IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2019, accepted August 19, 2019, date of publication August 27, 2019, date of current version September 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937747

A Novel Imperialist Competitive Algorithm With Multi-Elite Individuals Guidance for Multi-Object Unrelated Parallel Machine Scheduling Problem

MEI WANG AND GUOHUA PAN

Laboratory of Image Processing and Pattern Recognition, Yantai Vocational College, Yantai 264670, China
Yantai Public Security Bureau, Yantai 264670, China

Corresponding author: Mei Wang (wangmei336@163.com)

This work was supported by the Shandong Province Science and Technology Development Plan Project Foundation under Grant 2014GGX101030.

ABSTRACT In this study unrelated parallel machine scheduling problem (UPMSP) with preventive maintenance (PM) and sequence dependent setup times (SDST) is investigated. A novel imperialist competitive algorithm (NICA) with multi-elite individuals guidance is proposed to minimize makespan and total tardiness simultaneously. Initialization is done by two heuristics, each of which is built based on one objective. Multi-elite individuals guidance strategy is added in assimilation that colonies can move toward other imperialists, diversified strategies such as local search and estimation of distribution algorithm (EDA) are adopted based on solution quality in revolution and EDA is also used in imperialist competition. Empire aggression is added by local search of imperialist for plundering a randomly chosen colony. A number of experiments are conducted on the impact of new strategies and the comparisons among NICA and other algorithms. Computational results demonstrate the effectiveness and advantages of NICA in solving UPSMP with PM and SDST.

INDEX TERMS Preventive maintenance, setup times, imperialist competitive algorithm, multi-elite individual guidance, estimation of distribution algorithm.

I. INTRODUCTION

Parallel machine scheduling problem (PMSP) is a typical problem in the manufacturing process [1]–[5], which needs to optimize the target by rational allocation and scheduling of resources. Traditional PMSP often assumes that machines are continuously available throughout the production process, but due to the need for preventive maintenance (PM), the machine cannot be processed normally during the PM. Therefore, the assumption is not realistic in many actual production and manufacturing. Considering that PM on a regular basis can effectively prevent potential failures and avoid serious accidents in production, it is necessary to consider PM in unrelated parallel machine scheduling problems in order to get a better scheduling scheme.

In the past few years, PMSP with PM has attracted much attentions. In order to minimize makespan, Li *et al.* [6] proposed two mathematical programming models and designed two heuristics. Yoo and Lee [7] presented a dynamic programming approach to minimize makespan, (weighted) sum of completion times, maximum lateness and sum of lateness, respectively. For two-parallel-machine scheduling problem with machine-dependent availabilities, He *et al.* [8] built a mixed 0-1 programming model for small size problem and gave nine heuristics to solve large sized instances of the problem. Wang and Wei [9] designed a model and analyzed the complexity for PMSP with deteriorating maintenance to minimize the total absolute differences in completion times and the total absolute differences in waiting times.

For UPMSP with PM, Gara-Ali *et al.* [10] proposed several performance criteria and different maintenance systems and gave a new method to solve the problem with deteriorating and maintenance. In order to minimize total machine load, Yang *et al.* [11] applied the group balance principle to solve UPMSP with aging effects and PM and proved that the problem remains polynomially solvable when the maintenance frequency on every machine is given. Avalos-Rosales *et al.* [12] presented a mathematical

The associate editor coordinating the review of this article and approving it for publication was Baozhen Yao.

formulation for UPMSP with PM and SDST and designed an efficient meta-heuristic based on a multi-start strategy to solve larger instances. Tavana *et al.* [13] developed a three-stage maintenance scheduling model for UPMSP with aging effect and multi-maintenance activities. Wang and Liu [14] presented a multi-objective integrated optimization method with non-dominated sorting genetic algorithm-II (NSGA-II) to solve the multi-objective UPMSP with multi-resources PM.

In order to simplify the model, most studies assume that the setup times between jobs can be neglected or included in the processing time. But in actual production, setup times cannot be ignored, especially when the setup times are both sequence and machine dependent, such as chemical, printing, metal processing and semiconductor industries [15]. UPMSP with SDST has attracted some attention since the pioneering work of Parker *et al.* [16]. Kurz and Askin [17] presented several heuristics. Vallada and Ruiz [18] designed a genetic algorithm (GA) includes a fast local search and a local search enhanced crossover operator. Arnaout *et al.* [19] introduced an ant colony optimization (ACO) and tackled a special structure of the problem. Wang *et al.* [20] developed a hybrid EDA with iterated greedy search. Diana *et al.* [21] proposed an immune-inspired algorithm to solve this problem. Ezugwu and Akutsah [22] built an improved firefly algorithm refined with a local search. Fanjul-Peyro *et al.* [23] gave new mixed integer linear programs and a mathematical programming based algorithm to solve this problem. Caniyilmaz *et al.* [24] collected a real-life data from a factory and gave an artificial bee colony algorithm to solve UPMSP with SDST, processing set restrictions and due date. For UPMSP with SDST, machine eligibility restrictions and a common server, Bektur and Sarac [25] proposed a mixed integer linear programming model and designed a tabu search and a simulated annealing algorithm.

As mentioned above, many works have been finished on UPMSP with PM and UPMSP with SDST and most of them are just about the minimization of makespan as single objective. Few studies are about multi-objective optimization of the above two kinds of UPMSP; on the other hand, UPMSP with PM and UPMSP with SDST have been handled independently; however, PM and SDST are seldom simultaneously integrated into UPMSP [12]. PM and SDST are very common processing constraints in the real-life production process and the actual scheduling problem always has some conflicting objectives, thus, it is necessary to deal with multi-objective UPMSP with PM and SDST.

It also can be found that heuristics are the main method for UPMSP with PM, only GA is applied to solve it and meta-heuristics such as GA, ACO and EDA have applied to deal with UPMSP with SDST; however, the applications of meta-heuristics are not investigated fully and some algorithms such as imperialist competitive algorithm(ICA) are not used to solve UPMSP with PM or SDST.

ICA is a meta-heuristic based on the sociopolitical imperialist competition [26], which is an effective global optimization method with strong neighborhood search capability and flexible structure and can be combined easily with other algorithms [27]. In recent years, ICA has been successfully applied to solve many optimization problems including PMSP [28]–[37]. Although ICA has adopted to deal with PMSP, it is not utilized to handle UPMSP with PM or SDST. ICA has great potential to solve UPMSP with PM and SDST because of its notable features and the previous works on PMSP, so it is meaningful to consider the applications of ICA to UPMSP with PM and SDST.

In this paper, UPMSP with PM and SDST is considered and a novel imperialist competitive algorithm (NICA) with multi-elite individuals guidance is proposed to minimize total tardiness and makespan. In NICA, initialization is done by two heuristics, each of which is built based on one of two objectives. A new strategy of assimilation is given, where colonies can learn from other imperialists. Diversified strategies such as local search and EDA are adopted according to solution quality in revolution, and empire aggression is added by local search of imperialist for plundering a randomly chosen colony. A novel imperialist competition is designed, in which the weakest colony of the weakest empire is compared with a new solution generated by EDA and the better one will be allocated to the winning empire. A number of experiments are conducted on the impact of new strategies and the comparisons among NICA and other algorithms. Computational results demonstrate the effectiveness and advantages of NICA in solving UPSMP with PM and SDST.

The remainder of the paper is organized as follows. Problem under study are described in Sections 2. NICA for the problem is reported in Section 3. Numerical test experiments on NICA are shown in Section 4 and we summarize the conclusion and some research topics in the future in the last section.

II. PROBLEM DESCRIPTION

UPMSP with PM and SDST can be described as follows. Suppose n independent jobs $J_1, J_2, \cdots, J_n$ can be processed on m unrelated parallel machines $M_1, M_2, \cdots, M_m$. Each job is available at time zero. p_{ij} is the processing time of job J_j on machine M_i. d_j indicates the due date of J_j.

Jobs can be processed in an interval between two consecutive maintenance activities. The interval is called the processing one and its time length is u_i. w_i is the period of each maintenance. Thus, the maintenance periodic recycle of machine M_i is $T_i = w_i + u_i$. We use J_0 to denote maintenance activities [12].

The setup time is dependent on sequence and machine. s_{ijk} is the setup time for processing job J_k just after job J_j on machine M_i, s_{i0k} indicates the setup time of machine M_i to process the first job J_k after a maintenance activity, and s_{ij0} is the setup time of machine M_i to perform a maintenance activity just after the job J_j.

There are some constraints on jobs and machines:

1) Each job can be processed on only one machine at a time.

2) Operations cannot be interrupted.

3)If the processing of a job cannot be completed in a processing interval, the job can't be processed in the current interval and should be moved to the next interval for processing etc.

The goal of UPMSP with PM and SDST is to minimize the following two objectives simultaneously:

$$\min f_1 = C_{\max} = \max\left\{C_j | j = 1, 2, \cdots, n\right\} \quad (1)$$

$$\min f_2 = \sum_{j=1}^{n} \max\left\{C_j - d_j, 0\right\} \quad (2)$$

where the first objective f_1 is makespan and the second objective f_2 is total tardiness. C_j indicates the completion time of job J_j.

