A Self-Learning Discrete Jaya Algorithm for Multiobjective Energy-Efficient Distributed No-Idle Flow-Shop Scheduling Problem in Heterogeneous Factory System

Fuqing Zhao, Ru Ma, and Ling Wang

***Abstract*—In this study, a self-learning discrete Jaya algorithm (SD-Jaya) is proposed to address the energy-efficient distributed no-idle flow-shop scheduling problem (FSP) in a heterogeneous factory system (HFS-EEDNIFSP) with the criteria of minimizing the total tardiness (TTD), total energy consumption (TEC), and factory load balancing (FLB). First, the mixed-integer programming model of HFS-EEDNIFSP is presented. An evaluation criterion of FLB combining the energy consumption and the completion time is introduced. Second, a self-learning operators selection strategy, in which the success rate of each operator is summarized as knowledge, is designed for guiding the selection of operators. Third, the energy-saving strategy is proposed for reducing the TEC. The energy-efficient no-idle FSP is transformed to be an energy-efficient permutation FSP to search the idle times. The speed of operations which adjacent are idle times is reduced. The effectiveness of SD-Jaya is tested on 60 benchmark instances. On the quality of the solution, the experimental results reveal that the efficacy of the SD-Jaya algorithm outperforms the other algorithms for addressing HFS-EEDNIFSP.**

***Index Terms*—Energy-efficient distributed no-idle flow-shop scheduling problem (FSP), energy-saving strategy, heterogeneous factory system, Jaya algorithm, self-learning operation selection strategy (SLOS).**

I. Introduction

With the development of globalization, distributed manufacturing is becoming a typical scenario in intelligent manufacturing, and the energy-efficient distributed flow-shop scheduling problems (EEDFSPs) have attracted significant attention from the viewpoint of sustainable development and green manufacturing [1], [2]. In EEDFSP, the jobs are processed in distributed factories. Therefore, the assignment of factories and the processing sequence of jobs are both considered. In addition, the time consumption and the energy consumption are optimized simultaneously, which brings additional difficulty [3], [4]. In the various research literature, the studies of EEDFSP are based on the assumption that the processing mode and factories are identical. However, it is impossible to ensure that machines are exactly uniform, especially in the mechanical wear and tear. Mechanical wear and tear lead to increasing energy consumption. Besides, the load balancing between factories is not researched. Considering the impact of mechanical wear and tear and optimizing load balancing between factories on EEDFSP is significant for the research of the EEDFSP.

The flow-shop scheduling problem (FSP) [5]–[7], a classical NP-hard problem, is divided into a permutation flow shop problem [8] (PFSP), no-idle flow shop problem [9] (NIFSP), blocking flow shop problem [10] (BFSP), no-wait flow shop problem [11] (NWFSP), etc. The NIFSP generally exists in the manufacturing industry, such as integrated circuits, textiles, and pottery, since a two-machine NIFSP with the total completion time criterion is presented in [12]. Various methods are proposed for addressing the NIFSP with the optimization goals of minimizing the completion time ($C_{\max}$), the total tardiness (TTD), total flowtime, etc. In [13], a novel hybrid discrete particle swarm optimization (HDPSO) is proposed for addressing NIFSP with the criterion to minimize $C_{\max}$. The hybrid teaching-learning-based metaheuristic (HDTLM) is introduced to address the NIFSP with TTD criterion in [14]. According to [15], a bipopulation EDA that utilized two subpopulations generated by sampling the probability models is presented to balance the global exploration and the local exploitation for solving the NIFSP with the criteria of minimizing TTD. In [16], the mixed NIFSP with sequence-dependent setup time and total flowtime minimization is proposed.

With the development of manufacturing, the distributed flow shop scheduling problem (DFSP) becomes a typical scenario in manufacturing. The DFSP includes distributed PFSP [17] (DPFSP), distributed NIFSP [18] (DNIFSP), distributed BFSP [19], distributed NWFSP [20], etc. In various DFSP, the jobs are processed in F factories and each factory

Manuscript received January 16, 2021; revised April 17, 2021; accepted May 23, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62063021 and Grant 61873328; in part by the Lanzhou Science Bureau Project under Grant 2018-rc-98; and in part by the Public Welfare Project of Zhejiang Natural Science Foundation under Grant LGJ19E050001. This article was recommended by Associate Editor J. Chen. *(Corresponding authors: Fuqing Zhao; Ling Wang.)*

Fuqing Zhao and Ru Ma are with the School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China (e-mail: fzhao2000@hotmail.com; 312783672@qq.co).

Ling Wang is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: wangling@tsinghua.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2021.3086181.

Digital Object Identifier 10.1109/TCYB.2021.3086181

is assigned to one job at least. The F factories are totally identical. The DNIFSP as a common manufacturing model is proposed [21]. In [22], an iterated reference greedy is proposed to address DNIPFSP with the objective of minimizing the makespan. In [23], the distributed assembly no-idle flow shop scheduling problem (DANIFSP) is proposed with the objective of minimizing the makespan at the assembly stage. In [24], an effective scatter search-based memetic algorithm (SS-MA) is proposed to address the distributed assembly PFSP (DAPFSP). The above DFSP is based on the assumption of an identical factory. However, various distributed factories in actual manufacturing scenarios are heterogeneous. In [25], the distributed heterogeneous hybrid flow shop scheduling problem (DHHFSP) is proposed first. In DHHFSP, the machine number and processing capacity of machines are different. In [26], a distributed heterogeneous no-wait flow shop scheduling problem (DHNWFSP) with the criteria of minimizing the makespan is presented. In the DHNWFSP, the number of machines, machine technology, raw material supply, and transportation conditions is different in each factory. In [27], the energy-efficient scheduling of distributed flow shop with heterogeneous factories that shop types are varied in different factories are proposed for the first time. Research on the optimization of heterogeneous factories is more suitable with the actual production conditions.

Meanwhile, the research of FSP with energy efficient is meaningful for the sustainable development of the manufacturing industry [28]–[30]. To address the FSP with energy efficient, multiobjective evolutionary algorithms (MOEAs) are utilized. According to the evolutionary mechanism, the MOEA is divided into decomposition-based MOEA [31], domination-based MOEA [32], and indicator-based MOEA [33]. In [34], which is based on the thought of decomposition-based MOEA, a two-stage cooperative evolutionary algorithm (TS-CEA) considering the problem-specific knowledge is introduced to address the energy-efficient scheduling of no-wait flow-shop problem (EENWFSP). In [35], a hybrid ant colony algorithm (MHACO) is proposed to solve the energy-efficient scheduling for multiobjective two-stage flow shop with the optimization goal to minimize TEC and makespan. Other studies of the scheduling problem with energy efficient are summarized in [36]. In [37], a knowledge-based cooperative algorithm (KCA) is introduced to solve the EEDFSP. In [38], a multiobjective whale swarm algorithm (MOWSA) which holds advantages in population diversity and local search, is introduced to solve the EEDFSP. Meanwhile, a problem-dependent local search mechanism is designed. The energy-efficient multiobjective distributed no-idle FSP (EEDNIFSP) is only studied in [39], which proposes a collaborative optimization algorithm (COA) to solve the EEDNIFSP first. Except for COA, there is no optimization algorithm utilized to solve EEDNIFSP. Meanwhile, the mechanical wear and tear and the load balancing between factories are not considered.

Jaya algorithm as an algorithm-specific parameter-less evolutionary algorithm is proposed by Rao in [40] to address engineering optimization problems. After the proposal of Jaya, various variants of the Jaye algorithm are introduced in the literature. In [41], a Jaya-based parallel algorithm is proposed. An adaptive multiteam perturbation-guiding Jaya (AMTPG-Jaya) algorithm proposed in [42] divides the population into teams, and each team is updated by a formula to explore different areas of the search space. In [43], the Jaya algorithm with the xor operator is proposed to extend the Jaya algorithm for solving binary optimization problems. For the application of the Jaya algorithm, a detailed summary is made in [44]. In [45], the Jaya algorithm is utilized to address the permutation flow shop scheduling problem with the multiobjective of $C_{\max}$ and tardiness cost under due date constraints. The performance of Jaya is superior to the total enumeration method and simulated annealing (SA) algorithm. An improved Jaya (I-Jaya) algorithm proposed in [46] is presented to solve the flexible job-shop scheduling problem with transportation and setup times. Meanwhile, several problem-specific local search operators are proposed. In [47], the discrete Jaya (D-Jaya) algorithm is utilized to address the flexible job-shop rescheduling problem. Therefore, utilizing the Jaya algorithms to address the scheduling problems is feasible and effective.

According to the above description, the study of Jaya for addressing the energy-efficient distributed no-idle FSP in a heterogeneous factory system (HFS-EEDPFSP) with the criteria of minimizing the TTD, total energy consumption (TEC), and factory load balancing (FLB) is significant. The contributions are as follows.

1) The mixed-integer programming model of energy-efficient distributed no-idle FSP in a heterogeneous factory system (HFS-EEDNIFSP) is proposed. Factories are heterogeneous due to the different mechanical wear and tear of each machine. A new optimization goal, the FLB combined with the time consumption and energy consumption, is proposed.
2) A self-learning operation selection strategy (SLOS) is designed. In the algorithm and seven sequence-related operators, four speed-related operators are introduced. The historical success rate of each operator is summarized as knowledge to guide the algorithm for the self-learning operation selection.
3) The energy-saving strategy based on the characteristic of HFS-EEDNIFSP is utilized after the end of algorithm evolution. The energy-efficient no-idle FSP is transformed into an energy-efficient permutation FSP for searching the operation to reduce the speed and TEC.

