Co-Evolution With Deep Reinforcement Learning for Energy-Aware Distributed Heterogeneous Flexible Job Shop Scheduling

Rui Li, Wenyin Gong, *Member, IEEE*, Ling Wang, *Member, IEEE*, Chao Lu, and Chenxin Dong

***Abstract*—Energy-aware distributed heterogeneous flexible job shop scheduling (DHFJS) problem is an extension of the traditional FJS, which is harder to solve. This work aims to minimize total energy consumption (TEC) and makespan for DHFJS. A deep *Q*-networks-based co-evolution algorithm (DQCE) is proposed to solve this NP-hard problem, which includes four parts: First, a new co-evolutionary framework is proposed, which allocates sufficient computation to global searching and executes local search surrounding elite solutions. Next, nine problem features-based local search operators are designed to accelerate convergence. Moreover, deep *Q*-networks are applied to learn and select the best operator for each solution. Furthermore, an efficient heuristic method is proposed to reduce TEC. Finally, 20 instances and a real-world case are employed to evaluate the effectiveness of DQCE. Experimental results indicate that DQCE outperforms the six state-of-the-art algorithms for DHFJS.**

***Index Terms*—Co-evolution, deep *Q*-networks (DQNs), distributed heterogeneous flexible job shop scheduling (DHFJS) problem, energy-saving, multiobjective optimization.**

I. Introduction

With the development of global economic integration, the products are more refined with more operations and longer manufacturing cycles [1], [2]. However, the customers expect the enterprises to respond quickly to demand and reduce production time. Nevertheless, the traditional single-factory manufacturing mode cannot meet this growing request [3], [4]. Thus, the parts of the large equipment will be dispatched to multiple factories for parallel manufacturing to accelerate production, defined as distributed manufacturing [5]. Flexible job shop scheduling (FJS) is the classical shop scheduling problem, and distributed FJS (DFJS) is its extension [6], which divides the complex problem into three parts: 1) assigning all jobs to several factories; 2) determining the operations' processing sequence in different factories; and 3) choosing an appropriate machine for each operation [7]. Therefore, DFJS is more difficult than FJS, defined as an NP-hard problem [8]. The meta-heuristic algorithms are the mainstream method for DFJS, including the genetic algorithm [9], differential evolution algorithm [10], tabu search algorithm [11], chemical reaction algorithm [12], and estimation of distribution algorithm [13]. However, based on previous research, the factory type is homogeneous. Each factory contains the same machine type with the same processing and selectable machine for each operation. Nevertheless, this assumption cannot be realized in real-world distributed manufacturing due to the limitations of cost, space, and ability of workers. The number of selectable machines, machine processing time, or shop type differs in most situations. This distributed manufacturing environment is defined as a heterogeneous factory [14]. For example, during the machining stage in the process flow of large engineering equipment, the machine has compatibility, and some machines can process different scales of operations. Generally, the enterprise has multiple factories to increase its capacity and profit. Meanwhile, each operation has different processing times different groups of workers have different productivity. Thus, the distributed heterogeneous FJS (DHFJS) problem can be proposed by extracting those features from real-world manufacturing.

Recently, heterogeneous distributed scheduling problems received widespread concerns, and traditional shop scheduling problems were extended. Wang et al. [15] proposed the distributed heterogeneous welding flow shop scheduling. Shao et al. [16] studied the distributed heterogeneous hybrid flow shop problem. Chen et al. [17] presented a distributed heterogeneous flow shop scheduling. Zhao et al. [18], [19], [20] constructed a no-wait flow shop problem with multiple heterogeneous factories. Lu et al. [14] presented a distributed flow shop problem with different shop types. However, summarizing the current work about distributed heterogeneous manufacturing indicates no study on DHFJS due to its complexity and difficulty. Thus, the DHFJS should be studied to provide a theory guide for practical manufacturing. Moreover, with the growing global warming, controlling carbon dioxide emissions has become a hot topic

Manuscript received 13 March 2023; revised 7 June 2023; accepted 9 August 2023. Date of publication 6 September 2023; date of current version 19 December 2023. This work was supported by the National Natural Science Foundation of China under Grant 62076225 and Grant 62273193. This article was recommended by Associate Editor W. Shen. *(Corresponding authors: Wenyin Gong; Ling Wang.)*

Rui Li and Ling Wang are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: li-r23@mails.tsinghua.edu.cn; wangling@tsinghua.edu.cn).

