and d, where $\sum_{j=1}^{n} w_j(\sigma') = \sum_{j=1}^{n} w_j(\sigma^*) + p_d - p_b$. Because it is assumed that $p^R_{\max} \leq p^o_{\min}$, $\sum_{j=1}^{n} w_j(\sigma') \leq \sum_{j=1}^{n} w_j(\sigma^*)$; thus, it contradicts with the optimality of σ^*.

According to Property 5, each rework job in the optimal rescheduling cannot be scheduled earlier if the sequence of rework jobs remains unchanged. Therefore, the following property can be derived straight for work.

Property 6: If the sequence of rework jobs has been determined, then the optimal rescheduling of SRRO can be obtained by as early as possible inserting the rework jobs into the original schedule in order to that order sequence.

V. PSEUDO-POLYNOMIAL TIME ALGORITHM FOR A SPECIAL CASE OF SRRO

A special case of SRRO subject to the following two conditions is discussed in this section.

1) In schedule v' there exists only one machine idle time I, which is obtained by delaying the starting times of

original jobs as possible as late in v, i.e., $\overline{\Delta}_{[i]}(v) - I_{[i+1]} \geq \overline{\Delta}_{[n_o]}, i = 1, \cdots, n_o - 1$.

2) The maximum processing time for the rework jobs $p^R_{\max}$ is not larger than the minimum processing time of the original jobs $p^o_{\min}$, i.e., $p^R_{\max} \leq p^o_{\min}$.

An SRRO subject to $\overline{\Delta}_{[i]}(v) - I_{[i+1]} \geq \overline{\Delta}_{[n_o]}, i = 1, \cdots, n_o - 1$ means that there is only one idle time I of machine in schedule v' that is got by delaying the starting times of original jobs as late as possible in v.

Properties 5 and 6 can be used to determine which the rework jobs should be scheduled in this idle time I or after the last the original job in v' by SPT rule, respectively. In fact, for the n_R jobs, there are $n_R!$ alternative scheduling solutions, making the special case of SRRO NP-hard in the ordinary sense. The dynamic programming algorithm is designed as follows to implement the optimal merging for the special case, where t denotes time.

Algorithm A:

Input:

p_i for $i \in J_o$, p_j for $j \in J_R$, K, and v.

Preprocessing:

Compute $I = s_{[1]}(v) + \overline{\Delta}_{[1]}(v)$.

State variable:

$u_t = \{J_R, I, \sigma\}$ includes a set of candidate rework jobs J_R,the remaining idle time I,and the partial rescheduling σ.

Decision variable:

x_i means that subject to the constraint of maximum waiting times for the original jobs, the rework job j is inserted into the block (where a block is defined as a contiguous set of jobs with no idle time between jobs) of rework jobs before the first original job or after the last original job in σ by SPT rule.

State transition:

$$\begin{cases} J_R = J_R \setminus j, & I = I - p_j, \text{update } \sigma \text{ after job } j, \text{for } I \geq p_j \\ J_R = J_R \setminus j, & I = I, \text{update } \sigma \text{ after job } j, \text{for } I_r < p_i \end{cases}$$

Boundary condition $f(u_0)$:

$\sigma = v$, the total waiting time of the original jobs in schedule v.

Value function $f(u_t)$:

The total waiting time of all jobs in σ after job j is scheduled.

Recurrence equation:

$f(u_i) = \min\{f(u_{i-1}) + \delta(u_{i-1}, x_{i-1})\}$, where the total waiting times of all jobs increases $\delta(u_{i-1}, x_{i-1})\}$ units of time in σ from state u_{i-1} to u_i due to the determinatioin of x_{i-1}.

Time complexity of algorithm A:

The time complexity of algorithm A is $O(n_R^2 \log n_R I)$. Algorithm A is in fact a pseudo-polynomial time algorithm because the computing time depends on the length of the idle time I.

Based on the dynamic programming algorithm, the following corollary can be obtained

Corollary 1: Algorithm A can find an optimal rescheduling for a special SRRO subject to $\overline{\Delta}_{[i]}(v) - I_{[i+1]} \geq \overline{\Delta}_{[n_o]}, i = 1, \cdots, n_o - 1$ in pseudo-polynomial time.

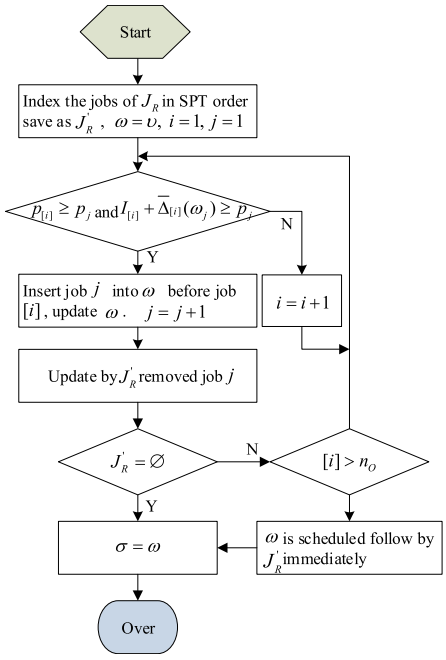

FIGURE 2. The flow chart of Algorithm B.

VI. HEURISTIC ALGORITHM FOR SRRO

In this section, a heuristic algorithm is designed for SRRO of properties 1-3, and it is proved to solve five special cases optimally. The procedure of the heuristic algorithm, denoted by algorithm B, is designed as follows.

Algorithm B:

Fig. 2 shows the flowchart of algorithm B.

Given K,v,p_i and p_j for $i \in J_o$, and $j \in J_R$.

Step 1. ω is the iterative variable with initial value equal to υ. Sorting the jobs of J_R by Shortest Processing Time (SPT) rule and saving them in J'_R. $I_{[i]}$ denotes the idle time before job $[i]$;

Step 2. Initialize $i = 1$ and $j = 1$.

Step 3. In ω, compute $\overline{\Delta}_{[i]}(\omega)$. if $p_{[i]} \geq p_j$ and $I_{[i]} + \overline{\Delta}_{[i]}(\omega_j) \geq p_j$; insert the job j into ω before the job $[i]$ and update $\omega, j = j + 1$, the job j is removed from J'_R, otherwise, $i = i + 1$ and go to Step 3.

Step 4. If $J'_R = \phi$, then ω is the solution of the rescheduling. If $J'_R \neq \phi$ and $[i] > n_o$, ω is scheduled followed by J'_R immediately and then the rescheduling solution σ is obtained.

Time complexity of algorithm B:

The sorting procedure needs $O(n_R log n_R)$ time and the looping steps require $O(2nn_o)$; thus, the time complexity of algorithm B is $O(2nn_o + n_R log n_R)$.

Corollary 2: Algorithm B can obtain an optimal rescheduling for the SRRO with identical processing time of rework jobs and $p^R_{max} \leq p^o_{min}$.