The remainder of this work is organized as follows. In Section II, the description of HFS-EEDNIFSP is presented. The Jaya algorithm and the discrete Jaya algorithm are introduced in Section III. The self-learning discrete Jaya algorithm (SD-Jaya) algorithm is proposed in Section IV. In Section V, the experimental analysis of SD-Jaya is presented. The SD-Jaya algorithm for the energy-efficient distributed no-idle FSP (EEDNIFSP) with the criteria of minimizing the TTD and TEC is presented in Section VI. The SD-Jaya algorithm for HFS-EEDNIFSP with the criteria of minimizing the TTD, TEC, and FLB is presented in Section VII. Finally, conclusions are summarized, and future work is suggested in Section VIII.

II. Description of HFS-EEDNIFSP

A. Notation

In this section, the notations and meaning utilized in this study are listed as follows.

- J Set of n jobs. $J = \{1, 2, \ldots, n\}$.
- M Set of m machines in each factor. $M = \{M_1, M_2, \ldots, M_m\}$.
- U Set of s speeds $U = \{U_1, U_2, \ldots, U_s\}$ satisfying $U_1 < U_2 < \ldots < U_s$.
- A Coefficient of mechanical wear and tear $A = \{A_1, A_2, \ldots, A_h\}$ satisfying $A_1 < A_2 < \cdots < A_h$.
- π^f Processing sequence of n^f jobs in factory f. $\pi^f = \{\pi_1^f, \pi_2^f, \ldots, \pi_{n^f}^f\}$.
- π Processing sequence of n jobs, $\pi = \{\pi_1^1, \pi_2^1, \ldots, \pi_{n^1}^1; \ldots; \pi_1^n, \pi_2^n, \ldots, \pi_{n^f}^n;\}$
- $O_{j,\ i}$ Operation of job j on the machine i.
- $S_{j,i}$ Start time of $O_{j,i}$.
- $C_{j,i}$ Completion time of $O_{j,i}$.
- $p_{j,i}$ Standard processing time of $O_{j,i}$.
- $v_{j,i}$ Processing speed $v_{j,i} \in \{U_1, U_2, \ldots, U_s\}$ of $O_{j,i}$.
- τ Due date tightness factor.
- d_j Due date of job j. $d_j = \tau \sum_{i=1}^{m} p_{j,m}$.
- $t_{j,i}$ Real processing time $t_{j,i} = (p_{j,i}/v_{j,i})$ of $O_{j,i}$.
- $a_{f,i}$ Coefficient of mechanical wear and tear of the machine i in factory f.
- PP_{j,U_u} Energy consumption when the machine i runs with speed U_u per unit time. $\mathrm{PP}_{j,U_u} = 4 \times {U_u}^2$kw.
- $C_{\max}^f$ Makespan of factory f.
- TEC^f Total energy consumption in factory f.
- TEC Total energy consumption of the heterogeneous factory system.
- TTD^f Total tardiness in factory f.
- TTD Total tardiness of heterogeneous factory system.
- Fl^f Loads in factory f.
- FLB Factory load balancing.
- $C_{i,l,f}$ Continuous variable denoting the completion time of the job at position l in π^f on machine j.

B. Mixed-Integer Programming Model of HFS-EEDNIFSP

For the HFS-EEDNIFSP, n jobs are assigned to F factories. Every machine in factoris has a coefficient of mechanical wear and tear. The following constraints are satisfied in the HFS-EENIPFSP. Jobs are processed on M_1 to M_m in the order. Once jobs are processed on the machine, idle time is not allowed between two consecutive operations. Machines are not allowed to stop until all the jobs are completed. One of s speeds is selected to process an operation until the operation is completed and the switching time of speeds is neglected for any two consecutive operations. The higher speed results in shorter real processing time but larger energy consumption. Meanwhile, the higher the coefficient of the mechanical wear and tear, the higher the energy consumption.

Minimize {TTD, TEC, FLB}

Subjectto

$$\sum_{f=1}^{F}\sum_{j=1}^{n} x_{j,l,f} = 1 \quad \forall l \tag{1}$$

$$\sum_{i=1}^{m} x_{j,l,f} \le 1 \quad \forall f, l \tag{2}$$

$$\sum_{u=1}^{s} y_{j,i,u} = 1 \quad \forall j, i \tag{3}$$

$$\sum_{u=1}^{h} z_{f,i,a} = 1 \quad \forall f, i \tag{4}$$

$$t_{j,i} = p_{j,i}\sum_{u=1}^{s}\frac{y_{j,i,k}}{U_u} \quad \forall j, i \tag{5}$$

$$C_{1,1,f} = \sum_{j=1}^{n} x_{j,1,f} \cdot t_{j,1} \quad \forall f \tag{6}$$

$$C_{i,l,f} = C_{i,l-1,f} + \sum_{j=1}^{n} x_{j,l,f} \cdot t_{j,i} \quad \forall i, l \ge 2, f \tag{7}$$

$$C_{i,l,f} \ge C_{i-1,l,f} + \sum_{j=1}^{n} x_{j,l,f} \cdot t_{j,i} \quad \forall i \ge 2, l, f \tag{8}$$

$$C_{\max} \ge C_{i,l,f} \quad \forall i, l, f \tag{9}$$

$$C_{i,l,f} \ge 0 \quad \forall i, l, f \tag{10}$$

$$x_{j,l,f} \in \{0, 1\} \quad \forall j, l, f \tag{11}$$

$$y_{j,i,u} \in \{0, 1\} \quad \forall j, i, u \tag{12}$$

$$z_{f,i,a} \in \{0, 1\} \quad \forall f, i, a. \tag{13}$$

Constrain (1) indicates that every job is assigned to one but only one position of the job sequence in a certain factory. Constraint (2) ensures that no more than one job is processed per machine simultaneously. Constraint (3) implies that only one speed is utilized when processing an operation. Constraint (4) indicates that each machine is divided into one but only one coefficient of mechanical wear and tear. Constraint (5) gives the real processing time when speed is determined. Constraints (6)–(10) ensure the process conforms to the characteristics of NIFSP. Constraint (11) means that if job j is assigned to the factory f at the position l in π^f, then $x_{j,l,f} = 1$; Otherwise $x_{j,l,f} = 0$. Constraint (12) means that if job j is processed at speed U_u on machine M_i, the $y_{j,i,u} = 1$; otherwise, $y_{j,i,u} = 0$. Constraint (13) means that if the machine i in the factory f is assigned to coefficient a, then $z_{f,i,a} = 1$; Otherwise, $z_{f,i,a} = 0$

$$F_{f,1,i} = t_{\pi_1^f,i+1} \quad i = 1, 2, 3, \ldots, m-1, f = 1, 2, \ldots, F \tag{14}$$

$$F_{f,j,i} = \max\left\{F_{f,j-1,i} - t_{\pi_j^f,i}, 0\right\} + t_{\pi_j^f,i+1} \quad j = 2, 3, \ldots, n^f, i = 1, 2, \ldots, m-1 \quad f = 1, 2, \ldots, F \tag{15}$$

$$C_{f,\max} = C_{f,n^f} = \sum_{i=1}^{m-1} F_{f,n^f,i} + \sum_{j=1}^{n} t_{\pi_j^f,1} \quad f = 1, 2, \ldots, F \tag{16}$$

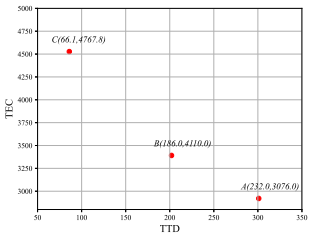


Fig. 1. Results of TTD and TEC.

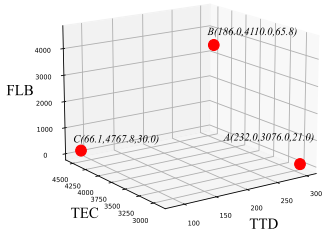


Fig. 2. Results of TTD, TEC, and FLB.

$$C_{f,\pi_j^f} = C_{f,\pi_{j+1}^f} - t_{\pi_{j+1}^f,m} \quad j = n^f - 1,\ n^f - 2, \ldots, 2 \tag{17}$$

$$\mathrm{TTD}^f = \sum_{j=1}^{n} \left(\max\left(C_{f,\pi_j^f} - d_{\pi_j^f} \right) \right) \tag{18}$$

$$\mathrm{TTD} = \max\left(\mathrm{TTD}^f \right), f = 1, \ldots, F \tag{19}$$

$$\mathrm{TEC}^f = \sum_{f=1}^{F} \sum_{i=1}^{m} \sum_{l=1}^{n} \left(\sum_{j=1}^{n} x_{j,l,f} \cdot t_{j,i} \cdot \sum_{u=1}^{s} \sum_{a=1}^{h} y_{j,i,u} \cdot \mathrm{pp}_{j,U_u} \cdot z_{f,i,a} \cdot A_h \right) \tag{20}$$

$$\mathrm{TEC} = \sum_{f=1}^{F} \mathrm{TEC}^f \tag{21}$$

$$\mathrm{Fl}^f = \alpha C_{f,\max} + (1-\alpha)\mathrm{TEC}^f \tag{22}$$

$$\mathrm{FLB} = \mathrm{std}\left(\mathrm{Fl}^1, \mathrm{Fl}^2, \mathrm{Fl}^f \right). \tag{23}$$

C. Objective Analysis of HFS-EEDNIFSP

There is an example to explain the relationship between the TTD, TEC, and FLB. In this explanation, jobs are processed in two factories in the heterogeneous factory system. The $U = \{1, 1.3, 1.55\}$. First, the processing speed is assigned to the maximum, random, and minimum. From Figs. 1 and 2, the TEC is decreased when the TTD is increased and the FLB change randomly. Thus, the TTD, TEC, and FLB are conflicting. The TTD, TEC, and FLB and to should be optimized simultaneously rather than aggregating into one objective using the linear weighted method.