We give an example with two machines and six jobs. $w_1 = 37$, $w_2 = 4$, $u_1 = 70$, $u_2 = 86$. Due dates of six jobs are 4, 89, 46, 35, 98, 45, respectively. pm is the processing time matrix, S_1 is the setup times matrix on machine 1, and S_2 indicates the setup times matrix on machine 2.

$$pm = \begin{bmatrix} 1 & 87 & 28 & 32 & 38 & 9 \\ 4 & 21 & 68 & 17 & 43 & 48 \end{bmatrix} \quad (3)$$

$$S_1 = \begin{bmatrix} 0 & 2 & 2 & 3 & 9 & 7 & 9 \\ 6 & 0 & 1 & 8 & 1 & 3 & 9 \\ 1 & 4 & 0 & 7 & 3 & 7 & 8 \\ 3 & 7 & 3 & 0 & 2 & 3 & 5 \\ 4 & 3 & 8 & 3 & 0 & 5 & 2 \\ 8 & 8 & 3 & 7 & 9 & 0 & 5 \\ 1 & 8 & 8 & 1 & 2 & 2 & 0 \end{bmatrix} \quad (4)$$

$$S_2 = \begin{bmatrix} 0 & 3 & 7 & 3 & 7 & 6 & 3 \\ 2 & 0 & 5 & 1 & 6 & 1 & 7 \\ 3 & 6 & 0 & 7 & 7 & 6 & 2 \\ 4 & 7 & 6 & 0 & 9 & 6 & 9 \\ 3 & 3 & 7 & 3 & 0 & 1 & 7 \\ 8 & 5 & 8 & 5 & 6 & 0 & 9 \\ 4 & 7 & 4 & 1 & 7 & 9 & 0 \end{bmatrix} \quad (5)$$

III. NICA FOR UPMSP WITH PM AND SDST

In ICA, the empire easily fall into local optimum if colonies only move toward its imperialist and it is necessary to make individuals learn from other imperialists in assimilation. Revolution is often implemented in a single way and seldom done using the diversified methods, it is important to make full use of imperialist as good solution. EDA is an emerging stochastic group evolution algorithm based on statistical learning principle [38] and has strong global exploration ability. In this study, a new NICA by introducing EDA into ICA is proposed, which is described in the following sub-sections.

A. ENCODING AND DECODING

UPMSP with PM and SDST consists of two sub-problems: machine assignment and scheduling.Two-string representation is often used in the previous works [39]–[42]; however, two strings of a solution are often dependent with each other. In this study, a novel two-string representation is proposed, in which a solution is composed of two independent strings.

For the problem with n jobs and m machines, each solution consists of a scheduling string$[\pi_1, \pi_2, \cdots, \pi_n]$ and a machine assignment string $\left[M_{\theta_1}, M_{\theta_2}, \cdots, M_{\theta_n}\right]$. where M_{θ_j} indicates the parallel machine assigned for job J_j, $1 \leq \theta_j \leq m$, $\pi_i \in \{1, 2, \cdots, n\}$. The assigned machine and the processing order for each job can be determined according to the machine assignment string and scheduling string respectively.

Fig. 1 (a), shows the Gantt chart of a possible solution with a machine assignment string $[M_1, M_2, M_1, M_1, M_2, M_2]$ and a scheduling string [3, 5, 1, 2, 6, 4]. As shown in Fig.1 (a), when the processing of job J_6 is considered, it can be found that the processing cannot finish in the first processing interval, so job J_6 should be processed in the next interval.

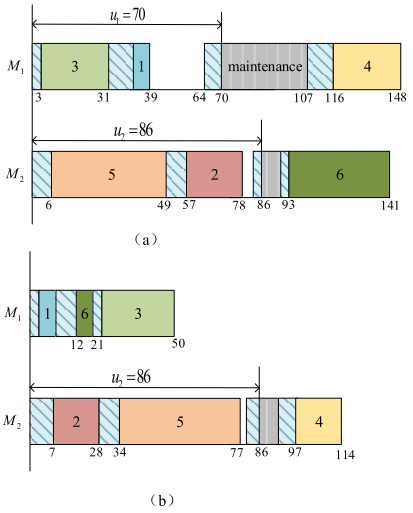


FIGURE 1. Gantt charts of the solution.

B. INITIALIZATION

In this section, initial population is first produced by two heuristics and a random way, then initial empires are constructed. To hybrid EDA with ICA, probability matrices of EDA are also initialized.

Heuristic is often used to generate initial population. In general, a heuristic can only produce a solution. In this study, heuristics 1 and 2 are adopted, each of which produces $N/3$ initial solutions, where N represents population size.

Heuristic 1 is shown in Algorithm 1. For the example, a string [3, 5, 1, 2, 6, 4] is first randomly obtained. Start with J_3, we calculate $\bar{C}_{3,1,1}$ and $\bar{C}_{3,1,2}$ and obtain $\bar{C}_{min,3}$ of 31. J_3 is allocated to the first position on M_1. Then for J_5, $\bar{C}_{min,5}$ is decided and equal to 49, and J_5 is allocated on M_2. For J_1, we find that $\bar{C}_{1,1,1} = \bar{C}_{1,2,1} = \bar{C}_{min,1} = 39$, so two positions can be selected and position 1 is chosen, so J_1 is inserted on the left of J_3 and the scheduling string becomes [1, 3, 5, 2, 6, 4]. The final scheduling string is [1, 6, 3, 2, 5, 4] and machine assignment string is $[M_1, M_2, M_1, M_2, M_2, M_1]$. C_{max} is 114. Fig.1(b) shows the Gantt chart of the obtained solution.

Algorithm 1 Heuristic 1

Randomly produce a scheduling string $[\pi_1, \pi_2, \cdots, \pi_n]$
for π_1 to π_n **do**
  for each position l on each machine M_i, $i = 1, 2, \cdots, m$ **do**
    Calculate $\bar{C}_{e,l,i}$ if inserting π_j into position l of M_i
  end for
  Compute the minimum value $\bar{C}_{min,e}$ of all possible $\bar{C}_{e,l,i}$.
  Decide all machines and positions meeting $\bar{C}_{e,l,i} = \bar{C}_{min,e}$.
  if more than one position has $\bar{C}_{e,l,i} = \bar{C}_{min,e}$ **then**
    Randomly choose a machine M_i^* and a position l^*.
  else
    Directly select the machine M_i^* and the position l^*.
  end if
  Allocate π_j into position l^* of M_i^*.
  if l^* is not the last position on M_i^* **then**
    Adjust the position of π_j on scheduling string by inserting π_j into the position of J_u, which is processed on position $l^* + 1$ of M_i^*.
  end if
end for

Heuristic 2 is similar with heuristic 1 and the difference between them is the computation of index. In heuristic 2, total tardiness $\bar{T}_{e,l,i}$ of all allocated jobs is first decided and then the minimum value $\bar{T}_{min,e}$ of all $\bar{T}_{e,l,i}$ is obtained.

Unlike the exiting heuristics, heuristics 1 and 2 can produce many solutions. In this way, initial population P is generated, in which $2N/3$ solutions are produced by two heuristics and the remained solutions are randomly obtained.

To construct N_{imp} initial empires, cost c_k for solution k is newly defined by

$$c_k = rank_k \times D + \sum_{j=1}^{D} \frac{\left| f_{k,j} - f_j^{\min} \right|}{f_j^{\max} - f_j^{\min}} \tag{6}$$

where $rank_k$ represents the *rank* value defined by Deb *et al.* [43] of the solution k. D is the number of objective functions. $f_{k,j}$ indicates the objective f_j of solution k. $f_j^{\max}$ is the maximum f_j of all solutions in P. $f_j^{\min}$ is the minimum f_j of all solutions in P.

After cost of each solution is computed, all solutions are sorted in the ascending order of cost and the first N_{imp} solutions are chosen as imperialists and other solutions are colonies. There are N_{col} colonies, $N_{col} = N - N_{imp}$. Then the normalized cost $\bar{c}_k$ and total number NC_k of colonies are calculated [26]. Finally, NC_k colonies are randomly allocated into empire k.

The optimization of EDA starts with initial probability matrices. m $(n+1) \times n$ probability matrices $\rho^i(g)$ are used to describe the probability of two adjacent jobs on machine M_i, $i = 1, 2, \cdots, m$. The probability matrix $\rho^i(g)$ is described as follows

$$\rho^i(g) = \begin{bmatrix} \rho_{0,1}^i(g) & \cdots & \rho_{0,n}^i(g) \\ 0 & \cdots & \rho_{1,n}^i(g) \\ \vdots & \vdots & \vdots \\ \rho_{n,1}^i(g) & \cdots & 0 \end{bmatrix}, \quad i = 1, 2, \cdots, m \tag{7}$$

where $\rho_{j,k}^i(g)$ represents the probability that job J_k is on the right of job J_j on machine M_i at the gth generation, $\rho_{0,j}^i(g)$ represents the probability that job J_j is first job on machine M_i at the gth generation.

The initial probability matrices $\rho^i(0)$ are uniformly produced in order to ensure the uniform sampling of the solution space.

$$\rho^i(0) = \begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \\ 0 & \frac{1}{n-1} & \cdots & \frac{1}{n-1} \\ \frac{1}{n-1} & 0 & \cdots & \frac{1}{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n-1} & \frac{1}{n-1} & \cdots & 0 \end{bmatrix}, \quad i = 1, 2, \cdots, m \tag{8}$$

C. ASSIMILATION

Assimilation is often implemented by moving each colony toward its imperialist. Global search between colony and imperialist is frequently used in assimilation. If colonies only learn from its imperialist, the empire will easily fall into local optimum. In this study, a new strategy based on multi-elite individuals guidance is proposed.

Suppose that H_k is the set of all colonies in empire k. The detailed steps of assimilation for empire k are described in Algorithm 2.

Algorithm 2 Assimilation

for each colony $\lambda \in H_k$ **do**
  Generate a random number α in [0,1].
  if $\alpha < 0.5$ **then**
    Colony λ moves toward its imperialist k. Two global search operations between λ and its imperialist k are executed sequentially.
  else
    Choose one solutions k^* from the other imperialists by tournament selection
    Colony λ learn from imperialist k^*. Two global search operations between λ and imperialist k^* are executed sequentially.
  end if
end for

Two global search operators are used for two strings. For the machine assignment string, a random number string $RS = \{rs_1, rs_2, \cdots, rs_n\}$ with length of n is first generated,

start with the first element of rs_1, and for each element rs_i, if $rs_i > 0.5$, then M_{θ_i} of colony λ is replaced with that of imperialist k.

For the scheduling string, randomly select a segment ϕ from the scheduling string of imperialist k, then we decide the position s of the first job of the segment ϕ in colony λ, delete all jobs of ϕ from the scheduling string of colony λ and insert the segment ϕ into position s sequentially.

