Energy-Efficient Shop Scheduling Using Space-Cooperation Multi-Objective Optimization

Jiepin Ding, *Student Member, IEEE*, Jun Xia, Yaning Yang, *Student Member, IEEE*, Junlong Zhou, *Member, IEEE*, Mingsong Chen, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Since Industry 5.0 emphasizes that manufacturing enterprises should raise awareness of social contribution to achieve sustainable development, more and more meta-heuristic algorithms are investigated to save energy in manufacturing systems. Although non-dominated sorting-based meta-heuristics have been recognized as promising multi-objective optimization methods for solving the energy-efficient flexible job shop scheduling problem (EFJSP), it is hard to guarantee the quality of the Pareto front (e.g., total energy consumption, makespan) due to the lack of population diversity. This is mainly because an improper individual comparison inevitably reduces population diversity, thus limiting exploration and exploitation abilities during population updates. To achieve efficient population evolution, this paper introduces a novel space-cooperation multi-objective optimization (SCMO) method that can effectively solve EFJSP to obtain scheduling schemes with better trade-offs. By cooperatively evaluating the similarity among individuals in both the decision space and objective space, we propose a space-cooperation population update method based on a three-vector representation that can accurately eliminate repetitive individuals to derive higher-quality Pareto solutions. To further improve search efficiency, we propose a difference-driven local search, which selectively changes the positions of operations with higher differences to search for neighbors effectively. Based on the Taguchi method, we conduct experiments to obtain a suitable parameter combination of SCMO. Comprehensive experimental results show that, compared to state-of-the-art methods, our SCMO method achieves the highest HV and NR and the lowest IGD, with an average of 0.990, 0.952, and 0.001, respectively. Meanwhile, compared to traditional local search approaches, our difference-driven local search obtains twice the HV on instance Mk12 and reduces the solving time from 1521 s to 475 s.**

***Index Terms*—Difference-driven local search, energy-efficient job shop scheduling problem (EFJSP), multi-objective optimization, Pareto front, space-cooperation population update.**

Received 12 January 2024; revised 12 November 2024; accepted 23 November 2024. Date of publication 11 December 2024; date of current version 5 June 2025. This work was supported in part by the Natural Science Foundation of China under Grant 62272170, in part by the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software under Grant 22510750100, and in part by Shanghai Trusted Industry Internet Software Collaborative Innovation Center. Recommended for acceptance by B. Parhami. *(Corresponding author: Mingsong Chen.)*

Jiepin Ding, Jun Xia, and Mingsong Chen are with the MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China (e-mail: 52205902006@stu.ecnu.edu.cn; mschen@sei.ecnu.edu.cn).

Yaning Yang is with the MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China, and also with the School of Physics and Electronic Information Engineering, Ningxia Normal University, Guyuan 756500, China.

Junlong Zhou is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA.

Digital Object Identifier 10.1109/TSUSC.2024.3506822

I. Introduction

ALONG with the prosperity of green manufacturing, more and more manufacturing enterprises are raising awareness to reduce energy consumption to promote sustainable development of manufacturing systems [1], [2], as the manufacturing industry consumes almost a third of global energy and generates approximately 28% greenhouse gas emissions, according to the report of the International Energy Agency [3]. As a critical part of manufacturing systems, the Energy-efficient flexible job shop scheduling problem (EFJSP) has been investigated by both industry and academia [4], [5], [6], where an EFJSP involves a set of flexible machines with different levels of energy consumption to address different manufacturing requirements. In this situation, except for minimizing makespan, making full use of such machines is becoming a major concern of manufacturing systems to reduce total energy consumption.

Various meta-heuristic algorithms make efforts on machine tools to save energy, which can be summarized into four categories [7], [8], [9]: turning on/off machines, machine speed adjustment, time-of-use electricity-based scheduling, and proper machine scheduling. The first is to adjust the operating state of the machines by comparing unload energy consumption with the energy generated by turning the machine on and off [10]. Since frequent turn-on/off behaviors will seriously deteriorate the usage life of machines [11], the frequency of turning on/off machines is worthy of attention. The second is to tune the machine speeds [12], which is based on the assumption that a higher machine speed reduces processing time while incurring more energy consumption. The third is to take advantage of time-of-use electricity pricing [13], where shifting the manufacturing process from peak to non-peak time can effectively reduce electricity costs. The fourth is proper machine scheduling [14], which applies to various scheduling problems. Although executing more jobs requires more energy, the large amount of energy consumption caused by manufacturing enterprises is mainly due to the inefficient use of machines. Therefore, we focus on proper machine scheduling to obtain a better Pareto front for EFJSP, aiming to minimize energy consumption without increasing the makespan as much as possible. Since the problem of resource allocation for multiple jobs has been proven to be NP-hard [15], it is challenging to figure out the optimal Pareto front for EFJSP,

especially when taking into account the setup time/energy and transportation time/energy.

To obtain an optimal Pareto front for complex EFJSP, various non-dominated sorting-based meta-heuristic methods have been investigated in machine scheduling [16], [17]. By retaining more objective-specific knowledge for each solution, non-dominated sorting-based meta-heuristic methods can effectively optimize multiple objectives simultaneously. However, due to the lack of population diversity, existing non-dominated sorting-based meta-heuristic methods still suffer from notable energy consumption and unacceptable makespan, which hinders their deployment in real scenarios. This is mainly because i) existing non-dominated sorting-based meta-heuristic methods focus on designing efficient and problem-specific operators and ignoring population diversity, which severely limits the exploration potential of these methods to explore better solutions; and ii) the evaluation of similarity among individuals is inappropriate, which cannot accurately screen repetitive individuals and affect the effectiveness of evolutionary processes. Although most meta-heuristics employ local search to improve the quality of solutions, it is too time-consuming, especially for large-scale scenarios. Therefore, *how to properly maintain population diversity to generate high-quality Pareto front for EFJSP in a reasonable solving time is becoming a major challenge in the design of non-dominated sorting-based meta-heuristic methods.*

To facilitate the efficient search for optimal Pareto front, this paper presents a novel non-dominated sorting-based meta-heuristic method named space-cooperation multi-objective optimization (SCMO), which aims to reduce energy consumption for manufacturing systems. This paper makes the three major contributions as follows:

- We schedule an energy-efficient flexible job shop that can properly describe realistic manufacturing scenarios. This studied scheduling shop takes both setup and transportation time/energy into consideration, where a multi-objective optimization model is constructed to minimize makespan and total energy consumption simultaneously.
- We design a new solution representation scheme to accurately filter out repetitive individuals by evaluating the similarity among individuals in decision space. Based on mating selection in objective space, SCMO fully utilizes individuals that are good at different objectives to support efficient population updates.
- By prioritizing operations with large differences in completion time, we propose a difference-driven local search based on the critical path that can selectively transpose positions of operations with more significant differences to improve search efficiency.

Comprehensive experiments on well-known benchmarks show that SCMO outperforms all investigated state-of-the-art methods by up to 52.78% in terms of Hypervolum (HV) against NSGA-II. Compared with traditional local search methods, our proposed difference-driven critical operator selection can improve HV by 94.55% on instance Mk12 and reduce the solving time from 1521 s to 475 s. Such improvements can be further enhanced when the instance scale increases.

The remainder of this paper is structured as follows. Section II gives the related work of energy-efficient scheduling solutions and non-dominated-based meta-heuristic methods. Section III describes an energy consumption model and formulations for EFJSP. In Section IV, we propose a space-cooperation multi-objective optimization method to address EFJSP. Based on the experimental setup described in Section V, Section VI conducts experiments and analyzes results. Section VII proposes conclusions and future work.

II. Related Work

To ensure sustainable development of manufacturing systems, reducing energy consumption in the manufacturing process is essential, since significant energy waste will increase the economic costs of manufacturing enterprises and cause serious environmental pollution [18], [19]. Therefore, in addition to minimizing the common optimization objective (i.e., makespan), manufacturing systems strive to reduce energy consumption to promote green manufacturing [20].

To achieve energy-efficient scheduling, extensive investigations about non-dominated sorting have been conducted in multi-objective trade-offs, which can facilitate meta-heuristic-based methods to search for the Pareto front. Table I compares the related works of multi-objective scheduling. For example, to reduce total energy consumption and total tardiness simultaneously, Pan et al. [21] presented a two-population approach based on non-dominated sorting to solve the parallel machine scheduling problem, where the non-dominated sorting genetic algorithm-II and memetic differential evolution evolve cooperatively on two populations in parallel to produce better solutions. In [24], Luo et al. presented an adaptive neighborhood search operator to address the classic distributed flexible job shop scheduling problem with worker arrangement, which can not only improve the exploitation ability of memetic algorithm, but also avoid trapping into local optima. Moreover, to better solve FJSP with sequence-dependent setup time/cost, Li et al. [5] developed an elitist hybrid algorithm based on non-dominated sorting that fully utilizes machine learning approaches to extract more non-dominated solutions from promising search regions. Although existing meta-heuristic methods based on non-dominated sorting perform well on multi-objective optimization, they still suffer from the problem of coarse-grained trade-offs among different optimization objectives.

To achieve finer-grained trade-offs, various non-dominated sorting-based meta-heuristic methods strive to maintain population diversity from different perspectives. Aiming at ensuring population diversity, Turkilmaza et al. [17] designed the dispatching rules to create an initial population and introduced new random individuals to the next generation population at a specific rate in the elite selection stage. Dai et al. [29] proposed a novel method named EGA, which designs an acceptance probability module and an annealing rate module to avoid trapping into local optima. In this approach, inferior solutions have a low acceptance rate rather than being discarded directly, where the acceptance rate decreases as the number of iterations increases. In [27], Gong et al. designed a new fitness ranking method that assigns different levels for individuals with the same non-dominated rank to alleviate the issue of sparse objective distribution. Although the above methods improve

TABLE I
COMPARISON OF META-HEURISTIC APPROACHES FOR MULTI-OBJECTIVE SCHEDULING

Problem	Reference	Objective Function	Proposed Solution	Mechanism for Population Diversity
PMSP	[21]	total energy consumption and total tardiness	two-population optimization algorithm	knowledge-based search operator
OSSP	[22]	total tardiness and energy consumption	multi-objective brain storm optimizer	finding candidate solutions by a population evolution and an external archive evolution
JSP	[23]	makespan, total weighted tardiness, and total weighted earliness	island model memetic algorithm	similarity- and-quality based replacement method
	[24]	makespan, maximum workload of machines and workload of workers	improved memetic algorithm	adaptive neighborhood search operator
FJSP	[25]	makespan and total electricity cost	artificial bee colony algorithm	genetic operators for employed bees
	[26]	makespan and stability	non-dominated ranking genetic algorithm	-
	[5]	makespan and total setup costs	elitist non-dominated sorting hybrid algorithm	estimation of distribution algorithm-based machine learning mechanism
	[17]	makespan and total tardiness	hybrid genetic algorithm-hypervolume contribution measure	generating new random individuals following a specific rate
	[27]	makespan, labor cost and green production-related factors	non-dominated ensemble fitness ranking method	assigning different ranks for individuals with the same non-dominated level
	[28]	total tardiness and energy consumption	knowledge-based cuckoo search algorithm	a reinforcement learning method to self-adaptively control parameters
EFJSP	[29]	energy consumption and makespan	enhanced genetic algorithm	acceptance probability module and annealing rate module
	ours	makespan and total energy consumption	space-cooperation multi-objective optimization (SCMO) method	space-cooperation (including objective space and decision space) population update based on three-vector representation

