Dual-Objective Mixed Integer Linear Program and Memetic Algorithm for an Industrial Group Scheduling Problem

Ziyan Zhao, *Student Member, IEEE*, Shixin Liu, *Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, and Abdullah Abusorrah, *Senior Member, IEEE*

***Abstract*—Group scheduling problems have attracted much attention owing to their many practical applications. This work proposes a new bi-objective serial-batch group scheduling problem considering the constraints of sequence-dependent setup time, release time, and due time. It is originated from an important industrial process, i.e., wire rod and bar rolling process in steel production systems. Two objective functions, i.e., the number of late jobs and total setup time, are minimized. A mixed integer linear program is established to describe the problem. To obtain its Pareto solutions, we present a memetic algorithm that integrates a population-based nondominated sorting genetic algorithm II and two single-solution-based improvement methods, i.e., an insertion-based local search and an iterated greedy algorithm. The computational results on extensive industrial data with the scale of a one-week schedule show that the proposed algorithm has great performance in solving the concerned problem and outperforms its peers. Its high accuracy and efficiency imply its great potential to be applied to solve industrial-size group scheduling problems.**

***Index Terms*—Insertion-based local search, iterated greedy algorithm, machine learning, memetic algorithm, nondominated sorting genetic algorithm II (NSGA-II), production scheduling.**

Manuscript received September 12, 2020; revised October 23, 2020; accepted November 18, 2020. This work was supported by the China Scholarship Council Scholarship, the National Key Research and Development Program of China (2017YFB0306400), the National Natural Science Foundation of China (62073069) and the Deanship of Scientific Research (DSR) at King Abdulaziz University (RG-48-135-40). Recommended by Associate Editor Tao Yang. *(Corresponding authors: Shixin Liu and MengChu Zhou.)*

Citation: Z. Y. Zhao, S. X. Liu, M. C. Zhou, and A. Abusorrah, "Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 6, pp. 1199–1209, Jun. 2021.

Z. Y. Zhao is with the State Key Laboratory of Synthetical Automation for Process Industries and the College of Information Science and Engineering, Northeastern University, Shenyang 110819, China, and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark NJ 07102 USA (e-mail: neuzhaoziyan@163.com).

S. X. Liu is with the State Key Laboratory of Synthetical Automation for Process Industries and the College of Information Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: sxliu@mail.neu.edu.cn).

M. C. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark NJ 07102 USA, and also with Center of Research Excellence in Renewable Energy and Power Systems, King Abdulaziz University, Jeddah 21481, Saudi Arabia (e-mail: zhou@njit.edu).

A. Abusorrah is with the Department of Electrical and Computer Engineering, Faculty of Engineering, and Center of Research Excellence in Renewable Energy and Power Systems, King Abdulaziz University, Jeddah 21481, Saudi Arabia (e-mail: aabusorrah@kau.edu.sa).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2020.1003539

I. Introduction

As an important branch of production scheduling problems, group scheduling problems (GSPs) have been increasingly concerned in recent years because of their extensive industrial applications [1]–[7]. In GSP, jobs to be scheduled are from several groups (or families). Those from a group have the same production requirements. In many industrial processes, jobs from a group are processed in serial batches [8], [9]. A so-called serial batch means that jobs within it have to be processed consecutively and thus its processing time equals the summation of the jobs'.

Setup time, which means the time for preparing a machine to satisfy production requirements, is sometimes non-ignorable. It can be divided into two classes, i.e., sequence-independent setup time [10] and sequence-dependent one [3]. The former means that setup time is only determined by a task, i.e., job or batch, to be processed; while the latter means that it depends on two consecutive tasks. In GSP, setup time does not exist between two consecutive jobs in a batch, but between two consecutive batches from different groups. Since it requires extra energy and labor cost, minimizing total setup time (denoted as $\widetilde{S}$) is an essential optimization objective in many scheduling problems [11]. Lee *et al.* [12] study a scheduling problem originated from a polyvinyl chloride sheet manufacturing process to minimize $\widetilde{S}$. Chen investigates a single machine scheduling problem by considering sequence-dependent setup time and minimizing $\widetilde{S}$ [13].

Two important classes of constraints, i.e., release time and due time, are commonly considered in scheduling problems [14]. The former is the time when a task becomes available for processing; while the latter is the time before which a task is expected to be completed. A scheduling problem considering the constraints of release time and sequence-dependent setup time, as well as an objective of minimizing makespan, is studied in [15], where a beam search algorithm is presented to solve it. A job is considered as a late one if it is completed later than its due time. The total number of late jobs (denoted as $\widetilde{N}$) is an important objective to be minimized in many scheduling problems [16]. A branch-and-bound method is designed in [17] to solve a single machine scheduling problem with periodic maintenance constraints and an objective of minimizing $\widetilde{N}$. A single machine scheduling problem with a time-dependent learning effect is studied in [18] to minimize

$\widetilde{N}$, which is solved by a branch-and-bound method and two heuristics.

In some scheduling problems, a scheduler makes decisions based on multiple optimization objectives. Thus, many researchers focus on multi-objective scheduling problems [19]–[22]. Different from a single-objective one that is to find an optimal or near-optimal solution, a multi-objective one aims to find an optimal or near-optimal Pareto set (or Pareto front) containing its non-dominated solutions [19]. Multi-objective evolutionary algorithms (MOEAs) [23]–[28] are effective and popular to solve multi-objective optimization problems. Among them, a nondominated sorting genetic algorithm II (NSGA-II) [23] is a well-known and widely used one, which effectively solves some flexible job-shop scheduling problems [29] and flow-shop ones [30]. Recently, memetic algorithms which integrate population-based global search and individual-based local heuristic search show great performance in solving multi-objective optimization problems [31]–[33], and thus motivates this work.

This work proposes and studies a new bi-objective serial-batch GSP (BSGSP) with a goal to minimize both $\widetilde{S}$ and $\widetilde{N}$ by considering sequence-dependent setup time, release time, and due time. The work in [3] and [4] tackles a similar problem but considers only a single objective function, i.e., minimizing $\widetilde{N}$. This work extends it to a bi-objective optimization problem to suit industrial needs. Besides [3] and [4], some similar but different problems are studied in literature [33]–[35]. [33] investigates a bi-objective parallel flow-shop scheduling problem with release time to minimize both $\widetilde{N}$ and total flowtime. [34] considers a serial-batch GSP with release time, sequence-independent setup time, and an aim to minimize makespan. [35] tackles a serial-batch scheduling problem on a single machine with learning effect to minimize $\widetilde{N}$.

This work attempts to make the following contributions:

1) It proposes a new BSGSP to minimize $\widetilde{S}$ and $\widetilde{N}$, which is never addressed in existing work to the best of authors' knowledge;

2) It establishes a mixed-integer linear program (MILP) to describe BSGSP in a rigorous way; and

3) It develops a new algorithm integrating NSGA-II and two individual-based improvement methods, i.e., an insertion-based local search and an iterated greedy algorithm, to solve BSGSP. In addition, this work compares the proposed algorithm with three competitive peers by extensive experiments to show its great performance and readiness for industrial applications.

Section II describes the concerned problem and formulates it into an MILP. A new algorithm is presented to solve the concerned problem in Section III. Experimental results are shown and analyzed in Section IV. Section V draws the conclusions and discusses some future research issues.

II. Problem Description and Modeling

A. Industrial Problem

A wire rod and bar rolling process plays a key role in steel production systems [3]. Steel ingots with short and thick shapes are processed into steel billets with long and thin shapes by going through a rolling mill as shown in Fig. 1. This work considers a single-machine scheduling problem on a rolling mill. Typically, a job in a rolling process is not just to deal with one steel ingot, but to continuously handle a set of steel ingots from a customer order, which have the same production and delivery requirements. A rolling mill has many positions to equip rolling stands as shown in Fig. 2 where 17 positions are available [36]. A rolling stand contains a pair of rollers in either parallel or vertical directions. The whole mechanical structure consisting of the equipped rolling stands is called a roll pass of a rolling mill. A roll pass can deal with the jobs with a range of specifications. Note that the rolling stands are not identical. Equipping different rolling stands on a position can lead to different size of steel billets. Apparently, the specification requirement of a job determines the roll pass of a rolling mill. If two jobs requiring different roll passes are processed consecutively, workers have to spend setup time in switching the equipped roll pass between them. In other words, some rolling stands need be equipped on or removed from the rolling mill. The setup time can be regarded as a linear function of the number of different rolling stands among the ones making up two roll passes [3].

Fig. 1. Schema of a rolling process.