Unlike the existing assimilation [26]–[34], the above assimilation make colonies guided by multi-elite individuals, which can avoid the algorithm falling into local optimum. Meanwhile, inspired by the above view, a new method is also proposed by using diversified strategies such as local search and EDA in revolution.

D. REVOLUTION

Revolution is another way to generate new solutions. In general, revolution of colony is implemented by local search like mutation of GA and all colonies are changed in a single way. In this study, the diversified methods are adopted based on solution quality.

The detailed steps of revolution in empire k are as follows. For each colony $\lambda \in H_k$, generate a random number β, if $\beta < p_r$, then if c_λ is less than γ colonies, conduct four neighborhood structures on λ sequentially; else EDA is executed on λ, a new solution z is obtained, if z dominates colony λ, then replace colony λ with z. Where c_λ is the cost of colony λ and we set $\gamma = 0.8N_{col}$ based on experiments.

Neighborhood structures $\mathcal{N}_1$,$\mathcal{N}_2$,$\mathcal{N}_3$,$\mathcal{N}_4$ are used and described below. Neighborhood structure $\mathcal{N}_1$ generates new solutions by randomly choosing a machine M_i and two jobs J_j and J_k on M_i, and then exchange them on scheduling string. $\mathcal{N}_1$ is just used to change scheduling string. Neighborhood structure $\mathcal{N}_2$ also acts on scheduling string. Randomly choose a machine M_i and two jobs J_j and J_k on M_i, and then decide all jobs between J_j and J_k on M_i and reverse their sequence on scheduling string. Neighborhood structure $\mathcal{N}_3$ is shown below. Randomly select machines M_{i_1} and M_{i_2}, choose a job J_j on M_{i_1} and a job J_k on M_{i_2} stochastically, swap J_j and J_k on scheduling string and swap machines of two jobs on machine assignment string. Neighborhood structure $\mathcal{N}_4$ is done in the following way. As done in $\mathcal{N}_3$, machines M_{i_1} and M_{i_2}, a job J_j on M_{i_1} and a job J_k on M_{i_2} are first randomly decided, then insert J_j on the right of J_k on scheduling string and assign J_j from M_{i_1} to M_{i_2}.

For a scheduling string $[3, 5, 1, 2, 6, 4]$ and a machine assignment string $[M_1, M_1, M_2, M_1, M_1, M_2]$. When $\mathcal{N}_1$ and $\mathcal{N}_2$ are applied, J_5 and J_4 are chosen on M_1. For $\mathcal{N}_3$ and $\mathcal{N}_4$, J_5 on M_1 and J_6 on M_2 are selected. Fig.2 gives the examples of $\mathcal{N}_1$,$\mathcal{N}_2$,$\mathcal{N}_3$,$\mathcal{N}_4$.

EDA generates a new solution by sampling the probability model. Θ is the set of all jobs.

The detailed steps of EDA for new solution are given in Algorithm 3.

In the above procedure, the chosen good colonies are improved by local search and the selected poor solutions are made better by global search of EDA, that is, two methods are used according to solution quality and global search ability and local search ability can be balanced well.

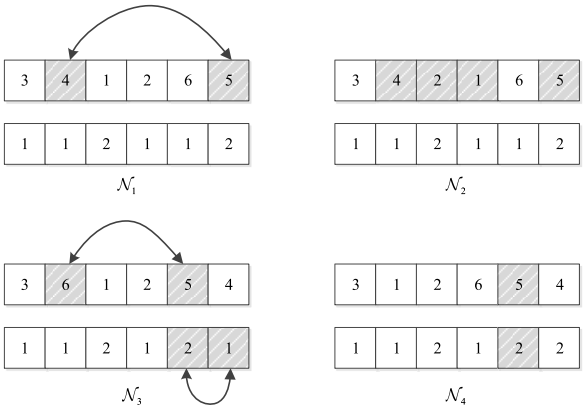


FIGURE 2. Examples for four neighborhood structures.

Algorithm 3 EDA Generate a New Solution

1: Calculate workload of all machines and decide the minimum workload of all machines.
2: **if** more than one machine has the minimum workload **then**
3: Randomly choose a machine M_i.
4: **else**
5: Directly select the machine M_i with minimum workload.
6: **end if**
7: Determine the last job on machine M_i, supposing the job is J_l. If there is no job, set a virtual job 0.
8: Select J_j by using roulette wheel based on probabilities on lth or $0th$ row of $\rho^i(g)$ and assign J_j to M_i. $\Theta = \Theta \backslash \{J_j\}$.
9: Let the jth column of $\rho^i(g)$, $i = 1, 2, \cdots, m$ to be zeros and normalize each row of the matrix.
10: **if** Θ is not empty **then**
11: Go to Step1.
12: **else**
13: Stop the algorithm.
14: **end if**

E. EMPIRE AGGRESSION AND IMPERIALIST COMPETITION

A new step named empire aggression is added into NICA by local search of imperialist for plundering a randomly chosen colony.

Empire aggression is as follows. For each imperialist k, randomly select one of four neighborhood structures $\mathcal{N}_1$, $\mathcal{N}_2$,$\mathcal{N}_3$, $\mathcal{N}_4$ and acts on the imperialist k, a new solution z is obtained, then a colony λ is randomly chosen, if z dominates colony λ, then colony λ is displaced by solution z and this colony is moved from its empire to empire k.

After the process of empire aggression, the cost of all countries is calculated again and imperialist of each empire is updated if a colony dominates its imperialist.

To implement imperialist competition, total cost TC_k is first defined by

$$TC_k = c_k + \xi \frac{\sum_{i \in H_k} c_i}{NC_k} \tag{9}$$

where ξ is a positive number. We set $\xi = 0.1$.

Then the normalized total cost NTC_k, power TP_k and a probability vector V are first computed by

$$NTC_k = \max_{l \in Q} \{TC_l\} - TC_k \tag{10}$$

$$TP_k = NTC_k / \sum_{l \in Q} NTC_l \tag{11}$$

$$V = TP - R = \left[TP_1 - r_1, TP_2 - r_2, \cdots, TP_{N_{imp}} - r_{N_{imp}}\right] \tag{12}$$

where $R = \left[r_1, r_2, \cdots, r_{imp}\right]$ is a random number vector, r_i follows uniform distribution in [0,1] and Q is the set of all imperialists.

Finally, a winning empire k with the biggest $TP_k - r_k$ is decided, EDA is applied and a new solution z is generated, if z dominates the weakest colony of the weakest empire, then the weakest colony is replaced with z and moved from the weakest empire to empire k; otherwise, the weakest colony is directly included into empire k.

In the above procedure, EDA is used to try to avoid adding directly the worst colony into the winning empire because the weakest colony is a worse solution and difficult to be improved even if it is included into a strong empire.

F. ALGORITHM DESCRIPTION

NICA is constructed by incorporating EDA into revolution and imperialist competition, respectively; moreover, there are different purposes. EDA is adopted in revolution to update colony and EDA is applied to avoid the direct inclusion of the weakest colony into the winning empire. Meanwhile, multi-elite individual guidance strategy is added in assimilation that colonies can move toward other imperialists. These are the great differences between NICA and other ICAs [33], [44], [45]. In fact, ICA is seldom hybridized with EDA. This study gives an effective hybrid method.

The detailed procedure of NICA is shown in Algorithm 4.

The stopping condition is max_it, which is maximum number of objective function evaluations. Fig. 3 shows the flow chart of NICA.

With respect to the updating probability matrices, at each generation, we choose $0.1 \times N$ individuals with smallest cost from population P as elite solutions according to Wang and Zheng [18]. Probability matrices are updated by the usage of of these elite solutions.

$$\rho^i_{j,k}(g+1) = (1-\alpha)\,\rho^i_{j,k}(g) + \frac{\alpha}{0.1 \times N} \sum_{z=1}^{0.1 \times N} I^z_{i,j,k} \tag{13}$$

Algorithm 4 NICA

1: Produce an initial population P by heuristics and random way, let $g = 1$.
2: Construct initial empires and probability matrices.
3: **while** The stopping criterion is not met **do**
4: Execute assimilation and revolution in each empire.
5: Perform empire aggression, calculate cost of all countries and exchange.
6: Execute imperialist competition and update the probability matrices, $g = g + 1$.
7: **end while**

where $\alpha \in (0, 1)$ is the learning rate, and $I^z_{i,j,k}$ is a 0-1 variable. In the zth solution, If job J_k is right after job J_j on machine M_i, $I^z_{i,j,k}$ is one; Otherwise $I^z_{i,j,k}$ is equal to zero.

IV. COMPUTATIONAL EXPERIMENTS

Extensive experiments are conducted on a set of problems to test the performance of NICA for UPMSP with PM and SDST. All experiments are implemented by using matlab2015b and run on 16.0G RAM 2.80GHz CPU PC.

A. TEST INSTANCES, METRICS AND COMPARATIVE ALGORITHMS

Test sets of small, medium and large scale [11,15] can be selected to evaluate the algorithmic performance. There exist 66 instances. For small instances, $n \in \{10, 12\}$ and $m \in \{2, 3\}$. For medium instances, $n \in \{20, 30\}$ and $m \in \{2, 3, 4\}$. For large instances, $n \in \{50, 100, 150, 200\}$ and $m \in \{10, 15, 20\}$. $p_{ij} \in [1, 99]$. There are three types of setup times, which are chosen from $[1, 9]$, $[1, 99]$ and $[1, 124]$. The processing time and setup times of small and large scales are taken from http://www.cima.uadec.mx/instancias/. The medium scale is generated randomly according to the literature [12]. $w_i \in [1, 99]$, the length of processing interval is decided by

$$u_i = (1 + \delta) \times \max_{\forall j} \left\{s_{i0j} + p_{ij} + s_{ij0}\right\} \tag{14}$$

Duedate d_j of job J_j is computed by

$$d_j = (1 + 2\delta) \times \sum_{i=1}^{m} p_{ij}/m \tag{15}$$

where δ is a random number between 0 and 1.