Proof: Rework jobs have identical processing time and their release times are zero; thus, rescheduling is independent of orders. Algorithm B inserts the rework jobs into the original schedule as much as possible before the original jobs and forms a feasible active schedule. According to Property 6, an optimal rescheduling is obtained for SRRO in case the processing times of rework jobs are identical.

Corollary 3: Algorithm B can find an optimal rescheduling for a special case of SRRO where $p^R_{max} \le p^o_{min}$,$\overline{\Delta}_{[i]}(\upsilon) - I_{[i+1]} \ge \overline{\Delta}_{[n_o]}, i = 1, \ldots, n_o - 1$ and $\sum_{j=n_R+1}^{n} p_j \le I + \varepsilon, \varepsilon \le max\{p_j | j \in J_R\}$.

Proof: The SRRO with $p^R_{max} \le p^o_{min}$ and $\overline{\Delta}_{[i]}(\upsilon) - I_{[i+1]} \ge \overline{\Delta}_{[n_o]} i = 1, \ldots, n_o - 1$ is the case in Section 5, where there exists only one machine idle time I in ν', which is obtained by delaying the starting times of original jobs as late as possible in ν. Based on it, two cases will be discussed if $\sum_{j=n_R+1}^{n} p_j \le I + \varepsilon, \varepsilon \le max\{p_j | j \in J_R\}$ as follows:

Case 1. $\sum_{j=n_R+1}^{n} p_j \le I$: a rescheduling can be obtained by algorithm B, where all rework jobs can be arranged in I in SPT order.

Case 2. $I < \sum_{j=n_R+1}^{n} p_j \le I + \varepsilon, \varepsilon \le max\{p_j | j \in J_R\}$: a rescheduling can be obtained by algorithm B, where only the rework job with the maximum processing time is arranged after the last original job while others are scheduled in I by SPT rule.

The rescheduling for each of the cases is proved to be optimal by a simple job interchanging argument.

The following theorem is a discrimination condition of the optimal solution of algorithm B for a general SRRO.

Theorem 2: A rescheduling obtained by algorithm B is optimal for SRRO if there exists no idle time in the rescheduling before the latest rework job.

Proof: Algorithm B produces a rescheduling σ with no idle time before the latest rework job. This theorem is proved by the following contradictions.

Supposed that an optimal rescheduling σ' can be obtained for SRRO by interchanging job $[d]$ and job $[d+1]$ in σ.

Case 1. There are rework job $[d]$ and rework job $[d+1]$. Then the objective values in σ' and σ respectively are $\sum w_i(\sigma) = \sum w_{[j]}(\sigma) + w_{[d]} + w_{[d+1]} = \sum w_{[j]}(\sigma) + 2C_{[d-1]}(\sigma) + p_{[d]}, i \in J, j \in J \setminus \{d, d+1\}$ and $\sum_i (\sigma') = w_{[j]}(\sigma) + w_{[d+1]}(\sigma') + w_{[d]}(\sigma') = \sum w_{[j]}(\sigma) + 2C_{[d-1]}(\sigma) + p_{[d+1]}, i \in J, j \in J \setminus \{d, d+1\}$.

According to algorithm B, the rework jobs are arranged by SPT rule, $p_{[d]} \le p_{[d+1]}$; thus, $\sum_{j \in J} w_i(\sigma) \le \sum_{i \in J} w_i(\sigma')$; then it is in contradiction with the optimal rescheduling σ'.

Case 2. There are original job $[d]$ and original job $[d+1]$. In this study, original jobs always maintain their sequence; thus, an infeasible rescheduling σ' is obtained.

Case 3. There are rework job $[d]$ and rework job $[d+1]$. It means that $p_{[d]} \le p_{[d+1]}$ and $p_{[d]} \le I_{[d+1]} + K - w_{[d+1]}$. Hence, $\sum w_{[i]}(\sigma') \ge \sum w_{[j]}(\sigma) + 2C_{[d-1]}(\sigma) + p_{[d+1]}, i \in J, j \in J \setminus \{d, d+1\}$. Thus, $\sum w_i(\sigma) \le \sum w_i(\sigma'), i \in J$; then it is in contradiction with the optimal rescheduling σ'.

Case 4. There are rework job $[d+1]$ and an original job. According to algorithm A in the rescheduling σ', $w_{[d]}(\sigma') = w_{[d]}(\sigma) + p_{[d+1]} > K$; thus, σ' is an infeasible rescheduling.

TABLE 1. Five special cases of SRRO.

Corollaries	Features of the cases
Corollary 2	Identical processing time for rework jobs and $p^R_{max} \le p^o_{min}$
Corollary 3	$\overline{\Delta}_{[i]}(\nu) - I_{[i+1]}, i = 1, \ldots, n_o - 1$ and $\sum_{j=n_R+1}^{n} p_j \le I + \varepsilon, \varepsilon \le max\{p_j | j \in J_R\}$
Corollary 4	No idle time in original schedule
Corollary 5	$\Delta_j \ge max\{p_i | i \in J_R\}$,for $j \in J_o$
Corollary 6	$I_{[j]} + \Delta_{[j]}(\nu) < min\{p_i | i \in J_R\}$ for $[j] \in J_o$

Hence, an optimal solution σ is obtained with no idle time before the latest rework job.

According to Theorem 2, the following two corollaries explain that the two cases of SRRO can be solved optimally.

Corollary 4: Algorithm B can find an optimal rescheduling for the SRRO with no idle time within the original schedule in polynomial time.

Proof: If there is no idle time within the original schedule, then the rescheduling σ obtained by algorithm B should have no idle time within it. Thus, according to Theorem 2, σ is optimal.

Corollary 5: Algorithm B can get an optimal rescheduling for SRRO in polynomial time if $\Delta_j \ge max\{p_i | i \in J_R\}$, for $j \in J_o$.

Proof: $\Delta_j \ge max\{p_i | i \in J_R\}$, for $j \in J_o$ means that all of the starting times of original jobs can be delayed and the delayable time of each job is not less than the maximum processing time of a rework job. Therefore, regardless of whether the idle times exists ahead of the original jobs, algorithm B can obtain a rescheduling with no idle time before the last rework job. By Theorem 2, the obtained rescheduling is an optimal solution for the case.

Corollary 6: Algorithm B can find an optimal rescheduling for SRRO in polynomial time, if $I_{[j]} + \Delta_{[j]}(v) < min\{p_i | i \in J_R\}$ for $[j] \in J_o$.

Proof: $I_{[j]} + \Delta_{[j]}(v) < min\{p_i | i \in J_R\}$ for $[j] \in J_o$ means that no rework job can be arranged by inserting into the original schedule, and all rework jobs can be arranged after the last original job only. It is proved by a simple job interchanging argument.

Table 1 summarizes that algorithm B can solve the five special cases of SRRO optimally in polynomial time.