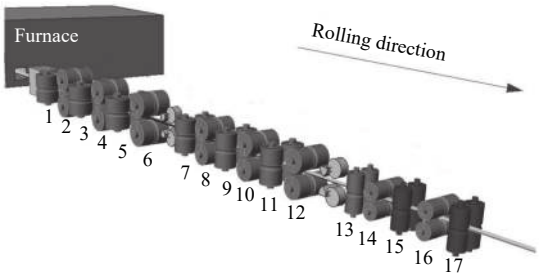


Fig. 2. Schema of a rolling mill.

As a medium process of steel production systems, a rolling process has time limits from its upstream and downstream ones. A job in it can only be handled after being released by an upstream process and is expected to be completed before the due time required by a downstream process. A job becomes a late one if it is completed after its due time. According to industrial production rules, the jobs in a rolling process are assigned into some batches in advance. The ones in the same batch require the same roll pass and have to be processed continuously [3]. Realizing optimal scheduling for such an industrial process is very important for steel plants. In the scheduling problem, schedulers and practitioners must focus on minimizing two objective functions, i.e., the number of late jobs ($\widetilde{N}$) and total setup time ($\widetilde{S}$). Reducing $\widetilde{S}$ tends to continuously process the batches from the same family and thus may increase $\widetilde{N}$; while reducing $\widetilde{N}$ needs to process as many jobs as possible before their due time and thus may increase $\widetilde{S}$, which results from frequent changes of machine

states to satisfy production requirements. Hence, minimizing both $\widetilde{S}$ and $\widetilde{N}$ causes a conflict.

B. Problem Transformation

To model and solve the above industrial problem, we transform it into a new single machine scheduling problem, i.e., a bi-objective serial-batch group scheduling problem (BSGSP). It has the following characteristics. J jobs from F families are to be scheduled on a single machine. They have been arranged into B serial non-preemptive batches in advance. The ones assigned into a batch are from the same family. Thus, $J \geq B \geq F$. There is no setup time between two consecutive jobs from the same batch. However, sequence-dependent setup time is necessary between two consecutive batches if they are from different families. It is to change the machine states to satisfy different production requirements the two batches. The batches and jobs to be scheduled have their release time, processing time, and due time requirements. $\widetilde{S}$ and $\widetilde{N}$ are two objective functions to be minimized.

$\widetilde{S}$ is only impacted by a batch sequence, i.e., an order in which all batches are processed on a rolling mill, while $\widetilde{N}$ depends on not only a batch sequence but also job sequences in the batches. Thus, intuitively, BSGSP aims to find a batch sequence and a job sequence in each batch. However, by using the first stage of a two-stage decomposition method proposed in [3], the number of late jobs within a batch b (denoted by $\widetilde{N}_b$) can be derived from its start time (denoted by t_b), which only depends on a batch sequence. To understand the details of the decomposition method, one can refer to [3]. Here we directly use its conclusion. Let N_b be the number of jobs in batch b. The method [3] constructs a mapping from t_b to $\widetilde{N}_b$. It divides the timeline into N_b+1 slots with lower and upper bounds denoted by $\check{t}_{b,n}$ and $\hat{t}_{b,n}$, respectively, where $n \in \mathbb{N}_b = \{0,1,\ldots,N_b\}$. If $\check{t}_{b,n} \leq t_b < \hat{t}_{b,n}$, then $\widetilde{N}_b = n$. After using it, we can see that both $\widetilde{S}$ and $\widetilde{N}$ are determined by a batch sequence only. In other words, the need for considering job sequences in the batches is eliminated and the goal of BSGSP is to optimize a batch sequence and minimize $\widetilde{S}$ and $\widetilde{N}$. Note that we aim to obtain the Pareto solutions of BSGSP since it is a bi-objective optimization problem.

C. Mathematical Model

In this section, an MILP is established to describe BSGSP. A virtual batch with index 0 is introduced as the first one in a schedule to represent the initial state of a machine and $P_0 = 0$. For an easy description, we define two sets: $\mathbb{B} = \{0,1,\ldots,B\}$ and $\mathbb{B}^+ = \{1,2,\ldots,B\}$. $\mathbb{R}$ means the set of real numbers. $\mathcal{F}_1$ and $\mathcal{F}_2$ represent two objective functions, i.e., $\widetilde{S}$ and $\widetilde{N}$, respectively. Besides, we define some notations in Table I and three groups of decision variables as follows:

$$x_{b,b'} = \begin{cases} 1, & \text{if batch } b' \text{ is immediately after batch } b \\ 0, & \text{otherwise} \end{cases}$$

$$y_{b,n} = \begin{cases} 1, & \text{if } \widetilde{N}_b = n \\ 0, & \text{otherwise} \end{cases}$$

$t_b \in \mathbb{R}$: start time of batch b.

Then, an MILP is proposed as follows:

TABLE I
NOTATIONS

Notation	Definition
R_b	Release time batch b
P_b	Processing time of batch b
$S_{b,b'}$	Setup time between two consecutive batches b and b'
D_j	Due time of job j
$\check{t}_{b,n}$	A lower bound of a start time window leading to n late jobs in batch b
$\hat{t}_{b,n}$	An upper bound of a start time window leading to n late jobs in batch b

$$\min \quad \mathcal{F}_1 = \widetilde{S} = \sum_{b\in\mathbb{B}} \sum_{b'\in\mathbb{B}^+} S_{b,b'} x_{b,b'} \tag{1}$$

$$\min \quad \mathcal{F}_2 = \widetilde{N} = \sum_{b\in\mathbb{B}^+} \widetilde{N}_b = \sum_{b\in\mathbb{B}^+} \sum_{n\in\mathbb{N}_b} n y_{b,n} \tag{2}$$

$$\text{s.t.} \sum_{b'\in\mathbb{B}^+} x_{b,b'} \leq 1, \; b \in \mathbb{B} \tag{3}$$

$$\sum_{b\in\mathbb{B}} x_{b,b'} = 1, \; b' \in \mathbb{B}^+ \tag{4}$$

$$t_b \geq t_{b'} + P_{b'} + S_{b',b} + \mathcal{M}(x_{b',b} - 1), \quad b \in \mathbb{B}^+, b' \in \mathbb{B} \tag{5}$$

$$t_0 = \min_{b\in\mathbb{B}^+} R_b \tag{6}$$

$$t_b \geq R_b, \; b \in \mathbb{B}^+ \tag{7}$$

$$\sum_{n\in\mathbb{N}_b} y_{b,n} = 1, \; b \in \mathbb{B}^+ \tag{8}$$

$$t_b \geq \sum_{n\in\mathbb{N}_b} \check{t}_{b,n} y_{b,n}, \; b \in \mathbb{B}^+ \tag{9}$$

$$t_b \leq \sum_{n\in\mathbb{N}_b} \hat{t}_{b,n} y_{b,n}, \; b \in \mathbb{B}^+ \tag{10}$$

$$x_{b',b}, y_{b,n} \in \{0,1\}, \; b' \in \mathbb{B}, b \in \mathbb{B}^+, n \in \mathbb{N}_b \tag{11}$$

$$t_b \in \mathbb{R}, \; b \in \mathbb{B}. \tag{12}$$

The objective functions in (1) and (2) minimize $\widetilde{S}$ and $\widetilde{N}$, respectively. Equations (3)–(7) together ensure a feasible batch sequence. Specifically, (3) means that there is at most one batch immediately after a batch. Only the last one is not followed by anyone. Equation (4) ensures that there is exactly one batch immediately before a batch to be scheduled. Note that the virtual batch as the first one is not restricted by it. Equation (5) ensures that at most one batch can be processed at a time, where $\mathcal{M}$ is a sufficiently large positive number. If $x_{b,b'} = 1$, we rewrite (5) as

$$t_b \geq t_{b'} + P_{b'} + S_{b',b} \tag{13}$$

where $b \in \mathbb{B}^+$ and $b' \in \mathbb{B}$. Otherwise, it is automatically relaxed. Equation (6) defines that the virtual batch starts from the earliest release time among all the batches to be scheduled. Equation (7) ensures that a batch is available only after its

release time. Equations (8)–(10) together map t_b to $\widetilde{N}_b$. Specifically, (8) ensures that a batch starts within exactly one time slot. Equations (9) and (10) provide its start time with lower and upper bounds. Equations (11) and (12) define the ranges of decision variables.