Two metrics are used. Metric DI_R [46] is applied to measure the convergence performance by computing the distance of the non-dominated set Ω_l relative to a reference set Ω^*.

$$DI_R(\Omega_l) = \frac{1}{|\Omega^*|} \sum_{y \in \Omega^*} \min\left\{\sigma_{xy} | x \in \Omega_l\right\} \tag{16}$$

where σ_{xy} is the distance between a solution x and a reference solution y in the normalized objective space. The reference set Ω^* is composed of the non-dominated set solutions in $\bigcup_l \Omega_l$.

The smaller $DI_R(\Omega_l)$ is, the better the algorithm is. $DI_R(\Omega_l) = 0$ means that algorithm l provides all members of the set Ω^*.

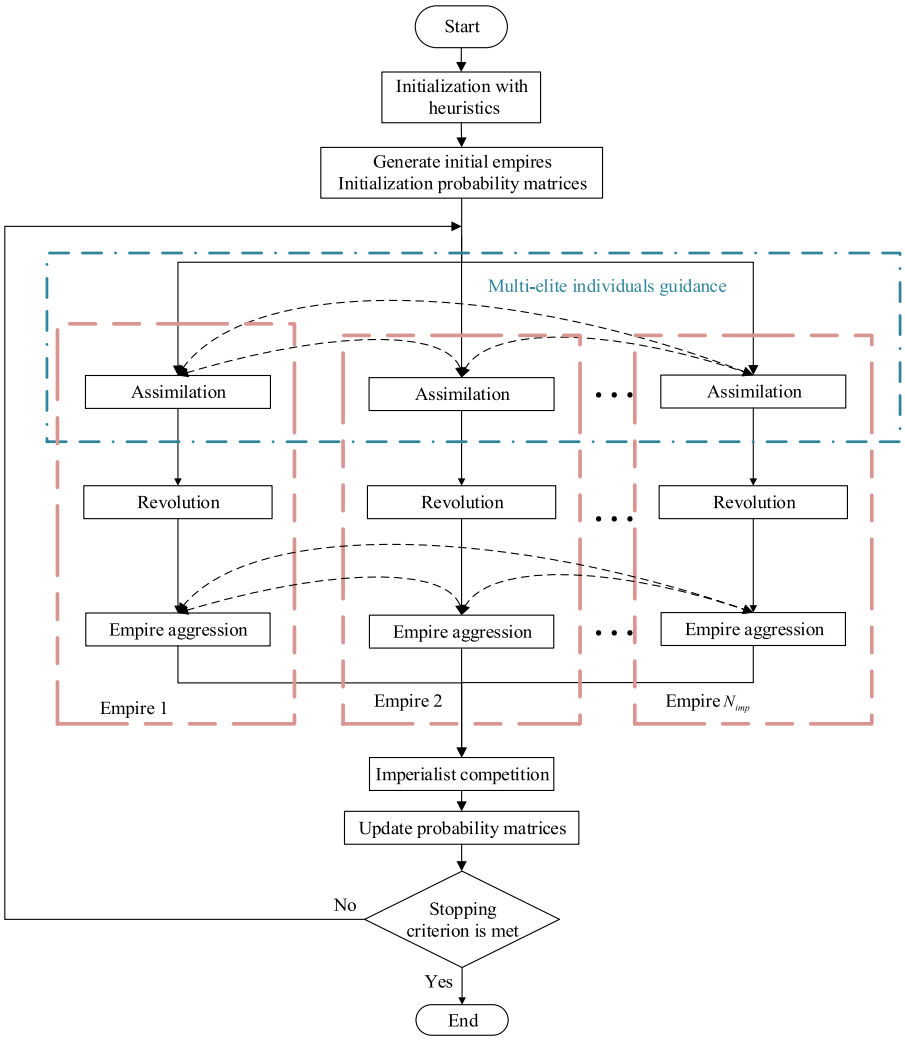


FIGURE 3. The flow chart of the NICA.

Metric ρ [47] indicates the ratio of number of the elements in the set $\{x \in \Omega_l \,|x \in \Omega^*\}$ to $|\Omega^*|$.

As stated above, multi-objective UPMSP with PM and SDST is seldom investigated and there are no existing comparative algorithms. In this study, we compare NICA with multi-objective multi-point simulated annealing (MOMSA [48]) and multi-objective harmony search (MOHS) algorithm [49].

MOMSA is proposed for solving multi-objective UPMSP by simultaneously minimising makespan, total weighted completion time and total weighted tardiness. The effectiveness of MOMSA is verified. Meanwhile, MOMSA can be directly applied to solve multi-objective UPMSP with PM and SDST by considering setup times and maintenance in decoding process.

As for MOHS, it presented to minimize the makespan, tardiness penalties and the purchasing cost of machines simultaneously. The effectiveness of MOHS is verified by the comparison with multi-objective particle swarm optimization, NSGA-II, and multi-objective ACO. MOHS can also be directly used to solve multi-objective UPMSP with PM and SDST by adding setup times and maintenance in decoding process.

TABLE 1. Parameters and their levels.

Parameter	Factor level 1	2	3
max_it	50000	100000	150000
N	60	100	140
N_{imp}/N	5%	10%	15%
p_r	0.3	0.4	0.5
α	0.05	0.1	0.15

B. PARAMETER SETTINGS

NICA has five important parameters: maximum number of objective function evaluations max_it, population scale N, the ratio of number of imperialists to population scale N_{imp}/N, revolutionary probability p_r and learning rate α.

A three-level design of experiment is designed on an instance with 20 machines and 200 jobs. Table 1 gives the

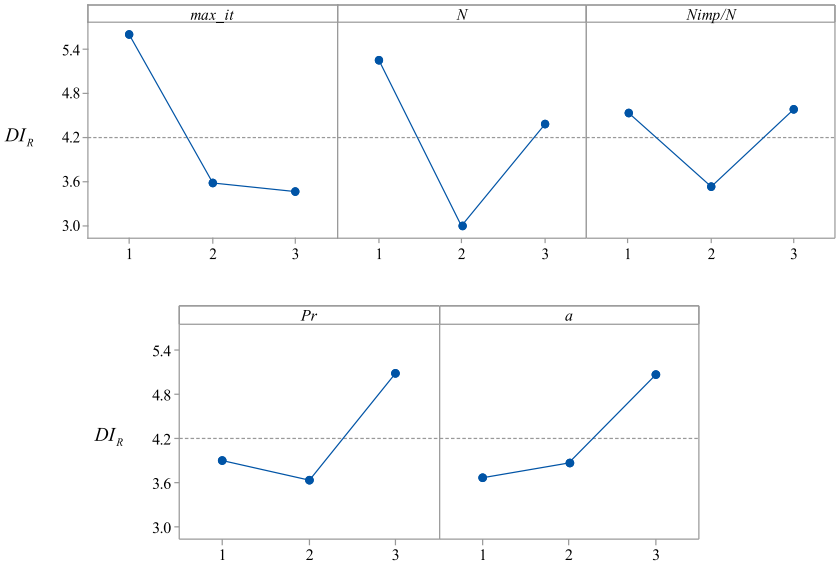


FIGURE 4. Main effect plot of DI_R.

TABLE 2. The orthogonal array $L_{27}(3^5)$.

test	Factor max_it	N	N_{im}/N	p_r	α	DI_R
1	1	1	1	1	1	7.1719
2	1	1	1	1	2	7.4340
3	1	1	1	1	3	5.3581
4	1	2	2	2	1	1.2813
5	1	2	2	2	2	2.7740
6	1	2	2	2	3	5.3252
7	1	3	3	3	1	6.9085
8	1	3	3	3	2	6.0239
9	1	3	3	3	3	8.1721
10	2	1	2	3	1	3.6524
11	2	1	2	3	2	4.0758
12	2	1	2	3	3	6.7036
13	2	2	3	1	1	1.3809
14	2	2	3	1	2	1.0899
15	2	2	3	1	3	4.7550
16	2	3	1	2	1	6.2098
17	2	3	1	2	2	1.5876
18	2	3	1	2	3	2.6529
19	3	1	3	2	1	3.0956
20	3	1	3	2	2	5.4946
21	3	1	3	2	3	4.2586
22	3	2	1	3	1	0.8150
23	3	2	1	3	2	4.6413
24	3	2	1	3	3	4.8614
25	3	3	2	1	1	2.4762
26	3	3	2	1	2	1.7654
27	3	3	2	1	3	3.6311

settings of each parameter at each level. Orthogonal arrays of different parameter levels are given in Table 2. For each group of parameters, NICA runs independently 20 times. DI_R is used to evaluate the results of each parameter combination, where the reference set Ω^* is composed of the non-dominated solutions from the union set of all sets Ω of NICA. The results are shown in Table 2. Table 3 describes the average DI_R of each parameter. The main effect plot of mean is shown in Fig. 4.

TABLE 3. Computational results.

Level	max_it	N	N_{imp}/N	p_r	α
1	5.605	5.249	4.526	3.896	3.666
2	3.568	2.992	3.521	3.631	3.876
3	3.449	4.381	4.575	5.095	5.080
Delta	2.157	2.258	1.055	1.464	1.414
Rank	2	1	5	3	4

From Table 3, we can see that N has the greatest influence on the result of the algorithm. Meanwhile, when max_it increases from 5×10^4 to 10^5, the results of NICA improve a lot. However, when max_it is greater than 10^5, NICA improves little. Therefore, in order to balance the performance and time consumption of the algorithm, we set $max\_it = 10^5$. The other parameters are as follows: $N = 100, N_{im}/N = 0.1, p_r = 0.4, \alpha = 0.05$.

C. STRATEGY EFFECTIVENESS ANALYSIS

NICA is built by the combination of EDA and ICA, inclusion of empire aggression and the usage of two heuristics in initialization.