VII. GENETIC ALGORITHM WITH ADAPTIVE LOCAL SEARCH SCHEME

To further solve general cases of SRRO, a GA with adaptive local search scheme (GAA) is developed in this study to enhance the strong global searching ability. The adaptive local search algorithm is designed with three kinds of local search & moves, considering the characteristics of SRRO. The local search & moves involve inversion, transfer and swap.

A. CHROMOSOMAL REPRESENTATION

A valid scheduling sequence is expressed as a chromosome. For example, the sequence of {1 8 2 9 3 6 4 7 5} is a valid

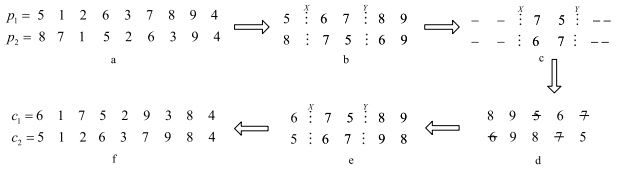

FIGURE 3. Order crossover operation schematic diagram.

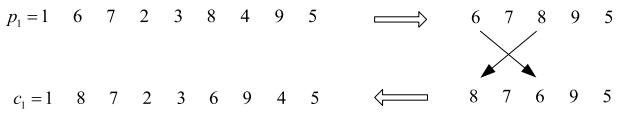

FIGURE 4. The process of mutation with local move for an individual.

chromosome, where, $J_o = \{1, 2, 3, 4\}$ is the set of original jobs, and $J_R = \{5, 6, 7, 8, 9\}$ is the set of rework jobs.

B. GENERATION OF INITIAL POPULATION

Algorithm B is applied to generate a set of rescheduling sequences as the initial population.

Order Crossover Local Move: For the rework jobs, a double-point order crossover with local move is considered in GAA. Fig. 3 shows an schematic example of the operation of order crossover. The detailed steps are as follows:

- a Two parents $p_1 = \{5\ 1\ 2\ 6\ 3\ 7\ 8\ 9\ 4\}$ and $p_2 = \{8\ 7\ 1\ 5\ 2\ 6\ 3\ 9\ 4\}$ are given.
- b The scheduling sequences of rework jobs in parents are shown and two points of crossover are selected.
- c The segments of genes between two points of crossover are exchanged, and then the partial genes of child are determined.
- d Based on b- from the first gene on the right of the second point of crossover, all genes are sequenced in order, and the genes repeated within c are deleted
- e Based on the partial genes of child in c, from the first position on the left of the second point of crossover, the remaining rework jobs in d are determined one by one.
- f All rework jobs are arranged into the original schedule as earlier as possible in the order of e according to properties 1 and 6. Two children $c_1 = \{6\ 1\ 7\ 5\ 2\ 9\ 3\ 8\ 4\}$ and $c_2 = \{5\ 1\ 2\ 6\ 3\ 7\ 9\ 8\ 4\}$ are finally generated.

C. MUTATION WITH LOCAL MOVE

Given a parent rescheduling sequence the partial scheduling of rework jobs in parent is known and two rework jobs are selected randomly. A new scheduling sequence of rework jobs by interchanging two positions of rework jobs is selected. A child is obtained with rework jobs inserted into the original schedule as earlier as possible in the order of the new scheduling sequence after the interchanging. Fig. 4 explains an example of the process of mutation with local movement for an individual.

D. SELECTION OF CHROMOSOMES FOR LOCAL MOVE

A chromosome i is selected randomly with probability P_i. Selection of local move is carried out by a roulette wheel.

E. ADAPTIVE LOCAL SEARCH

The adaptive local search combines inversion, transfer and swapping of three local search & moves and adopts an adaptive learning mechanism. All rework jobs are divided into some blocks of rework jobs in a rescheduling sequence. According to the characteristic of SRRO, the block structure is used as neighborhood structure. A new rescheduling sequence is generated by adjusting the sequence of rework jobs between two blocks of rework jobs selected randomly. Then the rescheduling sequence with a better solution is obtained by the local search.

An initial rescheduling sequence {7 8 1 2 6 3 4 9 5} is known. Then the rework jobs may be divided into three blocks {7 8 | 6 | 9 5}. With respect to the example, the three methods of local search are explained as follows.

1) LOCAL SEARCH BASED ON INVERSION

Two blocks of rework jobs are selected randomly, for example, Block $-1 = (7\ 8)$ and Block $-3 = (9\ 5)$. Two positions u and v are generated randomly in terms of the two selected blocks, where u and v are no less than 1 and no more than the number of rework jobs in each block. Suppose $u = 1$ and $v = 1$; then, the first position in Block-1 and the first position in Block-3 are determined and inversion is implemented for rework jobs between the two positions 7 and 9 (involving two rework jobs on the two positions). A new sequence of rework jobs {96875} is obtained, hence algorithm B for $J'_R = \{9\ 6\ 8\ 7\ 5\}$ is performed. A new rescheduling sequence is finally generated.

2) LOCAL SEARCH BASED ON TRANSFER

Similar to the above method, two blocks of rework jobs u and v are generated. Suppose Block- $1 = (7\ 8)$ and Block-$2 = (6)$ are selected and $u = 1$ and $v = 1$. The rework job on the $v = 1$ position in Block-2 is scheduled ahead of the rework jobs on the $u = 1$ position in Block-1. The rework jobs scheduled after the rework jobs on the $u = 1$ position in Block-1 are shifted right. A new subsequence of rework jobs {6 7 8} is obtained by transferring. A new sequence of rework jobs {6 7 8 9 5} is obtained, and then a new rescheduling sequence is generated by algorithm B for $J'_R = \{6\ 7\ 8\ 9\ 5\}$.

3) LOCAL SEARCH BASED ON SWAP

Two blocks of rework jobs,u and v are also generated by the same method as the above case. Suppose Block-$2 = (6)$ and Block-$3 = (9\ 5)$ are selected and $u = 1$ and $v = 2$. The rework job 6 on the $u = 1$ position in Block-2 and the rework job 5 on the $v = 2$ position in Block-3 are interchanged. The sequence of rework jobs {78596} is generated, hence a new rescheduling sequence is obtained by algorithm B for $J'_R = \{7\ 8\ 5\ 9\ 6\}$.

4) ADAPTIVE LOCAL SEARCH

Let $\lambda_{move}(t)$ denote the sum of all λ values obtained through iterations of t. Based on the inversion, transfer and swap, three local search strategies are used in GAA. p_{inve}, p_{tran},and p_{swap} indicate the probabilities adopted for these three strategies

respectively, and $p_{inve} + p_{tran} + p_{swap} = 1$. The initial values of p_{inve}, p_{tran} and p_{swap} are identical, which means that the probabilities adopted for the three strategies are equal. The strategy with a better performance should be used more than the other strategies in the algorithm. Therefore, p_{inve}, p_{tran}, and p_{swap} are determined by adaptive learning in GAA. λ indicates the degree of solution improvement after a local search, and its mathematical expression is as follows:

$$\lambda = \frac{f_{prior} - f_{after}}{f_{prior}}$$

where f_{prior} is the value of solution for the best individual in the population and f_{after} is the value of solution after local search for the best individual in the population. λ is computed only if $f_{after} < f_{prior}$. Then p_{inve}, p_{tran} and p_{swap} are recalculated. Figure 5 shows the flowchart for the adaptive local search.