III. Memetic Algorithm

A. Encoding and Decoding

According to the problem description, we know that the concerned problem is to find batch sequences that satisfy the constraints and lead to non-dominated objective function values. Thus, we use a non-repetitive integer permutation of the elements from $\mathbb{B}$ to encode a batch sequence. For example, when scheduled three batches, a permutation $\pi = \langle 1,3,2\rangle$ means that batch 1 is the first one to be processed followed by batches 3 and 2.

In a decoding procedure, given a batch sequence, decision variables $x_{b,b'}$ become known. For the above instance, all the $x_{b,b'} = 0$, where $b, b' \in \{1,2,3\}$, except for $x_{1,3} = 1$ and $x_{3,2} = 1$. Then, we calculate $\widetilde{S}$ by (1). For obtaining $\widetilde{N}$, t_b needs to be calculated first for each batch $b \in \mathbb{B}$. According to (7) and (13), t_b can be obtained via

$$t_b = \max\{R_b, t_{b'} + P_{b'} + S_{b',b}\} \tag{14}$$

where b' is the previous batch of b. Note that when $b = 0$, t_0 is determined by (6). Then, t_b is mapped to $\widetilde{N}_b$ by (8)–(10). Finally, we calculate $\widetilde{N}$ via (2).

B. Algorithm Design

We use each circle point in Fig. 3 to represent a solution of BSGSP. The hollow ones mean the solutions that are dominated by at least one of the others. The solid ones mean the solutions that constitute a Pareto front and are not dominated by the others. NSGA-II [23], [24] is a well-known MOEA, which is a population-based algorithm. However, when solving BSGSP with it, we find that the obtained Pareto front is far away from the ideal one. Thus, we design a novel memetic algorithm in Algorithm 1 by combining NSGA-II and two individual improvement methods, i.e., insertion based local search (ILS) and iterated greedy algorithm (IGA) [37] to solve BSGSP. Both ILS and IGA are single-solution-based algorithms that start from an initial solution and return a local optimal solution. For an easy description, we use NIMA to denote the proposed NSGA-II and Individual-improvement-based memetic algorithm.

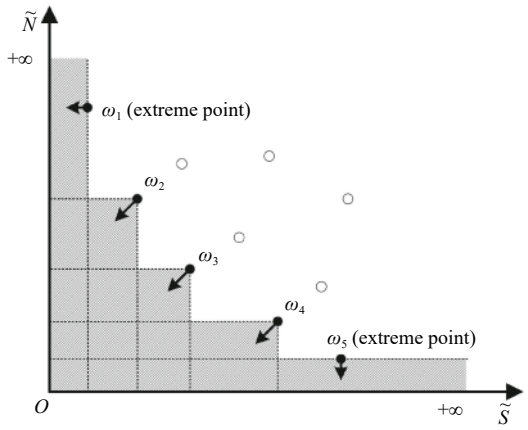


Fig. 3. Search space illustration of the proposed ILS.

In Algorithm 1, NSGA-II (lines 1–8) is used for evolution of the population, while ILS (lines 9–13) and IGA (lines 9–23) are used to further improve the individuals in a Pareto front. NIMA starts from random initial population $\mathbb{P}_0$. First, the individuals in $\mathbb{P}_0$ are sorted by a fast non-dominated sorting approach [23]. It compares the solutions in a population and gives each individual a rank according to the number of solutions that dominate it. The fewer solutions an individual is dominated by, the higher rank it has. After that, let $t = 0$ and a main loop starts for evolution. In it, an offspring population $\mathbb{Q}_t$ is generated according to $\mathbb{P}_t$ by using selection, crossover, and mutation operators. Next, $\mathbb{M}_t$ is obtained by merging $\mathbb{P}_t$ and $\mathbb{Q}_t$, which is then sorted by the fast non-dominated sorting approach. A new population $\mathbb{P}_{t+1}$ for the next generation is selected from $\mathbb{M}_t$ according to the individuals' ranks and crowded distances [23]. Then, update the iteration index by $t = t + 1$. Till now, an iteration of a population-based global search, i.e., NSGA-II, is done. After every M iterations, we collect the non-repetitive solutions in the Pareto front of $\mathbb{P}_t$ into a set Ω and improve them by using ILS and IGA. Actually, ILS (line 13) is used as a step of IGA. Let $|\Omega|$ denote the size of Ω. In an IGA section, the solutions in Ω denoted by ω_i, $i \in \{1,2,\ldots,|\Omega|\}$, are first sorted in terms of $\widetilde{S}$ from small to large. It is to find two extreme points of the Pareto front, i.e., ω_1 and $\omega_{|\Omega|}$, which have minimum $\widetilde{S}$ and $\widetilde{N}$, respectively, as shown in Fig. 3. ILS is conducted on each ω_i. If ω_i is an extreme point, i.e., $i = 1$ or $i = |\Omega|$, we further execute the following steps of IGA to explore better solutions. First, the solution after ILS is recorded as the current best one denoted by ω_i^*. Let $k = 0$ be a counter. A subloop starts and continues until k equals a given parameter K. At the beginning of this subloop, the counter is incremented by 1. Then, destruction and construction steps are executed to obtain a new solution ω_i', which is then improved by ILS. A simulated annealing acceptance criterion [37] is used to determine whether ω_i' is accepted or not. ω_i^* is updated once a better solution than it is found. When the subloop terminates, ω_i is updated by ω_i^*. When the main loop ends, the non-repetitive solutions in the Pareto front of $\mathbb{P}_t$ is selected and output.

Algorithm 1: Outline of NIMA

Input: A random initial population $\mathbb{P}_0$, a parameter M to activate local search

Output: A Pareto set Ω

1 Apply fast non-dominated sorting to $\mathbb{P}_0$ to get its individuals' ranks;

2 $t \leftarrow 0$;

3 while not termination **do**

4 $\mathbb{Q}_t \leftarrow$ An offspring population whose individuals are generated by conducting crossover and mutation operations;

5 $\mathbb{M}_t \leftarrow \mathbb{P}_t \cup \mathbb{Q}_t$;

6 Apply fast non-dominated sorting to $\mathbb{M}_t$ to get its

individuals' ranks;
7 $\mathbb{P}_{t+1} \leftarrow$ A new population selected from $\mathbb{M}_t$ based on individuals' ranks and crowded instances;
8 $t \leftarrow t+1$;
9 **if** *t is divisible by* M **then**
10 $\Omega \leftarrow$ A set of non-repetitive solutions in the Pareto front of $\mathbb{P}_t$;
11 Sort the solutions in Ω as one objective function value;
12 **for** $i = 1$ to $\Omega.size$ **do**
13 $\omega_i \leftarrow ILS(\omega_i)$;
14 **if** $i == 1$ or $i == \Omega.size$ **then**
15 $\omega_i^* \leftarrow \omega_i$;
16 $k \leftarrow 0$;
17 **while** $k < K$ **do**
18 $k \leftarrow k+1$;
19 $\omega_i' \leftarrow Des/Construction(\omega_i)$;
20 $\omega_i' \leftarrow ILS(\omega_i')$;
21 $\omega_i \leftarrow Acceptance(\omega_i, \omega_i')$;
22 $\omega_i^* \leftarrow Update(\omega_i^*, \omega_i')$;
23 $\omega_i \leftarrow \omega_i^*$;
24 $\Omega \leftarrow$ A set of non-repetitive solutions in the Pareto front of $\mathbb{P}_t$;
25 **return** Ω.

C. Selection, Crossover and Mutation

In NIMA, three operators, i.e., binary tournament selection (BTS) [23], partially mapped crossover (PMX) [38], and reciprocal exchange mutation (REM) [33], are used. Their details are introduced as follows.

1) A BTS is to select a parent individual. It randomly selects two individuals in the current population and compare them with each other. If they have different ranks, the higher-rank one is selected. Otherwise, we randomly select one of them.

2) A PMX operator is used to generate two offspring individuals (denoted by O1 and O2) after selecting two parent individuals (denoted by P1 and P2) by twice runs of BTS. An example of its procedure is illustrated in Fig. 4. First, two cut points on parent individuals are randomly chosen. The portion between the cut points is inherited by the offspring individuals. Here, the one of P1 (resp. P2) is inherited by O2 (resp. O1). Then, the mappings of the genes between the cut points are constructed. The rest genes of the offspring individuals are obtained by mapping the ones of the parent individuals to corresponding ones. Here, the genes mapped from the ones of P1 (resp. P2) are inherited by O1 (resp. O2).

3) An REM operator is to generate an offspring individual (O1) based on a parent one (P1). Fig. 5 shows its procedure, where two genes of the parent individual are selected and swapped to generate an offspring one.