Wenyin Gong and Chao Lu are with the School of Computer Science, China University of Geosciences, Wuhan 430074, China (e-mail: wygong@cug.edu.cn; luchao@cug.edu.cn).

Chenxin Dong is with the School of Mechanical and Automotive Engineering, Qingdao Hengxing University of Science and Technology, Qingdao 266100, China (e-mail: deyangxuanyi@hotmail.com).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2023.3305541.

Digital Object Identifier 10.1109/TSMC.2023.3305541

that cannot be ignored [21]. Saving energy is another important objective in modern manufacturing. Thus, studying an efficient energy-saving strategy makes sense for the DHFJS.

Knowledge-driven neighborhood structures can improve the convergence of population in scheduling problems [22]. The memetic framework, which executes local search after offspring generation, has been widely utilized due to its efficiency [23]. However, various experiments indicate that the memetic framework is inefficient for the distributed shop scheduling problem with multiple objectives due to the following reasons: First, the distributed shop scheduling with heterogeneous factories is complex, and the objective space is quite large. Second, the algorithm requires more global search to keep sufficient diversity to find more nondominated solutions. Nevertheless, directly applying local search to the population will consume much computation and sharply reduce the diversity. Moreover, when the population falls into local optima, it cannot explore more potential objective space to find more nondominated solutions. Thus, inspired by the interdependence of living things in nature, a co-evolutionary framework is designed to solve the problem. Similar to cooperative evolutionary [24], [25], competitive evolutionary [26], [27], and memetic evolutionary [28], [29], a co-evolutionary framework based on parasitic behavior also abstracts a relationship from nature, which has never been proposed for distributed shop scheduling problems.

Deep-Q-network (DQN) is a powerful unsupervised learning technique that can automatically learn the data distribution from the environment and optimize the decision to find the globally optimal solution [30]. Moreover, DQN can efficiently solve the strategies selection problem with large state space without defining the states set. Due to its advantage, DQN has been widely applied in different problems, including reversible data hiding for color images [31], vehicle-to-network communications [32], joint offloading and resource allocation [33], battery energy storage systems [34], gait pattern controller for humanoid robots [35], edge computing [36], Internet of Things [37], scheduling agile earth observation satellites [38], and FJS [39], [40]. Thus, considering the large state space of DHFJS, DQN is applied to learn and decide how to select an optimal operator.

This study employs a DQNs-based co-evolution algorithm (DQCE) to solve the energy-aware DHFJS problem, in which the total energy consumption (TEC) and makespan are minimized. Then, nine knowledge-driven local search strategies are proposed to enhance the convergence. The contributions of this work are summarized as follows.

1) *Problem Extension:* The DHFJS is first studied based on the real-world manufacturing scheduling problem. It considers the heterogeneous factories with different processing times of each operation.
2) *Co-Evolutionary Framework:* In order to efficiently balance limited computations between the global and local search for distributing shop scheduling problems, a novel co-evolutionary framework based on parasitic behavior is proposed. The global and local searches are dispatched to two populations. The knowledge of global search is transferred to the elite population for cooperative evolution.
3) *Local Search Operators:* In order to solve the heterogeneous factory constraint, a ranking-based factory selection operator is designed based on problem knowledge. Each operation has a higher rate to select the factory with a smaller average processing time. Meanwhile, the operator ensures that other factories have the probability to be selected. This design balances exploration and exploitation.
4) *DQN-Based Operators Selection Framework:* In order to choose the optimal operator for each solution, DQN is employed to learn the mapping between solution space and operator. First, the scheduling of DHFJS is fed into the network. Then, the network provides the decision for the co-evolutionary algorithm by abstracting the features of the solution. Next, the scheduling is optimized by the selected operator and gives feedback to the network. Then, the network learns from the collected data and continuously strengthens its decision-making ability to enhance the autonomy of the algorithm.
5) *Algorithm Design:* A novel algorithm DQCE is proposed, which combines the DQN-based operator selection framework with the co-evolutionary algorithm. When solving different scales of DHFJS, it has strong autonomy and can decide the optimal operator selection to assist the co-evolution to local search.