$\lambda_{inv}(t)$, $\lambda_{tra}(t)$, and $\lambda_{swap}(t)$ indicate the levels of solution improvement after the inversion, transfer and swap strategies of local search in t generation. Hence, the mathematical expressions of the probabilities adopted for the three strategies are as follows:

$$\begin{aligned}
p_{inve}(t+1) &= p_{inve}(t) + \tau \cdot \lambda_{inv}(t) \\
p_{tran}(t+1) &= p_{tran}(t) + \tau \cdot \lambda_{tra}(t) \\
p_{swap}(t+1) &= p_{swap}(t) + \tau \cdot \lambda_{swa}(t) \\
p_{inve}(t+1) &= \frac{p_{inve}(t+1)}{p_{inve}(t+1) + p_{tran}(t+1) + p_{swap}(t+1)} \\
p_{tran}(t+1) &= \frac{p_{tran}(t+1)}{p_{inve}(t+1) + p_{tran}(t+1) + p_{swap}(t+1)} \\
p_{swap}(t+1) &= 1 - p_{inve}(t) - p_{tran}(t+1)
\end{aligned}$$

where τ is defined as the relative influence of the levels of solution improvement. Therefore, not only the three design strategies can improve the quality of solutions by mutual cooperation but also increase the probabilities adopted by mutual competition.

Figure 5 shows the flowchart for the adaptive local search.

F. GAA

With the above design considerations, GAA is outlined below and Fig. 6 shows its flowchart.

1) INITIALIZATION

Set the population size *pop_size*, probability of crossover p_c, probability of mutation p_m, probability of replacement p_r, genetic iterations t, number of local search, and other parameters. Let $t = 0$.

2) GENERATION OF INITIAL POPULATION GUIDED BY THE RULE

The initial population is generated by algorithm B.

3) CHECK FOR OPTIMALITY

If an individual in the population satisfies that all rework jobs are arranged ahead of the first original job or after the last original job or there is no idle time of machine in the rescheduling sequence, then this individual is optimal according to Corollary 6 and Theorem 2.

Otherwise, enter the following steps.

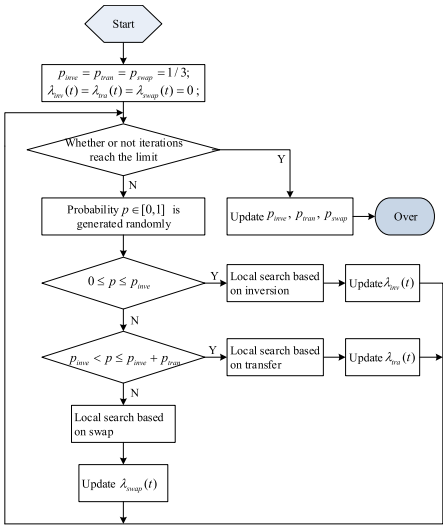

FIGURE 5. Flow chart for adaptive local search.

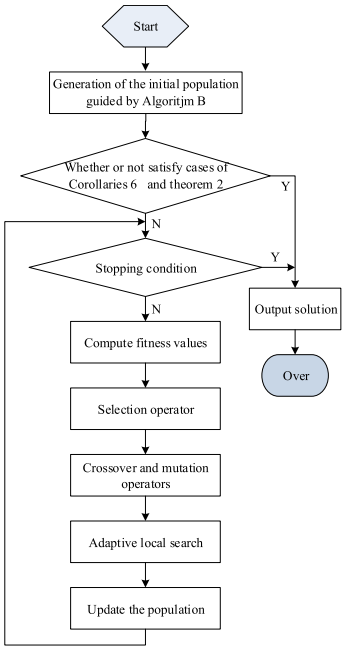

FIGURE 6. Flow chart for genetic algorithm with adaptive local search.

4) EVOLUTION OF THE POPULATION

Selection: The function of fitness values is computed. The parents are selected by roulette wheel, and the parents will carry out genetic arithmetic.

Crossover: Each pair of chromosomes in parents carries out order crossover with local move according to the probability of crossover p_c.

Mutation: Each chromosome after crossover carries out mutation with local move by the probability of mutation p_m.

Update the population: The interchange population and the progeny population are sorted in non-descending order

TABLE 2. Four relative factors for the problem generator.

Factor	Description	Level	Implication of level
n	Problem size, the sum of number for all jobs	20 40 80	Problem size from smaller to larger
n_R	Set of rework jobs, total number of rework jobs	$0.25n$ $0.50n$ $0.75n$	Number of rework jobs from fewer to larger
FIT	Frequency of v idle time within	0.1 0.3 0.5 0.7	The number of idle times in original schedule from smaller to larger

of their objective values. The first $pop\_size * p_r$ individuals are selected to replace the individuals in the parent population in the bottom $pop\_size * p_r$; then the parent population of the next generation is obtain.

Local search: The adaptive local search is implemented by the method explained in Section 7.6.

5) STOPPING RULE

If the number of iterations is equal to the specified number, the individual with the maximum fitness value is outputted and the computation is completed. Otherwise, go back to the procedure of ***Evolution of the population.***

VIII. COMPUTATIONAL EXPERIMENTS

The experimental study is aimed at evaluating the performance of MILP, algorithm B, GA and GAA, and the difference between GA and GAA with adaptive local search. MILP was solved using CPLEX, while algorithm B, GA, and GAA were coded using C++. All experiments were worked on a computer with 2GB RAM, AMD Athlon(tm) II X4 650 CPU with 3.19 GHz.

A. PROBLEM INSTANCE GENERATOR

In the literature, standard problem instances of SRRO cannot be found. In this paper, a problem generator is developed as follows.

To have a more general problem generator, four parameters were considered. Table 2 describes the parameter details of SRRO and the scales or levels considered in this study.