D. Insertion-based Local Search

An ILS [37], [39] is used to further improve the solutions in the obtained Pareto front. Meanwhile, it is an important step of IGA. The shadows and arrows in Fig. 3 illustrate its search space. For a solution in the current Pareto front, we explore the space that dominates it. If it is an extreme point, we additionally explore the space that does not dominate it but

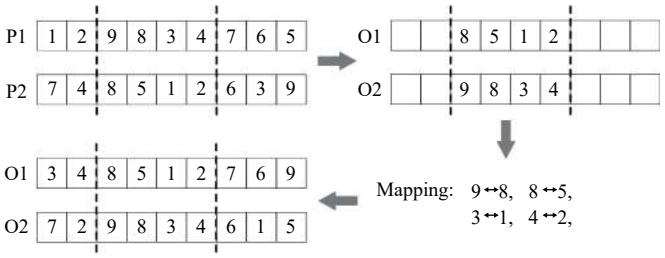


Fig. 4. Illustration of PMX operator.

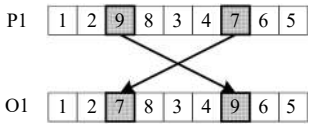


Fig. 5. Illustration of REM operator.

leads to smaller single objective function value than it. Algorithm 2 shows its procedure that is conducted on each non-repetitive solution in a Pareto front. In it, the best solution π^* is initialized by the input solution π. An iteration starts to randomly select a gene without repetition and remove it from π. Then, the best position is selected to reinsert it. When doing it, the removed position is initially marked as the best one followed by going through all the positions and comparing them with the marked one. Once a better position is found, it is marked as the best one. Let π^M and π^P be two solutions obtained by inserting the removed gene into the marked position (denoted by m) and another one (denoted by p), respectively. According to different input solution π, p is considered as a better position than m if it satisfies the following conditions:

1) For $\pi = \omega_1$, a) π^P has smaller $\widetilde{S}$ than π^M, or b) π^P has the same $\widetilde{S}$ as π^M but smaller $\widetilde{N}$ than it.

2) For $\pi = \omega_{|\Omega|}$, a) π^P has smaller $\widetilde{N}$ than π^M, or b) π^P has the same $\widetilde{N}$ as π^M but smaller $\widetilde{S}$ than it.

3) For others, π^P dominates π^M.

If the best position is not the removed one, we consider that the solution π after reinsertion is better than π^*, i.e., $\pi \prec \pi^*$. Then, π^* is updated by π. The search procedure terminates if π^* is not updated in an iteration or 10 iterations are performed.

Algorithm 2: Insertion-based local search

Input: A current solution π
Output: A local optimal solution π^*
1 *improve* = *true*;
2 *iteration* = 0;
3 $\pi^* = \pi$;
4 **while** *improve* & *iteration* < 10 **do**
5 *improve* = *false*;
6 *iteration* ← *iteration* + 1;
7 **for** $i = 1$ to B **do**
8 Select and remove a gene randomly from π without repetition;
9 $\pi \leftarrow$ Insert the gene into the best position;
10 **if** $\pi \prec \pi^*$ **then**
11 $\pi^* = \pi$;

12 $improve = true$;
13 return π^*.

E. Destruction and Construction

ILS provides a local optimal solution, which is broken out by destruction and construction steps of IGA as shown in Algorithm 3. In a destruction step, the batches from I random families are removed from a solution π, where I is a parameter to be adjusted. The removed ones constitute a subsequence called π^D and the remaining ones are in a subsequence called π^R. In a construction step, the batches in π^D are reinserted into the best positions of π^R one by one to construct a new solution. The so-called best position is defined depending on the input solution π.

1) If $\pi = \omega_1$, the best position means the one that leads to a subsequence (or sequence) with minimum $\widetilde{S}$ after insertion. Smaller makespan is considered as a tie-breaker.

2) If $\pi = \omega_{|\Omega|}$, the best position means the one that leads to a subsequence (or sequence) with minimum makespan after insertion. Smaller $\widetilde{S}$ is considered as a tie-breaker.

Note that we use makespan rather than $\widetilde{N}$ to evaluate a subsequence since $\widetilde{N}$ of a subsequence can be significantly changed by subsequent insertions.

Algorithm 3: Destruction and construction

Input: A current solution π
Output: A new solution π'
1 Select I families randomly;
2 Remove the batches from the selected families;
3 $\pi^D \leftarrow$ A subsequence consists of the removed batches;
4 $\pi^R \leftarrow$ A subsequence consists of the remaining batches;
5 for $i = 1$ to $\pi^D.size$ **do**
6 $\pi' \leftarrow$ Insert π_i^D into the best position of π^R;
7 return π'.

F. Acceptance Criterion

A new local optimal solution ω_i' is obtained after destruction, construction, and local search procedures. An acceptance criterion is used to determine whether it is accepted to replace the current one ω_i and applied for the next iteration or not. Let Δ mean the distance between ω_i' and ω_i, which is defined based on different extreme points. 1) When $i = 1$, we use $\widetilde{S}^C$ and $\widetilde{S}^N$ to denote $\widetilde{S}$ of ω_i and ω_i', respectively. Then, $\Delta = \widetilde{S}^N - \widetilde{S}^C$. 2) When $i = |\Omega|$, $\widetilde{N}^C$ and $\widetilde{N}^N$ refer to $\widetilde{N}$ of ω_i and ω_i', respectively. Then, $\Delta = \widetilde{N}^N - \widetilde{N}^C$. If $\Delta \le 0$, we accept ω_i' directly. Otherwise, we use an a simulated annealing-like acceptance criterion, which is commonly used for scheduling problems [37], [40]. To adopt it for the concerned problem, a constant temperature $\mathcal{T}$ is calculated as: $\mathcal{T} = \frac{T}{10 \cdot N} \sum_{i=1}^{B} P_i$, where T is a parameter to be adjusted. If a random number $\rho \in [0,1]$ satisfies $\rho \le \exp\left(-\frac{\Delta}{\mathcal{T}}\right)$, ω_i' is accepted. Otherwise, we reject it and use ω_i for the next iteration.

G. Computational Time Complexity

Since ILS and IGA are enabled per M iterations, we analyze the worst-case time complexity of NIMA in M iterations for BSGSP, which is denoted by C. A decoding procedure for calculating $\widetilde{S}$ and $\widetilde{N}$ consumes time $O(B+J)$, which can be simplified as $O(J)$ since $J \ge B$. Then, according to [23], NSGA-II in an iteration of NIMA requires time $O(U^2 J)$, where U means the population size. Based on [41], an ILS has the time complexity $O(B^2 J)$ when it is conducted on an individual from a Pareto set. An IGA takes time $O(B^2 JK)$ when it is used to improve an extreme point, since ILS is performed K times within it and has the highest complexity among all steps. In M iterations of NIMA, NSGA-II runs M times first. Then, we use ILS to improve the solutions in the Pareto front and IGA to improve two extreme points. Note that U solutions are in the Pareto front in the worst case. Thus, $C = O(U^2 JM + B^2 JU + B^2 JK) = O(J(U^2 M + B^2(K+U)))$.

IV. Experimental Results

A. Experimental Design

We use 14 400 experiments to test the proposed NIMA. The experiments are conducted on a dataset from a wire rod and bar rolling process [3]. The work in [3] uses the same industrial dataset to study a single-objective optimization problem while this work uses it to consider a bi-objective one. Nine groups of instances with different B and J are used. Their scales range from $B = 20$ and $J = 80$ to $B = 40$ and $J = 240$. The maximum scale is equivalent to one-week workload in a factory [3]. Each group contains twenty instances and thus $9 \times 20 = 180$ ones are solved in total.

To show the effectiveness of NIMA, we compare it with three MOEAs, i.e., NSGA-II [23], NSGA-III [26], and a recent peer denoted by NMMA, which is an NSGA-II and Mutation-local-search-based Memetic Algorithm [32]. For all the algorithms, they share the same population size, crossover and mutation rates, which are 100, 1 and 0.1, respectively.

The algorithms are all coded in C++ and run on a laptop computer with 32 GB of RAM and an Intel Core i7-8850H, 2.60 GHz processor. When solving an instance, for a fair comparison, all the algorithms share the same termination criterion which is the limited running time when an NSGA-II takes for 8000 iterations. Table II demonstrates the average termination time for solving an instance with given B and J. To reduce the impact of the randomness on algorithm performance, we run each of them 20 times to solve an instance. The average solutions are summarized and compared with each other. Therefore, we perform $180 \times 4 \times 20 = 14400$ experiments in this work for algorithm comparisons.