The effectiveness of DQCE is validated on 20 instances and a real-world case. The DQCE is compared to six state-of-the-art algorithms. Experiment results indicate the superiority of DQCE for solving DHFJS.

The other sections of the article are organized as follows. Section II introduces the problem statement and the MILP model of DHFJS. Section III introduces our approach in detail. Experimental results and discussion are presented in Section IV. Finally, Section V describes the conclusion and future work.

II. Problem Description and MILP Model

A. Problem Description

The DHFJS should solve the following three subproblems: 1) *Factory Assignment (FA):* determining the processing factory for all jobs; 2) *Operation Sequencing:* arranging a reasonable sequence for all operations assigned to each factory; and 3) *Machine Selection (MS):* selecting a suitable machine for each operation from flexible machines set. DHFJS includes n_f heterogeneous factories and n jobs dispatched to different factories, where every job has n_i operations. Moreover, the total operations of a job must be assigned to the same factory. As for the same operation $O_{i,j}$, it has the same selectable machine set but different processing times in different factories.

Before solving DHFJS, the following assumptions are described.

1) Initially, all jobs, heterogeneous factories, and machines are available.
2) Every machine cannot process two operations at the same time, and it cannot be interrupted.

3) Data and results (i.e., workload and processing time) should have specific values.
4) Each operation cannot be processed on two machines simultaneously.
5) Different factories have the same flexible machines set for every operation. However, every operation's processing time varies on the same machine of different factories.
6) Setup and transportation time, and dynamic events are irrespective.

B. MILP Model

The MILP model is the mathematical model of an optimization problem, in which some decision variables must be an integer, and the others are real numbers. Meanwhile, the constraints and objective functions are linear. The notations and constraints are described in the supplementary file. The objective functions are introduced as follows:

DHFJS aims to minimize two objectives, including TEC and max completion time, and their decision expressions are stated as follows.

1) Makespan Decision Expression: Makespan is the maximum completion time of scheduling, representing an enterprise's production profit. Moreover, the makespan $C_{\max}$ objective is described

$$\min\ F_1 = C_{\max} = \max\left\{F_{f,k,t}\right\} \forall f \in F, k \in H_f, t \in P_{f,k}. \quad (1)$$

2) TEC Decision Expression: TEC during processing in all factories is regarded as a critical green indicator, representing a factory's carbon emissions. TEC is stated as follows:

$$\min\ F_2 = \text{TEC} = G_P + G_I \quad (2)$$

$$G_P = \sum_{i\in I}\sum_{j\in J_i}\sum_{f\in F}\sum_{k\in M}\sum_{t\in P_{f,k}} \mathbf{X}_{i,j,f,k,t} \cdot T_{i,j,f,k} \cdot W_O \quad (3)$$

$$G_I = \sum_{f\in F}\sum_{k\in K}\sum_{t\in P'_{f,k}} W_I \cdot \left(B_{f,k,t+1} - F_{f,k,t}\right). \quad (4)$$

III. Our Approch: DQCE

A. Motivations of DQCE

The motivation of the parasitic behavior-based co-evolutionary framework (PCE) for the solved problem is indicated as follows.

1) The number of combinations of DHFJS is $(\sum n_i)! \cdot m! \cdot n_f!$, which is too hard to solve. Under limited computation resources, the global search can generate a considerable perturbation, which can approach potential objective space. Thus, more computational resources should be allocated to the global search than the local search. Therefore, the global and local searches are put into two types of populations.
2) Since DHFJS is first studied, the previous works have not designed an efficient local search for this problem. Thus, nine neighborhood structures based on problem features are designed to accelerate algorithmic convergence.