The problem size $n \in \{20, 40, 80\}$ and the number of rework jobs $n_R \in \{0.25n, 0.50n, 0.75n\}$ are initially determined. Then, the number of original jobs $n_O = n - n_R$ is determined. The processing times of n_O original jobs and n_R rework jobs are integers randomly generated between 1 and 100. The original jobs are numbered from 1 to n_O in the order of their generations. The distribution of idle time (FIT) within the original schedule has four levels, i.e., 0.1, 0.3, 0.5, and 0.7. Then, the expected idle times within the original schedule is equal to $FIT \times n_O$; thus, $FIT \times n_O$ idle times are generated in the original schedule. For instance, $FIT = 0.1$, $n_O = 40$; then $0.1 \times 40 = 4$ idle times are generated in the original schedule. The lengths of the idle times follow the identical independent distribution the same as the processing times of the jobs. The generated idle times are randomly arranged in the original schedule. The release times of original jobs are considered one by one according to the order of original jobs and the inserted idle times. T indicates the current available time of machine, and its initial value is 0. The current available time of machine for the $i-th$ original job T_i is equal to the complete time of the $(i-1)-th$ original job plus the length of idle time ahead of the ith original job, i.e., $T_i = C_{i-1} + I_i$. Then, the release time of the $i-th$ original job is an integer generated between T_i and $T_i \times (i+1)/(i+3)$.

To ensure the feasibility of the original schedule, K should not be smaller than the maximum waiting time of original jobs in the original schedule. If K is extremely large, it will result in that all rework jobs can be arranged ahead of the first original job; then computational experiments are trivial. Thus, the upper limit of waiting time of original jobs K is equal to the maximum waiting time of original jobs in the generated original schedule.

In this study, there are $3 \times 3 \times 2 \times 4 = 72$ configurations for different problem cases for each algorithm. For every possible configuration, 10 problem instances are generated. In each algorithm, 720 problem instances are tested.

B. GAA CONFIGURATION

A better configuration of parameters for GAA is determined in this subsection. GAA refers to three probability parameters: crossover p_c, mutation p_m, and replacement p_r. Through experiments and experience, the levels of parameters for GAA are chosen as $p_c \in \{0.4, 0.5, 0.6, 0.7\}$, $p_m \in \{0.1, 0.2, 0.3, 0.4\}$, and $p_r \in \{0.1, 0.3, 0.5\}$. The number of combinations for different levels of parameters for GAA is equal to $4 \times 4 \times 3 = 48$. However, an excessively large number of experiments may give rise to difficulty to achieve results, hence, the uniform design experimentation of mixed level proposed by Fang [20] is used to select representative candidate combinations of parameters for GAA testing. Uniform design experimentation only considers a uniform distribution of test points in the range to ensure the variety. Compared with the test points of orthogonal design, the test points of uniform design are more evenly distributed and are more representative. GAA develops the table for the uniform design of mixed level with 12 parameter combinations. The uniform design experimentation of mixed level is carried out using the DPS software of Jouppi [21]. Table 3 provides the detail of the 12 parameter combinations, where n_par indicates the number of parameter combinations, n_c, n_m, and n_r are the numbers of three levels of parameters selected respectively.

In the problem generation, given the problem size, the levels for the other three factors of the problem generator have $3 \times 2 \times 4 = 24$ combinations. Let $n = 40$; according to above design of the problem generator, 5 problem instances are randomly generated for each combination of factors by the problem generator. Therefore, there are 120 problem instances generated to test the performance for every candidate combination of parameters for GAA. Each of the problem instances run 30 times independently. For each of 12 candidate

TABLE 3. Parameter combinations of algorithms.

n_{par}	n_c	n_m	n_r	n_{par}	n_c	n_m	n_r
1	4	4	2	7	1	1	2
2	3	4	3	8	2	2	1
3	2	1	3	9	1	4	1
4	2	3	2	10	3	2	2
5	3	3	1	11	4	1	1
6	1	3	3	12	4	2	3

TABLE 4. Parameter combinations of GAA to solve the case.

n_{par}	n_{cmv}	p_{cmv}	n_{par}	n_{cmv}	p_{cmv}
1	3542	98.40%	7	3503	97.30%
2	3539	98.30%	8	3506	97.40%
3	3499	97.20%	9	3514	97.60%
4	3524	97.90%	10	3578	99.40%
5	3524	97.90%	11	3532	98.10%
6	3521	97.80%	12	3539	98.30%

combinations of parameters for GAA, 3600 experiments are carried out.

The size of population, iterations, and the number of local searches are 50, 30, and, 10 respectively. n_{cmv} indicates the number of occurrences for the incumbent minimum value of the objective function. The proportion of occurrences in incumbent minimum value of the objective function p_{cmv} is equal to n_{cmv} divided by the total number of experiments, i.e., 3600. p_{cmv} is used to evaluate candidate combinations of parameters for GAA, i.e., the larger the p_{cmv}, the better the combinations of parameter. Table 4 presents the results of GAA under the 12 candidate combinations of parameters. Obviously, the combination of parameter for $n_{par} = 10$ outperforms the others, i.e., $p_c = 0.6$, $p_m = 0.2$ and $p_r = 0.3$.

C. COMPUTATIONAL RESULTS

Based on the above experimental design, the 720 problem instances are generated in Subsection 8.1 by the problem instance generator and solved using MILP (CPLEX), heuristic (Algorithm B), GA, and GAA showed that MILP using CPLEX can obtain optimal solutions.

1) PERFORMANCE ANALYSIS OF FOUR METHODS

N_C and $P_C$indicate the number and proportion of optimal solutions obtained by the CPLEX software in tolerable time respectively. The tolerable time is set as less than 10,000 secs in the experiments. N_H and P_H are the number and proportion of optimal solutions obtained by the heuristic algorithm B. N_G and P_G denote the number and proportion of optimal solutions obtained by GA. N_G and P_G indicate the number and proportion of optimal solutions obtained by GAA. T_C, T_H, T_G and T_G indicate the average running time of CPLEX, heuristic algorithm, GA, and GAA. A_H, A_G and A_G are the average relative error of the heuristic algorithm, GA, and GAA respectively. S_H, S_G and S_G are the standard deviation of the heuristic algorithm, GA, and GAA respectively.

Table 5 presents a summary of the performance of four methods for all problem instances. It is shown that almost all problem instances are solved optimally by the MILP of SRRO with CPLEX in tolerable time. GAA shows an excellent performance in the experiments, i.e., both the quality and time of solving are satisfactory. Table 6ÍC8 summarize the performance of the four methods for the problem instances respectively.

Table 6 shows that the solving performances of four methods deteriorate with increasing problem size. The solving quality of GAA is better than the heuristic and GA with respect to each scale or level of the problems. In particular, when the MILP of SRRO is solved by the CPLEX software, the average running time is obviously longer for the problem with 80 jobs, where the longest running time is 8056.44 secs. Contrastly the solving times of the heuristic, GA, and GAA remain relatively stable.

In Table 7, the performances of the four methods are better with 0.25 proportion than of 0.50 and 0.75 proportion rework jobs. When the proportions of rework jobs are 0.5 and 0.75, the average running time is significantly longer when solving the MILP of SRRO with the CPLEX. The performances of the heuristic algorithm, GA, and GAA are relatively stable, although there is a slight decrease. The overall effect of GAA is better than those of the heuristic algorithm and GA.