III. Jaya Algorithm and Discrete Jaya Algorithm

In the Jaya algorithm, P initial solutions are randomly generated between the maximum and the minimum of each design variable. The population size is p, $(p = 1, 2, \ldots, P)$ and the number of design variable is d, $(d = 1, 2, \ldots, D)$. $O(x_{g,p,d})$ is the value of the dth variable for the pth candidate during the gth iteration. Then, $x_{g+1,p,d}$, which is the offspring of $x_{g,p,d}$ is generated by (24). $x_{g+1,p,d}$ is accepted when the fitness of $x_{g+1,p,d}$ is better than $x_{g,p,d}$. Otherwise, $x_{g,p,d}$ is reserved

$$x_{g+1,p,d} = x_{g,p,d} + r_1\left(x_{g,\mathrm{best},d} - \left|x_{g,p,d}\right|\right) - r_2\left(x_{g,\mathrm{worst},d} - \left|x_{g,p,d}\right|\right) \tag{24}$$

where $x_{g,\mathrm{best},d}$ and $x_{g,\mathrm{worst},d}$ represent the best and the worst solutions in the current population, respectively. r_1 and r_2 are random numbers in the range of 0–1.

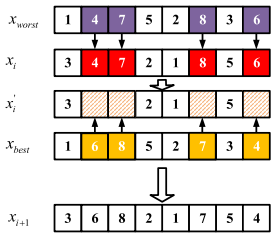


Fig. 3. Mutation strategy.

The mutation strategy in [43] is proposed to utilize the Jaya algorithm for addressing discrete optimization problems. In each generation, the individual with the best performance x_{best} and the individual with the worst performance x_{worst} are selected. The individual x_i is compared with x_{worst}. When jobs in the same positions are same, the positions are made empty. Then jobs at the corresponding positions of x_{best} are selected to fill the empty positions of x_i'. As shown in Fig. 3, the job sequences of x_{best}, x_i, and x_{worst} are $\{1, 4, 7, 5, 2, 8, 3, 6\}$, $\{3, 4, 7, 2, 1, 8, 5, 6\}$, and $\{1, 6, 8, 5, 2, 7, 3, 4\}$, respectively. The positions 2, 3, 6, 8 of x_i are the same as the jobs in positions of x_{worst}. Then x_i' is set as {3, none, none, 2, 1, none, 5, none}. A new individual $x_{i+1} = \{3, 6, 8, 2, 1, 7, 5, 4\}$ is generated.

IV. SD-Jaya Algorithm

A. Initialization Strategy

In the SD-Jaya algorithm, all jobs are distributed to factories randomly and ensure that there are at least two jobs in each subfactory. The speed of $O_{j,i}$ is initialized randomly in the range of U_1 to U_s. Then NEH is utilized to initialize the sequences in each factory. When the factory satisfies the criteria that if $\mathrm{TTD}^j > \mathrm{TTD}^i, i = 1, 2, 3, \ldots, F, \quad i \neq j$, the factory j is selected to be a critical factory. The TTD of HFS-EEDNIFSP is equivalent to the TTD of the critical factory.

In the critical factory, the population generation strategy (PG) is proposed to generate a population. In the population PG, N jobs are randomly selected in π^c, and N is a random integer number between two and the length of π^c. The positions of the selected jobs are set to empty. Then, the selected jobs are randomly reordered to generate a reference job sequence π^r. Insert the jobs into the original position of π^c in the order of π^r. As shown in Fig. 4, suppose the $\pi^c = \{8, 6, 5, 1, 2\}$, the job number selected randomly is 3

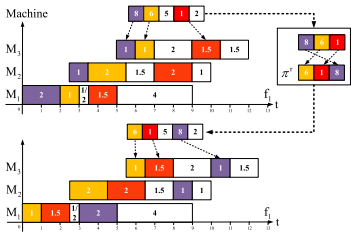


Fig. 4. Initialization strategy.

at the position 1, 2, and 4. The reference sequence generated randomly is π^r and $\pi^r = \{6, 1, 8\}$. Job 6 is to insert at position 1, job 1 is to insert at position 2, and job 8 is to insert at position 4. A new individual $\pi_{p_1} = \{6, 1, 5, 8, 2\}$ is generated. The remaining individuals are generated in the same way. The pseudocode of PG is shown as Algorithm 3 (supplementary material).

B. Operators and Self-Learning Operation Selection Strategy

1) Sequence-Related Operators: For tuning the job sequence, there are seven kinds of sequence-related operators, including 1) single factory swapping (SS); 2) single factory insertion (SI); 3) local insertion in the single factory (LSI); 4) local swapping in the single factory (LSS); 5) remove and reinserted in the single factory (RSI); 6) distributed factory swapping (DS); and 7) distributed factory insertion (DI). The operators are described as follows. π_b is before π_a. f_c is the critical factory and f_r is a random factory.

SS: In f_c, select two jobs π_b and π_a randomly and swap their position.

SI: In f_c, select two jobs π_b and π_a randomly. Then insert π_a in front of π_b.

LSI: In f_c, the job π_j, $(j = 1, 2, \ldots, n^c)$ is selected from π^c and $\pi^{c'} = \{\pi_1, \pi_2, \ldots, \pi_{j-1}, \pi_{j+1}, \ldots, \pi_{n^c}\}$. New individuals are generated by inserting π_j into every position in $\pi^{c'}$ except for position j. Then, the individual with the smallest TTD is accepted.

LSS: In f_c, the job π_j, $(j = 1, 2, \ldots, n^c)$ is selected from π^c. New individuals are produced by swapping π_j with all jobs in π^c except for the jobs next to π_j. The individual with the smallest TTD is reserved.

RSI: In the job sequence, the duplicate jobs are removed. Then missing jobs are inserted into all possible positions, and the best sequence is selected. The job sequence until all missing jobs are reinserted.

DS: The job π_j, $(j = 1, 2, \ldots, n^c)$ in f_c and the other job π_i, $(i = 1, 2, \ldots, n^r)$ in f_r are selected randomly, then swap π_j and π_i.

DI: The job π_j, $(j = 1, 2, \ldots, n^c)$ in f_c and the other job π_i, $(i = 1, 2, \ldots, n^r)$ in f_r are selected randomly. π_c is inserted in front of π_r.

2) Speed-Related Operators: Four speed-related operators, including 1) acceleration of single operation (SAS); 2) deceleration of single operation (SDS); 3) acceleration of random operations (SAR); and 4) deceleration of random operations (SDR) are utilized to tune the speed. R is a random integer number between 2 and M.

SAS: In f_r, an operation is selected randomly and accelerated to a higher level.

SDS: In f_r, an operation is selected randomly and decelerated to a lower level.

SAR: A job is selected randomly in f_c. Then R operations are accelerated to a higher level randomly.

SDR: A job is selected randomly in f_c. Then R operations are decelerated to a lower level randomly.

3) Self-Learning Operators Selection Strategy: For the above operators, the historical success rate of each operator is summarized as knowledge to guide the algorithm for the self-learning operation selection. When the success rate of the operator is zero, this operator will never be chosen. Therefore, it is necessary to transform the historical success rate into the knowledge that guides the self-learning selection of operators. Knowledge is described as (25) and (26). Finally, the operator is selected according to sk_g

$$\mathrm{sr}_g(\mathrm{op}_{i,j}) = \frac{\mathrm{se}_g(\mathrm{op}_{i,j})}{\mathrm{ss}_g(\mathrm{op}_{i,j})} + 0.5 \tag{25}$$

$$\mathrm{sk}_g(\mathrm{op}_{i,j}) = \frac{\mathrm{sr}_g(\mathrm{op}_{i,j})}{\sum_{i=1}^{O} \mathrm{sr}_g(\mathrm{op}_{i,j})} \tag{26}$$

where $\mathrm{ss}_g(\mathrm{op}_{i,j})$ is the executed number of the operator j in part i during generation g. $\mathrm{se}_{\mathrm{op}}$ is the success number of the operator. sr is the knowledge. sk_g is the selection rate.

The self-learning operator selection strategy, including $\mathrm{op}_1 = \{\mathrm{SS}, \mathrm{SI}\}$, $\mathrm{op}_2 = \{\mathrm{DS}, \mathrm{DI}\}$, $\mathrm{op}_3 = \{\mathrm{SAS}, \mathrm{SDS}\}$, and $\mathrm{op}_4 = \{\mathrm{SAR}, \mathrm{SDR}\}$. The rp_i, $i = 1, 2, \ldots, 4$ is a random number between 0 and 1. If $\mathrm{rp}_i < \mathrm{sp}_{i,1}$, the $\mathrm{op}_{i,1}$ is executed. Else, the $\mathrm{op}_{i,2}$ is selected.