Three variants of NICA are constructed. NICA1 is similar with NICA except that initial population is generated randomly. The difference between NICA and NICA2 lies in the elimination of EDA from NICA. In NICA3, colonies only can move toward its imperialist in assimilation. The parameter settings are as same as NICA. Each algorithm runs independently 20 times. Tables 4 and 5 show the computational results of NICA and its three variants and Fig. 5 and 6 describes the boxplot of four NICAs of the computational results. Table 9 shows the statistic results. We set confidence level as 95%. When $p\_value < 0.05$, the difference between algorithms is significant.

TABLE 4. Computational results of four NICAs on metric DI_R.

Instance	NICA	NICA1	NICA2	NICA3	Instance	NICA	NICA1	NICA2	NICA3
10×2(1_9)	0.0000	0.3825	0.1460	0.0000	50×15(1_9)	0.2043	10.0000	0.0000	0.2590
10×2(1_99)	0.0000	0.0000	0.4800	0.0000	50×15(1_99)	0.7840	8.0536	0.2562	3.3930
10×2(1_124)	0.0000	0.0000	0.0000	0.0000	50×15(1_124)	1.7976	9.0410	0.2830	2.5694
10×3(1_9)	0.0000	0.0000	0.0000	3.3152	50×20(1_9)	0.0477	11.8955	5.0031	0.4497
10×3(1_99)	0.0000	0.0000	0.0000	0.0000	50×20(1_99)	0.0000	8.7074	2.0417	2.3822
10×3(1_124)	0.0000	0.0000	0.0000	0.0000	50×20(1_124)	0.3301	8.8831	1.5311	2.1880
12×2(1_9)	0.0000	0.0000	0.0000	0.0039	100×10(1_9)	0.0000	6.3490	2.2251	7.9039
12×2(1_99)	0.0000	0.0000	0.1968	0.0000	100×10(1_99)	0.2322	5.2647	0.3458	6.0342
12×2(1_124)	0.0000	0.0000	0.0000	0.0000	100×10(1_124)	0.2806	9.4852	0.9484	9.1783
12×3(1_9)	0.0000	0.0000	0.0000	0.0000	100×15(1_9)	0.6405	9.5939	0.0000	2.4701
12×3(1_99)	0.0000	0.0000	0.0000	0.0000	100×15(1_99)	0.0000	12.5962	1.1528	5.2234
12×3(1_124)	0.0000	0.0000	0.0000	0.0000	100×15(1_124)	0.3237	12.4443	0.4740	5.5834
20×2(1_9)	0.0000	0.3795	0.6592	1.6707	100×20(1_9)	0.0000	8.2289	0.6493	1.2719
20×2(1_99)	1.0110	2.2082	0.1437	3.1516	100×20(1_99)	0.1206	12.4146	0.0539	3.8011
20×2(1_124)	4.6431	2.8205	3.5728	5.9318	100×20(1_124)	0.0530	13.0413	0.4685	4.0801
20×3(1_9)	1.6904	0.7165	1.4484	1.7683	150×10(1_9)	0.0000	1.8022	1.2948	10.4275
20×3(1_99)	0.0000	2.7823	1.7507	4.4656	150×10(1_99)	0.3788	8.4613	1.0355	8.8508
20×3(1_124)	1.8117	2.1596	0.0782	4.3984	150×10(1_124)	0.4544	6.6253	0.0000	10.0427
20×4(1_9)	3.1032	0.0000	0.6002	0.8600	150×15(1_9)	0.2046	8.7413	0.8010	6.5614
20×4(1_99)	0.4522	1.2779	0.9523	3.1438	150×15(1_99)	0.4652	8.9968	0.4576	10.4382
20×4(1_124)	3.7423	2.3807	3.1629	0.8483	150×15(1_124)	0.4052	8.7039	0.2385	6.9621
30×2(1_9)	0.1077	0.0802	0.2131	3.5726	150×20(1_9)	0.0371	3.0067	0.2810	4.1015
30×2(1_99)	2.4269	1.3504	4.7049	7.4319	150×20(1_99)	0.3803	9.7331	0.3291	4.0516
30×2(1_124)	0.4598	1.3767	1.1579	8.2878	150×20(1_124)	0.0000	11.5918	1.4708	5.5979
30×3(1_9)	0.1591	0.3036	0.3679	2.6852	200×10(1_9)	0.0727	2.1183	0.3581	6.6099
30×3(1_99)	0.0000	2.9710	1.1159	8.2740	200×10(1_99)	0.0000	9.1799	0.3609	9.8850
30×3(1_124)	0.9373	2.0141	0.5731	6.9018	200×10(1_124)	0.0000	10.1328	1.5130	11.0151
30×4(1_9)	0.4278	1.0437	1.1178	2.7501	200×15(1_9)	0.2619	7.6809	0.4539	8.8713
30×4(1_99)	0.3297	2.8043	1.3630	7.1927	200×15(1_99)	0.3958	11.3594	0.3608	5.8255
30×4(1_124)	0.0000	2.5666	1.4828	8.0796	200×15(1_124)	0.2921	12.3418	0.1180	6.7241
50×10(1_9)	0.0000	4.9155	1.3527	3.0004	200×20(1_9)	0.7737	11.0259	0.9844	4.9045
50×10(1_99)	0.0000	5.5101	2.6848	6.5826	200×20(1_99)	0.3396	12.1416	0.8722	3.1242
50×10(1_124)	1.7367	4.7583	0.1669	6.3164	200×20(1_124)	0.0000	11.5774	1.0819	4.8273

TABLE 5. Computational results of four NICAs on metric ρ.

Instance	NICA	NICA1	NICA2	NICA3	Instance	NICA	NICA1	NICA2	NICA3
10×2(1_9)	1.0000	0.5000	0.6667	1.0000	50×15(1_9)	0.0000	0.0000	1.0000	0.0000
10×2(1_99)	1.0000	1.0000	0.6667	1.0000	50×15(1_99)	0.4000	0.0000	0.6000	0.0000
10×2(1_124)	1.0000	1.0000	1.0000	1.0000	50×15(1_124)	0.3333	0.0000	0.3333	0.3333
10×3(1_9)	1.0000	1.0000	1.0000	0.6667	50×20(1_9)	0.5000	0.0000	0.5000	0.0000
10×3(1_99)	1.0000	1.0000	1.0000	1.0000	50×20(1_99)	1.0000	0.0000	0.0000	0.0000
10×3(1_124)	1.0000	1.0000	1.0000	1.0000	50×20(1_124)	0.7500	0.0000	0.2500	0.0000
12×2(1_9)	1.0000	1.0000	1.0000	0.8750	100×10(1_9)	1.0000	0.0000	0.0000	0.0000
12×2(1_99)	1.0000	1.0000	0.8000	1.0000	100×10(1_99)	0.6667	0.0000	0.3333	0.0000
12×2(1_124)	1.0000	1.0000	1.0000	1.0000	100×10(1_124)	0.7500	0.0000	0.2500	0.0000
12×3(1_9)	1.0000	1.0000	1.0000	1.0000	100×15(1_9)	0.0000	0.0000	1.0000	0.0000
12×3(1_99)	1.0000	1.0000	1.0000	1.0000	100×15(1_99)	1.0000	0.0000	0.0000	0.0000
12×3(1_124)	1.0000	1.0000	1.0000	1.0000	100×15(1_124)	0.4000	0.0000	0.6000	0.0000
20×2(1_9)	1.0000	0.5000	0.5000	0.0000	100×20(1_9)	1.0000	0.0000	0.0000	0.0000
20×2(1_99)	0.2000	0.2000	0.7000	0.0000	100×20(1_99)	0.3333	0.0000	0.6667	0.0000
20×2(1_124)	0.5000	0.0000	0.5000	0.0000	100×20(1_124)	0.7500	0.0000	0.2500	0.0000
20×3(1_9)	0.0000	0.8571	0.4286	0.1429	150×10(1_9)	1.0000	0.0000	0.0000	0.0000
20×3(1_99)	1.0000	0.3333	0.0000	0.0000	150×10(1_99)	0.8750	0.0000	0.1250	0.0000
20×3(1_124)	0.3333	0.0000	0.8333	0.0000	150×10(1_124)	0.0000	0.0000	1.0000	0.0000
20×4(1_9)	0.5000	1.0000	0.7500	0.0000	150×15(1_9)	0.6250	0.0000	0.3750	0.0000
20×4(1_99)	0.8000	0.0000	0.6000	0.0000	150×15(1_99)	0.5000	0.0000	0.5000	0.0000
20×4(1_124)	0.0000	0.0000	0.6667	0.3333	150×15(1_124)	0.5000	0.0000	0.5000	0.0000
30×2(1_9)	0.7500	0.2500	0.0000	0.0000	150×20(1_9)	0.6667	0.0000	0.3333	0.0000
30×2(1_99)	0.0000	0.8333	0.0000	0.1667	150×20(1_99)	0.6000	0.0000	0.4000	0.0000
30×2(1_124)	0.6000	0.4000	0.0000	0.0000	150×20(1_124)	1.0000	0.0000	0.0000	0.0000
30×3(1_9)	0.3529	0.2941	0.4706	0.0000	200×10(1_9)	0.8571	0.0000	0.1429	0.0000
30×3(1_99)	1.0000	0.0000	0.2000	0.0000	200×10(1_99)	1.0000	0.0000	0.0000	0.0000
30×3(1_124)	0.3750	0.1250	0.5000	0.0000	200×10(1_124)	1.0000	0.0000	0.0000	0.0000
30×4(1_9)	0.4000	0.3000	0.3000	0.0000	200×15(1_9)	0.6000	0.0000	0.4000	0.0000
30×4(1_99)	0.6667	0.0000	0.3333	0.0000	200×15(1_99)	0.5000	0.0000	0.5000	0.0000
30×4(1_124)	1.0000	0.0000	0.0000	0.0000	200×15(1_124)	0.5000	0.0000	0.5000	0.0000
50×10(1_9)	1.0000	0.0000	0.0000	0.0000	200×20(1_9)	0.4545	0.0000	0.4545	0.0909
50×10(1_99)	1.0000	0.0000	0.0000	0.0000	200×20(1_99)	0.5000	0.0000	0.5000	0.0000
50×10(1_124)	0.4000	0.0000	0.6000	0.0000	200×20(1_124)	1.0000	0.0000	0.0000	0.0000

It can be found that NICA has smaller DI_R than NICA1 on 49, and is only worse than NICA1 on 6 instances. As for metric ρ, NICA1 is better than or equal to NICA1 on 63 instances. The inclusion of two heuristics really improves the performance of NICA. With respect to NICA2, it is inferior to NICA on two metrics, DI_R of NICA2 is worse

TABLE 6. Computational results of three algorithms on metric DI_R.

