IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 1, 2019, accepted June 20, 2019, date of publication June 26, 2019, date of current version July 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924998

Multi-Objective Energy-Efficient Interval Scheduling in Hybrid Flow Shop Using Imperialist Competitive Algorithm

RUI ZHOU, DEMING LEI, AND XINMIN ZHOU
School of Automation, Wuhan University of Technology, Wuhan 430070, China
Corresponding author: Xinmin Zhou (zhouxinmin2003@163.com)

This work was supported by the National Natural Science Foundation of China under 61573264.

ABSTRACT Energy-efficient hybrid flow shop scheduling problem has attracted much attention in deterministic case; however, uncertainty is seldom considered in previous works. In this paper, energy-efficient interval hybrid flow shop scheduling problem (EIHFSP) is investigated, and a new imperialist competitive algorithm with empire grouping (EGICA) is proposed to minimize total energy consumption and makespan simultaneously. Groups of empires are obtained by defining normalized cost and normalized total cost in interval case, constructing initial empires, and grouping all empires. Assimilation is implemented in a new way, and an adaptive revolution is adopted in each group. Two-phase imperialist competition is newly proposed, and an adaptive search of member from archive is adopted. A number of computational experiments are conducted. The computational results demonstrate that the EGICA has promising advantages on solving the EIHFSP.

INDEX TERMS Hybrid flow shop scheduling problem, imperialist competitive algorithm, energy-efficient scheduling, interval processing time.

I. INTRODUCTION

As a classical production scheduling problem, hybrid flow shop scheduling problem (HFSP) extensively exists in many real-life manufacturing industries such as electronics, paper, textile, petrochemical, airplane engine and semiconductor [1], [2]. Hybrid flow shop possesses some advantages including flexibility, the increasing capacities and avoidance of bottleneck because of the redundance of machines at some stages. In the past decades, a number of results on various HFSP such as multi-objective hybrid flow shop scheduling problem (MOHFSP) and energy-efficient hybrid flow shop scheduling problem (EHFSP) have been obtained.

MOHFSP has been considered extensively. Jungwattanakit *et al.* [3] proposed some heuristics and a genetic algorithm (GA) for the problem with unrelated machines, setup times and dual criteria. Naderi *et al.* [4] solved MOHFSP with sequence-dependent setup times, transportation times and two objectives using an improved simulated annealing. Mousavi *et al.* [5] presented a bi-objective local search algorithm with three phases. Rashidi *et al.* [6] proposed an improved hybrid parallel GA. Cho *et al.* [7] reported a parallel GA with four different versions of local search strategies for reentrant HFSP. Karimi *et al.* [8] developed a multi-phase GA for bi-objective hybrid flexible flow shop group scheduling problem. Tran and Ng [9] applied a hybrid water flow algorithm for MOHFSP with limited buffers. Various practical constraints such as preventive maintenance [10], family setup times [11] and assembly [12] are investigated on MOHFSP. Other meta-heuristics are also applied, which include tabu search [10], [13], colonial competitive algorithm (CCA [14]), neighborhood search [12], [15], shuffled frog-leaping algorithm [16] and firefly algorithm [17] etc.

EHFSP often can be treated as a special MOHFSP because of the inclusion of energy-related objective and has attracted much attention in recent years. Dai *et al.* [18] developed a genetic-simulated annealing algorithm to minimize makespan and total energy consumption. Luo *et al.* [19] presented a novel ant colony optimization for EHFSP with electricity consumption cost. Tang *et al.* [20] introduced an improved particle swarm optimization for energy-efficient dynamic scheduling in flexible flow shop. Lin *et al.* [21] proposed teaching-learning-based optimization (TLBO) for the

The associate editor coordinating the review of this manuscript and approving it for publication was Shih-Wei Lin.

integration of processing parameter optimization and EHFSP with the minimization of makespan and carbon footprint. Lei *et al.* [22] developed a novel TLBO to minimize total tardiness treated as key objective and total energy consumption. Li *et al.* [23] presented an energy-aware multi-objective optimization algorithm for HFSP with setup energy consumptions. Zeng *et al.* [24] applied a hybrid non-dominated sorting genetic algorithm-II (NSGA-II) for flexible flow shop scheduling with total electricity consumption and material wastage. Meng *et al.* [25] proposed an improved GA with a new energy-conscious decoding method.

In the previous works on MOHFSP and EHFSP, processing conditions and data are fixed and deterministic and there are very few considerations on uncertainty.Take EHFSP as an example, uncertainty is just investigated in a special EHFSP [26], which is fuzzy flow shop scheduling problem with total energy consumption and tardiness penalty; on the other hand, scheduling problems with uncertainty have been extensively discussed in the past decades [26]–[36]; however, energy-related objective is seldom adopted. Lei and Guo [27] presented a dynamical neighborhood search for minimizing interval carbon footprint and makespan in dual-resource constrained interval job shop scheduling. Wang *et al.* [28] proposed a non-dominated GA for batch scheduling with uncertainties, energy consumption and tardiness. Uncertainty always exists in the real-life manufacturing process, the obtained schedule may be valid if uncertainty is neglected in scheduling problems and energy consumption itself is uncertain, so it is necessary to focus on energy-efficient scheduling problem with uncertainties.

Generally, uncertainties are modeled by using fuzzy theory, stochastic theory and interval number theory. There are some advantages for the usage of interval number. The lower bound and upper bound of interval are only required to indicate uncertain processing conditions, decision-maker prefers using interval number to indicate his expected performance and the obtained interval results can be understood easily, so it is a good choice to apply interval number for uncertainty. In the past decade, there are some works related to interval scheduling in parallel machines [30], flow shop [37] and job shop [27], [38], [39].

As stated above, meta-heuristics such as GA and TLBO have been applied to solve EHFSP; however, as an algorithm inspired by the sociopolitical behaviors, imperialist competitive algorithm (ICA) [40] is seldom used to deal with EHFSP. ICA possesses some new features such as good neighborhood search ability, effective global search property and good convergence rate [41] and also has the extensive applications to many production scheduling problems in single machine [42], parallel machines [43], flow shop [44], [45], job shop [46], [47] and open shop [48] etc, so it is necessary to investigate the advantages of ICA on solving EHFSP.

In this study, we investigate energy-efficient interval hybrid flow shop scheduling problem (EIHFSP) with interval processing time. A novel imperialist competitive algorithm with empire grouping (EGICA) is proposed to minimize interval total energy consumption and interval makespan. In EGICA, normalized cost and normalized total cost are defined in interval case, initial empires are constructed and all empires are divided into several groups, then each group is evolved independently, in which assimilation of colony can be done by moving toward imperialist of other empire, an adaptive revolution is adopted and an adaptive search of member of archive is added into a newly defined two-phase imperialist competition. Many computational experiments are conducted. The computational results demonstrate that the new strategies of EGICA are effective and EGICA has promising advantages on solving EIHFSP.

The remainder of the paper is organized as follows. Operators for interval scheduling in Section II and EIHFSP is described in Section III. The detailed steps of EGICA for EIHFSP are shown in Section IV. The computational experiments are depicted in Section V and the conclusions are concluded in the last Section. We also discuss the future research topics in the final Section.

TABLE 1. Notations and descriptions.

Notation	description
$A^L(A^R)$	the left (right) limit of interval A
$\bar{A}$	middle value of interval A, $\bar{A} = \left(A^L + A^R\right)/2$
$P(A \leq B)$	possibility degree of $A \leq B$
S_k	the set of unrelated machines at stage k, $\lvert S_k \rvert \geq 1$
M_{kj}	the jth machine of stage k
V	the set of speeds, $V = \{v_1, v_2, \cdots, v_d\}$
η_{ikj}	interval processing requirement of job J_i on M_{kj}
$\eta_{ikj}^{L(R)}$	the left (right) limit of η_{ikj}
p_{ikjl}	interval processing time
E_{kjl}	energy consumption per unit time when M_{kj} runs at speed v_l
SE_{kj}	energy consumption per unit idle time of M_{kj}
C_i	interval completion time of job J_i
C_{max}	interval makespan
TEC	interval total energy consumption
y_{ikjl}	if job J_i is processed on machine M_{kj} at speed v_l, then y_{ikjl} is 1; otherwise y_{ikjl} is 0
I_{kj}	the sum of all idle periods of machine M_{kj}
N_{im}	the number of imperialists
N_{col}	the number of colonies, $N_{col} = N - N_{im}$
NC_k	the number of colonies possessed by imperialist k
TC_k	total cost of empire k
$\overline{TC}_k$	normalized total cost of empire k

II. OPERATORS FOR INTERVAL SCHEDULING

An interval number $A = \left[A^L, A^R\right]$ represents a bounded set of real numbers between A^L and A^R.

For interval scheduling problem, interval number is used to indicate the range of processing time or completion time of job.Lei [38], [39] defined three operators to build interval schedule.

For two intervals $A = \left[A^L, A^R\right]$ and $B = \left[B^L, B^R\right]$, operators $A + B$ and $A \vee B$ are defined by

$$A + B = \left[A^L + B^L, A^R + B^R\right] \tag{1}$$

When A is the beginning time of a job and B denotes the interval processing time of the job, then $A + B$ is the completion time of the job, so addition operator is used to calculate interval completion time.

$$A \vee B = \left[\max\left\{A^L, B^L\right\}, \max\left\{A^R, B^R\right\}\right] \tag{2}$$

where $A \vee B$ indicates the max operator of A and B.