C. Acceptance Criterion

For the above operations, there are three kinds of acceptance criteria, including 1) acceptance criterion 1 (AC1); 2) acceptance criterion 2 (AC2); and 3) acceptance criterion 3 (AC3).

AC1: The new individual is accepted if $(\mathrm{TTD}_{p_{i+1}}, \mathrm{TEC}_{p_{i+1}}) \succ (\mathrm{TTD}_{p_i}, \mathrm{TEC}_{p_i})$; Otherwise, the new individual is discarded.

AC2: The new individual is accepted if $\mathrm{TTD}_{p_{i+1}} < \mathrm{TTD}_{p_{i+1}}$; Otherwise, discard the new individual.

AC3: The new individual is accepted if $\mathrm{TTD}_{p_{i+1}} < \mathrm{TTD}_{p_i}$ or $\mathrm{TEC}_{p_{i+1}} < \mathrm{TEC}_{p_i}$; Otherwise, discard the new individual.

D. Load-Balancing Adjustment and Energy-Saving Strategies

A load-balancing adjustment strategy is introduced to reduce FLB. A job π_i, $(i = 1, 2, \ldots, n^c)$ in f_c and the other job π_j, $(j = 1, 2, \ldots, n^{\min})$ in $f_{\min}$ are selected randomly, then insert π_i in front of π_j. The pseudocode is shown as Algorithm 4 (supplementary material).

In the energy-saving strategy, the EEDNIFSP is converted to the EEDPFSP to search the operations be decelerated. As shown in Fig. 5, the EEDNIFSP is converted to the EEDPFSP, the idle time $I_{j,i}$ of each machine is calculated. The pseudocode is shown in Algorithm 1. The speed of the operation $O_{i,j}$ is

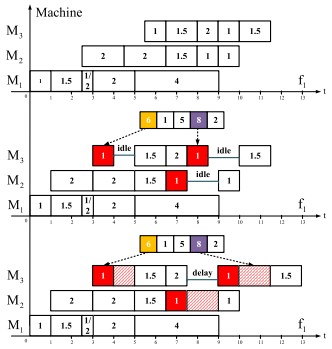


Fig. 5. Problem transformation.

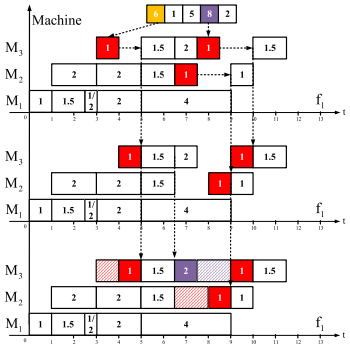


Fig. 6. Energy saving.

Algorithm 1: Calculate the Idle Time

```
for i frome 1 to M do
    for j from 1 to J − 1 do
        I_{j,i} = S_{j+1,i} − C_{j,i}
    end
end
```

adjusted when the idle time exists. However, not every operation before the idle time is decelerated. As shown in Fig. 5, when the operation of job eight in machine three is decelerated, the delay time in machine three is generated which reduces the TEC but increases the TTD. The energy-saving strategy ensures that TEC is reduced under TTD not increased.

As shown in Fig. 6, after converting the EEDNIFSP to the EEDPFSP and calculating the idle time of each machine. The operations $O_{i,j}$ is delayed $I_{j,i}$ to be processed. Then, calculate $\mathrm{Id}_{j,i}$ which is the difference of the start time of the job on M_{i+1} and the completion time on the M_i, $i = 1, 2, \ldots, m-1$. When the $\mathrm{Id}_{j,i} > 0$, the job j is decelerated until the termination condition: $\mathrm{ntim}_{j,i} - \mathrm{tim}_{\mathrm{j.i}} \leq I_{j,i}$ is satisfied. When the $\mathrm{Id}_{j,i} \leq 0$, the job $j-1$ is decelerated until the termination condition: $\mathrm{ntim}_{j,i} - \mathrm{tim}_{\mathrm{j.i}} \leq I_{j,i}$ is satisfied. The pseudocode is shown in Algorithm 2.

Algorithm 2: Energy Saving

```
for i from 1 to m do
    for j from 1 to n do
        if I_{j,i} > 0 then
            S_{j,i} = S_{j,i} + I_{j,i};
            C_{j,i} = C_{j,i} + I_{j,i};
        end
    end
end
for i from 1 to m − 1 do
    for j from 1 to n do
        Id_{j,i} = S_{j,i+1} − C_{j,i};
    end
end
for i from 1 to m do
    if I_{j,i} > 0 then
        Decelerate job 1 until the termination condition:
        ntim_{j,i} − tim_{j.i} ≤ I_{j,i} is satisfied.;
        for j from 2 to n do
            if I_{j,i} > 0 and Id_{j,i} > 0 then
                Decelerate job j until the termination
                condition: ntim_{j,i} − tim_{j.i} ≤ I_{j,i} is satisfied.
            else
                Decelerate the job j − 1 until the
                termination condition:
                ntim_{j,i} − tim_{j.i} ≤ I_{j,i} is satisfied.
            end
        end
    end
end
```

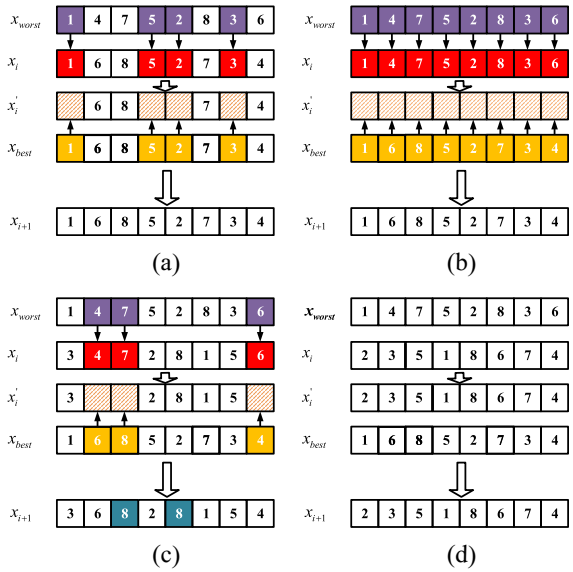


Fig. 7. Cases of SD-Jaya. (a) Case 1. (b) Case 2. (c) Case 3. (d) Case 4.

E. Framework of SD-Jaya

According to the mutation strategy of D-Jaya described in Section III, there are five cases when producing a new individual. Case 1 is that the new individual is the same as x_{best} when $x_i = x_{\mathrm{best}}$ as shown in Fig. 7(a).

Case 2 is $x_i = x_{\mathrm{worst}}$, the whole sequence of x_{worst} is replaced by x_{best} as shown in Fig. 7(b). Fig. 7(c) shows case 3, the job sequence with repetitive jobs is generated, which is an infeasible job sequence. As shown in Fig. 7(d), case 4, the new individual is the same as the old one due to the jobs of

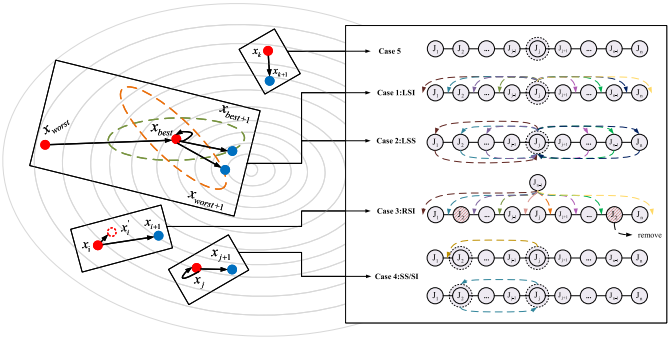


Fig. 8. Evolutionary process.

TABLE I
SCALES OF TA INSTANCES

Ta	61-70	71-80	81-90	91-100	101-110	111-120
Num	1	2	3	4	5	6
n	100	100	100	200	200	500
m	5	10	20	10	20	20

x_i and x_{worst} are not same in any position. A new feasible job sequence is generated in Fig. 3, which belongs to case 5.

As shown in Fig. 8, strategies for different cases are targeted. For case 1, the LSI and AC2 are executed, which is a local search of the insertion neighborhood about the best individual. As shown in Fig. 8, after the LSI, the $x_{\text{best}+1}$, which outperforms the x_{best} is produced. For case 2, the LSS and AC2 are executed, which is a local search of the swapping neighborhood about the best individual. Prefer the RSI and AC2 when the individual belongs to case 3. To increase the diversity of the population, when the individual belongs to case 4, the one of SS and SI is selected according to the self-learning operators selection strategy. The AC2 is utilized as the receiving criterion. When the individual is a new feasible solution that belongs to case 5, the AC2 is directly executed. Cases 1–3 are beneficial to individuals to explore more promising search areas and cases 4 and 5 maintain population diversity. The flowchart of SD-Jaya is shown in Fig. 9. The pseudocode is shown in Algorithm 5 (supplementary material).

V. EXPERIMENTAL ANALYSIS OF SD-JAYA

In this section, the benchmark instances, where $f = \{2, 3, 4, 5, 6, 7\}$, $n = \{100, 200, 500\}$, $m = \{5, 10, 20\}$, and $\tau = \{1, 2, 3\}$ are utilized to analyze the influence of parameters and the effectiveness of strategies proposed in SD-Jaya. The benchmark instances consist of 60 Ta problems. The scales of instances are shown in Table I.