Instance	NICA	MOMSA	MOHS	Instance	NICA	MOMSA	MOHS
10×2(1_9)	0.0000	2.6912	1.6351	50×15(1_9)	0.3933	10.0000	0.0000
10×2(1_99)	0.0212	3.3447	1.0609	50×15(1_99)	0.2890	10.1319	1.6053
10×2(1_124)	0.0000	0.0000	0.0000	50×15(1_124)	0.1089	9.0468	0.6931
10×3(1_9)	0.2072	8.4859	0.0000	50×20(1_9)	0.5590	10.0000	0.0000
10×3(1_99)	0.0000	5.6463	2.1024	50×20(1_99)	1.0110	11.0446	0.5909
10×3(1_124)	0.0000	0.0000	0.0000	50×20(1_124)	0.2761	8.4349	1.2385
12×2(1_9)	0.0000	1.4324	2.1581	100×10(1_9)	0.0000	1.7919	8.7100
12×2(1_99)	0.4475	3.8867	0.0000	100×10(1_99)	0.0000	2.8958	10.3288
12×2(1_124)	0.0000	1.3341	0.0000	100×10(1_124)	0.0000	3.7777	11.1843
12×3(1_9)	0.0000	11.5818	0.0000	100×15(1_9)	0.0000	4.2518	4.2268
12×3(1_99)	0.1040	11.2290	0.1040	100×15(1_99)	0.0000	3.3999	7.2465
12×3(1_124)	0.0000	10.4323	0.4152	100×15(1_124)	0.0000	3.7997	6.2226
20×2(1_9)	2.2574	6.0520	0.1333	100×20(1_9)	0.0000	4.2071	3.3536
20×2(1_99)	0.0850	4.0544	1.1221	100×20(1_99)	3.2661	10.5749	3.2661
20×2(1_124)	0.0000	11.3641	7.9837	100×20(1_124)	0.0000	10.1383	8.5274
20×3(1_9)	0.5733	7.2591	0.0986	150×10(1_9)	0.0000	5.9242	10.3052
20×3(1_99)	0.5922	8.6500	1.5853	150×10(1_99)	0.6868	3.1913	8.5808
20×3(1_124)	1.4804	5.1416	1.7202	150×10(1_124)	0.0000	3.5127	11.7134
20×4(1_9)	3.3244	6.5003	0.1601	150×15(1_9)	0.0000	3.8193	9.5188
20×4(1_99)	0.0000	4.4580	3.7910	150×15(1_99)	0.0000	9.8245	9.0907
20×4(1_124)	1.6373	4.9586	2.1883	150×15(1_124)	1.5014	2.1535	9.2158
30×2(1_9)	0.1121	1.0350	2.5899	150×20(1_9)	0.0000	3.2177	9.1028
30×2(1_99)	0.6167	2.4422	2.5550	150×20(1_99)	0.0000	4.4694	7.4879
30×2(1_124)	0.0000	9.7382	2.9347	150×20(1_124)	0.0000	9.3124	10.0068
30×3(1_9)	0.7801	2.4328	0.8346	200×10(1_9)	0.0000	1.0405	7.2231
30×3(1_99)	0.0000	3.7650	4.6557	200×10(1_99)	0.0000	1.7449	8.3192
30×3(1_124)	0.0000	9.0756	6.4662	200×10(1_124)	0.0000	2.8798	8.5723
30×4(1_9)	1.4599	5.8679	0.2006	200×15(1_9)	0.0000	6.1653	10.2354
30×4(1_99)	0.0000	8.1080	2.3859	200×15(1_99)	0.0000	4.3656	12.4223
30×4(1_124)	0.0000	6.7953	5.0432	200×15(1_124)	0.0000	7.8485	10.8799
50×10(1_9)	0.0000	8.0455	1.9108	200×20(1_9)	0.0790	1.8284	6.5147
50×10(1_99)	0.0000	11.9349	6.5149	200×20(1_99)	0.0000	6.0408	8.6687
50×10(1_124)	0.0000	9.2539	2.6606	200×20(1_124)	0.7999	2.1502	12.5339

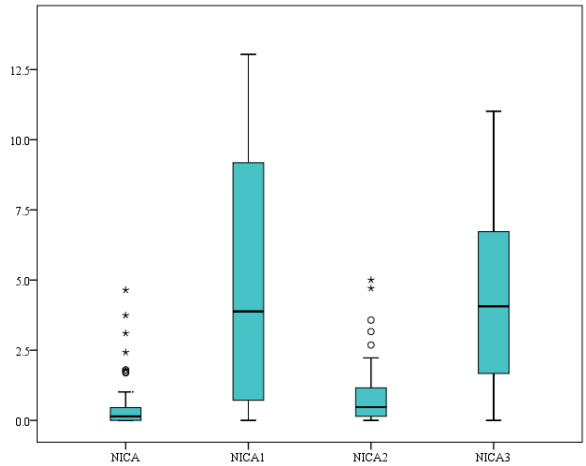


FIGURE 5. Boxplot of four NICAs on metric DI_R.

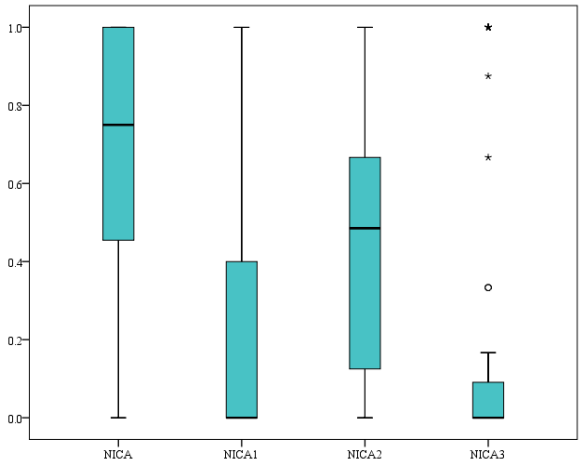


FIGURE 6. Boxplot of four NICAs on metric ρ.

than or equal to that of NICA2 on 47 instances and ρ of NICA1 is bigger than or equal to that of NICA2 on 52 instances, that is, the hybridization of ICA with EDA are effective and efficient. As stated in Tables 4 and 5, NICA performs better than NICA3. NICA produces smaller DI_R than or identical DI_R with NICA3 on 64 instances and obtains bigger ρ than NICA3 on 63 instances. The multi-elite individuals guidance strategy is really necessary. The same conclusions also can be drawn by the statistical results in Table 9 and Fig. 5 and 6, thus, three new strategies of NICA have positive impacts on its performances.

D. COMPARATIVE ANALYSIS WITH OTHER ALGORITHMS

To analyze the superiority of NICA in solving UPMSP with PM and SDST, we compare NICA with MOMSA and MOHS. The parameters of MOMSA and MOHS are directly adopted from Lin and Ying [48] and Shahidi-Zadeh *et al.* [49] expect the termination condition. Three algorithms have the same termination condition: $max\_it = 10^5$. Each algorithm runs independently 20 times. Tables 6, 7 and 8 list the results and computational time of NICA and two comparative algorithms. Fig. 7 and 8 shows the boxplot of the computational results.

TABLE 7. Computational results of three algorithms on metric ρ.

Instance	NICA	MOMSA	MOHS	Instance	NICA	MOMSA	MOHS
10×2(1_9)	1.0000	0.3333	0.3333	50×15(1_9)	0.0000	0.0000	1.0000
10×2(1_99)	0.8000	0.4000	0.6000	50×15(1_99)	0.6667	0.0000	0.3333
10×2(1_124)	1.0000	1.0000	1.0000	50×15(1_124)	0.6000	0.0000	0.4000
10×3(1_9)	0.6667	0.0000	1.0000	50×20(1_9)	0.0000	0.0000	1.0000
10×3(1_99)	1.0000	0.2500	0.2500	50×20(1_99)	0.6000	0.0000	0.4000
10×3(1_124)	1.0000	1.0000	1.0000	50×20(1_124)	0.6000	0.0000	0.4000
12×2(1_9)	1.0000	0.3750	0.3750	100×10(1_9)	1.0000	0.0000	0.0000
12×2(1_99)	0.8333	0.3333	1.0000	100×10(1_99)	1.0000	0.0000	0.0000
12×2(1_124)	1.0000	0.6667	1.0000	100×10(1_124)	1.0000	0.0000	0.0000
12×3(1_9)	1.0000	0.0000	1.0000	100×15(1_9)	1.0000	0.0000	0.0000
12×3(1_99)	0.8000	0.0000	0.8000	100×15(1_99)	1.0000	0.0000	0.0000
12×3(1_124)	1.0000	0.0000	0.0000	100×15(1_124)	1.0000	0.0000	0.0000
20×2(1_9)	0.2727	0.1818	0.9091	100×20(1_9)	1.0000	0.0000	0.0000
20×2(1_99)	0.8000	0.0000	0.2000	100×20(1_99)	0.5000	0.0000	0.5000
20×2(1_124)	1.0000	0.0000	0.0000	100×20(1_124)	1.0000	0.0000	0.0000
20×3(1_9)	0.4000	0.0000	0.6000	150×10(1_9)	1.0000	0.0000	0.0000
20×3(1_99)	0.8750	0.0000	0.1250	150×10(1_99)	0.7778	0.2222	0.0000
20×3(1_124)	0.5000	0.0000	0.5000	150×10(1_124)	1.0000	0.0000	0.0000
20×4(1_9)	0.6667	0.0000	0.6667	150×15(1_9)	1.0000	0.0000	0.0000
20×4(1_99)	1.0000	0.0000	0.0000	150×15(1_99)	1.0000	0.0000	0.0000
20×4(1_124)	0.5714	0.0000	0.4286	150×15(1_124)	0.6667	0.3333	0.0000
30×2(1_9)	0.8571	0.1429	0.0000	150×20(1_9)	1.0000	0.0000	0.0000
30×2(1_99)	0.5714	0.1429	0.2857	150×20(1_99)	1.0000	0.0000	0.0000
30×2(1_124)	1.0000	0.0000	0.0000	150×20(1_124)	1.0000	0.0000	0.0000
30×3(1_9)	0.6250	0.0000	0.3750	200×10(1_9)	1.0000	0.0000	0.0000
30×3(1_99)	1.0000	0.0000	0.0000	200×10(1_99)	1.0000	0.0000	0.0000
30×3(1_124)	1.0000	0.0000	0.0000	200×10(1_124)	1.0000	0.0000	0.0000
30×4(1_9)	0.4000	0.0000	0.7000	200×15(1_9)	1.0000	0.0000	0.0000
30×4(1_99)	1.0000	0.0000	0.0000	200×15(1_99)	1.0000	0.0000	0.0000
30×4(1_124)	1.0000	0.0000	0.0000	200×15(1_124)	1.0000	0.0000	0.0000
50×10(1_9)	1.0000	0.0000	0.0000	200×20(1_9)	0.9167	0.0833	0.0000
50×10(1_99)	1.0000	0.0000	0.0000	200×20(1_99)	1.0000	0.0000	0.0000
50×10(1_124)	1.0000	0.0000	0.0000	200×20(1_124)	0.6667	0.3333	0.0000