Max operator is applied to compute the beginning time of job when interval schedule is built.

Interval numbers are often compared according to possibility degree, which represents certain degree that one interval number is larger or smaller than another. Jiang et al. [49] provided a possibility degree-method given by

$$P(A \le B) = \begin{cases} 0 & A^L \ge B^R \\ 0.5 \cdot \dfrac{B^R - A^L}{A^R - A^L} \cdot \dfrac{B^R - A^L}{B^R - B^L} & B^L \le A^L < B^R \le A^R \\ \dfrac{B^L - A^L}{A^R - A^L} + 0.5 \cdot \dfrac{B^R - B^L}{A^R - A^L} & A^L < B^L < B^R \le A^R \\ \dfrac{B^L - A^L}{A^R - A^L} + \dfrac{A^R - B^L}{A^R - A^L} \cdot \dfrac{B^R - A^R}{B^R - B^L} & \\ +0.5 \cdot \dfrac{A^R - B^L}{A^R - A^L} \cdot \dfrac{A^R - B^L}{B^R - B^L} & A^L < B^L \le A^R < B^R \\ \dfrac{B^R - A^R}{B^R - B^L} + 0.5 \cdot \dfrac{A^R - A^L}{B^R - B^L} & B^L \le A^L < A^R < B^R \\ 1 & A^R \le B^L \end{cases} \tag{3}$$

Lei [39] defined the following relations based the above formula.

1. $A <_{pd} B$ if $P(A \le B) > 0.5$ or $P(B \le A) < 0.5$.
2. $A >_{pd} B$ if $P(A \le B) < 0.5$ or $P(B \le A) > 0.5$.
3. $A =_{pd} B$ if $P(A \le B) = 0.5$.

where $A < (>)_{pd} B$ indicates that A is less (greater)than B according to possibility degree, $A =_{pd} B$ denotes that A is equal to B.

Ranking operator is used to compare result of interval scheduling.

III. PROBLEM DESCRIPTION

EIHFSP is described as follows. There are n jobs $J_1, J_2, \cdots, J_n$ and m stages, each of which consists of some unrelated parallel machines. There is a set V of d different processing speeds for each machine. Each job J_i is processed in terms of the same production flow: stage 1, stage 2, $\cdots$, stage m, as shown in Figure 1.

When a job is processed at a stage, its processing must be executed on an assigned machine at a selected speed. The speed of a machine cannot be changed during the execution of a job.

Unlike EHFSP, EIHFSP has the interval processing requirement η_{ikj} of job J_i on M_{kj}, $\eta_{ikj} = \left[\eta_{ikj}^L, \eta_{ikj}^R\right]$, so when job J_i is processed on a machine $M_{kj} \in S_k$ at speed v_l, the interval processing time p_{ikjl} is defined as η_{ikj}/v_l.

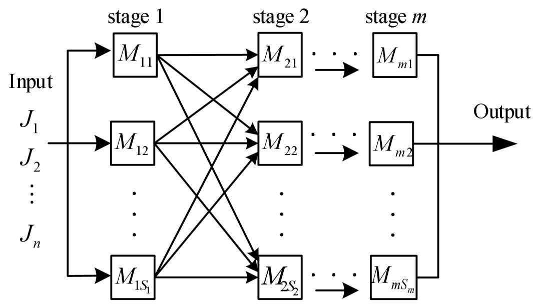

FIGURE 1. Schematic diagram of EIHFSP.

The processing of a job can skip some stages; however, it must be processed at one stage at least.

It is often assumed that energy consumption increases and processing time decreases when a job is processed on a machine at a higher speed [50]. Lei *et al.* [22] gave the detailed description on this assumption for EHFSP. This assumption is also adopted in EIHFSP.

The constraints of EIHFSP are the following. All machines and jobs are available from time zero. Each job can only be processed on one machine at a time. Each machine cannot process more than one job at a time. Preemption is not allowed and buffer size is not limited etc.

EIHFSP is composed of three sub-problems: speed selection which decides an appropriate processing speed of machine for each job; machine assignment which selects a parallel machine at each stage for job; and scheduling.

The goal of EIHFSP is to minimize simultaneously the following two objectives under the condition that constraints are all met and sub-problems are solved.

$$f_1 = C_{\max} = C_1 \vee C_2 \vee \ldots \vee C_n \tag{4}$$

$$f_2 = TEC = \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{j=1}^{S_k} \sum_{l=1}^{d} E_{kjl} p_{ikjl} y_{ikjl} + \sum_{k=1}^{m} \sum_{j=1}^{S_k} SE_{kj} I_{kj} \tag{5}$$

where $C_{\max} = \left[C_{\max}^L, C_{\max}^R\right]$, $C_{\max}^L$ and $C_{\max}^R$ indicate the left and right limit of makespan.

Max operation of intervals is also used to compute interval makespan. A machine M_{kj} may runs at speed v_l or be idle at any time. To calculate TEC, let $TEC = [0, 0]$, then for each machine M_{kj}, decide all jobs processed on this machine at the corresponding speeds and all idle periods, start with $[0, 0]$, compute $TEC = TEC + E_{kjl} \times p_{ikjl}$ for each job J_i processed on this machine at speed v_l, and calculate $TEC = TEC + SE_{kj} \times I_{kj}$ for all idle periods of M_{kj}.

Table 2 shows an illustrative example of EIHFSP. There are six jobs, two stages, three machines at stage 1 and two machines at stage 2. $v_1 = 1.0$, $v_2 = 1.5$ and $v_3 = 2.0$, $E_{1jl} = 4 \times v_l^2$, $E_{2jl} = 3 \times v_l^2$, $SE_{1j} = 1$, $SE_{2j} = 1.5$. Interval in Table 2 is processing requirement of job on its

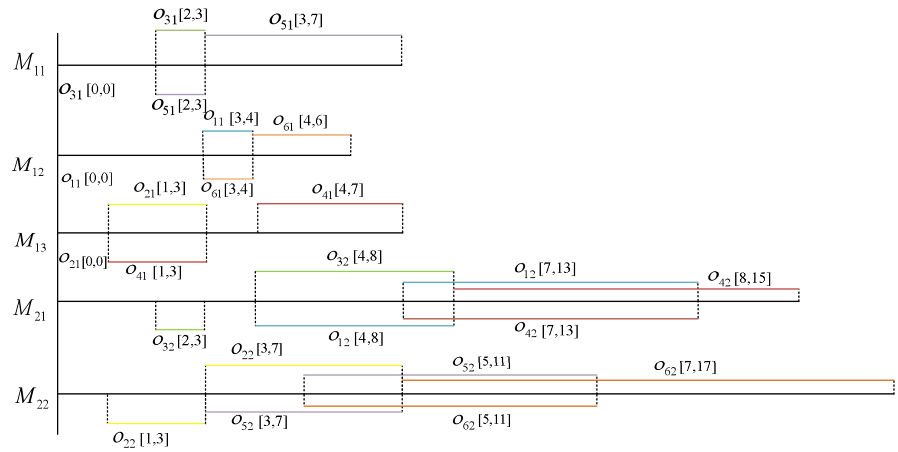

FIGURE 2. A schedule of the example.

TABLE 2. An illustrative example of the problem.

Job	Stage 1			Stage 2	
	M_{11}	M_{12}	M_{13}	M_{21}	M_{22}
J_1	$[3,6]$	$[4.5,6]\ (v_2)$	$[2,5]$	$[3,5]\ (v_1)$	$[1,2]$
J_2	$[3,5]$	$[1,4]$	$[1,3]\ (v_1)$	$[4,7]$	$[3,6]\ (v_2)$
J_3	$[4,6]\ (v_3)$	$[3,7]$	$[2.5,5]$	$[2,5]\ (v_1)$	$[1,4]$
J_4	$[5,7]$	$[3,4.5]$	$[4.5,6]\ (v_2)$	$[2,4]\ (v_3)$	$[5,7]$
J_5	$[1,4]\ (v_1)$	$[4,5]$	$[5,7]$	$[1,5]$	$[2,4]\ (v_1)$
J_6	$[3,6]$	$[2,4]\ (v_3)$	$[2,4]$	$[3,4]$	$[2,6]\ (v_1)$

processing machine. For example, job J_1 is processed on M_{12} at speed v_2, its interval processing time p_{1122} is [3], [4], that is, the actual value of p_{1122} may any ones in [3], [4] and is not fixed.

Table 2 also gives the velocities in parentheses for the chosen machines of jobs at each stage and Figure 2 describes a schedule of the example. In Figure 2, segment under the line is the beginning time and the segment above the line represents the completion time for an operation. This kind of graphical description is proposed by Lei [39]. On each segment, operation and its beginning time or completion time are listed, for example, $o_{22}[1, 3]$ indicates the beginning time [1,3] of o_{22}. o_{ik} indicates the operation of job J_i at stage k.