[1] The symbol "—" indicates that such a part is not mentioned in the original work.
[2] JSP: job shop scheduling problem; FSSP: flow shop scheduling problem; FJSP: flexible job shop scheduling problem; PMSP: parallel machine scheduling problem; OSSP: open shop scheduling problem; EFJSP: energy-efficient flexible job shop scheduling problem.

population diversity by increasing the randomness of individual selection, a comparison between individuals being limited to an objective space is not comprehensive, which will severely affect the exploration ability of algorithms. This is mainly because two individuals close in an objective space are not necessarily close in a decision space. To alleviate "premature" convergence issues, Qing-dao-er-ji and Wang [30] first defined a similar degree to achieve efficient individual selection, including similarity and concentration. Specifically, similarity measures the number of different elements at the corresponding positions of two individuals, and concentration calculates the overall similarity of each individual to the others in the population. By taking the similarity between individuals into account, Kurdi [23] developed a similarity-and-quality based replacement method to improve individual comparison efficiency, where if a new individual is better than the best or better than the worst and the similarity is below a given threshold, this new individual will replace the worst individual. Due to improper representation, the above methods cannot accurately distinguish repetitive individuals, seriously affecting evolutionary efficiency.

To the best of our knowledge, our SCMO method is the first attempt to cooperatively evaluate the similarity among individuals in both objective space and decision space based on a novel three-vector representation and selectively change positions for potential operations to improve search efficiency. Based on the synergy of these contributions, our approach can not only obtain a higher quality Pareto front in a reasonable time but also have more stabilization for solving EFJSP.

III. MULTI-OBJECTIVE OPTIMIZATION MODEL FOR EFJSP

A. Problem Description

Assume that a manufacturing system consists of a set of flexible machines $M = \{M_1, M_2, \ldots, M_k, \ldots, M_m\}$, and a set of jobs (i.e., manufacturing requirements) $J = \{J_1, J_2, \ldots, J_i, \ldots, J_n\}$. Each job J_i of the system contains n_i operations, which are sequentially ordered in the form of $J_i = < O_{i,1}, O_{i,2}, \ldots, O_{i,j}, \ldots, O_{i,n_i} >$. Here, an operation $O_{i,j}$ can only select one machine from its corresponding candidate machine set $M_{i,j} \subseteq M$ and $M_{i,j} \neq \emptyset$, and a machine can only execute one operation at a time. No interruption is allowed for operations during the execution process. If two consecutive operations (e.g. $O_{i,j-1}$ and $O_{i,j}$) in the same job are performed by different machines (e.g., M_w and M_k), the setup time $st_{i,j,k}$ of $O_{i,j}$ must be considered [31]. The location distance between different machines cannot be ignored, as each machine is located in different workshops. In this case, the transportation time exists when two successive operations in the same job are assigned to different machines. Note that for two consecutive operations within the same job, transportation must not start until the

TABLE II
DEFINITION OF NOTATIONS

Notations	Definition
i, u	The job indices.
j, v	The operation indices.
k, w	The flexible machine indices.
n	The number of jobs.
m	The number of flexible machines.
n_i	The number of successive operations of J_i.
$pt_{i,j,k}$	The processing time of $O_{i,j}$ on M_k.
$st_{i,j,k}$	The setup time of $O_{i,j}$ before performed on M_k.
$tt_{i,j,w,k}$	The transportation time caused by two successive operations (i.e., $O_{i,(j-1)}$ and $O_{i,j}$) that moves from M_w to M_k.
$S_{i,j}$	The start time of $O_{i,j}$.
$C_{i,j}$	The completion time of $O_{i,j}$.
pe_k	The processing energy per unit processing time on M_k.
se_k	The setup energy per unit setup time on M_k.
ue_k	The unload energy per unit unload time on M_k.
te	The transportation energy per unit transportation time.
ae	The auxiliary energy per unit time.
$\lambda_{i,j,k}$	An indicator variable, where $\lambda_{i,j,k} = 1$ if M_k is available to perform $O_{i,j}$, otherwise, $\lambda_{i,j,k} = 0$.
$\theta_{i,j,k}$	An indicator variable, where $\theta_{i,j,k} = 1$ if $O_{i,j}$ is assigned to M_k, otherwise, $\theta_{i,j,k} = 0$.
$\delta_{i,j,q,k}$	An indicator variable, where $\delta_{i,j,q,k} = 1$ if $O_{i,j}$ is processed in the q^{th} position on M_k, otherwise, $\delta_{i,j,q,k} = 0$.

TABLE III
SETTINGS OF THE EFJSP EXAMPLE

Job	Operation	Overhead		
		M_1	M_2	M_3
J_1	$O_{1,1}$	$T(2,8)$	$T(3,7)$	-
	$O_{1,2}$	-	$T(1,10)$	$T(2,7)$
	$O_{1,3}$	$T(2,6)$	$T(1,9)$	$T(3,7)$
J_2	$O_{2,1}$	$T(2,5)$	$T(1,8)$	$T(1,7)$
	$O_{2,2}$	$T(3,9)$	-	$T(2,10)$
	$O_{2,3}$	$T(2,6)$	$T(2,8)$	-
J_3	$O_{3,1}$	$T(2,12)$	$T(3,10)$	$T(2,14)$
	$O_{3,2}$	-	$T(1,8)$	$T(2,6)$
	$O_{3,3}$	$T(1,11)$	$T(3,8)$	-
Energy per unit time		$E(1,3)$	$E(2,2)$	$E(1,2)$

previous operation is completed. Unlike transportation, unoccupied machines can prepare for upcoming operations in advance, even if the previous operation of the upcoming operation is not completed at this time, since the setup time of the upcoming operation is independent of its previous operation. Taking energy consumption during the manufacturing processes into account, we define the problem above as the Energy-efficient Flexible Job Shop Scheduling Problem (EFJSP), where all notations used in this paper are defined in Table II.

To facilitate the understanding of energy-efficient scheduling processes, we consider an example of EFJSP involving three jobs $J = \{J_1, J_2, J_3\}$ and three machines $M = \{M_1, M_2, M_3\}$, whose settings are shown in Table III. For a job J and its operation O, we use the notion $T(x, y)$ to indicate that the setup time and the processing time of O on a specific machine (e.g., M_1, M_2 or M_3) are x and y, respectively. For example, operation $O_{1,1}$ needs 2 units of setup time and 8 units of processing time to execute on M_1. We use the notion $E(m, n)$ to indicate that the setup energy and processing energy of O on a specific machine (e.g., M_1, M_2 or M_3) are m and n per unit time, respectively. For example, the setup and processing procedures for each operation performed on M_1 consume 1 unit setup energy and 3 units

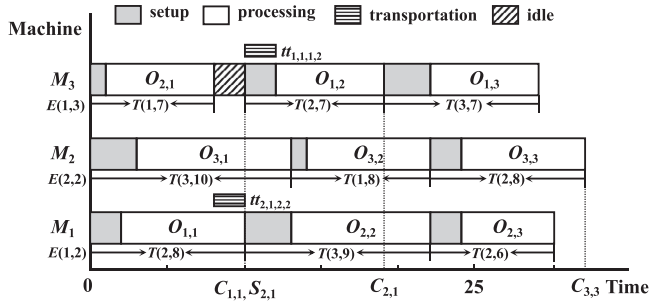

Fig. 1. Gantt chart of the example instance.

processing energy per unit time. In addition, the transportation time between M_1 and M_2, M_1 and M_3, M_2 and M_3 is 3, 2 and 2, respectively. The transportation energy between M_1 and M_2, M_1 and M_3, M_2 and M_3 is 1, 2 and 1 per unit time, respectively. The following subsection will detail the modeling of energy consumption.

Fig. 1 shows a Gantt chart of the example instance mentioned in Table III. As shown in the figure, each machine is selected to process operations. For example, operation $O_{1,1}$ is assigned to machine M_1 at time 0, where $T(2, 8)$ indicates that M_1 needs 2 and 8 unit time to complete setup and processing procedures, respectively. Since setup and processing procedures cannot be performed in parallel, the completion time of $O_{1,1}$ (i.e., $C_{1,1}$) is 10. Then, $O_{1,1}$ will be transported to M_3 to execute the next operation $O_{1,2}$ once $O_{1,1}$ is complete and its corresponding transportation time is 2. Note that the setup and transportation are not conflicting, while the processing must be performed after the transportation ends. In this case, we can start the setup procedure of $O_{1,2}$ during transportation to save time. Therefore, the start time and the completion time of $O_{1,2}$ are 10 and 17, respectively. The time before the M_3 starts to execute $O_{1,2}$ after executing $O_{2,1}$ is called the idle time. Since operation $O_{3,3}$ is the last one to complete, $C_{3,3} = 32$ is the total completion time of three jobs.

B. Energy Consumption Model

Since different machines have different manufacturing capabilities and locations, their setup time/energy, processing time/energy, and transportation time/energy are different even for the same operation. By referring to a real case of flexible workshops in Nanjing [29], we construct an energy consumption model that includes the following five situations [29]: i) machine preparation before processing, ii) machine processing for operations, iii) machine waiting to perform subsequent operations, iv) transporting an operation from one machine to another, and v) power to support the daily work of all workshops, each of which is described as follows.