The level of speed is $U = \{1.00, 1.30, 1.55, 1.75, 2.10\}$, and the level about coefficient of mechanical wear and tear is $A = \{1, 1.2, 1.4\}$. The α is set as 0.5. All algorithms are coded in python and run on a PC of Intel Core Xeon CPU E5-2630L 0 @ 2.00 GHz, 2.00 GHz (two processors)/16-GB(RAM) with the termination conditions as $n \times m \times 0.3$ s to ensure the fairness of the experiment. Each instance is run five times independently. In SD-Jaya, the selected parameter is $p \in \{5, 10, 15, 20\}$. The 60 benchmark instances with two factories and $\tau = 1$ are selected to test the performance of each parameter selection to obtain the Pareto archives A_i $(i = 1, 2 \ldots, 4)$. Then the final Pareto archives B which combines all 4 Pareto archives eliminated the dominated individuals. The evaluation metric response value (RV) is calculated to evaluate the performance of each parameter selection,. $\text{RV}_i = \text{CON}(i)/|A_i|$ and $\text{CON}(i) = |A_i^{'}|/|B|$. Where $A_i^{'} = A_i \cap B$. The larger value implies that the SD-Jaya is more effective under that parameter selection. From the result $\text{RV} = \{0.092, 0.079, 0.105, 0.053\}$, the p is set as 15.

TABLE II
C METRIC OF SD-JAYA, SD-JAYA-NPG, AND SD-JAYA-NS
$S_1 = C(\textit{SD-Jaya}, \textit{SD-Jaya-NPG})$, $S_1^{'} = C(\textit{SD-Jaya-NPG}, \textit{SD-Jaya})$,
$S_2 = C(\textit{SD-Jaya}, \textit{SD-Jaya-NS})$, $S_2^{'} = C(\textit{SD-Jaya-NS}, \textit{SD-Jaya})$

Num	initialization strategy		self-learning operation selection	
	S_1	$S_1^{'}$	S_2	$S_2^{'}$
1	**0.339**	0.294	0.253	**0.304**
2	**0.354**	0.317	**0.466**	0.254
3	0.326	**0.350**	**0.489**	0.173
4	**0.414**	0.245	**0.311**	0.200
5	**0.399**	0.244	**0.394**	0.334
6	0.297	**0.308**	**0.308**	0.201
Average	**0.355**	0.293	**0.370**	0.244

The overall nondominated vector generation (ONVG) and C Metric are utilized to evaluate the performance of the obtained Pareto set.

ONVG: The number of nondominated individuals in the final Pareto archive.

C Metric: The dominance relationship between the individuals in two Pareto archives A and B. The bigger C Metric is, the better the algorithm is. The calculation method of $C(A, B)$ is shown as follows:

$$C(A, B) = |\{b \in B | \exists a \in A, a \succ b \text{ or } a = b\}|/|B|.$$

A. Effectiveness of Population Generation Strategy

To analyse the effectiveness of the PG, the SD-Jaya is compared with SD-Jaya-NPG (the SD-Jaya algorithm removed the PG). In SD-Jaya-NPG, the population is initialized randomly. To evaluate the performance of SD-Jaya and SD-Jaya-NPG, the C Metric is calculated in all instance. As shown in Table II, although S_1 is smaller than $S_1^{'}$ on $\{n = 100, m = 20\}$ and $\{n = 500, m = 20\}$. The average of S_1 is bigger than $S_1^{'}$, which reflects the percentage of the solutions in SD-Jaya-NPG are dominated by or the same as the solutions in SD-Jaya is bigger than the percentage of the solutions in SD-Jaya are dominated by or the same as the solutions in SD-Jaya-NPG. The PG is effective for addressing HFS-EEDNIFSP.

B. Effectiveness of Self-Learning Operators Selection Strategy

The SD-Jaya without the SLOS called SD-Jaya-NS is compared with SD-Jaya to analyze the effectiveness of the SLOS. In SD-Jaya-NS, the operations of SS, SI, SRA, SRD, DS, DI, SSA, and SSD are selected randomly.

TABLE III
WILCOXON'S TEXT OF SD-JAYA, SD-JAYA-NS, SD-JAYA-NSP, AND SD-JAYA-NSQ WITH $f = 2$ FOR HFS-EEDNIFSP

SD-Jaya vs	τ	R+	R-	Z	p-value	$\alpha = 0.05$
SD-Jaya-NS	1	1303.5	526.50	-2.86	4.24E-03	**Yes**
	2	1187.0	466.00	-2.86	4.18E-03	**Yes**
	3	1259.5	510.50	-2.83	4.70E-03	**Yes**
SD-Jaya-NSP	1	1606.0	164.00	-5.44	5.27E-08	**Yes**
	2	1602.5	227.50	-5.06	4.17E-07	**Yes**
	3	1447.0	383.00	-3.92	9.00E-05	**Yes**
SD-Jaya-NSQ	1	1242.0	588.00	-2.41	1.61E-02	**Yes**
	2	1333.0	497.00	-3.08	2.09E-03	**Yes**
	3	1194.0	636.00	-2.05	4.00E-02	**Yes**

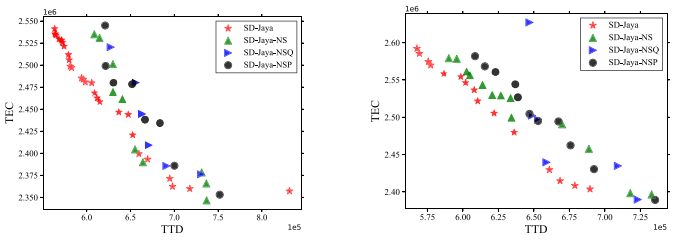


Fig. 9. Pareto front of four algorithms. (a) $\tau = 1, f = 5$, and $Ta = 111$. (b) $\tau = 1, f = 5$, and $Ta = 112$.

TABLE IV
ONVG OF SD-JAYA, SD-JAYA-NS, SD-JAYA-NSP, AND SD-JAYA-NSQ WITH $f = 2$ FOR HFS-EEDNIFSP

Num	SD-Jaya	SD-Jaya-NS	SD-Jaya-NSQ	SD-Jaya-NSP
1	**374.3**	350.3	342.8	345.6
2	**434.7**	390.1	386.8	400.3
3	**490.2**	448.9	424.3	453.9
4	**406.6**	372.8	349.3	371.9
5	**418.6**	401.9	355.1	392.2
6	**268.6**	237.8	198.9	235.5
Average	**398.8**	367.0	342.9	366.6

The C Metric is calculated in all instance. As shown in Table II, S_2 is outperformed $S_2^{'}$ in all instances except $\{n = 100, m = 50\}$ which illustrates that the Pareto set of SD-Jaya is superior to SD-Jaya-NS. The SLOS is significant for SD-Jaya. To analyze the characteristics of SLOS, the SLOS is divided into the SLOS of sequence-related operators (SLOS-SQ) and SLOS of speed-related operators (SLOS-SP). The effects of SLOS-SP optimizing TEC and SLOS-SQ optimizing TTD are analyzed, respectively. Therefore, the relative percentage increase (RPI) is introduced. The smaller the RPI, the better the performance of the algorithm

$$\text{RPI} = (T_r - T_b)/T_b * 100 \tag{27}$$

where T_r is the TEC of the rth solution in the algorithm, and T_b is the best solution among all algorithms.

In Fig. 9, the solutions in the Pareto front of SD-Jaya dominate the solutions in the Pareto front of SD-Jaya-NS, SD-Jaya-NSQ, and SD-Jaya-NSP. The number of nondominated solutions obtained by SD-Jaya is the largest from Table IV. Based on the above analysis, the SLOS is effective in addressing HFS-EEDNIFSP.

TABLE V
RPI OF SD-JAYA, SD-JAYA-NE, AND SD-JAYA-NL
$S_3 = C$(SD-JAYA, SD-JAYA-NE), $S_3^{'} = C$(SD-JAYA-NE, SD-JAYA), $S_4 = C$(SD-JAYA, SD-JAYA-NL), $S_4^{'} = C$(SD-JAYA-NL, SD-JAYA)

f	energy saving		load-balancing adjustment	
	S_3	$S_3^{'}$	S_4	$S_4^{'}$
2	**1.403**	4.862	**19018**	49077
3	**1.163**	6.357	**502.89**	1156.8
4	**0.857**	6.708	**246.77**	574.77
5	**0.632**	6.678	**119.15**	372.16
6	**0.558**	6.625	**81.983**	268.45
7	**0.505**	6.348	**66.080**	220.00
Average	**0.853**	6.263	**3339.2**	8611.6

C. Effectiveness of Energy-Saving Strategy

The SD-Jaya without the energy-saving strategy called SD-Jaya-NE is compared with SD-Jaya on the optimization goal of minimizing TEC to analyze the effectiveness of the energy-saving strategy. In SD-Jaya-NE, the energy-saving strategy is removed. To evaluate the performance of SD-Jaya and SD-Jaya-NE, the RPI is utilized to evaluate the effect of the energy-saving strategy.

As shown in Table V, the mean RPI of SD-Jaya on all scales and $\tau = \{1, 2, 3\}$ with the termination conditions as $n \times m \times 0.3$ s is smaller than the mean RPI of SD-Jaya-NE, which means that the SD-Jaya is outperformed SD-Jaya-NE on TEC. According to Fig. 21 (supplementary material) and Fig. 10. (a), the mean RPI of SD-Jaya is significantly better than the mean RPI of SD-Jaya-NE within a 95% confidence interval.