For machine M_{21}, three jobs are processed sequentially, for operation o_{32}, its last operation o_{31} is completed at [2, 3], the earliest available time of M_{21} is [0,0], so the beginning time of o_{32} is [2,3] ($= [2, 3] \vee [0, 0]$) and the completion time of o_{32} is obtained by addition operation of $[2, 3]+[2, 5]$. To calculate makespan, the completion times of J_4 and J_6 are [8,15] and [7,17] and makespan is obtained by max operator of two completion times, so interval makespan is $f_1 = [8, 17]$. For machine M_{21}, one idle period exists, the length of period is [2,3], the interval energy consumption on idle period is [3,4.5], energy consumptions for processing three jobs are [6,15], [9,15] and [12,24], respectively and total energy of M_{21} is [30, 58.5]. f_2 is the sum of energy consumption of all machines. $f_2 = [167, 300]$.

For the multi-objective optimization problem with the minimization of f_1, f_2, Pareto dominance is often used. For solutions x and y, if $f_i(x) \leq f_i(y)$ for $i = 1, 2$ and $f_i(x) < f_i(y)$ for $i = 1$ or 2, then $x \succ y$, which means that x dominates y or y is dominated by x. For EIHFSP, ranking operator is used to decide the domination relation between solutions.

IV. EGICA FOR EIHFSP

In this section, the basic principle of ICA is first introduced and then EGICA is applied to solve EIHFSP.

A. INTRODUCTION TO ICA

In ICA, a country represents a solution of the problem and solutions in population P are categorized into two parts: imperialists and colonies, the former are some best solutions in P and the latter are all solutions of P except imperialists. The search of ICA starts with initial empires, then empires are often evolved independently by assimilation and revolution, and imperialist competition is done among all empires. The detailed steps are shown in Algorithm 1. r_k is random number following uniform distribution in [0, 1].

With respect to cost, the smaller the cost of a solution is, the better the solution is. $\bar{c}_k$, NC_k, TC_k, $\overline{TC}_k$ and PE_k are defined by Hosseini and Khaled [41].

B. DESCRIPTIONS ON EGICA

In the previous works on ICAs [26]–[36], empires are evolved independently and empire grouping is seldom considered. If all empires are divided into several groups, more relations can be occur among empires and high diversity of population can be kept, for example, assimilation of colony can be done by moving toward imperialist of other empire; on the other hand, two-phase imperialist competition is introduced to avoid falling local optima, in which empires of each group first compete each other and then competition among groups

Algorithm 1 ICA

Randomly produce an initial population P and calculate the cost of each solution in P.
Choose N_{im} solutions with smallest cost as imperialists, calculate the normalized cost $\bar{c}_k$ and NC_k and randomly allocate NC_k colonies for each imperialist k.
while termination condition is not met **do**
  Assimilation. In each empire, each colony moves toward its imperialist and is replaced with the newly generated solution if possible.
  Revolution. Perform revolution according to revolution probability U_R.
  Exchange. In each empire, compare each colony with its imperialist and replace the imperialist with the colony with smaller cost than its imperialist.
  Imperialist competition. Calculate TC_k, $\overline{TC}_k$ and power PE_k for each empire k, construct the vector $\left[PE_1 - r_1, PE_2 - r_2, \cdots, PE_{N_{im}} - r_{N_{im}}\right]$, decide an empire g with the biggest $PE_g - r_g$ and allocate the weakest colony of the weakest empire into empire g.
end while

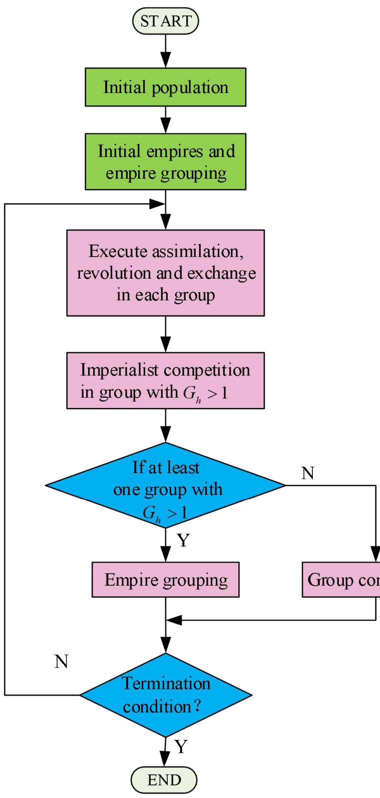

FIGURE 3. Flow chart of EGICA.

is done only when each group has one empire. These features are hardly adopted in ICA, so EGICA is constructed based on the above features.

The main steps of EGICA are shown in Algorithm 2, where s is the number of groups and G_h indicates the number of empires in group h. The termination condition is max_it. When a new solution is generated, $t = t + 1$. Figure 3 gives the flow chart of EGICA.

Algorithm 2 EGICA

Randomly produce an initial population P and construct initial archive Ω, $t = 1$.
Initial empires and empires grouping.
while termination condition is not met **do**
  for $h = 1$ to s **do**
    Execute new assimilation, revolution and exchange colony and its imperialist if possible in empires of group h.
    if $G_h > 1$ **then**
      Perform imperialist competition in group h.
    end if
  end for
  if at least one group with at least two empires **then**
    Divide all empires into s groups again.
  else
    Execute competition among groups.
  end if
end while

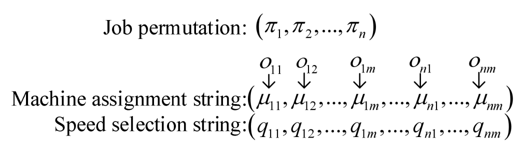

FIGURE 4. Encoding of EIHFSP.

C. ENCODING AND DECODING OF EIHFSP

EIHFSP is composed of scheduling, machine assignment and speed selection, so a three-string representation of EHFSP [22] is directly used, which can be directly applied to represent the solution of EIHFSP. The decoding procedure of EHFSP is also utilized to build schedule except that interval C_{max} and interval TEC are obtained according to interval processing time.

For EIHFSP with n jobs, m stages and d operation speeds for each machine, Figure 4 shows job permutation, machine assignment string and speed selection string. In these strings, $\pi_i \in \{1, 2, \ldots, n\}$, $\mu_{ik} \in S_k$ is the unrelated machine at stage k assigned to process job J_i, q_{ik} denotes the speed of machine $\mu_{ik} \in S_k$, $q_{ik} \in V$. The decoding procedure for each solution is shown in Algorithm 3.

Algorithm 3 Decoding Procedure

for $i = 1$ to n **do**
  for $k = 1$ to m **do**
    Job J_{π_i} is processed on machine $\mu_{\pi_i k}$ at speed $q_{\pi_i k}$
  end for
end for

For the example in Table 2, a possible solution consists of (3, 1, 2, 5, 6, 4), $(M_{12}, M_{21}, M_{13}, M_{22}, M_{11}, M_{21}, M_{13}, M_{21}, M_{11}, M_{22}, M_{12}, M_{22})$, $(v_2, v_1, v_1, v_2, v_3, v_1, v_2, v_3, v_1, v_1, v_3, v_1)$, and the corresponding schedule is given in Figure 2. Three strings are separate in the optimization procedure of EGICA, global search and neighborhood search are used independently to each string.

D. INITIAL EMPIRES AND EMPIRE GROUPING

Initial population P with N solutions are randomly generated, then normalized cost $\bar{c}_i$ for solution i is calculated by

$$\bar{c}_i = \max_{l \in P}\{rank_l\} - rank_i + 0.1 \times crowd_i \tag{6}$$

where $rank_i$ represents rank value of solution i obtained by non-dominated sorting [51] and $crowd_i$ is the crowding distance calculated using middle value of interval objectives f_1 and f_2.

Suppose that H_l is the set of solutions with rank l. When $|H_l| > 2$, for $j = 1, 2$, all solutions are first sorted in the ascending order of the middle value of the jth objective, $\tilde{f}_j^{g_j}$ is the $g_j th$ value in the obtained order, and for the $g_j th$ $(1 < g_j < |H_l|)$, if $\bar{f}_{i,j} = \tilde{f}_j^{g_j}$, then $crowd_i$ for solution i is computed by

$$crowd_i = \sum\nolimits_{j=1}^{2} \frac{\tilde{f}_j^{g_j+1} - \tilde{f}_j^{g_j-1}}{\bar{f}_j^{\max} - \bar{f}_j^{\min}} \tag{7}$$

where $\bar{f}_{i,j}$ is the middle value of f_j of solution i, $\bar{f}_j^{\max(\min)} = \max(\min)_{i \in H_l}\{\bar{f}_{i,j}\}$.

Obviously, $crowd_i$ is in [0,2] for solution i with $1 < g_j < |H_l|$.

For solution i with $g_j = |H_l|$ or 1, $crowd_i \in [2 \times crowd_{max}, 3 \times crowd_{max}]$, where $crowd_{max}$ denotes the maximum value of $crowd_i$ for all solutions with $1 < g_i < |H_l|$.

When solution i has g_j of 1 or $|H_l|$, it has the biggest or smallest $\bar{f}_{i,j}$ and is assigned high crowding distance as done by Deb et al. [51].

The above new definition on $\bar{c}_i$ can guarantee that each imperialist can be allocated the reasonable amount of colonies.