TABLE II
Termination Criterion

B	J	Time (s)	B	J	Time (s)
20	80	8.16	30	180	15.06
20	100	8.37	40	160	23.53
20	120	8.87	40	200	23.62
30	120	13.53	40	240	24.25
30	150	14.58			

B. Evaluation Metrics

Three kinds of metrics, i.e., extreme points, inverted

generational distance (IGD), and hypervolume, are used to evaluate the tested algorithms [42].

1) Extreme Points: Extreme points have the minimum single objective function values and thus are essential metrics to evaluate an algorithm. For many practical multi-objective scheduling problems, the solutions of the extreme points are important guides for practitioners.

2) IGD: Let $\mathcal{P}$ and $\mathcal{P}^*$ be a Pareto front obtained by solving an instance with an algorithm and an ideal one, respectively. Note that, since the real ideal one is unknown, we use the set of non-dominated solutions obtained by all tested algorithms in all runs to approximate $\mathcal{P}^*$. An IGD reflects the average distance between each pair of solutions $s \in \mathcal{P}^*$ and s', where s' is a solution in $\mathcal{P}$ and has minimum distance from s. Thus, it is calculated as $IGD(\mathcal{P},\mathcal{P}^*) = \sum_{s\in\mathcal{P}^*} \min_{s'\in\mathcal{P}} \mathcal{D}(s,s')/|\mathcal{P}^*|$, where $\mathcal{D}(s,s')$ means the Euclidean distance between s and s'. Note that, before calculating it, we normalize the objective function value $\mathcal{F}_i$ of a solution to $\mathcal{F}_i'$ by $\mathcal{F}_i' = (\mathcal{F}_i - \check{\mathcal{F}}_i)/(\hat{\mathcal{F}}_i - \check{\mathcal{F}}_i)$, where $i \in \{1,2\}$. $\check{\mathcal{F}}_i$ and $\hat{\mathcal{F}}_i$ represent the minimum and maximum single objective function values of the Pareto solutions obtained by all the runs of the algorithms [43]. Note that the smaller IGD, the better $\mathcal{P}$.

3) Hypervolume: Hypervolume is the volume of the region dominated by at least one solution in $\mathcal{P}$ and dominating a given reference point p. After normalization, p is set to (1,1).

C. Parameter Adjustment

Orthogonal experiments [43] are designed to adjust the four parameters of NIMA, i.e., 1) M: the number of iterations of NSGA-II before activating ILS and IGA; 2) K: the number of iterations of IGA for improving a Pareto solution; 3) I: the number of families to be removed in a destruction step; and 4) T: a temperature coefficient. For each of them, three candidates are given, i.e., $M \in \{100,300,500\}$, $K \in \{1,3,5\}$, $I \in \{1,2,3\}$, and $T \in \{1,5,10\}$. NIMAs with nine parameter combinations (denoted by G1–G9) are used to solve ten instances with $B = 30$ and $J = 150$. The average experimental results are shown in Table III. The parameters in G4 leading to the largest hypervolume are selected as the best combination and used for the following experiments. Note that we draw a box plot to show the hypervolume metrics of NIMA with different parameter combinations in Fig. 6. It shows that different combinations lead to similar hypervolumes, i.e., NIMA is parameter-insensitive.

TABLE III
PARAMETER ADJUSTMENT OF NIMA

No.	M	K	I	T	Hypervolume
G1	100	1	1	1	0.716
G2	100	3	2	5	0.718
G3	100	5	3	10	0.718
G4	300	3	3	1	**0.720**
G5	300	5	1	5	0.718
G6	300	1	2	10	0.717
G7	500	5	2	1	0.716
G8	500	1	3	5	0.717
G9	500	3	1	10	0.716

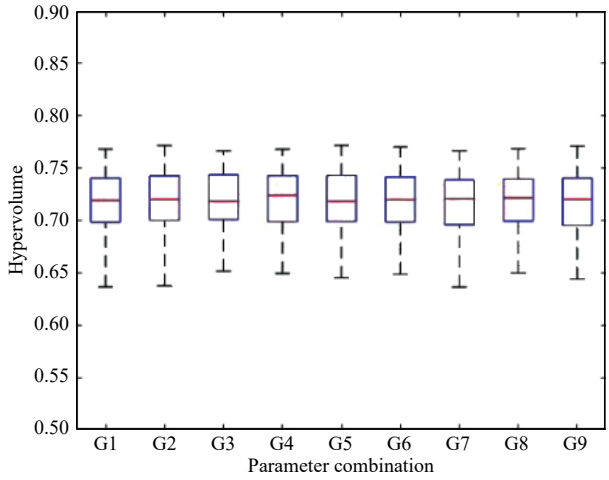


Fig. 6. Comparison of different parameter combinations.

D. Results and Comparisons

Tables IV and V compare $\widetilde{S}$ and $\widetilde{N}$ obtained by the four algorithms, denoted by $\widetilde{S}^a$ and $\widetilde{N}^a$, where $a \in \{\mathrm{i},\mathrm{ii},\mathrm{iii},\mathrm{iv}\}$, i.e., the extreme point comparisons. Note that $\widetilde{S}$ is measured in hours. The optimal solution or near-optimal one of $\widetilde{N}$ is denoted by $\widetilde{N}^*$ and given in [3], where CPLEX, i.e., a well-known commercial optimization solver, is used to solve an MILP. The optimal solution of $\widetilde{S}$ is denoted by $\widetilde{S}^*$ and can be obtained by using CPLEX to solving an MILP consisting of (1), (3)–(7), and a redundant constraint as follows;

$$x_{b,b'} = 1,\ b \in \mathbb{B},\ b' = \min \mathbb{L}_b. \tag{15}$$

In (15), $\mathbb{L}_b$ is a set of batches that are from the same family as batch b and have larger index than it. (15) gathers the batches from the same family and can improve the solving speed of CPLEX. The limited time of solving an instance with CPLEX is set to 7200 seconds (two hours). If its optimal solution cannot be obtained in 7200 seconds, CPLEX provides its near-optimal solution, which is a feasible one. G^a in Tables IV and V illustrate the gap between the solutions obtained by CPLEX and a tested algorithm, where $a \in \{\mathrm{i},\mathrm{ii},\mathrm{iii},\mathrm{iv}\}$. They are calculated as $G^a = (\widetilde{S}^a - \widetilde{S}^*)/\widetilde{S}^a \times 100\%$ in Table IV and $G^a = (\widetilde{N}^a - \widetilde{N}^*)/\widetilde{N}^a \times 100\%$ in Table V. We can see that the objective function values of the extreme points obtained by NIMA are very close to the optimal (or near-optimal) ones, and much better than those obtained by its peers. Note that, for some instances with $B = 40$, CPLEX cannot provide optimal solutions in limited time but near-optimal ones only. The reason why negative G^{i} appears for the group with $B = 40$ and $J = 200$ in Table V is that the solutions obtained by NIMA are better than CPLEX's in limited time.

Table VI demonstrates the IGD comparisons of the Pareto fronts obtained by the tested algorithms. NIMA gets the smallest IGD followed by NSGA-III, and is much better than NSGA-II and NMMA. Especially as the scale of the problem increases, NIMA can solve the instances with small IGDs but its peers' become larger and larger.

TABLE IV
$\widetilde{S}$ COMPARISON

B	J	$\widetilde{S}^*$	NIMA		NSGA-II		NSGA-III		NMMA	
			$\widetilde{S}^{\mathrm{i}}$	G^{i} (%)	$\widetilde{S}^{\mathrm{ii}}$	G^{ii} (%)	$\widetilde{S}^{\mathrm{iii}}$	G^{iii} (%)	$\widetilde{S}^{\mathrm{iv}}$	G^{iv} (%)
20	80	5.7	**5.8**	**1.50**	6.8 (+)	16.00	6.6 (+)	12.70	6.5 (+)	11.90
20	100	5.3	**5.4**	**1.60**	6.5 (+)	17.80	6.3 (+)	15.30	6.1 (+)	13.10
20	120	5.7	**5.8**	**1.40**	6.9 (+)	17.20	6.7 (+)	14.70	6.5 (+)	12.50
30	120	5.1	**5.1**	**1.60**	8.1 (+)	38.00	6.5 (+)	22.50	7.1 (+)	28.60
30	150	5.4	**5.4**	**0.50**	8.3 (+)	35.20	7.0 (+)	23.50	7.2 (+)	25.90
30	180	5.5	**5.6**	**1.10**	8.2 (+)	32.80	6.9 (+)	20.30	7.4 (+)	25.20
40	160	5.6	**5.7**	**1.00**	11.3 (+)	50.30	7.5 (+)	25.70	9.2 (+)	39.10
40	200	5.2	**5.2**	**1.40**	10.0 (+)	48.40	7.1 (+)	27.00	8.4 (+)	38.90
40	240	5.2	**5.3**	**1.70**	10.2 (+)	48.90	7.0 (+)	25.20	8.5 (+)	39.00