D. Effectiveness of Load-Balancing Adjustment Strategy

The load-balancing adjustment strategy is contributed to reducing the FLB. Therefore, the SD-Jaya without the load-balancing adjustment strategy called SD-Jaya-NL is compared to SD-Jaya to analyze the effectiveness of the load-balancing adjustment strategy with the optimization goal of minimizing FLB. In SD-Jaya-NL, the load-balancing adjustment strategy is removed. The RPI is adopted to analyze the effect of the load-balancing adjustment strategy. As shown in Table V, Fig. 10(b) and as shown in Fig. 22 (supplementary material). The mean RPI of SD-Jaya is not significantly better than the mean RPI of SD-Jaya-NL within a 95% confidence interval. The mean value of SD-Jaya has outperformed the mean value of SD-Jaya-NL.

VI. COMPARISON OF SD-JAYA WITH COA IN EEDNIFSP

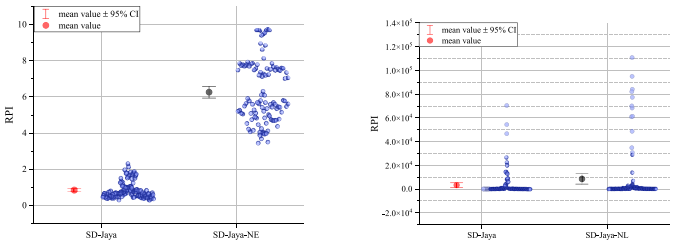
A. Numerical Results

The results of SD-Jaya in sloving EEDNIFSP with the optimization goal of minimizing the TTD and TEC are compared with the COA [39]. The level of speed is $U = \{1.00, 1.30, 1.55, 1.75, 2.10\}$. In [39], the result of COA is compared with NSGA-II and IMOEA/D. Therefore, only COA is selected to be a comparing algorithm.

The C Metric is utilized to evaluate the performances of SD-Jaya and COA on the 60 Ta problems with $\tau = \{1, 2, 3\}$

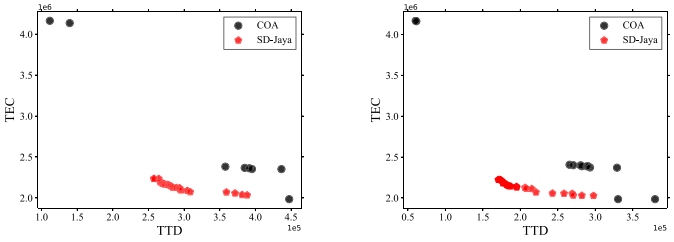
TABLE VI
C METRIC OF SD-JAYA AND COA FOR EEDNIFSP

n	m	$f=2$		$f=3$		$f=4$		$f=5$		$f=6$		$f=7$	
		A	A'	A	A'	A	A'	A	A'	A	A'	A	A'
100	5	**0.592**	0.000	**0.611**	0.003	**0.670**	0.000	**0.503**	0.000	**0.435**	0.029	**0.363**	0.014
100	10	**0.598**	0.000	**0.589**	0.003	**0.569**	0.010	**0.392**	0.000	**0.311**	0.023	**0.253**	0.020
100	20	**0.574**	0.000	**0.582**	0.004	**0.516**	0.002	**0.252**	0.009	**0.172**	0.010	**0.134**	0.023
200	10	**0.501**	0.000	**0.527**	0.000	**0.584**	0.000	**0.602**	0.002	**0.639**	0.004	**0.621**	0.000
200	20	**0.510**	0.000	**0.550**	0.000	**0.604**	0.003	**0.595**	0.003	**0.595**	0.000	**0.606**	0.000
500	20	**0.342**	0.000	**0.412**	0.000	**0.469**	0.000	**0.449**	0.000	**0.508**	0.000	**0.537**	0.000
Average		**0.520**	0.000	**0.545**	0.002	**0.569**	0.002	**0.466**	0.002	**0.443**	0.011	**0.419**	0.009



(a) (b)

Fig. 10. Interval plot of SD-Jaya, SD-Jaya-NE, and SD-Jaya-NL. (a) SD-Jaya and SD-Jaya-NE. (b) SD-Jaya and SD-Jaya-NL.



(a) (b)

Fig. 11. Pareto front for EEDNIFSP. (a) $\tau = 3, f = 6$ and $Ta = 111$. (b) $\tau = 3, f = 7$ and $Ta = 111$.

shown in Table VI and Tables XIII–XV (supplementary material). $A = C$(SD-Jaya, COA) and $A^{'} = C$(COA, SD-Jaya). In Tables XII, XIX, and XX (supplementary material), the Wilcoxon's texts of SD-Jaya, and COA are shown. In Figs. 23, 26, and 29 (supplementary material), the interval plots of SD-Jaya and COA with a 95% confidence interval are shown. In Figs. 24, 27, and 30 (supplementary material), the line chart of the SD-Jaya and COA in 60 Ta problems are shown. In Fig. 11 and Figs. 25, 28, and 31 (supplementary material) the Pareto fronts of SD-Jaya and COA are shown.

B. Analysis and Discussion

In Table VI, the mean of C(SD-Jaya, COA) is more significant than C(COA, SD-Jaya) in the $f = \{2, 3, 4, 5, 6, 7\}$ with the $\tau = \{1, 2, 3\}$, which means that the proportion of the solutions in the COA dominated by the solutions in Pareto set of SD-Jaya is bigger than the proportion of solutions in the SD-Jaya dominated by the solution in the Pareto set of COA. From Fig. 11, the Pareto front of COA is divided into three parts which include the solutions closed to the y-axis, the solution closed to the x-axis, and the solutions in the middle. In the initialization of COA, two individuals are assigned maximum speed, and minimum speed, respectively, which is beneficial for searching the solutions on the edge of the Pareto front and lead the Pareto front is divided into three parts. In SD-Jaya, the edge of the Pareto front does not be depicted. The solutions in the Pareto front of SD-Jaya dominate the solutions in the Pareto front of COA. In the supplementary material, the *p-value* in all scales is less than 0.05, which means that the SD-Jaya is outperformed COA under the 95% confidence interval from Figs. 23, 26, and 29. To $f = 6$ and $f = 7$ under $\tau = 1$ and $\tau = 2$, due to the result of the scale $\{n = 100, m = 20\}$ is small, the range of the best C Metric and the worst C Metric is extensive. However, the C Metric of SD-Jaya is more significant than the C Metric of COA in $f = 6$ and $f = 7$. In Figs. 24, 27, and 30, according to the line chart constituted by means of SD-Jaya and COA with $f = \{2, 3, 4, 5, 6\}$ in 60 Ta problems, the mean of C Metric of SD-Jaya is more significant than the mean of C Metric of COA in all 60 Ta problems. Based on the above analysis, the SD-Jaya is outperformed the COA in addressing EEDNIFSP.

VII. COMPARISON OF SD-JAYA WITH COA IN HFS-EEDNIFSP

A. Numerical Results

The results of SD-Jaya are compared with the COA in addressing HFS-EEDNIFSP. The speed setting is given as $U = \{1.00, 1.30, 1.55, 1.75, 2.10\}$. The A is set as $\{1, 1.2, 1.4\}$. The α is set as 0.5. The TEC is calculated by (20) and (21). The ageing grade of the machine is generated according to (29)

$$a_{f,i} = (f * i)\%\text{len(OL)}. \tag{28}$$

The C Metric are summarized in Table VII and Tables XVI–XVII (supplementary material). Tables XXI–XXIII (supplementary material), Wilcoxon's text of SD-Jaya and COA under 95% confidence interval are shown. In Figs. 12, 15, and 18, the interval plots of SD-Jaya and COA with a 95% confidence interval are shown. In Figs. 13, 16, and 19, the line chart of the SD-Jaya and COA in 60 Ta problems are revealed. In Fig. 32–34 (supplementary material), and Figs. 14, 17, and 20 the Pareto fronts of SD-Jaya and COA are shown. The ONVG of SD-Jaya and COA is shown in the Table VIII. $A = C$(SD-Jaya, COA) and $A^{'} = C$(COA, SD-Jaya).

TABLE VII
C METRIC OF SD-JAYA AND COA FOR HFS-EEDNIFSP

n	m	$f=2$		$f=3$		$f=4$		$f=5$		$f=6$		$f=7$	
		A	A'	A	A'	A	A'	A	A'	A	A'	A	A'
100	5	**0.464**	0.000	**0.586**	0.012	**0.576**	0.000	**0.483**	0.008	**0.406**	0.002	**0.321**	0.008
100	10	**0.473**	0.000	**0.505**	0.016	**0.449**	0.005	**0.356**	0.000	**0.259**	0.007	**0.181**	0.011
100	20	**0.535**	0.000	**0.475**	0.017	**0.398**	0.010	**0.177**	0.010	**0.098**	0.009	**0.102**	0.022
200	10	**0.315**	0.000	**0.491**	0.014	**0.518**	0.004	**0.579**	0.029	**0.524**	0.020	**0.565**	0.007
200	20	**0.299**	0.000	**0.496**	0.002	**0.552**	0.016	**0.470**	0.018	**0.525**	0.019	**0.496**	0.011
500	20	**0.131**	0.000	**0.232**	0.003	**0.329**	0.011	**0.438**	0.003	**0.501**	0.005	**0.453**	0.005
Average		**0.370**	0.000	**0.464**	0.011	**0.470**	0.008	**0.417**	0.011	**0.385**	0.010	**0.353**	0.011

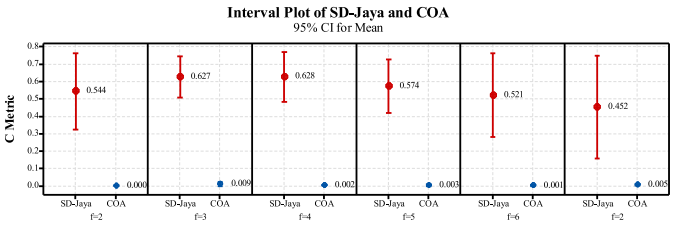


Fig. 12. Interval plot of SD-Jaya and COA with $\tau = 1$ for HFS-EEDNIFSP.