Algorithm 4 shows the detailed steps for the construction of initial empires and the grouping of empires, in which all empires are divided into s groups. POW_k and NC_k are defined by

$$POW_k = \bar{c}_k \Big/ \sum\nolimits_{j \in Q} \bar{c}_j \tag{8}$$

$$NC_k = round\,(POW_k \times N_{col}) \tag{9}$$

where $round(x)$ is an integer not exceeding x and being closest to x and Q is the set of all imperialists.

A new definition on $\overline{TC}_k$ is given, in which the coefficient $\frac{2N_{im}}{N-N_{im}}$ is not fixed.

$$\overline{TC}_k = \bar{c}_k + \frac{2N_{im}}{N - N_{im}} \sum\nolimits_{j \in \Theta_k} \bar{c}_j / NC_k \tag{10}$$

where Θ_k is the set of colonies in empire k.

Algorithm 4 Initial Empires and Initial Groups

1: Perform non-dominated sorting for all solutions in P.
2: Sort all solutions in each H_l in the ascending order of $\bar{f}_{i,j}$ and then calculate normalized cost for each solution.
3: Choose N_{im} solutions with the biggest normalized cost from P as imperialists and other solutions as colonies.
4: Compute the power POW_k of each imperialist k and the number NC_k of colonies possessed by imperialist k.
5: Randomly allocate NC_k colonies for each imperialist k.
6: Calculate the normalized total cost $\overline{TC}_k$.
7: Sort all empires in the descending order of $\overline{TC}_k$, suppose that $\overline{TC}_1 \geq \overline{TC}_2 \geq \cdots \geq \overline{TC}_{N_{im}}$.
8: Assign empire 1 to group 1, empire 2 to group 2, empire s to group s, empire $s+1$ to group 1 and so on.

In EGICA, normalized cost based on interval objectives is used to decide imperialists and colonies and all empires are divided into s groups, these two steps seldom exist in the previous ICAs [27]–[36].

E. ASSIMILATION AND REVOLUTION IN EACH GROUP

Assimilation and revolution are the main operators for producing new solutions. Assimilation is often done by moving colony toward its imperialist and colony seldom learns from imperialist of other empires. Revolution is often implemented using the same way as mutation of GA and a fixed revolution probability U_R is used to choose colonies. In this study, a new assimilation based on the number and $\overline{TC}_k$ of empires is executed in groups $1, 2, \cdots, s$, respectively, and an adaptive revolution probability is adopted.

Algorithm 5 shows the steps of assimilation and revolution in a group, where θ is a real number, β_1 is the number of neighborhood search for each chosen colony and random number α follows uniform distribution on [0, 1]. Suppose that empires $1, 2, \cdots, G_h$ are allocated into group h. $\overline{TC}_1 \geq \overline{TC}_2 \geq \cdots \geq \overline{TC}_{G_h}$.

Three global search operators between colony λ and a chosen imperialist are used [22], which are described as follows: for two solutions, if a random number $\alpha < \beta_2$, then operator on scheduling is executed; otherwise, two operators of other two strings are selected in the same probability and applied, where β_2 is a real number. Both β_2 and θ are set to be 0.7 based on experiments.

An adaptive revolution probability is defined by

$$U_R = U_0 \times e^{\left(\frac{t}{max\_it} - 1\right)} \tag{11}$$

where U_0 is set to be 0.35, t and max_it are the current number and maximum number of objective function evaluations.

Obviously, U_R increases with t. In the early stage, t is small, population P is not evolved well and exploration is the main focus of search. With the increasing of t, solution quality of P improves continuously and exploitation ability should be intensified to make good balance between exploration and exploitation, so a continuously increasing U_R is presented.

Algorithm 5 Assimilation and Revolution in Group h

```
Assimilation is done between each colony of empire 1 and its imperialist by global search operators.
if G_h ≥ 3 then
    for k = 2 to G_h do
        For each colony λ ∈ Θ_k, if a random number α < (k − 1)/G_h, then perform global search between λ
        and imperialist of empire 1; otherwise, apply global search between λ and its imperialist,
        produce a new solution z and update λ and archive Ω.
    end for
end if
if G_h = 2 then
    For each colony λ ∈ Θ_2, if a random number α < θ, then perform global search between λ and its
    imperialist; otherwise, apply global search between λ and imperialist of empire 1, produce a new
    solution z and update λ and Ω.
end if
for k = 1 to G_h do
    for each colony λ ∈ Θ_k do
        if A random number α < U_R then
            g = 1
            for j = 1 to β_1 do
                Apply N_g on λ, obtain a new solution z, compare z with λ and update λ and Ω.
                g = g + 1, let g = 1 if g = 5.
            end for
        end if
    end for
end for
```

Four neighborhood structures $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4$ are applied. The first one generates solutions by exchanging two randomly chosen jobs in scheduling string and the next three are *insert*, *change* and *speed* [22], respectively.

When a new solution z is compared with λ, the following conditions are tested: if solution z dominates λ or is non-dominated with λ, then replace λ with z.

External archive Ω is used to store non-dominated solutions generated by EGICA. Ω is updated in the following way. Solution z is added into Ω, all members of Ω are compared each other and the dominated ones are removed from Ω.

F. TWO-PHASE IMPERIALIST COMPETITION

In general, all empires compete each other after assimilation and revolution and the weakest colony of the weakest empire is directly added into the winning empire. In this study, two-phase imperialist competition is proposed, in which empires in each group compete each other and then groups compete each other when each group has only one empire. Algorithm 6 gives imperialist competition in each group h, where ξ is a real number and set to be 0.6 based on experiments.

Algorithm 6 First Phase of Imperialist Competition in Group h

```
for k = 1 to G_h do
    Compute normalized cost c̄_i, normalized total cost TC̄_k and power PE_k of empire k.
end for
Let power of empire as selection probability, apply roulette selection to choose a winning empire k_1
and choose an empire k_2 ≠ k_1 with smallest TC̄_{k_2}.
Randomly select a solution x ∈ Ω and directly add into empire k_1, delete the weakest colony from empire k_2.
if t ≤ ξ × max_it then
    Imperialists of empires k_1 and k_2 are chosen in the same probability, global search between x and
    the chosen imperialist is done once, a new solution z is obtained and decide if x and Ω are updated
    as done in Algorithm 5.
else
    N_2, N_3, N_4 acts on x respectively, when a new solution z is obtained, decide if x and Ω are renewed.
end if
```

As shown in Algorithm 6, a chosen member of Ω directly substitutes for the weakest colony and then an adaptive search is performed on the chosen member. The choosing of x from external archive Ω and the application of adaptive search are to keep high diversity in winning empire; moreover, no probability vector is constructed and roulette selection is executed, the competition of each group is independently, so there are at least s empires in most of search procedure and competition can be done fully.

When each group has only one empire, group competition is done in the same way of Algorithm 6 except that there are s empires, not G_h empires.

G. FEATURES ON EGICA

As shown above, EGICA has some different features from the existing ICAs [26]–[36]. (1) After population is divided into N_{im} empires, these empires are allocated into s groups. (2) Assimilation is implemented differently in different cases and groups to intensify the exploration ability and an adaptive revolution is adopted in each group to obtain a good exploitation.(3) Two-phase imperialist competition is adopted, which is first done independently in each group and then executed among groups when each group has only one empire.

In general, these features are beneficial to keep high diversity of population and avoid falling local optima, thus, EGICA is an effective method for EIHFSP.

V. COMPUTATIONAL EXPERIMENTS AND RESULTS

Extensive experiments are conducted on a set of problems to test the performance of EGICA for EIHFSP. All experiments are implemented by using Microsoft Visual C++ 2015 and run on 4.0G RAM 2.00GHz CPU PC.

A. INSTANCES, METRICS AND COMPARATIVE ALGORITHMS

44 instances are used, which are the combinations of $n = 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120$ and $m = 2, 4, 6, 8$. The detailed descriptions on their data except η_{ikj} are shown in paper [22]. For η_{ikj}, $\eta^L_{ikj} \in [1, 5]$ and $\eta^R_{ikj} = \eta^L_{ikj} + \varepsilon$, $\varepsilon \in [1, 4]$. η^L_{ikj} and ε are real number.

Metric $\mathcal{C}$ [52] is applied to compare the approximate Pareto optimal set respectively obtained by algorithms. $\mathcal{C}(L, B)$ measures the fraction of members of B that are dominated by members of L.

$$\mathcal{C}(L, B) = \frac{|\{b \in B : \exists h \in L, h \succ b\}|}{|B|} \tag{12}$$

Metric ρ [53] indicates the ratio of number of the elements in the set $\{x \in \Omega_l \,|x \in \Omega^*\}$ to $|\Omega^*|$, where The reference set Ω^* is composed of the non-dominated solutions in $\bigcup_l \Omega_l$.

The existing metrics for multi-objective optimization are difficult to be used directly in uncertain case. $\mathcal{C}$ and ρ are often applied for multi-objective scheduling algorithms and can be directly utilized in the interval case; moreover, they can be used to reveal the even distribution of non-dominated solutions and the distribution range of solutions and evaluate convergence, for example, $\rho = 1$ means that all non-dominated solutions of an algorithm belongs to the reference set, that is, the algorithm has better convergence than other algorithms.