- *Setup Energy Consumption (SEC):* A machine needs to prepare well in advance before performing different operations, such as positioning, clamping, loading workpieces, changing tools, and switching operations [29]. By multiplying the setup time of each operation and the setup energy consumed per unit setup time on specific machines, the

resulting SEC can be calculated as

$$SEC = \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} se_k \cdot st_{i,j,k} \cdot \theta_{i,j,k}. \tag{1}$$

- *Processing Energy Consumption (PEC):* The energy consumption generated when the machine performs the operations is PEC, which accounts for the majority of the total energy consumed throughout the product manufacturing process. Note that once a machine is on, whether or not it performs operations, it will consume a certain amount of energy [29]. The PEC can be formulated as

$$PEC = \sum_{k=1}^{m} ue_k \cdot \left(\sum_{i=1}^{n}\sum_{j=1}^{n_i} pt_{i,j,k} \cdot \theta_{i,j,k}\right) + \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} pe_k \cdot pt_{i,j,k} \cdot \theta_{i,j,k}. \tag{2}$$

- *Unload Energy Consumption (UEC):* Assume that a machine will remain powered on until all operations on the machine are completed. We set this assumption mainly for the following two reasons: i) switching the machine on and off will generate energy consumption, and ii) frequent turn-on and turn-off behaviors will seriously deteriorate the usage life of the machine. A machine is in an unloaded state during idle time before the next operation and after the current operation is completed. We define UEC as the energy consumed by machines in unloaded states, which can be formulated as

$$UEC = \sum_{k=1}^{m} ue_k \cdot \left(\max_{\forall i,j} C_{i,j} - \sum_{i=1}^{n}\sum_{j=1}^{n_i}(C_{i,j} - S_{i,j}) \cdot \theta_{i,j,k}\right). \tag{3}$$

- *Transportation Energy Consumption (TEC):* Due to the different locations of machines, if two successive operations of a job are not assigned to the same machine, there will be transportation time between the two operations. The energy consumed during the operation transportation process can be expressed as

$$TEC = \sum_{i=1}^{n}\sum_{j=2}^{n_i}\sum_{k=1}^{m}\sum_{w=1}^{m} te \cdot tt_{i,j,w,k} \cdot \theta_{i,(j-1),w} \cdot \theta_{i,j,k}. \tag{4}$$

- *Auxiliary Energy Consumption (AEC):* AEC is auxiliary energy needed to support the production environment, such as lighting and air-conditioning. We define AEC as

$$AEC = ae \cdot \max_{\forall i,j}\left(C_{i,j}\right). \tag{5}$$

C. Problem Formulation

The purpose of this paper is to obtain a scheduling schedule that can achieve a low total energy consumption and a makespan simultaneously. This is mainly because total energy consumption and makespan are the two major issues that manufacturing enterprises are most concerned about, and they need to be clearly defined to ensure long-term sustainable development of manufacturing systems. Let f_1 and f_2 be the makespan and total energy consumption, respectively. Based on the problem description of shop scheduling and the modeling of energy consumption, we formulate the EFJSP as follows:

$$\min f_1 = C_{max} = \max_{\forall i,j}(C_{i,j}), \tag{6}$$

$$\min f_2 = SEC + PEC + UEC + TEC + AEC, \tag{7}$$

$$1 \le \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} \lambda_{i,j,k} \le m, \tag{8}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} \theta_{i,j,k} = 1, \tag{9}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} \delta_{i,j,q,k} = 1, \tag{10}$$

$$S_{i,j} = \max(P_k,\ C_{i,(j-1)} + tt_{w,k} - st_{i,(j-1),k}), \tag{11}$$

$$C_{i,j} = S_{i,j} + st_{i,j,k} + pt_{i,j,k}, \tag{12}$$

$$C_{i,j} - C_{i,(j-1)} \ge \sum_{i=1}^{n}\sum_{j=1}^{n_i} pt_{i,j,k} \cdot \theta_{i,j,k}, \tag{13}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i} S_{i,j} \cdot \theta_{i,j,k} \cdot \delta_{i,j,q,k} \ge \sum_{i=1}^{n}\sum_{j=1}^{n_i} C_{u,v} \cdot \theta_{u,v,k} \cdot \delta_{u,v,q,k}. \tag{14}$$

Objective (6) is to minimize the total completion time of all operations, where f_1 records the completion time of the final operation. Objective (7) strives to minimize the total energy consumption restricted by SEC, PEC, UEC, TEC, and AEC. Constraint (8) means that each operation $O_{i,j}$ has a candidate machine set $M_{i,j} \subseteq M$. Constraint (9) indicates that each operation $O_{i,j}$ can ultimately be performed on only one machine in $M_{i,j}$. Constraint (10) ensures that each operation cannot be interrupted during execution. The start time $S_{i,j}$ in Constraint (11) takes the larger value of the progressive time (i.e., P_k) of M_k and the earliest available time of $O_{i,j}$ (i.e., $C_{i,(j-1)} + tt_{w,k} - st_{i,(j-1),k}$), where P_k indicates the completion time of the operation performed on M_k at the current moment. Once an operation is assigned, P_k will be updated according to its completion time. The notation w denotes the machine index that performs $O_{i,(j-1)}$. Since the setup time of an operation $O_{i,j}$ is independent of its previous operation $O_{i,(j-1)}$, $O_{i,j}$ can be set in advance before $O_{i,(j-1)}$ is transported to M_k. Constraint (12) means that the setup and process procedures can not be disturbed. Constraint (13) specifies the dependencies between operations in the same job, where each operation must wait until

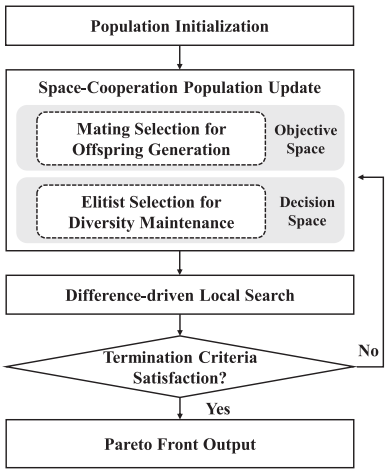

Fig. 2. Workflow of our method.

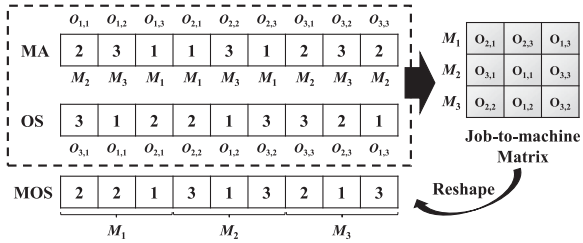

Fig. 3. An example of three-vector representation.

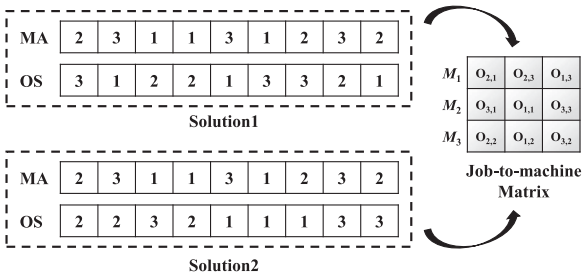

Fig. 4. An example of the same two solutions with different representations.

all its previous operations are completed. Constraint (14) ensures that each machine can perform only one operation at a time.

IV. Our Space-Cooperation Optimization Method

This section details our SCMO method for EFJSP, whose objective is to reduce energy consumption and makespan simultaneously. Fig. 2 illustrates the workflow of our method, which consists of three major parts: i) population initialization that randomly generates N individuals to form an initial population based on a three-vector representation; ii) space-cooperation population update that adopts mating selection in objective space to achieve efficient offspring generation and selects elitists in decision space to maintain population diversity; and iii) difference-driven local search that explores better neighbors by adjusting the machines or sequences of critical operations of non-dominated individuals. During the population evolution, we repeat the space-cooperation population update and local search until the termination criteria are met (e.g., the maximum number of iterations).

A. Solution Representation

Although two traditional vectors are sufficient to represent a scheduling scheme with machine assignment (MA) and operation sequencing (OS), we propose a machine-related operation sequencing vector (MOS) in this paper, which can accurately distinguish whether two solutions are identical. In our three-vector representation, each position of an MA represents an operation placed sequentially, and each element at the position represents the machine index that performs the operation represented by the corresponding position. In an OS vector, the element at each position represents a job index, and the number of occurrences of each element indicates the number of operations in the job to which that element corresponds. By adjusting an OS vector following operation sequences on each machine, we can get an MOS vector. Fig. 3 shows an example of three-vector representation, where three jobs are ready to be performed on three machines. As shown in the figure, each position of MA is represented as an operation for each job sequentially, and the first element 2 in MA indicates that $O_{1,1}$ is assigned to M_2. In OS, the first element 3 at the first position is $O_{3,1}$, and the second element 3 at the sixth position represents $O_{3,2}$. Therefore, the combination of MA and OS in the figure can constitute a solution, i.e., $O_{3,1}(M_2) \rightarrow O_{1,1}(M_2) \rightarrow O_{2,1}(M_1) \rightarrow O_{2,2}(M_3) \rightarrow O_{1,2}(M_3) \rightarrow O_{3,2}(M_3) \rightarrow O_{3,3}(M_2) \rightarrow O_{2,3}(M_1) \rightarrow O_{1,3}(M_1)$. Based on the solution, we record the execution sequencing of operations on each machine with a matrix, where the first row represents the execution sequencing of operations on M_1 as $O_{2,1} \rightarrow O_{2,3} \rightarrow O_{1,3}$. Then, we combine the corresponding job indices of operations by row into an MOS vector.

For population initialization, each operation randomly selects one machine from its candidate machine set to ensure the feasibility of a solution, and the operation sequencing is also randomly determined. Note that we only initialize an MA and OS for each individual (i.e., solution) in a population since an MA and OS can determine a solution. The reason why we introduce an MOS vector in our three-vector representation is that the same MA and different OS may indicate the same solution, which cannot accurately judge the consistency of the two solutions. Fig. 4 presents an example of the same two solutions with different representations, where their MA vectors are the same while OS vectors are different. Although traditional methods by comparing OS will conclude that the two solutions are different, the OS vectors of the two solutions are essentially the same because they can get the same matrix. The following subsection will detail the usage of a three-vector representation in this paper.

B. Space-Cooperation Population Update

1) Mating Selection for Offspring Generation: The main task of our EFJSP is to maintain population diversity in an objective space to obtain a complete Pareto front. To achieve this goal, we design objective reference sets (i.e., all non-dominated solutions at each iteration) for mating selection, which can

Algorithm 1: Mating Selection.

Input: i) P, population; ii) N, population size; iii) f_1, makespan; iv) f_2, total energy consumption.

1 $rank_1, rank_2, \ldots = non\text{-}dominated\_sorting(P)$;
2 $p_1 = min\_f_1(P)$, $p_2 = min\_f_2(P)$;
3 $Group_1, Group_2 = decomposition\_pop(P, p_1, p_2)$;
4 $R_1, R_2 = decomposition\_rank_1(rank_1, p_1, p_2)$;
5 **for** $l = 1, 2, \ldots, N$ **do**
6 **if** $p_l \in Group_1$ **then**
7 Obtain a similarity set S_l between p_l and all the individuals in R_2;
8 **end**
9 **else**
10 Obtain a similarity set S_l between p_l and all the individuals in R_1;
11 **end**
12 $S\_nor_l = normalization(S_l)$;
13 $G_l = roulette\_wheel\_selection(S\_nor_l)$;
14 **end**

Output: G_l

improve optimization objectives with poor performance by learning objective-specific knowledge. Algorithm 1 details the implementation of mating selection based on objective reference sets. Line 1 executes non-dominated sorting for population P and obtains a non-dominated set (i.e., individuals in $rank_1$). Line 2 searches for two individuals p_1 and p_2 with minimum f_1 and f_2 in the population P, respectively. Based on the midpoint M of p_1 and p_2, line 3 divides P into two groups $Group_1$ and $Group_2$, and line 4 divides $rank_1$ into two R_1 and R_2. After that, lines 5-14 perform N mating selection steps for each individual to support offspring generation. Specifically, if an individual p_l belongs to $Group_1$, line 7 calculates the similarity between p_l and all the non-dominated individuals in R_2 to obtain a similarity set S_l. In contrast, if an individual p_l belongs to $Group_2$, line 10 calculates the similarity between p_l and all the non-dominated individuals in R_1 to obtain a similarity set S_l. Based on the normalized similarity set S_nor_l obtained in line 12, line 13 selects a mating individual as a guiding individual G_l based on the roulette wheel method to achieve evolutionary guidance for p_l.