Table 8 explains that in the three *FIT* levels of 0.3, 0.5, and 0.7, the number of idle times in the original schedule basically did not affect the solving quality of the four methods. The solving quality of the four methods is more outstanding with $FIT = 0.1$ compared to the other three levels. With regard to the running time, the effect of solving the MILP of SRRO with CPLEX with $FIT = 0.1$ is better than with other levels. However, the different levels of *FIT* have almost no effect on the performance of the heuristic algorithm, GA and GAA. The overall effect of GAA is more satisfactory than in the heuristic algorithm on each *FIT* levels.

2) STABILITY ANALYSIS OF GAA

The above experiment results show that GAA is suitable for solving SRROs of large scale. To test the stability of GAA with increasing problem scale, the following experiments are carried out.

In above problem instances with 40 jobs, we selected the following combination of factors (n_R, *FIT*) for the problem: (0.75,0.7) with the longest average running time, (0.25, 0.7) with the shortest average running time, (0.50, 0.1) with the best rate of better solution and the smallest average relative error, and (0.50, 0.1) with the worst rate of better solution and largest average relative error. The performance of GAA with the optimal parameter combination of algorithm is tested along with the increasing problem scale.

Fig. 7 shows the changes in average running time of GAA with different total number of jobs. The solid line and the dotted line indicate the results of the factors combinations for the problem with the longest average running time and with the shortest average time, respectively.

TABLE 5. Performance of the four methods.

Situation of optimal solution obtained								Average running time				Average relative error			Standard deviation		
N_C	$P_C(\%)$	N_H	$P_H(\%)$	N_G	$P_G(\%)$	N_G	$P_G(\%)$	T_C	T_H	T_G	T_G	A_H	A_G	A_G	S_H	S_G	S_G
712	98.89	341	47.89	450	63.20	600	84.27	215.21	0.62	4.20	5.91	0.005	0.0009	0.0001	0.0133	0.0008	0.0007

TABLE 6. Effect of *n* on the performance of the four algorithms.

n	Situation of optimal solution obtained								Average running time				Average relative error			Standard deviation		
	N_C	$P_C(\%)$	N_H	$P_H(\%)$	N_G	$P_G(\%)$	N_G	$P_G(\%)$	T_C	T_H	T_G	T_G	A_H	A_G	A_G	S_H	S_G	S_G
20	240	100.00	153	63.75	240	100.00	240	100.00	2.39	0.33	2.01	3.15	0.005	0	0	0	0	0
40	240	100.00	111	46.25	111	46.25	217	90.42	6.88	0.57	2.94	4.10	0.005	0.081	0.013	0.0001	0.0008	0.0004
80	232	96.67	77	32.08	99	41.25	149	62.08	650.89	0.97	7.05	8.25	0.004	0.009	0.0003	0.0099	0.0058	0.0015

TABLE 7. Effect of n_R on the performance of the four algorithms.

n_R	Situation of optimal solution obtained								Average running time				Average relative error			Standard deviation		
	N_C	$P_C(\%)$	N_H	$P_H(\%)$	N_G	$P_G(\%)$	N_G	$P_G(\%)$	T_C	T_H	T_G	T_G	A_H	A_G	A_G	S_H	S_G	S_G
$0.25n$	240	100.00	138	57.50	147	61.25	214	89.17	58.47	0.61	2.95	4.32	0.008	0.0004	0.0002	0.0241	0.0021	0.0011
$0.50n$	235	97.92	99	42.13	168	71.49	186	79.15	340.14	0.60	4.82	5.96	0.004	0.0007	0.0001	0.0101	0.0009	0.0005
$0.75n$	237	98.75	104	43.88	135	56.96	200	83.33	250.44	0.65	4.83	7.79	0.003	0.0008	0.0001	0.0052	0.0010	0.0005

TABLE 8. Effect of *FIT* on the performance of the three algorithms.

FIT	Result of optimal solution obtained								Average running time				Average relative error			Standard deviation		
	N_C	$P_C(\%)$	N_H	$P_H(\%)$	N_G	$P_G(\%)$	N_G	$P_G(\%)$	T_C	T_H	T_G	T_G	A_H	A_G	A_G	S_H	S_G	S_G
0.1	180	100.00	115	63.89	130	72.22	167	92.78	83.90	0.61	4.56	5.74	0.002	0.0005	0.0000	0.0080	0.0005	0.0002
0.3	178	98.89	75	42.13	104	79.22	141	79.21	151.34	0.62	4.01	5.97	0.005	0.0009	0.0001	0.0121	0.0005	0.0006
0.5	176	97.78	75	42.61	115	65.34	149	84.66	352.34	0.62	5.00	5.74	0.006	0.0010	0.0001	0.0165	0.0009	0.0007
0.7	178	98.89	75	42.13	101	56.74	143	80.34	333.49	0.63	3.23	6.17	0.006	0.0011	0.0002	0.0148	0.0020	0.0011

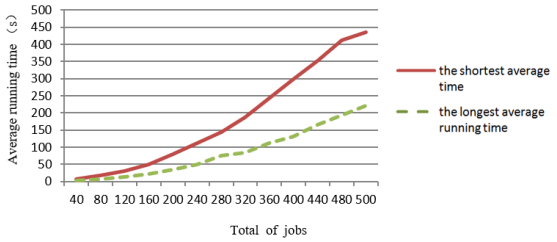

FIGURE 7. The average running time of GAA with different number of total jobs.

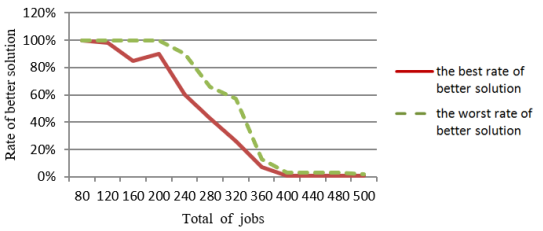

FIGURE 8. The rate of better solution of GAA changing with different number of total jobs.

Fig. 8 shows the changes in the rate of better solution for GAA with different total number of jobs. The solid line and the dotted line indicate the results of the factor combinations for the problem with the best rate of better solution and with the worst rate of better solution, respectively.

Fig. 9 shows the changes in the average relative error for GAA with different total number of jobs. The solid line and the dotted line indicate the result of the factor combinations for the problem with the largest average relative error and with the smallest average relative error, respectively.

The results of the above experiments suggest that with increasing total number of jobs, the trend of increase in the average running time is slow and the rate of better solution is

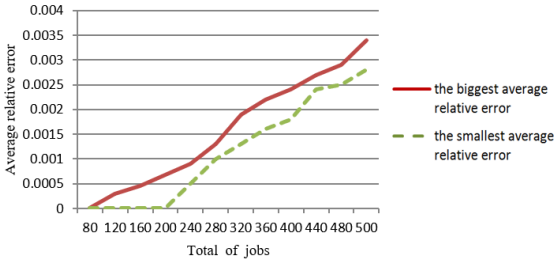

FIGURE 9. The average relative error of GAA changing with different number of total jobs.

declining. Although the average relative error is increasing, its value is on an acceptable level for GAA. Thus, GAA has

TABLE 9. The data of the actual example.