EIHFSP is seldom considered and it is difficult to find comparative algorithms. In this study, we choose a CCA [14] for MOHFSP, the classical multiobjective optimization algorithm named non-dominated sorting genetic algorithm-II (NSGA-II) and multi-objective tabu search method (MOTS) [10] that obtains better results for HFSP with preventive maintenance as comparative algorithms.

Karimi and Davoudpour [14] proposed a multi-objective CCA for HFSP with the minimization of makespan and total weighted tardiness. This CCA has good performance in solving MOHFSP and can be directly applied to solve EIHFSP after a speed selection string and neighborhood structure *speed* are added.

To apply MOTS to EIHFSP, steps on maintenance are deleted and a greedy rule in the original MOTS is still applied to allocate machines for each job. Speed selection string and *speed* are also adopted.

We revised NSGA-II for EIHFSP in the following way: crossovers between two individuals are also executed in terms of global search in Algorithm 5, one of four neighborhood structures is randomly chosen as mutation operator when mutation is for an individual.

The main parameters of EGICA are listed below: N, N_{im}, s, β_1, max_it. Taguchi method is used to decide the settings for parameters. The levels of each parameter are shown in Table 3. Table 4 gives the orthogonal array $L_{27}(3^5)$. EGICA with each combination runs 20 times for 90×6. We calculate ρ and the results of ρ and S/N ratio based on ρ are shown in Figure 5. S/N ratio is defined as $-10\log_{10}\left[(1+\epsilon-\rho)^2\right]$ where ϵ is a small number and set to be 0.001.

TABLE 3. Parameters and their levels.

Parameter	Factor level 1	2	3
N	70	80	90
N_{im}	5	6	7
s	1	2	3
β_1	7	8	9
max_it	90000	100000	110000

TABLE 4. The orthogonal array $L_{27}(3^5)$.

test	Factor N	N_{im}	s	β_1	max_it
1	1	1	1	1	1
2	1	1	1	1	2
3	1	1	1	1	3
4	1	2	2	2	1
5	1	2	2	2	2
6	1	2	2	2	3
7	1	3	3	3	1
8	1	3	3	3	2
9	1	3	3	3	3
10	2	1	2	3	1
11	2	1	2	3	2
12	2	1	2	3	3
13	2	2	3	1	1
14	2	2	3	1	2
15	2	2	3	1	3
16	2	3	1	2	1
17	2	3	1	2	2
18	2	3	1	2	3
19	3	1	3	2	1
20	3	1	3	2	2
21	3	1	3	2	3
22	3	2	1	3	1
23	3	2	1	3	2
24	3	2	1	3	3
25	3	3	2	1	1
26	3	3	2	1	2
27	3	3	2	1	3

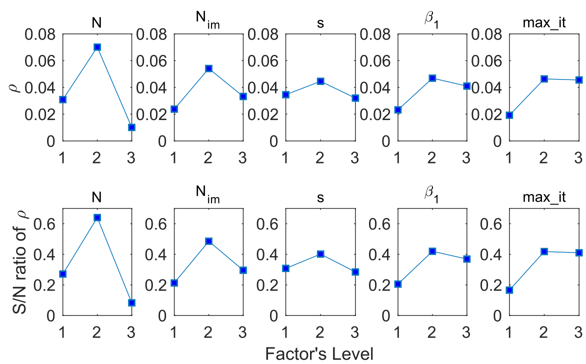

FIGURE 5. The mean ρ and the mean S/N ratio of ρ.

As shown in Figure 5, the best settings are $N = 80$, $N_{im} = 6$, $s = 2$, $\beta_1 = 8$, $max\_it = 10^5$.

The parameters of CCA are $N = 90$, $N_{im} = 9$, $max\_it = 10^5$.

For MOTS, the size of neighborhood solutions is set to be 350 and max_it is set to be 10^5 for all instances.

For NSGA-II, population scale $N = 100$, crossover probability $P_c = 0.8$, mutation probability $P_m = 0.1$ and maximum generation of $max\_it/100$.

TABLE 5. Computational results of EGICA and other algorithms on metric ρ.

Instance	EGICA	EGICA1	EGICA2	NSGA-II	CCA	MOTS
20×2	0.000	0.169	0.000	0.108	0.538	0.185
20×4	0.238	0.000	0.010	0.000	0.455	0.297
20×6	0.000	0.213	0.000	0.000	0.404	0.383
20×8	**0.565**	0.000	0.000	0.000	0.000	0.435
30×2	0.196	0.000	0.040	0.000	0.303	0.461
30×4	0.305	0.000	0.000	0.000	0.457	0.238
30×6	0.000	0.000	0.000	0.000	0.000	1.000
30×8	**0.522**	0.000	0.000	0.000	0.000	0.478
40×2	**0.508**	0.000	0.045	0.000	0.000	0.447
40×4	**0.639**	0.000	0.047	0.000	0.000	0.314
40×6	**0.662**	0.000	0.000	0.059	0.114	0.165
40×8	0.000	0.000	0.000	0.000	1.000	0.000
50×2	**0.544**	0.000	0.000	0.000	0.367	0.089
50×4	**0.967**	0.000	0.000	0.000	0.000	0.033
50×6	0.094	0.000	0.000	0.000	0.906	0.000
50×8	**0.575**	0.000	0.000	0.000	0.000	0.425
60×2	**0.789**	0.000	0.000	0.000	0.211	0.000
60×4	**0.806**	0.000	0.000	0.032	0.162	0.000
60×6	**0.782**	0.000	0.000	0.000	0.000	0.218
60×8	**0.615**	0.000	0.140	0.000	0.000	0.245
70×2	0.000	0.000	0.486	0.000	0.514	0.000
70×4	**1.000**	0.000	0.000	0.000	0.000	0.000
70×6	**1.000**	0.000	0.000	0.000	0.000	0.000
70×8	**1.000**	0.000	0.000	0.000	0.000	0.000
80×2	0.000	0.000	0.131	0.000	0.479	0.390
80×4	**0.967**	0.000	0.000	0.033	0.000	0.000
80×6	**1.000**	0.000	0.000	0.000	0.000	0.000
80×8	**1.000**	0.000	0.000	0.000	0.000	0.000
90×2	**0.567**	0.000	0.000	0.000	0.238	0.205
90×4	**1.000**	0.000	0.000	0.000	0.000	0.000
90×6	**1.000**	0.000	0.000	0.000	0.000	0.000
90×8	**1.000**	0.000	0.000	0.000	0.000	0.000
100×2	0.231	0.000	0.126	0.000	0.476	0.167
100×4	**1.000**	0.000	0.000	0.000	0.000	0.000
100×6	**1.000**	0.000	0.000	0.000	0.000	0.000
100×8	**1.000**	0.000	0.000	0.000	0.000	0.000
110×2	0.483	0.000	0.000	0.000	0.000	0.517
110×4	**1.000**	0.000	0.000	0.000	0.000	0.000
110×6	**1.000**	0.000	0.000	0.000	0.000	0.000
110×8	**1.000**	0.000	0.000	0.000	0.000	0.000
120×2	**0.637**	0.000	0.000	0.000	0.112	0.251
120×4	**1.000**	0.000	0.000	0.000	0.000	0.000
120×6	**1.000**	0.000	0.000	0.000	0.000	0.000
120×8	**1.000**	0.000	0.000	0.000	0.000	0.000

TABLE 6. Computational results of three EGICAs on metric $\mathcal{C}$.

Instance	$\mathcal{C}(E, E1)$	$\mathcal{C}(E1, E)$	$\mathcal{C}(E, E2)$	$\mathcal{C}(E2, E)$	$\mathcal{C}(E2, E1)$	$\mathcal{C}(E1, E2)$
20×2	0.000	0.943	**0.476**	0.300	0.981	0.031
20×4	**1.000**	0.000	0.284	0.691	0.973	0.000
20×6	0.000	1.000	0.238	0.575	0.932	0.000
20×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
30×2	**1.000**	0.000	**0.981**	0.000	0.185	0.673
30×4	**1.000**	0.000	**0.980**	0.029	1.000	0.000
30×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
30×8	**1.000**	0.000	**1.000**	0.000	0.000	0.968
40×2	**0.433**	0.237	**0.825**	0.117	0.605	0.183
40×4	**1.000**	0.000	**0.944**	0.011	1.000	0.000
40×6	**1.000**	0.000	**1.000**	0.000	0.000	1.000
40×8	**1.000**	0.000	**1.000**	0.000	0.426	0.067
50×2	**1.000**	0.000	**0.647**	0.185	1.000	0.000
50×4	**1.000**	0.000	**1.000**	0.000	0.000	1.000
50×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
50×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
60×2	**1.000**	0.000	**1.000**	0.000	1.000	0.000
60×4	**1.000**	0.000	**1.000**	0.000	0.955	0.000
60×6	**1.000**	0.000	**1.000**	0.000	0.120	0.436
60×8	**1.000**	0.000	**0.726**	0.125	0.000	1.000
70×2	**1.000**	0.000	0.000	1.000	1.000	0.000
70×4	**1.000**	0.000	**1.000**	0.000	0.921	0.017
70×6	**1.000**	0.000	**1.000**	0.000	0.000	1.000
70×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
80×2	**1.000**	0.000	0.000	1.000	0.358	0.712
80×4	**1.000**	0.000	**1.000**	0.000	1.000	0.000
80×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
80×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
90×2	**1.000**	0.000	**0.893**	0.000	0.000	1.000
90×4	**1.000**	0.000	**1.000**	0.000	1.000	0.000
90×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
90×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
100×2	**0.545**	0.182	0.367	0.723	0.000	1.000
100×4	**1.000**	0.000	**1.000**	0.000	1.000	0.000
100×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
100×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
110×2	**1.000**	0.000	**1.000**	0.000	0.880	0.071
110×4	**1.000**	0.000	**1.000**	0.000	1.000	0.000
110×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
110×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000
120×2	**1.000**	0.000	**1.000**	0.000	0.000	1.000
120×4	**1.000**	0.000	**1.000**	0.000	1.000	0.000
120×6	**1.000**	0.000	**1.000**	0.000	1.000	0.000
120×8	**1.000**	0.000	**1.000**	0.000	1.000	0.000

The above parameters are obtained based on experiments.