Based on the selected mating individuals, we set them as guiding individuals for all individuals in a population to generate offspring. To improve population diversity, we randomly select another individual $p_l{}'$ from P to participate in the evolution processes of MA. To guarantee the feasibility of offspring, we adopt the multi-point crossover for MA and the precedence operation crossover for OS [32], where MA only exchanges elements that are at the same position. As for the mutation operator, MA adopts a single-point mutation, which randomly selects an operation to re-select another one from its set of candidate machines [32]. Moreover, a pairwise swap is used for OS, where two elements in randomly selected positions are swapped [33].

2) Elitist Selection for Diversity Maintenance: Although the design of objective reference sets for mating selection can improve evolution efficiency, it still suffers from a lack of population diversity, since we ignore repetitive individuals during evolution processes. Especially after multiple rounds of

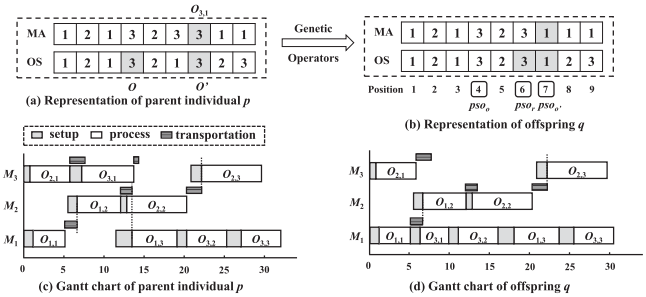

Fig. 5. An example of difference-driven local search.

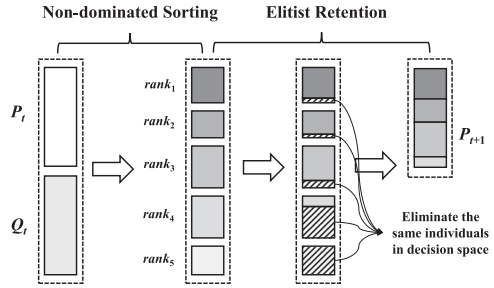

Fig. 6. Procedure of elitist selection.

iterations, a large number of repetitive individuals make the algorithm difficult to avoid trapping into local optima dilemma and even mask the effectiveness of our mating selection method. To maintain population diversity, traditional methods calculate crowding distances to discard similar individuals. However, the crowding distance is calculated from the perspective of objective space, which may mistakenly remove different individuals. This is because two individuals approaching or overlapping in the objective space may differ substantially in the decision space. Furthermore, if we directly compare the MA and OS vectors of two individuals using the Hamming distance, we may not be able to accurately screen all repetitive individuals, as illustrated in Fig. 4.

To address these issues, we introduce a sequence-transfer mechanism to maintain the diversity of the population. By converting mixed operation sequencing (i.e., OS) into machine-dependent operation sequencing (i.e., MOS), we can calculate the Hamming distance for both solutions based on MA and MOS to accurately find repetitive individuals. If two individuals have a Hamming distance of 0, they are repeated. Fig. 6 shows the procedure of elitist selection based on the sequence-transfer mechanism, which includes two parts: non-dominated sorting and individual retention. Given a population P_t and its offspring population Q_t in the t^{th} iteration, we assign a non-dominated rank $rank_i$ for each individual based on their optimization objectives, where the lower rank of an individual is, the better performance of the individual is. To ensure the effective evolution of a population, we preferentially retain individuals with low ranks. For all the individuals in $rank_1$, we calculate their Hamming distances based on the MA and MOS vectors. We kept all individuals without duplication in a new population P_{t+1}, while repetitive individuals retained only one. We will perform

the individual retention operation for the next rank until the size of population P_{t+1} reaches N.

C. Difference-Driven Local Search

Since the makespan of a solution is determined by its critical path, traditional critical path-based local searches try to change the sequence or machine of all critical operations to find better neighbors [34], where all the operations on a critical path are regarded as critical operations. Although adjusting the sequence or machine allocation of critical operations has proven effective in reducing makespan, it is time-consuming to traverse all the neighborhoods.

To improve search efficiency, we propose a difference-driven local search to selectively transpose positions of potential critical operations. Since a large difference $d_{i,j}$ between the completion time of operation $O_{i,j}$ traversing from left to right and traversing from right to left indicates a long delay for its subsequent operations, we give priority to operations whose difference are greater than a particular threshold. If the difference of an operation is greater than a threshold, then we will reassign the operation to another machine in the candidate machine set. Furthermore, if the subsequent operation of the operation is a critical operation, then we will adjust the sequence for the subsequent operation to search for better neighbors. Let $C_{i,j}$ be the completion time of $O_{i,j}$ traversing from left to right and $C^{-}_{i,j}$ be the completion time of $O_{i,j}$ traversing from right to left, the difference of operation $O_{i,j}$ is calculated as $d_{i,j} = |C_{i,j} - C^{-}_{i,j}|$. Based on this, we define the difference threshold as

$$d\_threshold = \max(d_{i,j}) * \mu, \tag{15}$$

where μ is a coefficient. We set coefficient μ ranging from 0 to 1, where a larger value of μ indicates a longer time of neighborhood exploration. Here, 0 means that all operations try to adjust their locations to search for better neighbors, and 1 means that only the operation with the largest difference performs a neighborhood search.

Fig. 5 gives an example of a difference-driven local search. Assume that an operation $O_{3,1}$ in parent individual p has the maximum difference in completion time, we attempt to adjust the position of $O_{3,1}$, where its representation is illustrated in Fig. 5(a). For MA, $O_{3,1}$ randomly chooses a machine M_1 from its candidate machine set $M_{3,1} = \{M_1, M_3\}$. For OS, we change the sequence only if the subsequent operation of $O_{3,1}$ is a critical operation. Since the subsequent operation $O_{3,2}$ belongs to the critical path, it is necessary to sequence operations to further explore the neighbors. We mark the positions of $O_{3,1}$ and $O_{3,2}$ as pso_o and $pso_{o'}$, respectively. After randomly selecting a position pso_r from the ranges of positions pso_o to $pso_{o'}$, we change the two positions (i.e., pso_r and $pso_{o'}$) to obtain an offspring q as shown in Fig. 5(b). From the Gantt charts of parent individual p and offspring q shown in Fig. 5(c) and (d), the makespan of offspring q is less than that of parent individual p.

V. Experimental Setup

To evaluate the effectiveness of our method, we conducted a series of experiments on a real-world instance named Real_case [29] and 20 EFJSP et al. [35] (Kc01-05) and Brandimarte [36] (Mk01-15). Note that the specific setup time, transport time, and energy per unit time are generated randomly based on [29]. Each experiment is performed in 30 independent runs to reduce the randomness of the experimental results. All the experimental results were obtained using Python from a Windows laptop with Intel 2.4 GHz CPU and 8 GB memory.

A. Performance Metrics

To achieve Pareto efficient solutions with low makespans and total energy consumption for EFJSPs, in this paper, we use the following performance metrics to evaluate the results obtained by different algorithms. Assume that RPF be **the reference Pareto front** formed after the non-dominant sorting of the Pareto solutions acquired by investigated algorithms.

1) The Non-dominated Ratio (NR) indicator [37] calculates the ratio between the number of Pareto solutions obtained by an algorithm but not dominated by the Pareto solutions in RPF and the number of Pareto solutions in RPF. The NR of Pareto front M is described as:
$$NR(M) = \frac{|RPF \cap M|}{|RPF|}, \tag{16}$$
where $|RPF|$ is the number of Pareto solutions in RPF. A larger $NR(\cdot)$ means that the Pareto solutions obtained by an algorithm cover most of the RPF, showing greater searchability.
2) The Inverted Generational Distance (IGD) indicator [38] measures the distance between the Pareto front M and RPF, which is described as:
$$IGD(M) = \frac{1}{|RPF|} \sum_{x \in |RPF|} \min_{y \in M} d(x, y), \tag{17}$$
where $d(x, y)$ is the Euclidean Distance between the solution x and solution y. Note that a smaller IGD indicates the Pareto solution obtained by an algorithm converges more to the RPF.
3) The Hypervolume (HV) [39] indicator calculates the volume of the region in objective space enclosed by the Pareto solutions of an algorithm.

B. Parameter Tuning

Since meta-heuristic-based methods are parameter-sensitive, we studied four key parameters to obtain a suitable parameter combination: i) crossover rate P_c, ii) mutation rate P_m, iii) coefficient of local search μ, and iv) population size N. Based on the Taguchi method [20], we conducted orthogonal experiments for these parameters on the Real_case of moderate-scale. Note that the maximum number of iterations is 500 [27]. Table IV shows the levels of four parameters for our approach, where four parameters with different levels can be combined to form an orthogonal array for subsequent experiments.

Fig. 7 illustrates the level trend of each parameter. As illustrated in the figure, we can observe that population size N has the most significant impact on NR and solving time. With the increase of N, solving time shows a straight uptrend while the

TABLE IV
LEVELS OF FOUR PARAMETERS FOR OUR APPROACH

Parameter	Level			
	1	2	3	4
P_c	0.7	0.8	0.9	1.0
P_m	0.1	0.2	0.3	0.4
μ	0.25	0.5	0.75	1
N	50	100	150	200

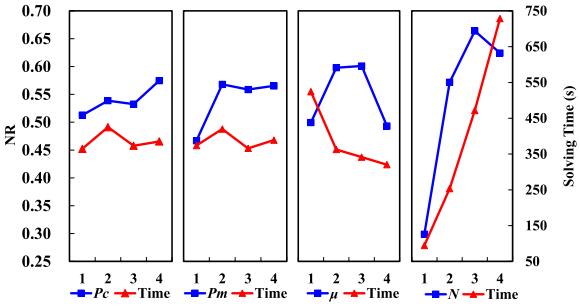

Fig. 7. Trend of Each Parameter Level in Our Approach.

uptrend of NR slows down. The results of the crossover rate P_c and the mutation rate P_m are comparatively stable. Due to an overall rise in P_c, more crossover operators help explore more potential solutions. For P_m, too few mutation operators will reduce NR. However, NR remains stable when P_m is greater than or equal to 0.2. Moreover, the value of μ at level two is enough to explore neighbors within a reasonable solving time. Therefore, we set four parameters as $P_c = 1$, $P_m = 0.2$, $\mu = 0.75$, and $N = 100$ for subsequent experiments to solve the proposed EFJSP.