TABLE 8. Computational time (seconds) of three algorithms.

Instance	NICA	MOMSA	MOHS	Instance	NICA	MOMSA	MOHS
10×2(1_9)	8.39	4.19	12.08	50×15(1_9)	22.81	26.85	25.23
10×2(1_99)	7.57	4.74	10.29	50×15(1_99)	21.03	23.88	31.12
10×2(1_124)	7.74	4.11	12.09	50×15(1_124)	19.76	28.3	25.97
10×3(1_9)	8.47	4.84	12.16	50×20(1_9)	25.61	29.35	36
10×3(1_99)	8.41	6.03	10.90	50×20(1_99)	22.66	23.68	32.51
10×3(1_124)	6.51	4.63	7.20	50×20(1_124)	23.01	22.72	29.71
12×2(1_9)	8.16	6.14	12.20	100×10(1_9)	28.91	24.72	38.42
12×2(1_99)	7.94	3.63	8.60	100×10(1_99)	28.07	24.04	38.17
12×2(1_124)	6.92	3.40	8.96	100×10(1_124)	28.22	23.98	38.57
12×3(1_9)	8.13	4.70	12.08	100×15(1_9)	35.99	30.4	40.19
12×3(1_99)	7.44	4.49	9.41	100×15(1_99)	34.94	25.51	37.83
12×3(1_124)	7.45	4.02	10.32	100×15(1_124)	34.36	26.18	39.83
20×2(1_9)	12.65	11.05	20.77	100×20(1_9)	41.18	46.82	40.3
20×2(1_99)	12.11	10.98	18.46	100×20(1_99)	38.45	26.47	39.94
20×2(1_124)	12.47	7.17	16.95	100×20(1_124)	39.11	26.55	38.04
20×3(1_9)	14.05	8.38	20.12	150×10(1_9)	41.14	30.35	45.23
20×3(1_99)	11.93	8.18	20.10	150×10(1_99)	41.45	43.69	46.08
20×3(1_124)	11.81	7.87	19.78	150×10(1_124)	41.23	40.21	44.66
20×4(1_9)	11.70	8.87	20.12	150×15(1_9)	38.92	44.19	44.75
20×4(1_99)	12.60	8.69	19.38	150×15(1_99)	37.98	47.55	44.49
20×4(1_124)	12.71	14.92	13.38	150×15(1_124)	41.05	42.57	45.88
30×2(1_9)	12.73	8.21	19.72	150×20(1_9)	43.68	49.95	45.44
30×2(1_99)	12.30	7.78	15.93	150×20(1_99)	43.70	44.07	44.09
30×2(1_124)	13.23	9.72	15.44	150×20(1_124)	44.15	42.27	45.02
30×3(1_9)	12.83	8.45	20.25	200×10(1_9)	48.08	55.43	57.68
30×3(1_99)	12.46	8.15	19.99	200×10(1_99)	46.94	53.16	56.12
30×3(1_124)	13.28	8.98	21.32	200×10(1_124)	47.31	54.53	56.75
30×4(1_9)	13.51	9.14	20.28	200×15(1_9)	57.07	56.7	62.66
30×4(1_99)	12.83	12.37	20.65	200×15(1_99)	58.07	60.59	60.17
30×4(1_124)	12.42	8.86	19.58	200×15(1_124)	60.82	52.03	64
50×10(1_9)	18.64	23.42	31.82	200×20(1_9)	61.10	61.13	62.18
50×10(1_99)	18.44	22.63	32.39	200×20(1_99)	59.34	53.49	61.65
50×10(1_124)	22.81	16.61	31.83	200×20(1_124)	61.55	53.01	63.67

As stated in Tables 6 and 7, MOMSA obtains better DI_R than NICA on only 2 instance and MOMSA is inferior to NICA on ρ on 62 instances. With respect to MOHS, NICA gets smaller results of DI_R than MOHS on 51 instances and has greater ρ than MOHS on 51 instances. Moreover, DI_R of NICA is 0 on 40 instances and NICA provides all members

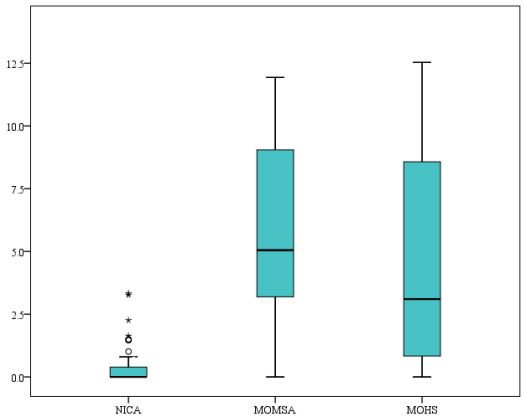


FIGURE 7. Boxplot of three algorithms on metric DI_R.

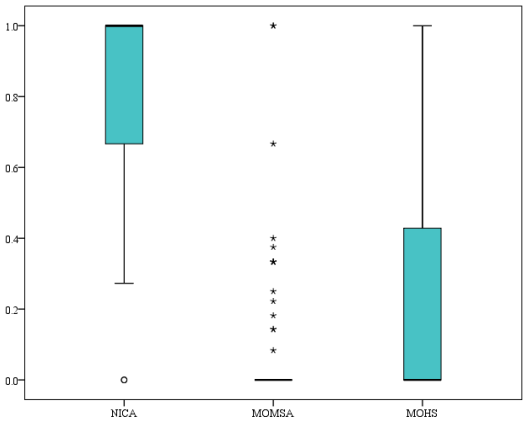


FIGURE 8. Boxplot of three algorithms on metric ρ.

of the set Ω^* on 40 instances. We also can find from the Fig. 7 and 8 that NICA has better performance than other algorithms. The statistical results in Table 9 are in agreement with this conclusion. Thus, NICA can get better results than other two algorithms on most of instances in similar computation times and has promising advantages in solving UPMSP with PM and SDST.

TABLE 9. Results of paired sample t-test

t-test	p-value (DI_R)	p-value (ρ)
t-test (NICA, NICA1)	0.000	0.000
t-test (NICA, NICA2)	0.007	0.001
t-test (NICA, NICA3)	0.000	0.000
t-test (NICA, MOMSA)	0.000	0.000
t-test (NICA, MOHS)	0.000	0.000

Colonies moving toward other imperialists can avoid the algorithm falling into local optimum. Meanwhile, different strategies in revolution are beneficial to make good balance between global search and local search. New added step and imperialist competition can make full use of good solutions. Besides, initialization can guarantee that the search of NICA starts with good initial population. Based on the above analysis, it can be concluded that NICA can effectively solve the UPMSP with PM and SDST.

V. CONCLUSION

UPMSP with PM and UPMSP with SDST are often studied; however, UPMSP with PM and SDST is seldom considered. In this paper, a new novel algorithm called NICA is proposed to solve UPMSP with PM, SDST and the minimization of makespan and total tardiness. Two heuristics are designed to initialize population. Multi-elite individuals guidance strategy is designed in assimilation that colonies can move toward other imperialists. EDA is adopted in revolution and imperialist competition for different purposes. A novel step named empire aggression is introduced by local search of imperialist for plundering a randomly chosen colony. Extensive experiments are conducted and the computational results show that NICA provides promising results for the considered UPMSP.

UPMSP with practical processing constraints such as SDST is an important one and extensively exists in the actual manufacturing systems. In the near future, we will continue to focus on this kind of problem and apply some meta-heuristics such as shuffled frog-leaping algorithm and teaching-learning-based optimization to solve it. Energy-efficient distributed scheduling in multiple factories is also our future topics. We will investigate distributed scheduling problems in unrelated parallel machines environment.

REFERENCES

[1] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," *Eur. J. Oper. Res.*, vol. 47, no. 3, pp. 271–292, Aug. 1990.

[2] R. A. F. Jans and Z. Degraeve, "An industrial extension of the discrete lot-sizing and scheduling problem," *IIE. Trans.*, vol. 36, pp. 47–58, Jan. 2004.

[3] C. Silva and J. M. Magalhaes, "Heuristic lot size scheduling on unrelated parallel machines with applications in the textile industry," *Comput. Ind. Eng.*, vol. 50, pp. 76–89, May 2006.

[4] M. Pattloch, G. Schmidt, and M. Y. Kovalyov, "Heuristic algorithms for lotsize scheduling with application in the tobacco industry," *Comput. Ind. Eng.*, vol. 39, pp. 235–253, Apr. 2001.