B. EFFECT OF NEW STRATEGIES IN EGICA

There are two new strategies of EGICA. The first one is empire grouping. EGICA1 is obtained from EGICA by removing empire grouping from EGICA. The second one is two-phase imperialist competition based on adaptive search of member of archive and roulette selection of winning empire. EGICA2 is produced, in which imperialist competition is executed in the way of ICA [41]. There is no adaptive search in EGICA2.

Table 5 shows the computations results of EGICA and other algorithms on metric ρ, in which the reference set Ω^* consists of the non-dominated solutions in the union set of archive or non-dominated solutions set of EGICA, EGICA1, EGICA2, CCA, NSGA-II and MOTS. Each algorithm randomly runs 20 times for each instance. Table 6 describes the results of three EGICAs on metric $\mathcal{C}$. To make the results statistically convincing, paired-sample t-test is done to compare EGICA with other algorithms. The p-value results of paired-sample t-test are shown in Table 7.

TABLE 7. Results of paired sample t-test.

t-test	p-value (ρ)	p-value ($\mathcal{C}$)
t-test (EGICA, EGICA1)	0.000	0.000
t-test (EGICA, EGICA2)	0.000	0.000
t-test (EGICA, NSGA-II)	0.000	0.000
t-test (EGICA, CCA)	0.000	0.000
t-test (EGICA, MOTS)	0.000	0.000

The term ‘t-test (A, B)’ means that a paired t-test is conducted to judge whether algorithm A gives a better sample mean than B. We assume a significance level of 0.05. There is significant difference between A and B in the statistical sense if the p-value is less than 0.05.

As shown in Tables 5 and 6, EGICA performs better than its two variants on most of instances. EGICA generates better ρ than EGICA1 and EGICA2 on 39 instances, $\mathcal{C}(E1, E)$ is less than $\mathcal{C}(E, E1)$ on 42 instances and EGICA has smaller $\mathcal{C}(E2, E)$ than $\mathcal{C}(E, E2)$ on 39 instances. The results

TABLE 8. Computational results of EGICA and its three comparative algorithms on metric $\mathcal{C}$.

Instance	$\mathcal{C}(E,N)$	$\mathcal{C}(N,E)$	$\mathcal{C}(E,C)$	$\mathcal{C}(C,E)$	$\mathcal{C}(E,M)$	$\mathcal{C}(M,E)$
20 × 2	0.230	0.580	**0.464**	0.385	**0.395**	0.320
20 × 4	**1.000**	0.000	0.067	0.807	**0.697**	0.144
20 × 6	**1.000**	0.000	**0.823**	0.530	0.047	0.647
20 × 8	**1.000**	0.000	**1.000**	0.000	0.316	0.636
30 × 2	**0.864**	0.233	0.293	0.618	**0.403**	0.034
30 × 4	**1.000**	0.000	**0.407**	0.338	**0.458**	0.117
30 × 6	**1.000**	0.000	**1.000**	0.000	0.000	1.000
30 × 8	**1.000**	0.000	**1.000**	0.000	0.125	0.458
40 × 2	**1.000**	0.000	**1.000**	0.000	**0.446**	0.123
40 × 4	**1.000**	0.000	**1.000**	0.000	0.113	0.689
40 × 6	**0.451**	0.000	**0.604**	0.111	**0.842**	0.074
40 × 8	**1.000**	0.000	0.000	1.000	**1.000**	0.000
50 × 2	**1.000**	0.000	**0.396**	0.140	**0.881**	0.000
50 × 4	**1.000**	0.000	**0.961**	0.000	**0.547**	0.016
50 × 6	**1.000**	0.000	0.033	0.769	**1.000**	0.000
50 × 8	**1.000**	0.000	**1.000**	0.000	0.233	0.468
60 × 2	**1.000**	0.000	**0.813**	0.118	**1.000**	0.000
60 × 4	**0.667**	0.000	**0.443**	0.220	**1.000**	0.000
60 × 6	**1.000**	0.000	**1.000**	0.000	**0.807**	0.000
60 × 8	**1.000**	0.000	**1.000**	0.000	**0.574**	0.156
70 × 2	**1.000**	0.000	0.000	1.000	**0.406**	0.321
70 × 4	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
70 × 6	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
70 × 8	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
80 × 2	**1.000**	0.000	0.000	1.000	0.130	0.613
80 × 4	**0.832**	0.000	**1.000**	0.000	**1.000**	0.000
80 × 6	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
80 × 8	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
90 × 2	**1.000**	0.000	**0.532**	0.387	**0.766**	0.042
90 × 4	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
90 × 6	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
90 × 8	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
100 × 2	**1.000**	0.000	0.148	0.631	**0.477**	0.111
100 × 4	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
100 × 6	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
100 × 8	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
110 × 2	**1.000**	0.000	**1.000**	0.000	0.000	0.333
110 × 4	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
110 × 6	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
110 × 8	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
120 × 2	**1.000**	0.000	**0.657**	0.000	**0.457**	0.000
120 × 4	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
120 × 6	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000
120 × 8	**1.000**	0.000	**1.000**	0.000	**1.000**	0.000

TABLE 9. Computational times of EGICA and its three comparative algorithms.

Instance	Running time (s)				Instance	Running time (s)			
	EGICA	CCA	NSGA-II	MOTS		EGICA	CCA	NSGA-II	MOTS
20 × 2	5.542	6.335	9.812	5.552	70 × 6	32.41	33.40	41.02	30.25
20 × 4	7.885	8.235	10.72	7.126	70 × 8	37.23	37.08	50.71	37.05
20 × 6	9.005	9.356	16.21	8.884	80 × 2	20.47	21.24	29.82	20.68
20 × 8	12.40	13.76	20.42	13.59	80 × 4	28.45	34.24	37.17	33.75
30 × 2	8.568	8.411	15.12	9.548	80 × 6	38.25	37.32	47.26	38.22
30 × 4	12.23	13.53	19.44	14.22	80 × 8	43.18	44.37	58.41	46.56
30 × 6	14.72	16.52	23.41	15.28	90 × 2	23.45	26.97	33.22	24.44
30 × 8	18.55	20.25	29.22	18.95	90 × 4	31.72	43.86	54.18	38.12
40 × 2	11.18	12.85	18.33	10.99	90 × 6	43.08	56.33	61.27	47.08
40 × 4	14.08	15.44	21.48	13.41	90 × 8	49.47	58.75	65.37	50.84
40 × 6	16.33	16.37	25.69	16.44	100 × 2	25.57	26.56	32.35	25.77
40 × 8	18.45	19.42	27.85	18.42	100 × 4	35.48	43.14	50.82	40.02
50 × 2	12.42	13.87	21.58	12.85	100 × 6	50.31	55.04	61.77	51.33
50 × 4	18.28	19.56	26.12	17.82	100 × 8	63.58	71.15	77.67	65.77
50 × 6	23.34	27.54	31.48	24.52	110 × 2	26.11	31.68	39.39	28.50
50 × 8	26.86	29.23	36.01	28.32	110 × 4	46.75	55.96	65.35	48.48
60 × 2	14.15	14.05	23.69	14.33	110 × 6	52.96	64.80	68.07	55.26
60 × 4	20.28	23.58	29.25	22.56	110 × 8	65.44	72.30	88.21	68.30
60 × 6	26.63	31.09	36.40	27.88	120 × 2	29.27	34.57	45.18	30.39
60 × 8	31.44	35.38	41.98	34.48	120 × 4	47.75	57.42	79.46	49.19
70 × 2	17.19	18.17	26.94	17.22	120 × 6	57.38	66.34	86.22	60.24
70 × 4	27.58	30.58	36.04	28.56	120 × 8	73.96	84.08	104.2	80.72

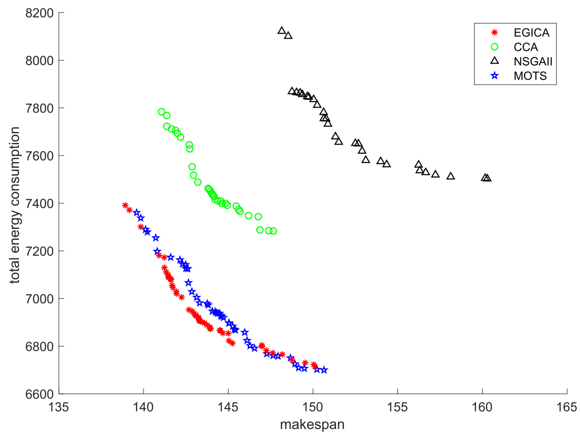

FIGURE 6. Distributions of non-dominated solutions of four algorithms for instance 60 × 8.

in Table 7 also demonstrate the notable performance difference between EGICA and its variants.