TABLE V
$\widetilde{N}$ COMPARISON

B	J	$\widetilde{N}^*$	NIMA		NSGA-II		NSGA-III		NMMA	
			$\widetilde{N}^{\mathrm{i}}$	G^{i} (%)	$\widetilde{N}^{\mathrm{ii}}$	G^{ii} (%)	$\widetilde{N}^{\mathrm{iii}}$	G^{iii} (%)	$\widetilde{N}^{\mathrm{iv}}$	G^{iv} (%)
20	80	5.0	**5.1**	**2.8**	12.8 (+)	60.9	6.0 (+)	16.6	9.8 (+)	48.9
20	100	14.6	**14.8**	**1.3**	25.9 (+)	43.5	15.8 (+)	7.7	21.9 (+)	33.3
20	120	25.7	**25.7**	**0.2**	42.3 (+)	39.3	26.8 (+)	4.0	37.6 (+)	31.7
30	120	8.6	**8.9**	**3.8**	40.8 (+)	78.9	11.4 (+)	24.6	33.4 (+)	74.2
30	150	23.5	**23.7**	**1.1**	65.5 (+)	64.2	26.0 (+)	9.7	57.3 (+)	59.1
30	180	43.9	**43.9**	**0.1**	89.3 (+)	50.8	46.3 (+)	5.2	81.0 (+)	45.8
40	160	14.7	**14.9**	**1.8**	76.3 (+)	80.8	20.7 (+)	28.9	66.5 (+)	77.9
40	200	32.9	**32.6**	**-0.7**	106.8 (+)	69.2	36.3 (+)	9.4	97.7 (+)	66.3
40	240	58.8	**59.2**	**0.7**	140.6 (+)	58.2	62.8 (+)	6.4	130.7 (+)	55.0

TABLE VI
IGD COMPARISON

B	J	NIMA	NSGA-II	NSGA-III	NMMA
20	80	**0.02**	0.08 (+)	0.05 (+)	0.06 (+)
20	100	**0.02**	0.09 (+)	0.05 (+)	0.06 (+)
20	120	**0.01**	0.10 (+)	0.04 (+)	0.07 (+)
30	120	**0.02**	0.19 (+)	0.06 (+)	0.13 (+)
30	150	**0.02**	0.20 (+)	0.05 (+)	0.15 (+)
30	180	**0.02**	0.21 (+)	0.05 (+)	0.16 (+)
40	160	**0.02**	0.29 (+)	0.06 (+)	0.22 (+)
40	200	**0.02**	0.30 (+)	0.06 (+)	0.25 (+)
40	240	**0.02**	0.30 (+)	0.06 (+)	0.25 (+)

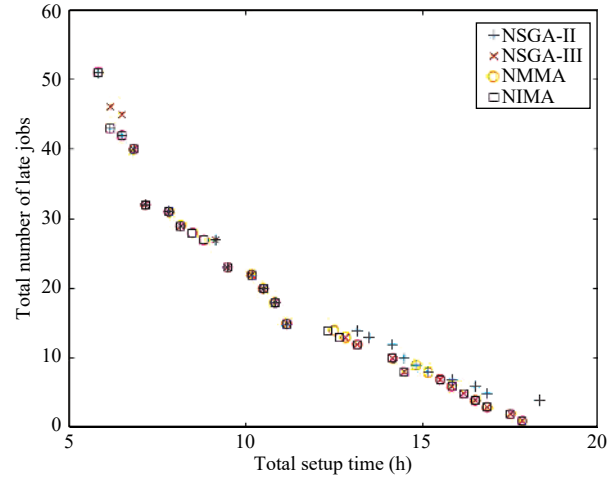
TABLE VII
HYPERVOLUME COMPARISON

B	J	NIMA	NSGA-II	NSGA-III	NMMA
20	80	**0.75**	0.66 (+)	0.71 (+)	0.70 (+)
20	100	**0.74**	0.64 (+)	0.70 (+)	0.68 (+)
20	120	**0.73**	0.61 (+)	0.69 (+)	0.65 (+)
30	120	**0.76**	0.52 (+)	0.70 (+)	0.59 (+)
30	150	**0.74**	0.49 (+)	0.68 (+)	0.56 (+)
30	180	**0.73**	0.48 (+)	0.67 (+)	0.55 (+)
40	160	**0.73**	0.39 (+)	0.67 (+)	0.47 (+)
40	200	**0.73**	0.38 (+)	0.67 (+)	0.45 (+)
40	240	**0.72**	0.38 (+)	0.66 (+)	0.44 (+)

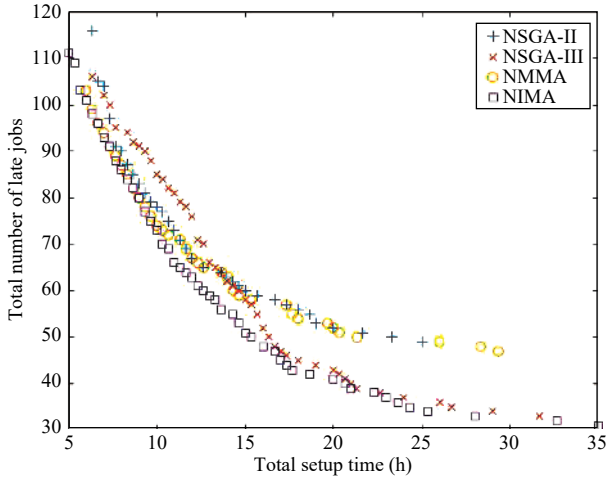
In terms of hypervolume metric, our comparison is shown in Table VII, which indicates that NIMA has better performance than its peers. Similar to IGD, the hypervolumes of the compared algorithms get worse and worse as the problem scale increases, but NIMA keeps a great one.

In addition, we adopt t-test with 38 degrees of freedom at a 0.05 level of significance to show the performance comparison of NIMA and its peers. The results in Tables IV–VII marked by using (+), (−), and (∼) mean that NIMA is significantly better than, significantly worse than, and statistically equivalent to its peers, respectively [44]. We can see that NIMA is significantly better than the compared algorithms for all groups of instances in terms of all evaluation metrics.

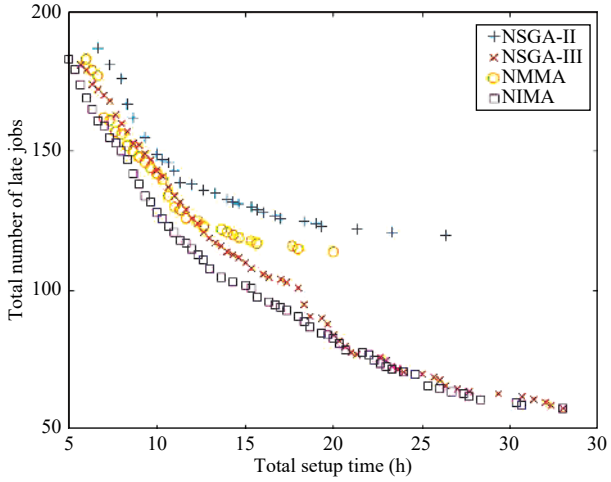
To clearly and intuitively illustrate the effectiveness of the proposed algorithm, in Fig. 7, we draw the final Pareto front obtained by NIMA and its peers when solving three randomly selected instances with different size, i.e., the ones with small ($B = 20$ and $J = 80$), medium ($B = 30$ and $J = 150$), and large ($B = 40$ and $J = 240$) scales. The so-called final Pareto front



(a) An instance with 20 batches and 80 jobs



(b) An instance with 30 batches and 150 jobs



(c) An instance with 40 batches and 240 jobs

Fig. 7. Final Pareto front comparison.

of an algorithm consists of the non-dominated solutions after merging the Pareto solutions of 20 independent runs. For the small-scale instance a), Fig. 7(a) shows that the difference of the final Pareto fronts obtained by the four tested algorithms is not significant. However, as problem scale grows, in Figs. 7(b) and (c), the final Pareto fronts obtained by NIMA are clearly better than those of its peers.