C. Comparison Algorithms

Since our SCMO is developed on top of a common multi-objective optimization framework (i.e., non-dominated sorting), we chose NSGA-II as a baseline. By applying space-cooperation optimization based on the three-vector representation scheme and designing difference-driven local search, our approach can stably and quickly obtain high-quality solutions. To further investigate the effectiveness of our SCMO approach, we compared it with four state-of-the-art meta-heuristic methods: i) hybrid self-adaptive multi-objective evolutionary algorithm based on decomposition (HPEA) [40], ii) enhanced genetic algorithm (EGA) [29], iii) hybrid genetic algorithm (HGA) [17], and iv) surprisingly popular algorithm-based adaptive Euclidean Distance-based topology learning particle swarm optimization (SpadePSO) [41]. The reasons why we chose these four meta-heuristic methods as baselines are as follows. First, all four meta-heuristic algorithms are multi-objective optimization methods, which are good at solving the multi-objective optimization problem investigated in this paper. Second, to demonstrate the superiority of our approach to enhancing population diversity, we chose the four meta-heuristic algorithms that focus on improving population diversity from different perspectives as baselines. Third, for fairness reasons, all four meta-heuristic algorithms take local search into account to improve the quality of solutions by spending a certain amount of time for neighbor exploration. Please note that since deep reinforcement learning is promising in shop scheduling [42], [43], [44], [45], we also chose a state-of-the-art deep reinforcement learning-based method [46] that performs well in solving multi-objective optimization scheduling problems as a baseline. The five baselines are described below.

1) HPEA [40] proposes two problem-specific initial rules and five problem-specific neighbor search operators to maintain the diversity of the population. Furthermore, it utilizes the Tchebycheff decomposition strategy for solution selection to balance convergence and diversity.
2) EGA [29] is an improved version of NSGA-II. It designs three modules based on a simulated annealing algorithm to avoid being trapped in premature convergence.
3) HGA [17] is also a modification of NSGA-II. It defines a hypervolume contribution measure that can evaluate the contribution of a solution in the Pareto front, aiming to discard extreme points from the front.
4) SpadePSO [41] is an improved particle swarm optimization (PSO). Unlike traditional PSO only uses fitness to evaluate particles, SpadePSO proposes an adaptive Euclidean distance-based topology that allows each particle to select connected particles based on the Euclidean distance.
5) HDRL [46] is a two-hierarchy deep reinforcement learning method, which contains a front controller deep Q network and a back actuator deep Q network to perform hierarchical decision-making on multiple optimization objectives.

In this paper, we conducted the experiments for HPEA, EGA, HGA, SpadePSO, and HDRL based on their original parameter settings as presented in [17], [29], [40], [41], [46], respectively. Table VI shows the parameter settings of all baselines. To avoid the effects of randomness, we repeated each experiment 30 times separately.

VI. EXPERIMENTAL RESULTS

A. Comparison With State-of-The-Arts

1) Performance Evaluation: Table V compares the HV achieved by SCMO and the six baselines. In Table V, we can observe that our SCMO achieves the best HV on all the instances compared to the six baselines. To enable a more intuitive comparison, Table V shows the HV difference between NSGA-II and SMCO, HPEA, EGA, HGA, SpadePSO, and HDRL using HV results. The HV difference between NSGA-II and our SMCO is calculated as:

$$\frac{HV(NSGA-II)}{HV(SCMO)} \times 100\% - 1, \tag{18}$$

Note that a negative HV difference means that our SCMO outperforms NSGA-II. From this table, we can find that the average HVs of HPEA, EGA, and HGA are close and better than NSGA-II, SpadePSO, and HDRL on all instances and outperform NSGA-II, SpadePSO, and HDRL, while HDRL has the worst performance on small-scale instances. In addition,

TABLE V
COMPARISON OF HV BETWEEN SCMO AND BASELINES

Instance	$n/m/L$	HV							HV Differences (%, against NSGA-II)					
		NSGA-II	HPEA	EGA	HGA	SpadePSO	HDRL	SCMO	HPEA	EGA	HGA	SpadePSO	HDRL	SCMO
Case_study	10/10/32	0.98	0.95	0.97	0.97	0.94	0.85	**1.00**	3.25	0.52	0.98	3.93	15.29	**-1.95**
Kc01	4/5/12	1.00	1.00	**1.00**	**1.00**	1.00	0.90	1.00	-0.01	**-0.04**	**-0.04**	0.09	11.11	-0.01
Kc02	8/8/27	0.95	0.91	**0.99**	0.97	0.80	0.76	0.99	3.77	**-4.33**	-2.21	17.98	25.00	-4.26
Kc03	10/7/29	0.95	0.94	0.97	0.95	0.76	0.72	**0.98**	0.62	-1.58	0.43	24.55	31.94	**-3.24**
Kc04	10/10/30	0.96	0.97	0.96	0.96	0.82	0.76	**1.00**	-1.01	-0.03	-0.45	17.58	26.32	**-3.92**
Kc05	15/10/56	0.72	0.77	0.72	0.80	0.58	0.60	**1.00**	-6.44	-0.54	-10.51	24.62	20.00	**-28.23**
Mk01	10/6/56	0.91	0.97	0.91	0.92	0.65	0.62	**1.00**	-6.35	0.33	-1.07	40.14	46.77	**-8.83**
Mk02	10/6/55	0.81	0.80	0.80	0.82	0.58	0.64	**1.00**	2.45	2.06	-1.10	40.05	26.56	**-18.54**
Mk03	15/8/150	0.47	0.47	0.38	0.42	0.24	0.42	**1.00**	-0.87	21.22	10.17	97.03	11.90	**-53.40**
Mk04	15/8/89	0.65	0.69	0.73	0.64	0.31	0.56	**1.00**	-5.81	-11.45	0.91	106.26	16.07	**-35.38**
Mk05	15/4/96	0.83	0.89	0.83	0.82	0.26	0.72	**1.00**	-6.37	0.48	1.90	214.82	15.28	**-16.83**
Mk06	10/15/150	0.39	0.39	0.44	0.49	0.12	0.48	**1.00**	-1.45	-12.96	-21.94	223.34	-18.75	**-61.44**
Mk07	20/5/100	0.69	0.81	0.77	0.68	0.37	0.78	**1.00**	-14.68	-10.20	1.16	84.39	-11.54	**-30.89**
Mk08	20/10/134	0.62	0.64	0.68	0.69	0.21	0.62	**1.00**	-3.22	-9.10	-9.63	199.27	0.00	**-37.89**
Mk09	20/10/235	0.48	0.52	0.47	0.55	0.10	0.60	**1.00**	-7.77	2.01	-12.12	379.24	-20.00	**-52.04**
Mk10	20/15/244	0.52	0.54	0.57	0.68	0.12	0.64	**1.00**	-3.48	-9.83	-23.95	323.02	-18.75	**-48.31**
Mk11	30/5/187	0.66	0.73	0.74	0.72	0.24	0.67	**0.98**	-9.76	-11.65	-9.19	170.25	-1.49	**-32.71**
Mk12	30/10/206	0.65	0.76	0.68	0.65	0.18	0.72	**1.00**	-14.30	-4.65	0.01	260.22	-9.72	**-34.83**
Mk13	30/10/212	0.47	0.66	0.52	0.53	0.12	0.63	**1.00**	-28.90	-9.65	-10.16	283.51	-25.40	**-52.78**
Mk14	30/15/283	0.59	0.64	0.65	0.63	0.10	0.75	**1.00**	-8.30	-9.53	-5.96	512.53	-21.33	**-41.20**
Mk15	30/15/279	0.57	0.68	0.55	0.65	0.11	0.82	**1.00**	-16.25	4.34	-11.81	418.78	-30.49	**-43.05**
Avg.		0.71	0.75	0.73	0.74	0.41	068	**0.99**	-5.95	-3.06	-4.98	163.89	4.23	**-29.03**

TABLE VI
THE PARAMETER SETTINGS OF SIX BASELINES

Method	Parameter
NSGA-II	Crossover rate 1; Mutation rate 0.2; No local search.
HPEA	Mutation rate 0.8; Size of success and failure memory 45.
EGA	Crossover rate 0.8; Two self-adaptive learning factors for crossover 0.3, 0.3; Annealing rate coefficient for mutation 2; Iteration number for the current temperature 10; Temperature threshold 50.
HGA	No improvement count (stopping condition) 50; Crossover rate 0.7; Mutation rate 0.03.
SpadePSO	Maximum value of velocity to 10% of the search range; Out-degree of each particle 2; Increasing velocity of out-degree 6; Number of expert particles 5.
HDRL	Number of training epochs 200; Replay memory sizes of C-DQN and A-DQN 16,200; Minibatch size for gradient descent 16; Initial value of and the decrement after each scheduling point 0.6,0.0001; Discount factor 0.95; Optimizer Adam.

the improvements achieved by our SCMO, HGA, and HPEA over NSGA-II are significant, which can reach 61.44%, 23.95%, and 28.90%, respectively. Especially for those instances with more than 100 operations, the superiority of our SCMO will be even more significant, where its HV differences are basically over 30.00%. Although HDRL can beat NSGA-II, HPEA, EGA, and HGA on relatively large-scale instances, it performs worse than our SCMO method on any scale. Since all non-dominant sorting-based methods such as HPEA, EGA, HGA, and SCMO outperform the traditional NSGA-II, subsequent experiments of performance evaluation no longer compare NSGA-II as a baseline.

To make a further performance comparison, Table VII shows the IGD and NR achieved by our SCMO approach and five baselines, i.e., HPEA [40], EGA [29], HGA [17], SpadePSO [41], and HDRL [46]. As shown in the table, our SCMO achieves the lowest average IGD and the highest average NR among the six approaches, which is consistent with the results presented in Table V.

2) Solving Time Evaluation: Table VIII compares the solving time achieved by our SCMO and six baselines for executing 21 benchmarks. The results show that SpadePSO and EGA have a similar solving time as NSGA-II and are much faster than the other three meta-heuristic algorithms (i.e., HPEA, HGA, and SCMO). Specifically, the solving time of SpadePSO, EGA, and NSGA-II is less than 261.44 s, 380.84 s, and 247.11 s for all the instances, respectively. Although the solving time of HDRL is very short, its training time is much more than that of the meta-heuristic methods, and its solution quality cannot be guaranteed. Combined with Table V and Table VIII, we can find that the superiority of local search is more significant on larger instances, although it consumes more solving time. Table VIII also lists the solving time ratios of our SCMO and four state-of-the-art algorithms against NSGA-II, which can directly prove the time efficiency of our SCMO. As shown in the table, we can find that the solving time of HPEA and HGA is skyrocketing with the increasing instance scales, far exceeding SCMO. Although HPEA and HGA consume a similar solving time as SCMO and even faster for small-scale instances, the solving time of our SCMO is relatively more stable compared to HPEA and HGA. For example, the solving time ratios of HPEA and HGA against NSGA-II can be up to 7.50 and 14.01, respectively, while the solving time ratio of SCMO against NSGA-II can be maintained at around 3.00 on large-scale instances. It means that the solving time growth rate of our SCMO is significantly lower than that of NSGA-II, and both solve times maintain a relatively stable proportion on large-scale instances.