When population is divided into N_{im} empires, empires are evolved independently and solutions in an empire just exchange information with those in the same empire. Empire grouping makes empires have frequent communications, as a result, the performance of EGICA is improved. In the usual imperialist competition, the weakest colony from the weakest empire is directly added into the winning empire; in EGICA, the weakest colony is eliminated from the winning empire and directly replaced with a member of archive and then adaptive search of the member is executed, in this way, the computing resource waste on the weakest colony is avoided and search efficiency is improved, so the new imperialist competition really improves the performance of EGICA.

C. RESULTS AND ANALYSES

Tables 5 and 8 describe the computational results of EGICA and its three comparative algorithms. Table 9 shows the computational times of four algorithms. Figure 6 gives the distributions of non-dominated solutions of four algorithms for instance 60 × 8. Because it is impossible to show the distribution of solutions in two-dimensional objective space using interval objectives, each point in Figure 6 is composed of middle values of f_1, f_2 of solutions. Figure 7 is the Gantt chart of a non-dominated solution of EGICA for instance 40 × 2, $f_1 = [18.9, 37.3]$, $f_2 = [835.7, 1692.2]$. The serial number of job is labeled on segment.

As shown in Tables 5 and 8, EGICA can provide better results than CCA, NSGA-II and MOTS on most of instances. EGICA has bigger ρ than NSGA-II on 43 instances and smaller $\mathcal{C}(N, E)$ than $\mathcal{C}(E, N)$ on 43 instances; moreover, EGICA provides all members of the reference set Ω^* on 17 instances. The statistical results in Table 7 also validate the performance difference between EGICA and NSGA-II. Figure 6 also shows the performance difference between

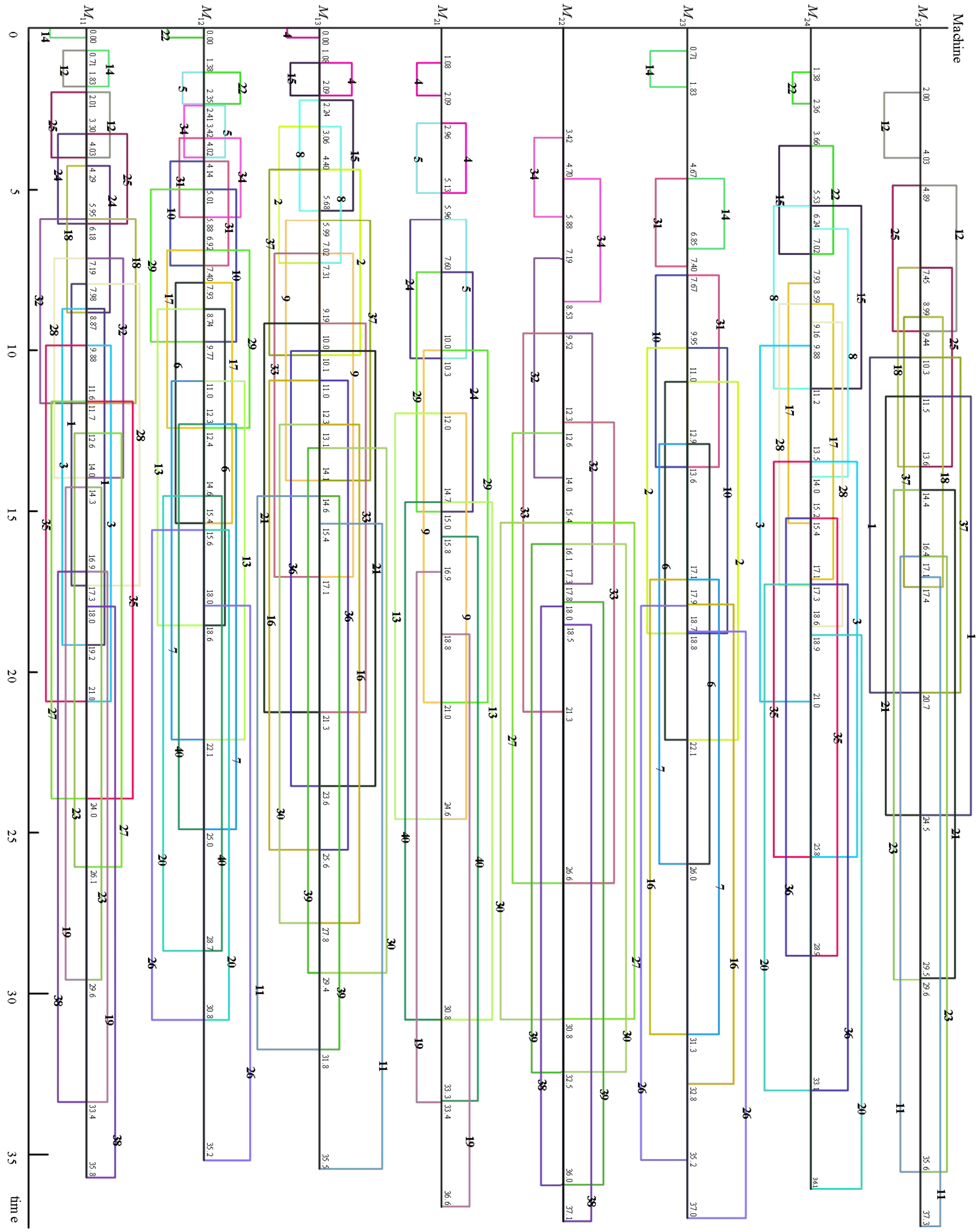

FIGURE 7. A non-dominated solution of EGICA for instance 40 × 2.

EGICA and NSGA-II. The same conclusion also can be drawn on EGICA and CCA, MOTS.

In EGICA, all empires are divided into s groups, the strongest empire of each group guides the search of other empires in the same group, empires in a group are not fixed and an empire can be allocated into different groups, these strategies can effectively keep high diversity, so it can be concluded that empire or sub-population grouping may be effective path to improve performance of multi-population algorithms like ICA; on the other hand, two-phase imperialist competition can effectively avoid premature. The above features result in the good performance of EGICA on solving EIHFSP, thus, EGICA is a very competitive method for EIHFSP.

VI. CONCLUSIONS

Energy-efficient scheduling is the main topic of scheduling research in recent years; however, the main works on energy-efficient scheduling are done in the deterministic case and uncertainty is seldom adopted in energy-efficient scheduling problem. In this study, an energy-efficient interval scheduling problem called EIHFSP is investigated and a new algorithm named EGICA is proposed to minimize two interval objectives. EGICA is composed of empire grouping, assimilation, adaptive revolution and two-phase imperialist competition. The computational experiments are conducted and results show that the effectiveness of empire grouping and two-phase imperialist competition and the promising advantages of EGICA on EIHFSP.

This paper provides some new strategies to construct a ICA with good performance. This is the theoretical contribution of our paper. We will investigate new strategies of EGICA on the applications of other scheduling problems, such as other energy-efficient scheduling problem with uncertainty. On the other hand, we have paid attention to distributed scheduling with energy related objective and tried to design a powerful scheduling algorithm, so energy-efficient distributed scheduling is also our future topic.

REFERENCES

[1] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 1–18, Aug. 2010.

[2] C. Low, C.-J. Hsu, and C.-T. Su, "A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines," *Comput. Oper. Res.*, vol. 35, no. 3, pp. 845–853, Mar. 2008.

[3] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, and F. Werner, "Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria," *Int. J. Adv. Manuf. Technol.*, vol. 37, nos. 3–4, pp. 354–370, May 2008.

[4] B. Naderi, M. Zadieh, A. K. G. Balagh, and V. Roshanaei, "An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness," *Expert Syst. Appl.*, vol. 36, no. 6, pp. 9625–9633, Aug. 2009.

[5] S. M. Mousavi, M. Mousakhani, and M. Zandieh, "Bi-objective hybrid flow shop scheduling: A new local search," *Int. J. Adv. Manuf. Tech.*, vol. 64, nos. 5–8, pp. 933–950, Feb. 2013.

[6] E. Rashidi, M. Jahandar, and M. Zandieh, "An improved hybrid multi-objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines," *Int. J. Adv. Manuf. Technol.*, vol. 49, nos. 9–12, pp. 1129–1139, Aug. 2010.

[7] H.-M. Cho, S.-J. Bae, J. W. Kim, and I.-J. Jeong, "Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm," *Comput. Ind. Eng.*, vol. 61, no. 3, pp. 529–541, Oct. 2010.

[8] N. Karimi, M. Zandieh, and H. R. Karamooz, "Bi-objective group scheduling in hybrid flexible flowshop: A multi-phase approach," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4023–4032, Jun. 2010.