From the above comparisons, we can draw the conclusions that NIMA can well solve BSGSP and outperforms its peers. According to the termination criterion as shown in Table II, NIMA can solve the concerned problems with practical scales in very short time. Since the largest-scale instance with $B = 40$ and $J = 240$ is equivalent to an actual scheduling problem in one week [3], which is a common scheduling period used in practice, NIMA has great potential to be applied to a factory.

V. Conclusions and Future Work

This work tackles a new bi-objective serial-batch group scheduling problem with release time, due time, and sequence-dependent setup time. It arises from a practical industrial production system and aims to find a batch sequence to minimize both the number of late jobs and total setup time. A mixed integer linear program is formulated to describe it. Then, we design a novel memetic algorithm, based on hybrid NSGA-II, insertion-based local search, and iterated greedy algorithm, to solve it. Computational results of many experiments show the great effectiveness of the presented algorithm by comparing the extreme points obtained by it with the optimal or near-optimal solutions and comparing its performance with its three peers'. Its high solution accuracy and speed prove its great potential to be applied in practice.

As future research, we plan to extend the considered problem with real-world constraints, such as uncertain release time and processing time [35], [45]. The proposed algorithm can be problem-specifically modified to solve various similar problems [46]–[66].

References

[1] J. S. Neufeld, J. N. D. Gupta, and U. Buscher, "A comprehensive review of flowshop group scheduling literature," *Comput. Oper. Res.*, vol. 70, pp. 56–74, Jun. 2016.

[2] R. Zhang, S. J. Song, and C. Wu, "Robust scheduling of hot rolling production by local search enhanced ant colony optimization algorithm," *IEEE Trans. Ind. Inf.*, vol. 16, no. 4, pp. 2809–2819, Apr. 2020.

[3] Z. Y. Zhao, S. X. Liu, M. C. Zhou, X. W. Guo, and L. Qi, "Decomposition method for new single-machine scheduling problems from steel production systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1376–1387, Jul. 2020.

[4] Z. Y. Zhao, S. X. Liu, M. C. Zhou, D. You, and X. W. Guo, "Heuristic scheduling of batch production processes based on petri nets and iterated greedy algorithms," *IEEE Trans. Autom. Sci. Eng.*, to be published. 2020. DOI: 10.1109/TASE.2020.3027532.

[5] G. Fortino, F. Messina, D. Rosaci, G. M. L. Sarné, and C. Savaglio, "A trust-based team formation framework for mobile intelligence in smart factories," *IEEE Trans. Ind. Inf.*, vol. 16, no. 9, pp. 6133–6142, Sep. 2020.

[6] Z. Y. Zhao, S. X. Liu, M. C. Zhou, X. W. Guo, and J. L. Xue, "Iterated greedy algorithm for solving a new single machine scheduling problem," in *Proc. IEEE 16th Int. Conf. Networking, Sensing and Control*, Banff, Canada, 2019, pp. 430–435.

[7] Z. Y. Zhao, S. X. Liu, M. C. Zhou, and X. W. Guo, "Intelligent scheduling for a rolling process in steel production systems," in *Proc. IEEE Int. Conf. Networking, Sensing and Control*, Nanjing, China, 2020, pp. 1–6.

[8] C. Schwindt and N. Trautmann, "Batch scheduling in process industries: An application of resource-constrained project scheduling," *OR-Spektrum*, vol. 22, no. 4, pp. 501–524, Nov. 2000.

[9] J. Pei, X. B. Liu, W. J. Fan, P. M. Pardalos, and S. J. Lu, "A hybrid BA-VNS algorithm for coordinated serial-batching scheduling with deteriorating jobs, financial budget, and resource constraint in multiple manufacturers," *Omega*, vol. 82, pp. 55–69, Jan. 2019.

[10] E. Mehdizadeh, R. Tavakkoli-Moghaddam, and M. Yazdani, "A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times," *Appl. Math. Model.*, vol. 39, no. 22, pp. 6845–6859, Nov. 2015.

[11] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 345–378, Oct. 2015.

[12] C. H. Lee, C. J. Liao, and C. W. Chao, "Scheduling with multi-attribute setup times," *Comput. Ind. Eng.*, vol. 63, no. 2, pp. 494–502, Sep. 2012.

[13] W. J. Chen, "Sequencing heuristic for scheduling jobs with dependent setups in a manufacturing system," *Int. J. Adv. Manuf. Technol.*, vol. 38, no. 1–2, pp. 176–184, Jul. 2008.

[14] T. C. E. Cheng, Y. H. Chung, S. C. Liao, and W. C. Lee, "Two-agent singe-machine scheduling with release times to minimize the total weighted completion time," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 353–361, Jan. 2013.

[15] M. C. Vélez-Gallego, J. Maya, and J. R. Montoya-Torres, "A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan," *Comput. Oper. Res.*, vol. 73, pp. 132–140, Sep. 2016.

[16] M. O. Adamu and A. O. Adewumi, "A survey of single machine scheduling to minimize weighted number of tardy jobs," *J. Ind. Manag. Optim.*, vol. 10, no. 1, pp. 219–241, Jan. 2014.

[17] M. Liu, S. J. Wang, C. B. Chu, and F. Chu, "An improved exact algorithm for single-machine scheduling to minimise the number of tardy jobs with periodic maintenance," *Int. J. Prod. Res.*, vol. 54, no. 12, pp. 3591–3602, Jun. 2016.

[18] Z. Y. Wang, C. M. Wei, and L. H. Sun, "Solution algorithms for the number of tardy jobs minimisation scheduling with a time-dependent learning effect," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3141–3148, Jan. 2017.

[19] H. T. Yuan, J. Bi, M. C. Zhou, Q. Liu, and A. C. Ammari, "Biobjective task scheduling for distributed green data centers," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 731–742, Apr. 2021.

[20] H. T. Yuan, H. Liu, J. Bi, and M. C. Zhou, "Revenue and energy cost-optimized biobjective task scheduling for green cloud data centers," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 817–830, Apr. 2021.

[21] X. W. Guo, S. X. Liu, M. C. Zhou, and G. D. Tian, "Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1091–1103, Jul. 2018.

[22] S. J. Wang, M. Liu, F. Chu, and C. B. Chu, "Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration," *J. Clean. Prod.*, vol. 137, pp. 1205–1215, Nov. 2016.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[24] Y. Hou, N. Q. Wu, M. C. Zhou, and Z. W. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.

[25] Q. F. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[26] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[27] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

[28] Q. Kang, X. Y. Song, M. C. Zhou, and L. Li, "A collaborative resource allocation strategy for decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 49, no. 12, pp. 2416–2423, Dec. 2019.

[29] E. Ahmadi, M. Zandieh, M. Farrokh, and S. M. Emami, "A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms," *Comput. Oper. Res.*, vol. 73, pp. 56–66, Sep. 2016.

[30] Y. Y. Han, D. W. Gong, X. Y. Sun, and Q. K. Pan, "An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 52, no. 8, pp. 2211–2231, 2014.

[31] X. Q. Wu and A. D. Che, "A memetic differential evolution algorithm for energy-efficient parallel machine scheduling," *Omega*, vol. 82, pp. 155–165, Jan. 2019.

[32] M. Villalobos-Cid, M. Dorn, R. Ligabue-Braun, and M. Inostroza-Ponta, "A memetic algorithm based on an NSGA-II scheme for phylogenetic tree inference," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 776–787, Oct. 2019.

[33] H. F. Wang, Y. P. Fu, M. Huang, G. Q. Huang, and J. W. Wang, "A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 113, pp. 185–194, Nov. 2017.

[34] W. J. Fan, J. Pei, X. B. Liu, P. M. Pardalos, and M. Kong, "Serial-batching group scheduling with release times and the combined effects of deterioration and truncated job-dependent learning," *J. Global Optim.*, vol. 71, no. 1, pp. 147–163, May 2018.

[35] J. Pei, B. Cheng, X. Liu, P. M. Pardalos, and M. Kong, "Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup time," *Ann. Oper. Res.*, vol. 272, no. 1-2, pp. 217–241, Jan. 2019.

[36] P. Szota, S. Mróz, H. Dyja, and A. Kawałek, "3D FEM modelling and experimental verification of the rolls wear during the bar rolling process," in *Proc. Mater. Sci. Forum*, vol. 706–709, pp. 1533–1538, Jan. 2012.

[37] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 2033–2049, Mar. 2007.

[38] A. Hussain, Y. S. Muhammad, M. N. Sajid, I. Hussain, A. M. Shoukry, and S. Gani, "Genetic algorithm for traveling salesman problem with modified cycle crossover operator," *Comput. Intell. Neurosci.*, vol. 2017, Article No. 7430125, 2017.