3) Stability Analysis: Table IX compares the standard deviation (STD) of HV achieved by our SCMO and five baselines on 21 benchmarks. From this table, we can find that all six algorithms can obtain low STD values, indicating that these algorithms can solve EFJSP stably. To make a more intuitive comparison, this table shows the p-values of HV achieved by five baselines against that by SCMO on 21 benchmarks. Note that the difference is considered significant if a p-value is less than 0.05. Specifically, the notion "*" represents the 5% significance level (i.e., $p \leq 0.05$), and the notion "**" represents the 1% significance level (i.e., $p \leq 0.01$). As shown in the table, the

TABLE VII
IGD AND NR COMPARISON OF DIFFERENT ALGORITHMS

Instance	$n/m/L$	IGD						NR					
		HPEA	EGA	HGA	SpadePSO	HDRL	SCMO	HPEA	EGA	HGA	SpadePSO	HDRL	SCMO
Real_case	10/10/30	0.028	0.066	0.017	0.476	0.514	**0.007**	0.067	0.000	0.133	0.000	0.000	**0.967**
Kc01	4/5/12	0.025	0.034	**0.001**	0.358	0.396	0.004	0.688	0.125	0.813	0.000	0.000	**0.875**
Kc02	8/8/27	0.036	0.087	0.024	0.851	0.762	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Kc03	10/7/29	0.056	0.060	0.038	0.751	0.794	**0.001**	0.000	0.000	0.077	0.000	0.000	**0.923**
Kc04	10/10/30	0.017	0.069	**0.009**	0.735	0.715	0.010	0.077	0.000	0.385	0.000	0.000	**0.538**
Kc05	15/10/56	0.139	0.143	0.026	0.721	0.647	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk01	10/6/56	0.035	0.098	0.040	0.785	0.735	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk02	10/6/55	0.075	0.136	0.075	0.825	0.655	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk03	15/8/150	0.397	0.460	0.397	1.104	0.328	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk04	15/8/89	0.176	0.382	0.157	0.995	0.352	**0.001**	0.150	0.000	0.100	0.000	0.000	**0.750**
Mk05	15/4/96	0.095	0.448	0.099	0.782	0.422	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk06	10/15/150	0.377	0.650	0.260	1.031	0.281	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk07	20/5/100	0.065	0.190	0.112	0.861	0.204	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk08	20/10/134	0.200	0.573	0.217	1.024	0.347	**0.000**	0.000	0.000	0.000	0.000	0.000	**1.000**
Mk09	20/10/235	0.343	0.685	0.217	1.119	0.165	**0.000**	0.000	0.000	0.000	0.000	0.126	**1.000**
Mk10	20/15/244	0.383	0.792	0.218	1.007	0.345	**0.000**	0.000	0.000	0.000	0.000	0.012	**1.000**
Mk11	30/5/187	0.137	0.362	0.147	0.793	0.284	**0.006**	0.000	0.000	0.029	0.000	0.000	**0.941**
Mk12	30/10/206	0.147	0.629	0.131	0.881	0.207	**0.000**	0.000	0.000	0.000	0.000	0.067	**1.000**
Mk13	30/10/212	0.428	0.717	0.312	0.984	0.358	**0.000**	0.000	0.000	0.000	0.000	0.052	**1.000**
Mk14	30/15/283	0.293	0.717	0.312	0.984	0.256	**0.000**	0.000	0.000	0.000	0.000	0.026	**1.000**
Mk15	30/15/279	0.406	0.821	0.201	1.066	0.156	**0.000**	0.000	0.000	0.000	0.000	0.045	**1.000**
Avg.		0.184	0.387	0.143	0.863	0.425	**0.001**	0.047	0.006	0.073	0.000	0.016	**0.952**

TABLE VIII
SOLVING TIME COMPARISON OF DIFFERENT ALGORITHMS

Instance	$n/m/L$	Solving Time (s)							Solving Time Ratios (%, against NSGA-II)					
		NSGA-II	HPEA	EGA	HGA	SpadePSO	HDRL	SCMO	HPEA	EGA	HGA	SpadePSO	HDRL	SCMO
Real_case	10/10/32	33.64	228.56	83.50	215.11	35.57	0.11	168.70	6.79	2.48	6.40	1.06	0.00	5.02
Kc01	4/5/12	20.88	115.11	70.19	46.97	21.29	0.05	116.20	5.51	3.36	2.25	1.02	0.00	5.56
Kc02	8/8/27	29.25	206.89	68.39	121.90	34.23	0.10	172.04	7.07	2.34	4.17	1.17	0.00	5.88
Kc03	10/7/29	32.36	216.75	69.33	136.93	35.23	0.10	207.27	6.70	2.14	4.23	1.09	0.00	6.41
Kc04	10/10/30	32.11	222.86	74.39	131.52	36.69	0.11	198.89	6.94	2.32	4.10	1.14	0.00	6.19
Kc05	15/10/56	53.66	374.94	97.61	316.59	58.30	0.19	247.56	6.99	1.82	5.90	1.09	0.00	4.61
Mk01	10/6/56	51.26	362.92	95.53	342.15	56.79	0.18	232.68	7.08	1.86	6.67	1.11	0.00	4.54
Mk02	10/6/55	49.23	354.40	94.08	316.55	57.31	0.19	228.30	7.20	1.91	6.43	1.16	0.00	4.64
Mk03	15/8/150	129.54	913.37	197.67	984.57	141.34	0.36	303.85	7.05	1.53	7.60	1.09	0.00	2.35
Mk04	15/8/89	81.94	569.02	133.52	515.24	91.45	0.28	236.46	6.94	1.63	6.29	1.12	0.00	2.89
Mk05	15/4/96	86.39	593.76	140.17	900.97	93.67	0.30	291.38	6.87	1.62	10.43	1.08	0.00	3.37
Mk06	10/15/150	128.98	883.35	200.55	717.30	140.18	0.37	292.28	6.85	1.55	5.56	1.09	0.00	2.27
Mk07	20/5/100	92.18	631.18	150.91	851.39	99.17	0.30	299.34	6.85	1.64	9.24	1.08	0.00	3.25
Mk08	20/10/134	119.61	869.01	194.11	923.59	129.70	0.36	364.74	7.27	1.62	7.72	1.08	0.00	3.05
Mk09	20/10/235	202.66	1434.96	308.50	2016.35	210.04	0.57	432.08	7.08	1.52	9.95	1.04	0.00	2.13
Mk10	20/15/244	204.03	1509.38	322.58	1676.09	215.72	0.57	444.37	7.40	1.58	8.21	1.06	0.00	2.18
Mk11	30/5/187	165.81	1204.50	260.43	2322.89	172.58	0.52	436.06	7.26	1.57	14.01	1.04	0.00	2.63
Mk12	30/10/206	181.10	1333.30	281.71	1567.67	189.30	0.54	422.16	7.36	1.56	8.66	1.05	0.00	2.33
Mk13	30/10/212	186.74	1342.13	291.76	1571.18	196.23	0.54	498.25	7.19	1.56	8.41	1.05	0.00	2.67
Mk14	30/15/283	247.11	1853.71	380.84	2064.14	261.44	0.63	573.90	7.50	1.54	8.35	1.06	0.00	2.32
Mk15	30/15/279	245.50	1804.45	379.93	1876.12	254.83	0.58	549.50	7.35	1.55	7.64	1.04	0.00	2.24
Avg.		113.05	810.69	185.51	934.06	120.53	0.33	319.81	7.01	1.84	7.25	1.08	0.00	3.64

p-values are almost all lower than 0.05, representing obvious differences between the HV values obtained by SCMO and the other five algorithms except for Kc01. Especially for larger instances, the advantages of SCMO over other algorithms are particularly significant.

Table X shows the T-test results concerning the IGD value obtained by SCMO and five baselines. As shown in the table, the STD of SCMO is lower than five baselines in 14 out of 21 instances, indicating that our SCMO approach can stably obtain high-quality solutions to solve the proposed EFJSP. Meanwhile, the p-values are almost all lower than 0.05, representing obvious differences between the IGD value obtained by SCMO and other algorithms. Especially for larger instances, the advantages of SCMO over other algorithms are particularly significant. This is mainly because the space-cooperation population update of SCMO can better explore and exploit potential search space, alleviating the instability of results caused by randomness.

B. *Effectiveness of Space-Cooperation Optimization*

To evaluate the effectiveness of key components in our SCMO (i.e., space-cooperation population update and difference-driven local search) on improving the performance of NSGA-II, we conducted ablation studies on three benchmark instances with different scales, i.e., Real_case, Mk05, and Mk12, each of which consists of 32, 96, and 206 operations, respectively. Table XI presents the contributions of key components based on ablation studies. Column 1 denotes the original NSGA-II without adopting a space-cooperation population update or difference-driven local search. Columns 4-6 indicate the HV values on three benchmarks. As shown in the table, we can find that each component achieves an improvement on NSGA-II in HV. Especially for large-scale instances, each component has a more significant improvement on the NSGA-II. For example, compared with the NSGA-II, the HV values of our SCMO can be improved from 0.351 to 1.000 on a large-scale instance Mk12.