J	$p_i(h)$	$r_i(h)$	J	$p_i(h)$	$r_i(h)$	J	$p_i(h)$	$r_i(h)$
1	0.26	0	21	0.61	9.05	41	0.53	17.68
2	0.25	0.28	22	0.31	9.54	42	0.1	0
3	0.51	0.54	23	0.6	9.69	43	0.12	0
4	0.3	1.08	24	0.43	10.12	44	0.12	0
5	0.42	1.2	25	0.4	11.06	45	0.13	0
6	0.35	1.63	26	0.44	10.83	46	0.14	0
7	0.64	2.21	27	0.41	11.93	47	0.14	0
8	0.46	2.25	28	0.49	12.39	48	0.14	0
9	0.52	3.09	29	0.63	12.44	49	0.15	0
10	0.34	3.44	30	0.29	13.34	50	0.16	0
11	0.48	3.57	31	0.35	13.25	51	0.16	0
12	0.28	4.42	32	0.28	14.09	52	0.16	0
13	0.27	4.75	33	0.62	13.9	53	0.17	0
14	0.57	4.79	34	0.6	15.02	54	0.17	0
15	0.25	5.42	35	0.39	15.47	55	0.18	0
16	0.63	5.6	36	0.4	16.08	56	0.18	0
17	0.5	6.47	37	0.28	16.24	57	0.18	0
18	0.62	7.21	38	0.32	16.23	58	0.18	0
19	0.59	7.37	39	0.49	17.14	59	0.18	0
20	0.55	8.47	40	0.6	17.27	60	0.19	0

TABLE 10. The solutions of four algorithms.

Algorithms	Reschedule	Solution
CPLEX	53 1,..., 10 47 11 ,..., 26 42 27 ,..., 33 46 34 ,..., 38 45 39,..., 41 55 44 43 52 51 59 49 60 56 58 50 54 48 57	339.01
GAA	53 1,..., 10 47 11 ,..., 26 42 27 ,..., 33 46 34 ,..., 38 45 39,..., 41 55 44 43 52 51 59 49 60 56 58 50 54 48 57	339.01
GA	42 1 ,..., 11 46 12 ,..., 26 53 27 ,..., 38 45 39 ,..., 41 43 47 55 43 51 59 49 60 48 50 54 56 58 57	342.17
The heuristic	42 1 ,..., 11 46 12 ,..., 26 53 27 ,..., 38 45 39 ,..., 41 43 47 55 43 51 59 49 60 48 50 54 56 58 57	342.17
The manual scheduling	1 42 2 ,..., 17 46 18 ,..., 24 47 25 ,..., 41 43 44 45 48 49 50 51 52 ,..., 60	388.00

a satisfactory stability for solving SRRO with increasing the total number of jobs.

D. AN EXAMPLE IN QUARTZ GLASS FACTORY

An actual example of one day's production (three shifts in 24 h) on a machine in a quartz glass factory in China shows that the better solution can significantly reduce costs and energy consumption. In this example, $N_O = 41$, $N_R = 19$, $J_O = 1, 2, \ldots, 41$, $J_R = 42, 43, \ldots, 60$, $v = 1, 2, \ldots, 41$, and $K = 0.69$ h. Table 9 presents the data of all jobs in the example.

Table 10 presents the solutions obtained by MILP (CPLEX), GA, GAA, heuristic algorithm B, and manual scheduling (i.e., relying on the experience of workers). The results suggest that the optimal solution obtained by MILP (CPLEX) and GAA is $388.00 - 339.01 = 48.99$ h per day less than that of the manual solution.

In the quartz glass factory, the annual production is 5000 tons per production line. The specific heat of quartz is 0.8 kJ/(kg·K) and the cooling rate of spare parts after preheating is approximately 30–40 °C/h. Therefore, the rescheduling obtained by MILP (CPLEX) and GAA will save 21.67×10^{12} to 22.39×10^{12} kJ of energy compared to the manual rescheduling per year per production line in this factory. Therefore, this research is very significant and necessary for achieving cost and energy savings.

IX. CONCLUSION

In this study, a rescheduling problem of importance to some manufacturing industries aiming at energy savings is investigated. The theoretical importance of the study resides on the MILP of SRRO, complexity analysis, formalization of structural properties, development of three algorithms, and proofs of the optimal solution for the six special cases of SRRO by the proposed heuristic algorithm. The extensively effective experiments were designed to test the performance of the MILP with CPLEX software, the proposed heuristic algorithm, GA and GAA. With experimental results, there is no doubt that the MILP with CPLEX software is the best choice for solving small-scale SRROs. The numerical experiments show that the performance and ability of GAA are excellent and its solutions are close to optimal ones. Therefore, for large-scale problems, GAA is a better choice than the proposed heuristic, except for the six special cases solved optimally by the heuristic. An example of real situation in industry shows that the proposed rescheduling methodology is effective for attaining significant energy savings. In future work, with respect to practical applications, rework jobs with stochastic release times and production environments such as flowshop and job shop need to be considered and investigated.

REFERENCES

[1] Leon, V. J, "Game-theoretic control and robust scheduling of job-shops in the presence of disruptions," Ph.D. dissertation, dissertation, Lehigh Univ., Bethlehem, PA, USA, May 1991.
[2] S. D. Wu, R. H. Storer, and P. C. Chang, "A rescheduling procedure for manufacturing systems under random disruptions," in *New Directions for Operations Research in Manufacturing*, G. Fandel, T.Gulledge, and A. Jone, Eds. Berlin, Germany: Springer, 1992, pp. 292–308.
[3] G. E. Vieira, J. W. Herrmann, and E. Lin, "Rescheduling manufacturing systems: A framework of strategies, policies, and methods," *J. Sched.*, vol. 6, no. 1, pp. 39–62, 2003.
[4] J. W. Herrmann, "Rescheduling strategies, policies, and methods," in *Handbook of Production Scheduling*, vol. 89. 2006, pp. 135–148.
[5] A. T. Unal, R. Uzsoy, and A. S. Kiran, "Rescheduling on a single machine with part-type dependent setup times and deadlines," *Ann. Oper. Res.*, vol. 70, pp. 93–113,Apr. 1997.
[6] N. G. Hall and C. N. Potts, "Rescheduling for new orders," *Oper. Res.*, vol. 52, no. 3, pp. 440–453, Jun. 2004.
[7] N. G. Hall, Z. Liu, and C. N. Potts, "Rescheduling for multiple new orders," *INFORMS J. Comput.*, vol. 19, no. 4, pp. 633–645, Nov. 2007.
[8] J. Yuan and Y. Mu, "Rescheduling with release dates to minimize makespan under a limit on the maximum sequence disruption," *Eur. J. Oper. Res.*, vol. 182, no. 2, pp. 936–944, Oct. 2007.
[9] H. Hoogeveen, C. Lenté, and V. T'kindt, "Rescheduling for new orders on a single machine with setup times," *Eur. J. Oper. Res.*, vol. 223, no. 1, pp. 40–46, Nov. 2012.