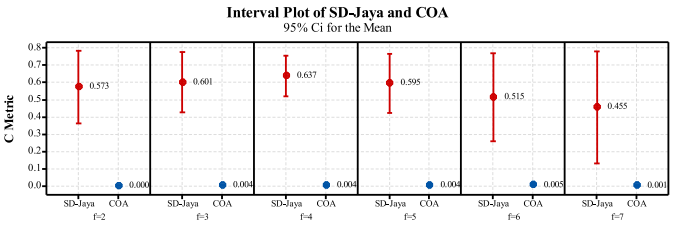


Fig. 15. Interval plot of SD-Jaya and COA with $\tau = 2$ for HFS-EEDNIFSP.

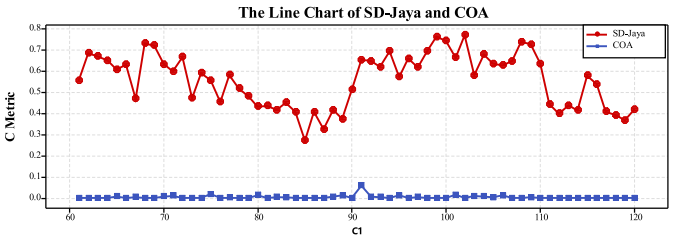


Fig. 13. Line chart of SD-Jaya and COA with $\tau = 1$ for HFS-EEDNIFSP.

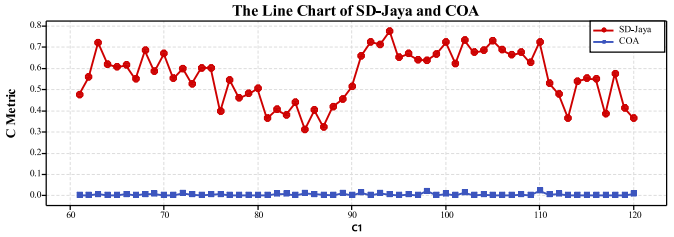


Fig. 16. Line chart of SD-Jaya and COA with $\tau = 2$ for HFS-EEDNIFSP.

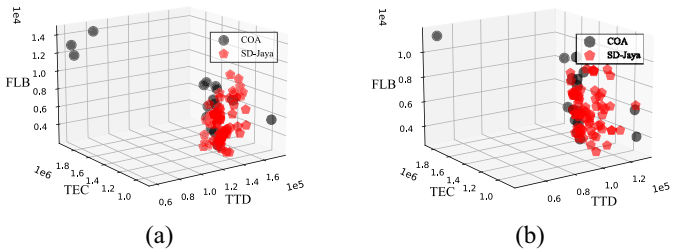


Fig. 14. Pareto front for HFS-EEDNIFSP with $\tau = 1$. (a) $f = 5$ and Ta = 109. (b) $f = 6$ and Ta = 101.

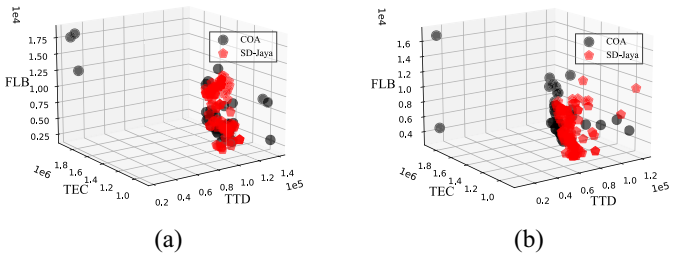


Fig. 17. Pareto front for HFS-EEDNIFSP with $\tau = 2$. (a) $f = 5$ and Ta = 105. (b) $f = 6$ and Ta = 108.

B. Analysis and Discussion

In Table VII, the $C(\text{SD} - \text{Jaya}, \text{COA})$ is more significant than $C(\text{COA}, \text{SD} - \text{Jaya})$ in the$f = \{2, 3, 4, 5, 6, 7\}$ with the $\tau = \{1, 2, 3\}$. From Tables XXI–XXIII (supplementary material), Wilcoxon's text indicates that the SD-Jaya is outperformed COA under the 95% confidence interval in $f = \{2, 3, 4, 5, 6, 7\}$. According to Figs. 12, 15, and 18, the algorithm performs differently at different scales, and SD-Jaya is significantly better than COA. To $f = 6$ and $f = 7$ under $\tau = 1$ and $\tau = 2$, due to the result of the scale $\{n = 100, m = 20\}$ is small, the range of the best C Metric and the worst C Metric is large. In Figs. 13, 16, and 19, and 21, the line chart constituted by means of SD-Jaya and COA with $f = \{2, 3, 4, 5, 6\}$ in 60 Ta problems indicate that the mean of C Metric of SD-Jaya is more significant than the mean of C Metric of COA in all 60 Ta problems. For three optimization goals, the Pareto front is a surface. The surface formed by the solution is closer to the origin, proving that the Pareto preface is better. Figs. 14, 17, and 20 shown that the surface formed by the red points is closer to the origin than the surface formed by the black points. The red points block out the black points, which means that the red points dominate the black points. In Table VIII, the ONVG of SD-Jaya and COA indicated that the number of nondominated solutions is more than the number of nondominated solutions of COA for addressing problems of the same scale. Based on the above analysis, the SD-Jaya outperformed the COA in addressing HFS-EEDNIFSP.

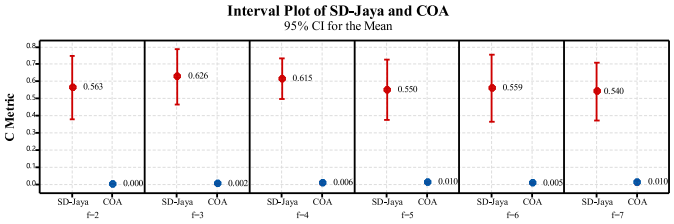


Fig. 18. Interval plot of SD-Jaya and COA with $\tau = 3$ for HFS-EEDNIFSP.

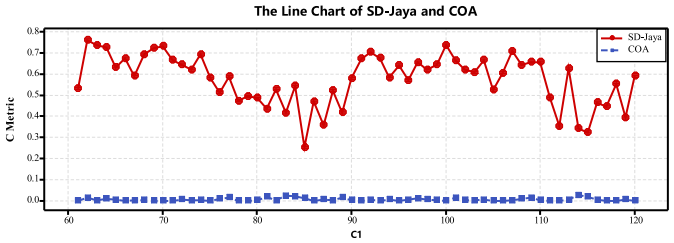


Fig. 19. Line chart of SD-Jaya and COA with $\tau = 3$ for HFS-EEDNIFSP.

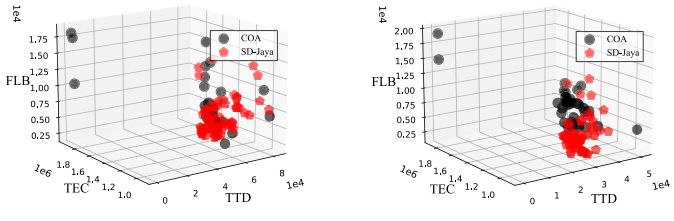


Fig. 20. Pareto front for HFS-EEDNIFSP with $\tau = 3$. (a) $f = 5$ and Ta $= 104$. (b) $f = 6$ and Ta $= 108$.

TABLE VIII
ONVG OF SD-JAYA AND COA

n	m	$\tau = 1$		$\tau = 2$		$\tau = 3$	
		COA	SD-Jaya	COA	SD-Jaya	COA	SD-Jaya
100	5	300.3	**337.0**	301.5	**344.0**	313.3	**365.3**
100	10	294.1	**409.3**	307.0	**423.5**	301.5	**455.6**
100	20	267.6	**423.8**	276.0	**472.8**	264.0	**530.1**
200	10	276.5	**369.8**	255.3	**405.6**	276.8	**413.8**
200	20	240.1	**406.0**	260.5	**434.1**	248.5	**424.1**
500	20	226.3	**244.8**	226.6	**268.1**	221.1	**233.6**
Average		267.5	**365.1**	271.1	**391.3**	270.8	**403.8**

VIII. CONCLUSION

In this study, the SD-Jaya algorithm is introduced to address the EEDNIFSP with the optimization objectives of minimizing the TTD and TEC and the HFS-EEDNIFSP with the optimization objectives of minimizing the TTD, TEC, and FLB. According to the features of the EEDNIFSP and HFS-EEDNIFSP, the population PG, SLOS, energy-saving strategy, and load-balancing adjustment strategy are designed, respectively. The effectiveness of all strategies is analyzed. On the quality of solution, the experimental results reveal that the efficacy of the SD-Jaya algorithm outperforms the COA, which is the only algorithm for addressing EEDNIFS.