TABLE IX
HV STD COMPARISON OF DIFFERENT ALGORITHMS AND P-VALUES AGAINST SCMO

Instance	$n/m/L$	HV STD						P-Values (against SCMO)				
		HPEA	EGA	HGA	SpadePSO	HDRL	SCMO	HPEA	EGA	HGA	SpadePSO	HDRL
Real_case	10/10/32	3.267	3.482	2.533	9.089	4.766	3.981	0.000**	0.000**	0.000**	0.000**	0.000**
Kc01	4/5/12	0.489	0.494	0.895	1.596	2.542	0.511	0.000**	0.883	0.601	0.000**	0.000**
Kc02	8/8/27	2.097	1.472	0.683	3.243	2.854	0.501	0.000**	0.000**	0.000**	0.000**	0.000**
Kc03	10/7/29	1.316	1.814	1.262	3.267	3.490	1.933	0.000**	0.000**	0.000**	0.000**	0.000**
Kc04	10/10/30	1.129	2.018	0.854	3.522	4.868	0.601	0.000**	0.001**	0.000**	0.000**	0.000**
Kc05	15/10/56	2.833	3.812	5.509	7.108	6.411	1.562	0.000**	0.000**	0.000**	0.000**	0.000**
Mk01	10/6/56	1.421	2.067	2.202	3.395	2.591	0.644	0.000**	0.000**	0.000**	0.000**	0.000**
Mk02	10/6/55	1.612	3.373	1.909	6.255	3.870	0.845	0.000**	0.000**	0.000**	0.000**	0.000**
Mk03	15/8/150	12.433	23.716	23.272	19.175	25.865	6.764	0.000**	0.000**	0.000**	0.000**	0.000**
Mk04	15/8/89	3.345	0.299	2.281	6.344	5.733	2.343	0.000**	0.000**	0.000**	0.000**	0.000**
Mk05	15/4/96	1.662	7.295	1.882	6.883	8.903	1.788	0.000**	0.000**	0.000**	0.000**	0.000**
Mk06	10/15/150	8.372	9.455	14.162	16.452	15.028	6.518	0.000**	0.000**	0.000**	0.000**	0.000**
Mk07	20/5/100	6.142	6.142	4.391	22.835	15.026	3.496	0.000**	0.000**	0.000**	0.000**	0.000**
Mk08	20/10/134	12.524	11.437	8.119	13.695	13.603	6.012	0.000**	0.000**	0.000**	0.000**	0.000**
Mk09	20/10/235	22.182	39.153	33.371	17.107	28.998	13.633	0.000**	0.000**	0.000**	0.000**	0.000**
Mk10	20/15/244	14.412	24.861	34.472	28.854	30.675	22.895	0.000**	0.000**	0.000**	0.000**	0.000**
Mk11	30/5/187	12.862	14.681	12.046	30.565	27.091	4.792	0.000**	0.000**	0.000**	0.000**	0.000**
Mk12	30/10/206	19.212	20.665	19.433	29.986	24.212	9.241	0.000**	0.000**	0.000**	0.000**	0.000**
Mk13	30/10/212	14.442	30.285	32.791	23.633	29.090	12.445	0.000**	0.000**	0.000**	0.000**	0.000**
Mk14	30/15/283	29.255	36.122	29.132	41.56	34.275	17.711	0.000**	0.000**	0.000**	0.000**	0.000**
Mk15	30/15/279	24.726	24.424	58.564	29.822	41.973	30.396	0.000**	0.000**	0.000**	0.000**	0.000**
Avg.		9.329	12.717	13.798	15.447	15.803	7.077	-	-	-	-	-

TABLE X
IGD STD COMPARISON OF DIFFERENT ALGORITHMS AND P-VALUES AGAINST SCMO

Instance	$n/m/L$	IGD STD						P-Values (against SCMO)				
		HPEA	EGA	HGA	SpadePSO	HDRL	SCMO	HPEA	EGA	HGA	SpadePSO	HDRL
Real_case	10/10/32	2.196	3.061	3.122	7.805	8.772	3.542	0.000**	0.000**	0.000**	0.000**	0.000**
Kc01	4/5/12	0.948	0.854	0.851	1.538	2.783	0.417	0.000**	0.653	0.580	0.000**	0.000**
Kc02	8/8/27	1.871	1.633	0.890	2.752	1.590	0.602	0.000**	0.000**	0.000**	0.000**	0.000**
Kc03	10/7/29	1.527	2.255	1.651	4.702	3.216	2.056	0.000**	0.000**	0.000**	0.000**	0.000**
Kc04	10/10/30	1.446	1.962	1.170	4.085	5.388	1.012	0.000**	0.002**	0.000**	0.000**	0.000**
Kc05	15/10/56	2.582	3.451	4.803	5.886	5.982	0.967	0.000**	0.000**	0.000**	0.000**	0.000**
Mk01	10/6/56	2.164	2.735	2.402	4.228	4.230	0.851	0.000**	0.000**	0.000**	0.000**	0.000**
Mk02	10/6/55	2.107	2.963	2.093	5.771	6.316	1.466	0.000**	0.000**	0.000**	0.000**	0.000**
Mk03	15/8/150	8.746	18.384	19.037	17.382	23.705	5.028	0.000**	0.000**	0.000**	0.000**	0.000**
Mk04	15/8/89	2.044	0.882	2.473	5.938	3.277	2.498	0.000**	0.000**	0.000**	0.000**	0.000**
Mk05	15/4/96	2.361	5.847	2.203	7.398	6.921	2.229	0.000**	0.000**	0.000**	0.000**	0.000**
Mk06	10/15/150	7.922	7.360	16.283	15.901	12.990	5.378	0.000**	0.000**	0.000**	0.000**	0.000**
Mk07	20/5/100	4.085	5.934	5.279	16.927	13.306	3.028	0.000**	0.000**	0.000**	0.000**	0.000**
Mk08	20/10/134	9.023	10.372	7.328	11.929	14.785	5.871	0.000**	0.000**	0.000**	0.000**	0.000**
Mk09	20/10/235	19.032	27.930	29.928	15.136	22.381	12.765	0.000**	0.000**	0.000**	0.000**	0.000**
Mk10	20/15/244	12.977	18.634	29.940	24.872	21.792	17.037	0.000**	0.000**	0.000**	0.000**	0.000**
Mk11	30/5/187	9.047	15.825	10.370	25.525	18.803	5.370	0.000**	0.000**	0.000**	0.000**	0.000**
Mk12	30/10/206	14.973	15.387	17.659	26.201	23.552	8.446	0.000**	0.000**	0.000**	0.000**	0.000**
Mk13	30/10/212	11.088	17.309	16.872	21.072	22.671	6.183	0.000**	0.000**	0.000**	0.000**	0.000**
Mk14	30/15/283	24.088	2.083	22.776	24.163	19.796	11.376	0.000**	0.000**	0.000**	0.000**	0.000**
Mk15	30/15/279	13.974	18.361	33.357	24.274	27.068	27.565	0.000**	0.000**	0.000**	0.000**	0.000**
Avg.		7.343	8.725	10.976	13.023	12.825	5.890	-	-	-	-	-

TABLE XI
ABLATION RESULTS CONSIDERING THE IMPACTS OF KEY COMPONENTS

Component			HV		
NSGA-II	Space-cooperation	LS	Real_case	Mk05	Mk12
√			0.943	0.638	0.351
√	√		0.977	0.830	0.514
√	√	√	0.999	1.000	1.000

To further investigate the effectiveness of critical operation selection based on differences, we conducted comparison experiments considering two cases: SCMO with or without difference-driven critical operation selection in local search (LS). Fig. 8 shows the histogram plots for the HV values and the solving time of two cases on three benchmarks with different scales, i.e., Real_case, Mk05, and Mk12. Note that each case is performed in 30 independent runs on each benchmark. The results in Fig. 8(a) clearly show that local search with difference-driven critical operation selection performs slightly better and is more stable than that without difference-driven critical operation selection. From Fig. 8(b), we can observe that difference-driven critical operation selection effectively accelerates the local search process. Furthermore, with the increase of instance scale, the solving time of local search with difference-driven critical operation selection grows significantly slower than that without difference-driven critical operation selection. It means that selectively adjusting the position of the critical operations can not only reduce local search time but also improve neighbor exploration efficiency.

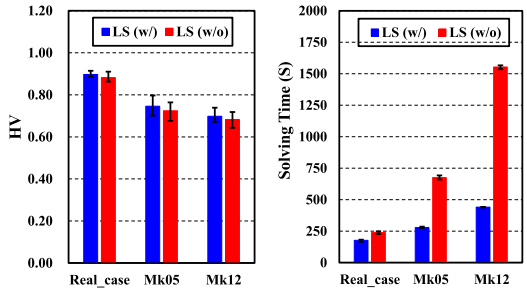

Fig. 8. Histogram plots for (a) HV values and (b) solving time of our SCMO considering difference-driven critical operation selection or not in local search.

VII. Conclusion and Feture Work

Due to the improper evaluation of similarity among individuals, existing non-dominated sorting-based meta-heuristic methods suffer from the problem of insufficient population diversity while solving the Energy-efficient Flexible Job Shop Scheduling Problem (EFJSP) with both setup and transportation processes. To address this issue, this paper presented a space-cooperation multi-objective optimization method named SCMO that enables efficient and effective resource allocation to reduce energy consumption without sacrificing time (i.e., makespan). Based on a three-vector representation, our SCMO method fully evaluates the similarity among individuals in objective and decision space, which can improve evolution efficiency by accurately eliminating repetitive individuals. Furthermore, we proposed a difference-driven local search to facilitate efficient search to explore better neighbors. Comprehensive experiments on various EFJSP instances demonstrated the effectiveness of SCMO over state-of-the-art methods from the perspectives of solution quality, stability, and solving time.

Although the SCMO method can effectively address the proposed EFJSP problem, it still has some limitations. First, it assumes that each operation in manufacturing systems is assigned a fixed processing time. However, this is not always true in real-world scenarios, where uncertain disruptions in manufacturing environments may dramatically affect the effectiveness of dispatching solutions. In the future, we plan to take into account various uncertain factors (e.g., machine breakdowns and urgent job arrivals) from real scenarios. Second, since manufacturing systems cannot fully handle unrecognized disruptions, how to reasonably arrange workers to improve the cooperation ability between machines during scheduling processes is also an interesting topic worthy of further study.

References

[1] R. Zhang and R. Chiong, "Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," *J. Cleaner Prod.*, vol. 112, no. 4, pp. 3361–3375, 2016.

[2] M. Chen, X. Zhang, H. Gu, T. Wei, and Q. Zhu, "Sustainability-oriented evaluation and optimization for MPSoC task allocation and scheduling under thermal and energy variations," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 2, pp. 84–97, Second Quarter 2018.

[3] G. Gong et al., "A two-stage memetic algorithm for energy-efficient flexible job shop scheduling by means of decreasing the total number of machine restarts," *Swarm Evol. Computation*, vol. 75, 2022, Art. no. 101131.

[4] J. Li, Y. Han, K. Gao, X. Xiao, and P. Duan, "Bi-population balancing multi-objective algorithm for fuzzy flexible job shop with energy and transportation," *IEEE Trans. Automat. Sci. Eng.*, vol. 21, no. 3, pp. 4686–4702, pp. 1–17, Jul. 2024, doi: 10.1109/TASE.2023.3300922.

[5] Z. Li, B. Qian, R. Hu, L. Chang, and J. Yang, "An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups," *Knowl.-Based Syst.*, vol. 173, no. 3, pp. 1097–1109, 2019.

[6] L. Meng, C. Zhang, X. Shao, and Y. Ren, "Milp models for energy-aware flexible job shop scheduling problem," *J. Cleaner Prod.*, vol. 210, no. 10, pp. 710–723, 2019.

[7] X. Gong, Y. Liu, N. Lohse, T. Pessemier, L. Martens, and W. Joseph, "Energy- and labor-aware production scheduling for industrial demand response using adaptive multiobjective memetic algorithm," *IEEE Trans. Ind. Inform.*, vol. 15, no. 2, pp. 942–953, Feb. 2019.

[8] L. He, R. Chiong, W. Li, S. Dhakal, Y. Cao, and Y. Zhang, "Multiobjective optimization of energy-efficient JOB-shop scheduling with dynamic reference point-based fuzzy relative entropy," *IEEE Trans. Ind. Inform.*, vol. 18, no. 1, pp. 600–610, Jan. 2022.