[10] B. Yang, "Single machine rescheduling with new jobs arrivals and processing time compression," *Int. J. Adv. Manuf. Technol.*, vol. 34, nos. 3–4, pp. 378–384, Aug. 2007.
[11] C. Zhao and H. Tang, "Rescheduling problems with deteriorating jobs under disruptions," *Appl. Math. Model.*, vol. 34, no. 1, pp. 238–243, Jan. 2010.
[12] Y.-D. Guo, Q. Wang, and M. Huang, "Rescheduling with release time to minimize sum of waiting time considering waiting constraint of original loads," *Acta Automatica Sinica*, vol. 39, no. 12, pp. 2100–2110, Mar. 2014.
[13] Z. Liu and Y. K. Ro, "Rescheduling for machine disruption to minimize makespan and maximum lateness," *J. Scheduling*, vol. 17, no. 4, pp. 339–352, Aug. 2014.
[14] L. Le and Z. Hong, "Single-machine rescheduling of minimizing the maximum lateness with the disruptive arrival of new jobs," *J. Syst. Eng.*, vol. 4, pp. 494–506, Mar. 2014.
[15] Q. Zhao and J. Yuan, "Rescheduling to minimize the maximum lateness under the sequence disruptions of original jobs," *Asia–Pacific J. Oper. Res.*, vol. 34, no. 05, Oct. 2017, Art. no. 1750024.
[16] Y. Guo, M. Huang, Q. Wang, and V. Jorge Leon, "Single-machine rework rescheduling to minimize maximum waiting-times with fixed sequence of jobs and ready times," *Comput. Ind. Eng.*, vol. 91, pp. 262–273, Jan. 2016.
[17] G. L. Nemhauser and M. W. P. Savelsbergh, "A cutting plane algorithm for the single machine scheduling problem with release times," in *Combinatorial Optimization: New Frontiers in the Theory and Practice* (NATO ASI series F: Computer and Systems Sciences), M. Akgül, H. Hamacher, and S. Tufecki Eds. Berlin, Germany: Springer, 1992, pp. 63–84.
[18] M. R. Garey, R. E. Tarjan, and G. T. Wilfong, "One-processor scheduling with symmetric earliness and tardiness penalties," *Math. Oper. Res.*, vol. 13, no. 2, pp. 330–348, May 1988.
[19] W. E. Smith, "Various optimizers for single-stage production," *Naval. Res. Log. Quart.*, vol. 3, no. 1, pp. 59–66, Mar. 1956.
[20] K. T. Fang, "The Uniform Design: Application of number-theoretic methods in experimental design," *Acta Math. Applicatae Sinica*, vol. 3, no. 4, 1980.
[21] N. P. Jouppi and A. Eustace, "Data processing system and method with small fully-associative cache and prefetch buffers," U.S. Patent 5 261 066 A, Sep. 11, 1993.
[22] D.-J. Wang, F. Liu, Y.-Z. Wang, and Y. Jin, "A knowledge-based evolutionary proactive scheduling approach in the presence of machine breakdown and deterioration effect," *Knowl.-Based Syst.*, vol. 90, pp. 70–80, Dec. 2015.
[23] D.-J. Wang, F. Liu, J.-J. Wang, and Y.-Z. Wang, "Integrated rescheduling and preventive maintenance for arrival of new jobs through evolutionary multi-objective optimization," *Soft Comput.*, vol. 20, no. 4, pp. 1635–1652, Apr. 2016.
[24] D. J. Wang, F. Liu, and Y. Jin, "A multi-objective evolutionary algorithm guided by directed search for dynamic scheduling," *Comput. Oper. Res.*, vol. 79, pp. 279–290, Mar. 2017.
[25] K. Gao, L. Wang, J. Luo, H. Jiang, A. Sadollah, and Q. Pan, "Discrete harmony search algorithm for scheduling and rescheduling the reprocessing problems in remanufacturing: A case study," *Eng. Optim.*, vol. 50, no. 6, pp. 965–981, Jun. 2018.
[26] P. Kunkun, P. Quan-Ke, and G. Liang, "An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process," *Comput. Ind. Eng.*, vol. 122, pp. 235–250, Aug. 2018.
[27] D. Wang, Y. Yin, and T. C. E. Cheng, "Parallel-machine rescheduling with job unavailability and rejection," *Omega*, vol. 81, pp. 246–260, Dec. 2018.
[28] Y. Yin, T. C. E. Cheng, and D.-J. Wang, "Rescheduling on identical parallel machines with machine disruptions to minimize total completion time," *Eur. J. Oper. Res.*, vol. 252, no. 3, pp. 737–749, Aug. 2016.
[29] T. Jiang and G. Deng, "Optimizing the low-carbon flexible job shop scheduling problem considering energy consumption," *IEEE Access*, vol. 6, pp. 46346–46355, 2018.
[30] T. Jiang, C. Zhang, and H. Zhu, "Energy-efficient scheduling for a job shop using grey wolf optimization algorithm with double-searching mode," *Math. Problems Eng.*, vol. 2018, Oct. 2018, Art. no. 8574892.
[31] T. Jiang, C. Zhang, and Q.-M. Sun, "Green job shop scheduling problem with discrete whale optimization algorithm," *IEEE Access*, vol. 7, pp. 43153–43166, 2019.

YANDONG GUO is currently an Associate Professor with Bohai University. Her research interests cover production planning and scheduling, optimization and decision, modeling and simulation of the chemical process, molecular simulation, and so on.

MIN HUANG is currently the Head of the Department of Artificial Intelligence and the Director of the Institute for Data Intelligence & Systems Engineering, College of Information Science and Engineering, Northeastern University, China. She has been recognized as the Distinguished Young Scholars by NSFC, Changjiang Scholarship Chair Professor of MOE in China. Her research interests cover the management of logistics and supply chain systems, the modeling, analytics and optimization for manufacturing and service systems, risk management, behavioral operations management, data analysis and machine learning, and so on.

QING WANG is currently an Associate Professor with the Department of Systems Engineering, College of Information and Engineering, Northeastern University. His research interests include production planning and scheduling, operations management, supply chain, mathematical modeling and optimization theories, e-commerce, and intelligent algorithm.

V. JORGE LEON is currently the Allen-Bradley Professor in Factory Automation at the Texas A&M University, Uvalde, TX, USA, where he holds a joint appointment with the Department of Engineering Technology and Industrial Distribution (ETID) and the Department of Industrial and Systems Engineering (ISEN). His teaching and research interests are in the areas of operations optimization with specific consideration to robustness, partial information-sharing, rescheduling, and multiple optimization criteria.