[5] H. Wang and A. Bahram, "Effective heuristic for large-scale unrelated parallel machines scheduling problems," *Omega*, vol. 83, pp. 261–274, Mar. 2019.

[6] G. Li, M. Q. Liu, S. P. c, and D. H. Xu, "Parallel-machine scheduling with machine-dependent maintenance periodic recycles," *Int. J. Prod. Econ.*, vol. 186, pp. 1–7, Apr. 2017.

[7] J. Yoo and I. S. Lee, "Parallel machine scheduling with maintenance activities," *Comput. Ind. Eng.*, vol. 101, pp. 361–371, Nov. 2016.

[8] J. He, Q. Li, and D. Xu, "Scheduling two parallel machines with machine-dependent availabilities," *Comput. Oper. Res.*, vol. 72, pp. 31–42, Aug. 2016.

[9] J. B. Wang and C. M. Wei, "Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties," *Appl. Math. Comput.*, vol. 217, pp. 8093–8099, Jun. 2011.

[10] A. Gara-Ali, G. Finke, and M. L. Espinouse, "Parallel-machine scheduling with maintenance: Praising the assignment problem," *Eur. J. Oper. Res.*, vol. 252, pp. 90–97, Jul. 2016.

[11] D. L. Yang, T. C. E. Cheng, S. J. Yang, and C. J. Hsu, "Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities," *Comput. Oper. Res.*, vol. 39, pp. 1458–1464, Jul. 2012.

[12] O. Avalos-Rosales, F. Angel-Bello, A. Álvarez, and Y. Cardona-valdés, "Including preventive maintenance activities in an unrelated parallel machine environment with dependent setup times," *Comput. Ind. Eng.*, vol. 123, pp. 364–377, Sep. 2018.

[13] M. Tavana, Y. Zarook, and F. J. Santos-Arteaga, "An integrated three-stage maintenance scheduling model for unrelated parallel machines with aging effect and multi-maintenance activities," *Comput. Ind. Eng.*, vol. 83, pp. 226–236, May 2015.

[14] S. J. Wang and M. Liu, "Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning," *J. Manuf. Sys.*, vol. 37, pp. 182–192, Oct. 2015.

[15] A. Allahverdi, J. N. Gupta, and T. Aldowaisan, "A review of scheduling research involving setup considerations," *Omega*, vol. 27, pp. 219–239, Apr. 1999.

[16] R. G. Parker, R. H. Deana, and R. A. Holmes, "On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent changeover costs," *AIIE Trans.*, vol. 9, pp. 155–160, Jun. 1977.

[17] M. E. Kurz and R. G. Askin, "Heuristic scheduling of parallel machines with sequence-dependent set-up times," *Int. J. Prod. Res.*, vol. 39, pp. 3747–3769, Jan. 2001.

[18] E. Vallada and R. Ruiz, "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times," *Eur. J. Oper. Res.*, vol. 211, pp. 612–622, Jun. 2011.

[19] J. P. Arnaout, G. Rabad, and R. Musa, "A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times," *J. Int. Manuf.*, vol. 21, no. 6, pp. 693–701, Dec. 2010.

[20] L. Wang, S. Wang, and X. L. Zheng, "A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 3, pp. 235–246, Jul. 2016.

[21] R. O. M. Diana, M. F. D. F. Filho, S. R. D. Souza, and J. F. D. A. Vitor, "An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation," *Neurocomputing.*, vol. 163, pp. 94–105, Sep. 2015.

[22] A. E. Ezugwu and F. Akutsah, "An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times," *IEEE Access*, vol. 4, pp. 54459–54478, 2018.

[23] L. Fanjul-Peyro, R. Ruiz, and F. Perea, "Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times," *Comput. Oper. Res.*, vol. 101, pp. 173–182, Jan. 2019.

[24] E. Caniyilmaz, B. Benli, and M. S. IIkay, "An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date," *Int. J. Adv. Manuf. Technol.*, vol. 77, pp. 2105–2115, Apr. 2015.

[25] G. Bektur and T. Sarac, "A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server," *Comput. Oper. Res.*, vol. 103, pp. 46–63, Mar. 2019.

[26] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 4661–4667.

[27] 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.

[28] A. Rabiee, M. Sadeghi, and J. Aghaei, "Modified imperialist competitive algorithm for environmental constrained energy management of microgrids," *J. Clean Prod.*, vol. 202, pp. 273–293, Nov. 2018.

[29] R. Akbari, M. Abbasi, F. Faghihi, S. M. Mirvakili, and J. Mokhtari, "A novel multi-objective optimization method, imperialist competitive algorithm, for fuel loading pattern of nuclear reactors," *Prog. Nuclear Energy*, vol. 108, pp. 391–397, Sep. 2018.

[30] A. Fathy and H. Rezk, "Parameter estimation of photovoltaic system using imperialist competitive algorithm," *Renew. Energy* vol. 111, pp. 307–320, Oct. 2018.

[31] D. M. Lei, M. Li, and L. Wang, "A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold," *IEEE Trans. Cybern.* vol. 49, no. 3, pp. 1097–1109, Mar. 2019.

[32] Z. X. Pan, D. M. Lei, and Q. Y. Zhang, "A new imperialist competitive algorithm for multiobjective low carbon parallel machines scheduling," *Math. Problems Eng.*, vol. 2018, pp. 1–13, Apr. 2018.

[33] 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.

[34] S. Hany, Z. Mostafa, F. Hamed, and M. Iraj, "Simulated imperialist competitive algorithm in two-stage assembly flow shop with machine breakdowns and preventive maintenance," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 230, no. 5, pp. 934–953, May 2016.

[35] M. Yazdani, S. M. Khalili, and F. Jolai, "A parallel machine scheduling problem with two-agent and tool change activities: An efficient hybrid metaheuristic algorithm," *Int. J. Comput. Int. Manuf.*, vol. 29, no. 10, pp. 1075–1088, Oct. 2016.

[36] M. Abedi, H. Seidgar, H. Fazlollahtabar, and R. Bijani, "Bi-objective optimisation for scheduling the identical parallel batch-processing machines with arbitrary job sizes, unequal job release times and capacity limits," *Int. J. Prod. Res.*, vol. 53, no. 6, pp. 1680–1711, Mar. 2015.

[37] P. Zhang, Y. L. Lv, and J. Zhang, "An improved imperialist competitive algorithm based rolling horizon strategy for photolithography machines scheduling," *IFAC-PapersOnLine.*, vol. 49, no. 12, pp. 1295–1300, 2016.

[38] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA, USA: Kluwer Press, 2002.

[39] J. Q. Gao, G. X. He, and Y. S. Wang, "A new parallel genetic algorithm for solving multiobjective scheduling problems subjected to special process constraint," *Int. J. Adv. Manuf. Technol.*, vol. 43, pp. 151–160, Jul. 2009.

[40] M. Afzalirad and J. Rezaein, "A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches," *Appl. Soft Comput.*, vol. 50, pp. 109–123, Jan. 2017.

[41] G. H. Wu, C. Y. Cheng, H. I. Yang, and C. T. Chena, "An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines," *Appl. Soft Comput.*, vol. 71, pp. 1072–1084, Oct. 2018.

[42] M. Tigane, M. Dahane, and M. Boudhar, "Multiobjective approach for deteriorating jobs scheduling for a sustainable manufacturing system," *Int. J. Adv. Manuf. Technol.*, vol. 101, pp. 1939–1957, Apr. 2018.

[43] K. Deb, S. S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSGA-II," in *Proc. Int. Conf. Parallel Problem Solving Nature*. New York, NY, USA: Springer, 2000 pp. 849–858.

[44] D. Li, C. Zhang, G. Tian, X. Shao, and Z. Li, "Multiobjective program and hybrid imperialist competitive algorithm for the mixed-model two-sided assembly lines subject to multiple constraints," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 48, no. 1, pp. 119–129, Jan. 2018.

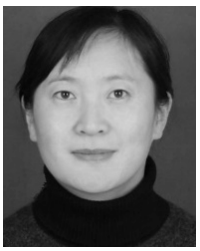
[45] A. H. Banisadr, M. Zandieh, and I. Mahdavi, "A hybrid imperialist competitive algorithm for single-machine scheduling problem with linear earliness and quadratic tardiness penalties," *Int. J. Adv. Manuf. Technol.*, vol. 65, nos. 5–8, pp. 981–989, 2013.

[46] J. D. Knowles and D. W. Corne, "On metrics for comparing nondominated sets," in *Proc. Congr. Evol. Comput.*, Honolulu, TX, USA, 2002, pp. 711–716.

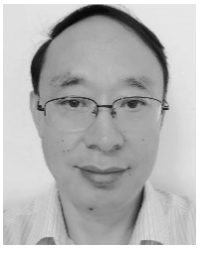
[47] D. Lei, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 37, nos. 1–2, pp. 157–165, Apr. 2008.

[48] S.-W. Lin and K.-C. Ying, "A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems," *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1065–1076, 2015.

[49] B. Shahidi-Zadeh, R. Tavakkoli-Moghaddam, A. Taheri-Moghadam, and I. Rastgar, "Solving a bi-objective unrelated parallel batch processing machines scheduling problem: A comparison study," *Comput. Oper. Res.*, vol. 88, pp. 71–90, Sep. 2017.



MEI WANG received the M.S. degree in electronic and information and the Ph.D. degree in communications and information systems from the Naval Aeronautical Engineering Institute, Yantai, China, in 2003 and 2008, respectively. She is currently a Professor with the Laboratory of Image Processing and Pattern Recognition, Yantai Vocational College, Yantai. Her current research interests include manufacturing systems intelligent optimization, production scheduling, image processing, object detection, and pattern recognition.



GUOHUA PAN graduated in computer and application from the Shandong University of Technology, Jinan, China, in 1988, where he is currently a Senior Engineer with the Yantai Public Security Bureau. His current research interests include scheduling, electronic information processing, video object detection, and other directions.