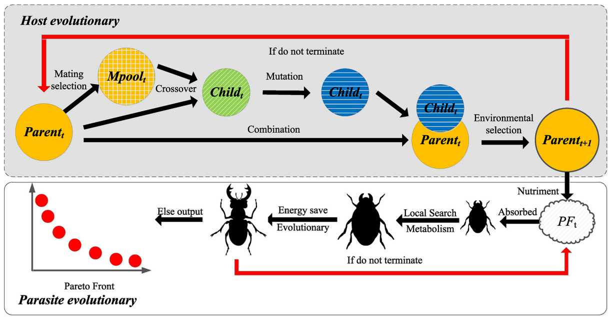


Fig. 1. PCE for distributed shop scheduling.

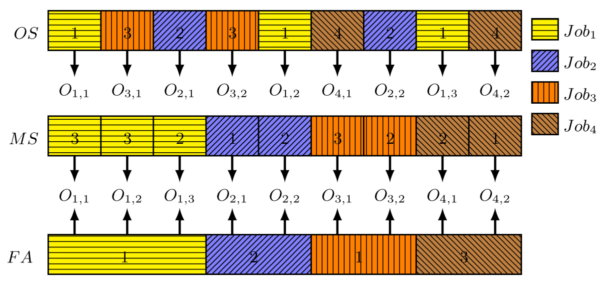
3) Due to its ability to rapidly learn the mapping between solutions and operators, DQN is applied to learn and select the best local search operator for each individual.
4) The previous works cannot provide an efficient energy-saving strategy. Based on observing the TEC objective, idle time is the key variable influencing TEC. Finding the idle space and inserting the latter operation into it can reduce TEC. Meanwhile, the former operation can be backward inserted into the latter idle space, thus further reducing the TEC.

B. Co-Evolutionary Framework Based on Parasitic Behavior

As shown in Fig. 1, the PCE divides the evolutionary process into two objects: 1) the host $\mathcal{H}$ and 2) the parasite $\mathcal{P}$. This design can let $\mathcal{H}$ sufficiently search nondominated solutions without early-maturing. First, the host $\mathcal{H}$ evolves one generation by NSGA-II [41]. Second, the parasite $\mathcal{P}$ absorbs the Pareto solutions from $\mathcal{H}$. Next, the parasite searches locally to find more potential nondominated solutions. Then, the parasite reduces TEC by adopting an energy-saving strategy. Finally, the parasite $\mathcal{P}$ outputs the final nondominated solutions set.

To the best of our knowledge, the concept of parasitic cooperative evolutionary has been proposed by [42], which is applied to solve the single optimization benchmark. However, PCE significantly differs from [42] for the following reasons.

1) Qin thinks the host should exchange many best fitness solutions with the same number of the worst solutions of the parasite. However, the parasite only absorbs the elite solutions from the host in this work as a one-directional action.
2) Qin considers the parasite as a possible behavior with an increasing rate like the mutation. If this behavior does not occur, these two swarms will evolve independently as a co-evolutionary algorithm with information interaction. Nevertheless, this work modifies this behavior as a specific action.
3) The size of the host is much bigger than the parasite, while the host and parasite have the same size in [42].
4) Qin solves the single optimization benchmark whereas PCE is designed for multi/many-objective distributed shop scheduling problem with a generality. Thus, PCE is a novel co-evolutionary framework distinguished from the previous work and is specific to distributed shop scheduling problem.



OS
MS
FA
$O_{1,1}$ $O_{3,1}$ $O_{2,1}$ $O_{3,2}$ $O_{1,2}$ $O_{4,1}$ $O_{2,2}$ $O_{1,3}$ $O_{4,2}$
$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{3,1}$ $O_{3,2}$ $O_{4,1}$ $O_{4,2}$
Job_1
Job_2
Job_3
Job_4

Fig. 2. Solution representation for DHFJS.