[9] T. H. Tran and K. M. Ng, "A hybrid water flow algorithm for multi-objective flexible flow shop scheduling," *Eng. Opt.*, vol. 45, no. 4, pp. 483–502, Apr. 2013.

[10] S. J. Wang and M. Liu, "Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method," *Int. J. Prod. Res.*, vol. 52, no. 5, pp. 1495–1508, Mar. 2014.

[11] M. Ebrahimi, S. M. T. F. Ghomi, and B. Karimi, "Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates," *Appl. Math. Model.*, vol. 38, nos. 9–10, pp. 2490–2504, May 2014.

[12] D. Lei and Y. Zheng, "Hybrid flow shop scheduling with assembly operations and key objectives: A novel neighborhood search," *App. Soft Comput.*, vol. 61, pp. 122–128, Dec. 2017.

[13] M. A. Bozorgirad and R. Logendran, "Bi-criteria group scheduling in hybrid flowshops," *Int. J. Prod. Econ.*, vol. 145, no. 2, pp. 599–612, Oct. 2013.

[14] N. Karimi and H. Davoupour, "Multi-objective colonial competitive algorithm for hybrid flowshop problem," *Appl. Soft Comput.*, vol. 49, pp. 725–733, Dec. 2016.

[15] D. Lei, "Two-phase neighborhood search algorithm for two-agent hybrid flow shop scheduling problem," *Appl. Soft Comput.*, vol. 34, pp. 721–727, Sep. 2015.

[16] D. Lei and X. Guo, "A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9333–9339, Dec. 2015.

[17] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 301–305, Apr. 2014.

[18] M. Dai, D. B. Tang, A. Giret, M. A. Salido, and W. D. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 5, pp. 418–429, 2013.

[19] H. Luo, B. Du, G. Q. Huang, H. Chen, and X. Li, "Hybrid flow shop scheduling considering machine electricity consumption cost," *Int. J. Prod. Econ.*, vol. 146, no. 2, pp. 423–439, 2013.

[20] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Comput. Ind.*, vol. 81, pp. 82–95, Sep. 2016.

[21] W. Lin, D. Yu, C. Zhang, X. Liu, S. Zhang, Y. Tian, S. Liu, and Z. Xie, "A multi-objective teaching–learning-based optimization algorithm to scheduling in turning process for minimizing makespan and carbon footprint," *J. Clean. Prod.*, vol. 101, no. 15, pp. 337–347, Aug. 2015.

[22] D. Lei, L. Gao, and Y. Zheng, "A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop," *IEEE Trans. Eng. Manage.*, vol. 65, no. 2, pp. 330–340, May 2018.

[23] J.-Q. Li, H.-Y. Sang, Y.-Y. Han, C.-Q. Wang, and K.-Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Clean. Prod.*, vol. 181, pp. 584–598, Apr. 2018.

[24] Z. Zeng, M. G. Hong, Y. Man, J. Li, Y. Zhang, and H. Liu, "Multi-object optimization of flexible flow shop scheduling with batch process—Consideration total electricity consumption and material wastage," *J. Clean. Prod.*, vol. 183, no. 10, pp. 925–939, May 2018.

[25] L. Meng, C. Zhang, X. Shao, Y. Ren, and C. Ren, "Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines," *Int. J. Prod. Res.*, vol. 57, no. 4, pp. 1119–1145, 2018.

[26] G.-S. Liu, Y. Zhou, and H.-D. Yang, "Minimizing energy consumption and tardiness penalty for fuzzy flow shop scheduling with state-dependent setup time," *J. Clean. Prod.*, vol. 147, no. 2017, pp. 470–484, Mar. 2017.

[27] D. Lei and X. Guo, "An effective neighborhood search for scheduling in dual-resource constrained interval job shop with environmental objective," *Int. J. Prod. Econ.*, vol. 159, pp. 196–303, Jan. 2015.

[28] J. Wang, F. Qiao, F. Zhao, and J. W. Sutherland, "Batch scheduling for minimal energy consumption and tardiness under uncertainties: A heat treatment application," *CIRP Ann.*, vol. 65, no. 1, pp. 17–20, 2016.

[29] J.-Q. Li and Q.-K. Pan, "Chemical-reaction optimization for solving fuzzy job-shop scheduling problem with flexible maintenance activities," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 4–17, Sep. 2013.

[30] C. T. Ng, T. C. E. Cheng, A. M. Bandalouski, M. Y. Kovalyov, and S. S. Lam, "A graph-theoretic approach to interval scheduling on dedicated unrelated parallel machines," *J. Oper. Res. Soc.*, vol. 65, no. 10, pp. 1571–1579, Oct. 2014.
[31] Y. Fu, J. Ding, H. Wang, and J. Wang, "Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in industry 4.0-based manufacturing system," *Appl. Soft Comput.*, vol. 68, pp. 847–855, Jul. 2018.
[32] K. Wang and S. H. Choi, "A holonic approach to flexible flow shop scheduling under stochastic processing times," *Comput. Oper. Res.*, vol. 43, no. 1, pp. 157–168, Mar. 2014.
[33] S. Wang, L. Wang, Y. Xu, and M. Liu, "An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3778–3793, Jun. 2013.
[34] J. Lin, "Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 186–196, Jan. 2019.
[35] F. P. Golneshini and H. Fazllahtabar, "Meta-heuristic algorithms for a clustering-based fuzzy bi-criteria hybrid flow shop scheduling problem," *Soft. Comput.*, pp. 1–20, Jan. 2019.
[36] J. Lin, L. Zhu, and Z.-J. Wang, "A hybrid multi-verse optimization for the fuzzy flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 127, pp. 1089–1100, Jan. 2019.
[37] Y. Han, D. Gong, Y. Jin, and Q.-K. Pan, "Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time," *Appl. Soft Comput.*, vol. 42, pp. 229–245, May 2016.
[38] D. Lei, "Interval job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 60, nos. 1–4, pp. 291–301, Apr. 2012.
[39] D. Lei, "Population-based neighborhood search for job shop scheduling with interval processing time," *Comput. Ind. Eng.*, vol. 61, no. 4, pp. 1200–1208, Nov. 2011.
[40] E. Atashpaz-Gagari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 4661–4667.
[41] S. Hosseini and A. A. Khaled, "A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research," *Appl. Soft Comput.*, vol. 24, pp. 1078–1094, Nov. 2014.
[42] S. Molla-Alizadeh-Zavardehi, R. Tavakkoli-Moghaddam, and F. H. H. Lotfi, "Hybrid metaheuristics for solving a fuzzy single batch-processing machine scheduling problem," *Sci. World J.*, vol. 5, Apr. 2014, Art. no. 214615.
[43] Z. Pan, D. Lei, and Q. Zhang, "A new imperialist competitive algorithm for multiobjective low carbon parallel machines scheduling," *Math. Problems Eng.*, vol. 2018, Apr. 2018, Art. no. 5914360.
[44] E. Shokrollahpour, M. Zandieh, and B. Dorri, "A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem," *Int. J. Prod. Res.*, vol. 49, no. 11, pp. 3013–3087, May 2010.
[45] H. Seidgar, M. Kiani, M. Abedi, and H. Fazlollahtabar, "An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem," *Int. J. Prod. Res.*, vol. 52, no. 4, pp. 1240–1256, Feb. 2014.
[46] S. Karimi, Z. Ardalan, B. Naderi, and M. Mohammadi, "Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm," *Appl. Math. Model.*, vol. 41, pp. 667–682, Jan. 2017.
[47] M. Zandieh, A. R. Khatami, and S. H. A. Rahmati, "Flexible job shop scheduling under condition-based maintenance: Improved version of imperialist competitive algorithm," *Appl. Soft Comput.*, vol. 58, pp. 449–464, Sep. 2017.
[48] S. M. Goldansaz, F. Jolai, and A. H. Z. Anaraki, "A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop," *Appl. Math. Model.*, vol. 37, no. 23, pp. 9603–9616, Dec. 2013.
[49] C. Jiang, X. Han, G. R. Liu, and G. P. Liu, "A nonlinear interval number programming method for uncertain optimization problems," *Eur. J. Oper. Res.*, vol. 188, no. 1, pp. 1–13, Jul. 2008.
[50] J.-Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 758–771, 2016.
[51] 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.
[52] E. Zitzler and L. Thiele, "Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[53] D. Lei, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 37, nos. 1–2, pp. 157–165, Apr. 2008.

RUI ZHOU received the bachelor's degree in automation from the Henan University of Technology, Zhengzhou, China, in 2018. He is currently pursuing the master's degree with the School of Automation, Wuhan University of Technology, Wuhan, China. His current research interest includes manufacturing systems intelligent optimization and scheduling.

DEMING LEI received the M.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 1996, and the Ph.D. degree in automation science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005. He is currently a Professor with the School of Automation, Wuhan University of Technology, Wuhan, China. He has published over 50 journal papers. His current research interests include intelligent system optimization and control, and production scheduling.

XINMIN ZHOU received the M.E. and Ph.D. degrees from the Huazhong University of Science and Technology, China, in 1989 and 2011, respectively. He is currently an Associate Professor with the Wuhan University of Technology. His research interests include control theory, power electronics, and port automation systems.