[9] G. Xie, X. Xiao, H. Peng, R. Li, and K. Li, "A survey of low-energy parallel scheduling algorithms," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 1, pp. 27–46, First Quarter 2022.

[10] N. Rakoviti, D. Li, N. Zhang, J. Li, L. Zhang, and X. Xiao, "Novel approach to energy-efficient flexible job-shop scheduling problems," *Energy*, vol. 238, 2022, Art. no. 121773.

[11] C. Lu, L. Gao, Q. Pan, X. Li, and J. Zheng, "A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution," *Appl. Soft Comput.*, vol. 75, pp. 728–749, 2019.

[12] K. Fang, N. A. Uhan, F. Zhao, and J. Sutherland, "Flow shop scheduling with peak power consumption constraints," *Ann. Operations Res.*, vol. 206, pp. 115–145, 2013.

[13] J. Ding, S. Dauzere-Peres, L. Shen, and Z. Lv, "A novel evolutionary algorithm for energy-efficient scheduling in flexible job shops," *IEEE Trans. Evol. Comput.*, vol. 27, no. 5, pp. 1470–1484, Oct. 2023.

[14] B. Zhou and Y. Lei, "Bi-objective grey wolf optimization algorithm combined levy flight mechanism for the FMC green scheduling problem," *Appl. Soft Comput.*, vol. 111, 2021, Art. no. 107717.

[15] M. Garey, D. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Int. J. Prod. Res.*, vol. 1, pp. 117–129, 1976.

[16] E. Yağmur and S. Kesen, "Bi-objective coordinated production and transportation scheduling problem with sustainability: Formulation and solution approaches," *Int. J. Prod. Res.*, vol. 61, no. 3, pp. 774–795, 2022.

[17] A. Turkyilmaza, O. Senvarb, I. Unalb, and S. Bulkan, "A hybrid genetic algorithm based on a two-level hypervolume contribution measure selection strategy for bi-objective flexible job shop problem," *Comput. Operations Res.*, vol. 141, 2022, Art. no. 105694.

[18] A. Oukil, A. El-Bouri, and A. Emrouznejad, "Energy-aware job scheduling in a multi-objective production environment–An integrated Dea-Owa model," *Comput. Ind. Eng.*, vol. 168, 2022, Art. no. 108065.

[19] D. 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.

[20] W. Zhang, J. Ding, Y. Wang, S. Zhang, and Z. Xiong, "Multi-perspective collaborative scheduling using extended genetic algorithm with interval-valued intuitionistic fuzzy entropy weight method," *J. Manuf. Syst.*, vol. 53, pp. 249–260, 2019.

[21] Z. Pan, D. Lei, and L. Wang, "A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling," *IEEE Trans. Cybern.*, vol. 56, no. 2, pp. 5051–5063, Jun. 2022.

[22] Y. Fu, M. Zhou, X. Guo, L. Qi, K. Gao, and A. Albeshri, "Multiobjective scheduling of energy-efficient stochastic hybrid open shop with brain storm optimization and simulation evaluation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 54, no. 7, pp. 4260–4272, Jul. 2024.

[23] M. Kurdi, "An improved island model memetic algorithm with a new cooperation phase for multi-objective job shop scheduling problem," *Comput. Ind. Eng.*, vol. 111, pp. 183–201, 2017.

[24] Q. Luo, Q. Deng, G. Gong, X. Guo, and X. Liu, "A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm," *Expert Syst. Appli.*, vol. 207, 2022, Art. no. 117984.

[25] E. Jiang and L. Wang, "Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices," *Knowl.-Based Syst.*, vol. 204, 2020, Art. no. 106177.
[26] E. Ahmadi, M. Zandieh, M. Farrokh, and S. Emami, "A multi objective optimization approach for flexible job shop scheduling problem under random machine break down by evolutionary algorithms," *Comput. Operations Res.*, vol. 73, pp. 56–66, 2016.
[27] G. Gong, Q. Deng, X. Gong, and D. Huang, "A non-dominated ensemble fitness ranking algorithm for multi-objective flexible job-shop scheduling problem considering worker flexibility and green factors," *Knowl.-Based Syst.*, vol. 231, 2021, Art. no. 107430.
[28] Z. Cao, C. Lin, and M. Zhou, "A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 1, pp. 56–69, 2021.
[29] M. Dai, D. Tang, A. Giret, and M. Salido, "Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints," *Robot. Comput.- Integr. Manuf.*, vol. 59, pp. 143–157, 2019.
[30] R. Q. Dao-Er Ji and Y. Wang, "A new hybrid genetic algorithm for job shop scheduling problem," *Comput. Operations Res.*, vol. 39, pp. 2291–2299, 2012.
[31] C. Lin, Z. Cao, and M. Zhou, "Learning-based grey wolf optimizer for stochastic flexible job shop scheduling," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 4, pp. 3659–3671, Oct. 2022.
[32] J. Ding, Y. Wang, S. Zhang, W. Zhang, and Z. Xiong, "Robust and stable multi-task manufacturing scheduling with uncertainties using a two-stage extended genetic algorithm," *Enterprise Inf. Syst.*, vol. 13, no. 10, pp. 1442–1470, 2019.
[33] Y. Yuan and H. Xu, "Multiobjective flexible job shop scheduling using memetic algorithms," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 1, pp. 336–353, Jan. 2015.
[34] L. Gao, X. Lin, X. Wen, C. Lu, and F. Wen, "A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem," *Comput. Ind. Eng.*, vol. 88, pp. 417–429, 2015.
[35] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic," *Math. Comput. Simul.*, vol. 60, no. 3/5, pp. 245–276, 2002.
[36] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Ann. Operations Res.*, vol. 41, pp. 157–183, 1993.
[37] C. Goh and K. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.
[38] C. Coello and N. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genet. Program. Evolvable Machines*, vol. 13, pp. 163–190, 2005.
[39] J. Zhou et al., "Resource management for improving soft-error and lifetime reliability of real-time mpsocs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 12, pp. 2215–2228, Dec. 2019.
[40] R. Li, W. Gong, and C. Lu, "Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time," *Comput. Ind. Eng.*, vol. 168, 2022, Art. no. 108099.
[41] X. Wu et al., "Incorporating surprisingly popular algorithm and euclidean distance-based adaptive topology into PSO," *Swarm Evol. Computation*, vol. 76, 2023, Art. no. 101222.
[42] Z. Zeng, X. Li, and C. Bai, "A deep reinforcement learning approach to flexible job shop scheduling," in *Proc. 2022 IEEE Int. Conf. Systems, Man, Cybern.*, 2022, pp. 884–890.
[43] Y. Feng, L. Zhang, Z. Yang, Y. Guo, and D. Yang, "Flexible job shop scheduling based on deep reinforcement learning," in *Proc. 2021 5th Asian Conf. Artif. Intell. Technol.*, 2022, pp. 660–666.
[44] X. Shang, "A study of deep learning neural network algorithms and genetic algorithms for FJSP," *J. Appl. Math.*, 2023, Art. no. 4573352.
[45] J. Guo, D. Lei, and M. Li, "Two-phase imperialist competitive algorithm for energy-efficient flexible job shop scheduling," *J. Intell. Fuzzy Syst.*, vol. 40, no. 6, pp. 12125–12137, 2021.
[46] J. Zhang et al., "An adaptive multi-objective multi-task scheduling method by hierarchical deep reinforcement learning," *Appl. Soft Comput.*, vol. 154, 2024, Art. no. 111342.

Jiepin Ding (Student Member, IEEE) received the BS and MS degrees from the Department of Information Management and Artificial Intelligence, Zhejiang University of Finance & Economics, Hangzhou, China, in 2016 and 2020, respectively. She is currently working toward the PhD degree with the Software Engineering Institute, East China Normal University, Shanghai, China. Her research interests include production scheduling, heuristic algorithms, and reinforcement learning.

Jun Xia received the BS degree in computer science and technology from Hainan University, Haikou, China, in 2016, the MS degree in computer science and technology from Jiangnan University, Wuxi, China, in 2019, and the PhD degree in software engineering institute from the East China Normal University, Shanghai, China, in 2023. He is currently a postdoctoral researcher with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, USA. His research interests include the areas of embedded systems, AIoT applications, and trustworthy computing.

Yaning Yang (Student Member, IEEE) is an associate professor with the School of Physics and Electronic Information Engineering, Ningxia Normal University, Guyuan, China. She is currently working toward the PhD degree with the Software Engineering Institute, East China Normal University, Shanghai, China. Her research interests include reinforcement learning, embedded systems, cloud/edge computing, and serverless computing.

Junlong Zhou (Member, IEEE) received the PhD degree in computer science from East China Normal University, Shanghai, China, in 2017. He was a visiting scholar with the University of Notre Dame, Notre Dame, IN, USA, from 2014 to 2015. He is currently an associate professor with the Nanjing University of Science and Technology, China. His research interests include embedded systems, the Internet of Things, and cyber-physical systems, where he has published 90 refereed papers, including 30 in premier IEEE/ACM Transactions. He received the Best Paper Award from IEEE iThings 2020. He was an Associate Editor of JSA (Elsevier), JCSC, and IET CPS; and a Guest Editor of six ACM/IET/Elsevier journals, such as ACM Transactions on Cyber-Physical Systems.

Mingsong Chen (Senior Member, IEEE) received the BS and ME degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006, respectively, and the PhD degree from the University of Florida, Gainesville, FL, USA, in 2010. He is currently a professor with the Software Engineering Institute, East China Normal University, Shanghai, China. His research interests include the area of design automation of cyber-physical systems, EDA, embedded systems, and formal verification techniques. Currently, he serves as the Director of the Engineering Research Center of Software/Hardware Co-design Technology and Application affiliated to the Ministry of Education, China, and the vice director with the technical committee of embedded systems of China Computer Federation (CCF).

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985, and the PhD degree in computer science from the University of Houston, in 1990. He is a SUNY distinguished professor with the State University of New York and a National Distinguished professor with Hunan University (China). He has authored or co-authored more than 1060 journal articles, book chapters, and refereed conference papers. He received several best paper awards from international conferences including PDPTA-1996, NAECON-1997, IPDPS-2000, ISPA-2016, NPC-2019, ISPA-2019, and CPSCom-2022. He holds over 75 patents announced or authorized by the Chinese National Intellectual Property Administration. Since 2020, he has been among the world's top few most influential scientists in parallel and distributed computing regarding single-year impact (ranked #2) and career-long impact (ranked #4) based on a composite indicator of the Scopus citation database. He was a 2017 recipient of the Albert Nelson Marquis Lifetime Achievement Award for being listed in Marquis Who's Who in Science and Engineering, Who's Who in America, Who's Who in the World, and Who's Who in American Education for over twenty consecutive years. He received the Distinguished Alumnus Award from the Computer Science Department at the University of Houston in 2018. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the 2022-2023 International Science and Technology Cooperation Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS fellow, an AAIA fellow, an ACIS fellow, and an AIIA Fellow. He is a member of Academia Europaea (Academician of the Academy of Europe).