Although the performance of the SD-Jaya outperforms other algorithms in addressing EEDNIFSP and HFS-EEDNIFSP, the research should be extended to the scheduling problems with blocking and no-wait manufacturing scenarios. In addition, more indicators affecting energy consumption should be mined and optimized to make the model cover the manufacturing scenarios.

REFERENCES

[1] G. Wang, X. Li, L. Gao, and P. Li, "An effective multi-objective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop," *Ann. Oper. Res.*, to be published.

[2] Y. Han, J. Li, H. Sang, Y. Liu, K. Gao, and Q. Pan, "Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106343.

[3] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization by NSGA-II and MOEA/D with large populations BT," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Oct. 2009, pp. 1758–1763.

[4] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 525–537, Jun. 2019.

[5] B. J. Lageweg, J. K. Lenstra, and A. Kan, "General bounding scheme for the permutation flow-shop probmem," *Oper. Res.*, vol. 26, no. 1, pp. 53–67, 1978.

[6] M. De Freitas Araujo, J. E. C. Arroyo, and L. B. Fialho, "Tabu search and iterated greedy for a flow shop scheduling problem with worker assignment BT," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Jul. 2020, pp. 1661–1668.

[7] B. Liu, L. Wang, and Y.-H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling BT—Special issue on memetic algorithms," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 18–27, Feb. 2007.

[8] B.-B. Li, L. Wang, and B. Liu, "An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 818–831, Jul. 2008.

[9] F. Zhao, L. Zhang, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion," *Expert Syst. Appl.*, vol. 146, pp. 1–21, May 2020.

[10] Y. Han, D. Gong, Y. Jin, and Q. Pan, "Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 184–197, Jan. 2019.

[11] F. Q. Zhao, S. Qin, Y. Zhang, W. M. Ma, C. Zhang, and H. B. Song, "A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem," *Expert Syst. Appl.*, vol. 126, pp. 321–339, Jul. 2019.

[12] L. Narain and P. C. Bagga, "Flowshop/no-idle scheduling to minimize total elapsed time," *J. Global Optim.*, vol. 33, no. 3, pp. 349–367, 2005.

[13] M. Cheng, S. Sun, and Y. Yu, "A note on flow shop scheduling problems with a learning effect on no-idle dominant machines," *Appl. Math. Comput.*, vol. 184, no. 2, pp. 945–949, 2007.

[14] W. Shao, D. Pi, and Z. Shao, "A hybrid discrete teaching-learning based meta-heuristic for solving no-idle flow shop scheduling problem with total tardiness criterion," *Comput. Oper. Res.*, vol. 94, pp. 89–105, Jun. 2018.

[15] J.-N. Shen, L. Wang, and S.-Y. Wang, "A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion," *Knowl. Based Syst.*, vol. 74, no. 1, pp. 167–175, 2015.

[16] F. L. Rossi and M. S. Nagano, "Heuristics for the mixed no-idle flowshop with sequence-dependent setup times and total flowtime criterion," *Exp. Syst. Appl.*, vol. 125, pp. 40–54, Jul. 2019.

[17] J. Pan, W. Zou, and J. Duan, "A discrete artificial bee colony for distributed permutation flowshop scheduling problem with total flow time minimization," in *Proc. 37th Chin. Control Conf. (CCC)*, 2018, pp. 8379–8383.

[18] C. Ling-Fang, W. Ling, and W. Jing-Jing, "A two-stage memetic algorithm for distributed no-idle permutation flowshop scheduling problem," in *Proc. 37th Chin. Control Conf. (CCC)*, 2018, pp. 2278–2283.

[19] S. Chen, Q. Pan, X. Hu, and M. F. Tasgetiren, "An iterated greedy algorithm for distributed blocking flowshop problems with makespan minimization," in *Proc. 39th Chin. Control Conf. (CCC)*, 2020, pp. 1536–1541.

[20] W. Shao, D. Pi, and Z. Shao, "A Pareto-based estimation of distribution algorithm for solving multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1344–1360, Jul. 2019.

[21] J. Gao, R. Chen, and W. Deng, "An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 3, pp. 641–651, 2013.

[22] K.-C. Ying, S.-W. Lin, C.-Y. Cheng, and C.-D. He, "Iterated reference greedy algorithm for solving distributed no-idle permutation flow-shop scheduling problems," *Comput. Ind. Eng.*, vol. 110, pp. 413–423, Aug. 2017.

[23] W. Shao, D. Pi, and Z. Shao, "Local search methods for a distributed assembly no-idle flow shop scheduling problem," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1945–1956, Jun. 2019.

[24] Y. X. Yang, P. Li, S. Wang, B. Liu, and Y. Luo, "Scatter search for distributed assembly flowshop scheduling to minimize total tardiness BT," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 861–868.

[25] Y. Li, X. Li, L. Gao, and L. Meng, "An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times," *Comput. Ind. Eng.*, vol. 147, Sep. 2020, Art. no. 106638.

[26] L. Haoran, L. Xinyu, and G. Liang, "A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem," *Appl. Soft Comput.*, vol. 100, Mar. 2021, Art. no. 106946.

[27] C. Lu, L. Gao, J. Yi, and X. Li, "Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China," *IEEE Trans. Ind. Informat.*, early access, Dec. 10, 2020, doi: 10.1109/TII.2020.3043734.

[28] 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.

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

[30] X. Li, C. Lu, L. Gao, S. Xiao, and L. Wen, "An effective multiobjective algorithm for energy-efficient scheduling in a real-Life welding shop," *IEEE Trans. Ind. Informat.*, vol. 14, no. 12, pp. 5400–5409, Dec. 2018.

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

[32] 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.

[33] E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search BT," in *Proc. 8th Int. Conf. Parallel Probl. Solving Nat. (PPSN)*, vol. 3242, Sep. 2004, pp. 832–842.

[34] F. Zhao, X. He, and L. Wang, "A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem," *IEEE Trans. Cybern.*, early access, Oct. 23, 2020, doi: 10.1109/TCYB.2020.3025662.

[35] X. Zheng, S. Zhou, R. Xu, and H. Chen, "Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimisation algorithm," *Int. J. Prod. Res.*, vol. 58, no. 13, pp. 4103–4120, 2020.

[36] K. Gao, Y. Huang, A. Sadollah, and L. Wang, "A review of energy-efficient scheduling in intelligent production systems," *Complex Intell. Syst.*, vol. 6, no. 2, pp. 237–249, 2020.

[37] J. Wang and L. Wang, "A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 5, pp. 1805–1819, May 2020.

[38] G. Wang, L. Gao, X. Li, P. Li, and M. F. Tasgetiren, "Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100716.

[39] J. Chen, L. Wang, and Z. Peng, "A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100557.

[40] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.

[41] H. Migallon, A. Jimeno-Morenilla, J. L. Sanchez-Romero, H. Rico, and R. V. Rao, "Multipopulation-based multi-level parallel enhanced Jaya algorithms," *J. Supercomput.*, vol. 75, no. 3, pp. 1697–1716, 2019.

[42] R. V. Rao, H. S. Keesari, P. Oclon, and J. Taler, "An adaptive multi-team perturbation-guiding Jaya algorithm for optimization and its applications," *Eng. Comput.*, vol. 36, no. 1, pp. 391–419, 2020.

[43] M. Aslan, M. Gunduz, and M. S. Kiran, "JayaX: Jaya algorithm with XoR operator for binary optimization," *Appl. Soft Comput. J.*, vol. 82, Sep. 2019, Art. no. 105576.

[44] M. Alsajri, M. A. Ismail, and S. Abdul-Baqi, "A review on the recent application of Jaya optimization algorithm BT," in *Proc. 1st Annu. Int. Conf. Inf. Sci. (AiCIS)*, Nov. 2018, pp. 129–132.

[45] A. K. Mishra, D. Shrivastava, B. Bundela, and S. Sircar, "An efficient Jaya algorithm for multi-objective permutation flow shop scheduling problem BT," in *Proc. Int. Conf. Adv. Eng. Optim. Intell. Techn. (AEOTIT)*, vol. 949, Aug. 2020, pp. 113–125.

[46] J.-Q. Li *et al.*, "An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times," *Knowl. Based Syst.*, vol. 200, Jul. 2020, Art. no. 106032.

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

Fuqing Zhao received the B.Sc. and Ph.D. degrees from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively.

Since 1998, he has been with the School of Computer Science Department, Lanzhou University of Technology, where he became a Full Professor in 2012. He has been as the Postdoctoral Fellow with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China, in 2009. He has been as a Visiting Scholar with the Exeter Manufacturing Enterprise Center, Exeter University, Exeter, U.K., and Georgia Tech Manufacturing Institute, Georgia Institute of Technology, Atlanta, GA, USA, from 2008 to 2019 and from 2014 to 2015, respectively. He has authored two academic book and over 50 refereed papers. His current research interests include intelligent optimization and scheduling.

Ru Ma received the B.S. degree from the School of Printing, Packaging Engineering and Digital Media Technology, Xi'an University of Technology, Xi'an, China, in 2018, and the M.S. degree from the School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou, China, in 2021.

Her current research interests include intelligent optimization and scheduling algorithms.

Ling Wang received the B.Sc. degree in automation and the Ph.D. degree in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively.

Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a Full Professor in 2008. He has authored five academic books and more than 300 refereed papers. His current research interests include intelligent optimization and production scheduling.

Prof. Wang is a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, the Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China. He is currently the Editor-in-Chief of *International Journal of Automation and Control*, and an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Swarm and Evolutionary Computation*.