[39] E. Vallada, F. Villa, and L. Fanjul-Peyro, "Enriched metaheuristics for the resource constrained unrelated parallel machine scheduling problem," *Comput. Oper. Res.*, vol. 111, pp. 415–424, Nov. 2019.

[40] R. Ruiz, Q. K. Pan, and B. Naderi, "Iterated Greedy methods for the distributed permutation flowshop scheduling problem," *Omega*, vol. 83, pp. 213–222, Mar. 2019.

[41] R. Ruiz and T. Stützle, "An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 1143–1159, Jun. 2008.

[42] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[43] Y. P. Fu, M. C. Zhou, X. W. Guo, and L. Qi, "Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 12, pp. 5037–5048, Dec. 2020.

[44] X. W. Guo, M. C. Zhou, S. X. Liu, and L. Qi, "Multiresource-constrained selective disassembly with maximal profit and minimal energy consumption," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 804–816, Apr. 2021.

[45] F. Yue, S. J. Song, Y. L. Zhang, J. N. D. Gupta, and R. Chiong, "Robust single machine scheduling with uncertain release times for minimising the maximum waiting time," *Int. J. Prod. Res.*, vol. 56, no. 16, pp. 5576–5592, May 2018.

[46] K. Z. Gao, F. J. Yang, M. C. Zhou, Q. K. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019.

[47] Y. M. Wang, X. P. Li, R. Ruiz, and S. C. Sui, "An iterated greedy heuristic for mixed no-wait flowshop problems," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1553–1566, May 2018.

[48] X. W. Guo, S. X. Liu, M. C. Zhou, and G. D. Tian, "Disassembly sequence optimization for large-scale products with multiresource constraints using scatter search and Petri nets," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2435–2446, Nov. 2016.

[49] G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, and J. Tan, "Modeling and planning for dual-objective selective disassembly using and/or graph and discrete artificial bee colony," *IEEE Trans. Industrial Informatics*, vol. 15, no. 4, pp. 2456–2468, Apr. 2019.

[50] M. Chen, M. C. Zhou, X. W. Guo, X. S. Lu, J. C. Ji, and Z. Y. Zhao, "Grey wolf optimizer adapted for disassembly sequencing problems," in *Proc. IEEE 16th Int. Conf. Networking, Sensing and Control*, Banff, Canada, 2019, pp. 46–51.

[51] X. W. Guo, M. C. Zhou, S. X. Liu, and L. Qi, "Lexicographic multiobjective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3307–3317, Jul. 2020.

[52] W. Luo, M. C. Zhou, X. W. Guo, H. P. Wei, L. Qi, and Z. Y. Zhao, "Improved artificial bee colony algorithm for solving a single-objective sequence-dependent disassembly line balancing problem," in *Proc. IEEE Int. Conf. Networking, Sensing and Control*, Nanjing, China, 2020, pp. 1–6.

[53] F. J. Yang, K. Z. Gao, I. W. Simon, Y. T. Zhu, and R. Su, "Decomposition methods for manufacturing system scheduling: A survey," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 2, pp. 389–400, Mar. 2018.

[54] J. Zhao, S. X. Liu, M. C. Zhou, X. W. Guo, and L. Qi, "Modified cuckoo search algorithm to solve economic power dispatch optimization problems," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 4, pp. 794–806, Jul. 2018.

[55] K. Z. Gao, Z. G. Cao, L. Zhang, Z. H. Chen, Y. Y. Han, and Q. K. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 4, pp. 904–916, Jul. 2019.

[56] Q. H. Zhu, Y. Qiao, and N. Q. Wu, "Optimal integrated schedule of entire process of dual-blade multi-cluster tools from start-up to close-down," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 2, pp. 553–565, Mar. 2019.

[57] F. J. Yang, N. Q. Wu, Y. Qiao, and R. Su, "Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via petri nets," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 270–280, Jan. 2018.

[58] J. Li, X. H. Meng, and X. Dai, "Collision-free scheduling of multi-bridge machining systems: A colored traveling salesman problem-based approach," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 139–147, Jan. 2018.

[59] J. Q. Li, H. Y. Sang, Q. K. Pan, P. Y. Duan, and K. Z. Gao, "Solving multi-area environmental/ economic dispatch by pareto-based chemical-reaction optimization algorithm," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 5, pp. 1240–1250, Sep. 2019.

[60] W. Q. Xiong, C. R. Pan, Y. Qiao, N. Q. Wu, M. X. Chen, and P. Hsieh, "Reducing wafer delay time by robot idle time regulation for single-arm cluster tools," *IEEE Trans. Autom. Sci. Eng.*, 2020. DOI: 10.1109/TASE.2020.3014078.

[61] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Trans. Cybernetics*, vol. 47, no. 11, pp. 3658–3668, November 2017.

[62] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, and J. Li, "Application-Aware Dynamic Fine Grained Resource Provisioning for a Virtualized Cloud Data Center," *IEEE Trans. Autom. Science and Engineering*, vlo. 14, no. 2, pp. 1172–1183, Apr. 2017.

[63] H. Yuan, J. Bi and M. Zhou, "Spatial Task Scheduling for Cost Minimization in Distributed Green Cloud Data Centers," *IEEE Trans Autom. Science and Engineering*, vol. 16, no. 2, pp. 729–740, April 2019.

[64] Y. Fu, M. Zhou, X. Guo and L. Qi, "Scheduling Dual-Objective Stochastic Hybrid Flow Shop With Deteriorating Jobs via Bi-Population Evolutionary Algorithm," *IEEE Trans Systems, Man, and Cybernetics: Systems*, vol. 50, no. 12, pp. 5037–5048, Dec. 2020.

[65] Z. Cao, C. Lin, M. Zhou, and R. Huang, "Scheduling Semiconductor Testing Facility by Using Cuckoo Search Algorithm with Reinforcement Learning and Surrogate Modeling," *IEEE Trans. Autom. Science and Engineering*, vol. 16, no.22, pp. 825–837, Apr. 2019.

[66] P. Zhang and M. Zhou, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy," *IEEE Trans. Autom. Science and Engineering*, vol. 15, no. 2, pp. 772–783, Apr. 2018.

Ziyan Zhao (Student Member, IEEE) received the B.S. degrees in 2015 and the M.S. degrees in 2017 respectively, from the Department of Information Science and Engineering, Northeastern University, where he is working toward the Ph.D. degree. From 2018, he works as a joint Ph.D. candidate with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. His research interests include production planning and scheduling and intelligent optimization.

Shixin Liu (Member, IEEE) received the B.S. degree in mechanical engineering from Southwest Jiaotong University, in 1990, the M.S. degree and the Ph. D. degree in systems engineering from Northeastern University, in 1993, and 2000. He is currently a Professor of the College of Information Science and Engineering, Northeastern University. His research interests include intelligent optimization algorithm, project management, and the theory and method of planning and scheduling. He has over 100 publications including 1 book.

MengChu Zhou (Fellow, IEEE) received his B.S. degree in control engineering from Nanjing University of Science and Technology, in 1983, M.S. degree in automatic control from Beijing Institute of Technology, in 1986, and Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY in 1990. He joined New Jersey Institute of Technology (NJIT), Newark, NJ in 1990, and is now a Distinguished Professor of Electrical and Computer Engineering. His research interests include Petri nets, intelligent automation, Internet of Things, big data, web services, and intelligent transportation. He has over 900 publications including 12 books, 600+ journal papers (450+ in IEEE transactions), 29 patents and 29 book-chapters. He is the Founding Editor of IEEE Press Book Series on Systems Science and Engineering and Editor-in-Chief of *IEEE/CAA Journal of Automatica Sinica*. He is a recipient of Humboldt Research Award for US Senior Scientists from Alexander von Humboldt Foundation, Franklin V. Taylor Memorial Award and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society, and Excellence in Research Prize and Medal from New Jersey Institute of Technology. He is a life member of Chinese Association for Science and Technology-USA and served as its President in 1999. He is a Fellow of International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), Chinese Association of Automation (CAA) and National Academy of Inventors (NAI).

Abdullah Abusorrah (Senior Member, IEEE) is a Professor in the Department of Electrical and Computer Engineering at King Abdulaziz University. He is the head of the Center for Renewable Energy and Power Systems at King Abdulaziz University. His research interests include energy systems, smart grid and system analyses. He received the Ph.D. degree in electrical engineering from the University of Nottingham in United Kingdom in 2007.