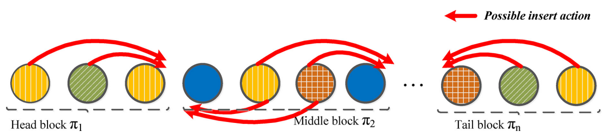


Fig. 3. N6 neighborhood structure. Different operations of a job have the same color. For example, the yellow operations belongA job I_1. $O_{1,1}$ and $O_{1,2}$ are on the same machine, and $O_{1,3}$ is on another machine.

C. Encoding and Decoding Schema

This work employs a three-level encoding schema to represent a solution for the problem. Fig. 2 shows the solution representation. Moreover, the encoding and decoding methods are stated as follows.

Encoding Schema: Three vectors are utilized to represent a solution to the problem, including the operation sequence (OS), FA, and MS. Moreover, the length of FA is n, and the length of OS and MS vectors equals the total number of operations. Significantly, the operations of every job cannot be dispatched to different factories.

Decoding Schema: According to the *FA* vector, all jobs are assigned to each heterogeneous factory. Next, the OS in each factory is extracted from the OS vector. Then, referring to the MS vector, every operation selects a selectable machine, and the processing of $O_{i,j}$ can be obtained. Finally, the start and completion times of all operations can be got, and the makespan and TEC of the total shop can be obtained.

D. Initialization Strategy

An initialization strategy is essential for combinatorial optimization problems [8]. Random initialization is adopted to attain a great diversity for training DQN. First, operation sequencing is randomly generated. Then, a machine is randomly selected for each operation from its candidate set $H_{i,j,f}$. Next, job index i is mod to n_f. Finally, permute this array to randomly assign a factory to each job.

E. Evolution and Environmental Selection

Sufficient permutation with a large step is crucial to efficiently generate offspring [21]. This work applies the precedence operation cross (POX) operator for OS and the uniform crossover (UX) operator is adopted for MS and FA. More details about POX and UX are stated in the supplementary file. It is worth noting that OS, MS, and FA adopt their crossover operators in each crossover step. Furthermore, each offspring obtained by the crossover step executes two mutation strategies with a rate P_m: 1) *MS Mutation:* randomly change the MS of two operations and 2) *OS Mutation:* randomly select two operations and swap their positions. After executing crossover and mutation operators, the environmental selection is referred to [41].

F. Knowledge-Driven Local Search Strategies

Inspired by [43], designing the problem-features-based local search operators can vastly reduce the randomness of local search and improve efficiency. The properties of DHFJS are related to its three subproblems: 1) OS; 2) MS; and 3) FA. Thus, changing MS and FA is critical for DHFJS to reduce makespan. According to these subproblems, nine local search strategies $\mathcal{N}_i$, $i \in [1, 9]$ are designed to find more nondominated solutions, as follows.

1) $\mathcal{N}_1$ (N6): N6 is a strong neighborhood structure designed for job shop scheduling problems by changing the OS in critical blocks. A block comprises several adjacent critical operations in the whole path on the same machine [44]. Moreover, N6 executes the following steps to reduce makespan. As shown in Fig. 3, for the head block π_1 in the critical path, an operation except for the tail operation is randomly selected and is inserted into the tail of π_1. For the middle block π_2, two middle operations are randomly selected, one operation is inserted into the head of π_2, and another is inserted into the tail of π_2. Finally, for the tail block π_n, an operation except for the head operation is randomly inserted into the front of π_n.

2) $\mathcal{N}_2$ (Swap in All Factories): In order to increase the diversity, the two-point exchange neighborhood is adopted by randomly swapping two operations in *OS*.

3) $\mathcal{N}_3$ (Swap in Critical Factory): This strategy selects two critical operations in the critical factory. Then, their positions are swapped to reduce makespan.

4) $\mathcal{N}_4$ (Insert in All Factories): Two operations are randomly selected among all operations, and the latter is inserted into the front of the former to increase diversity.

5) $\mathcal{N}_5$ (Insert in the Critical Factory): Focusing on the operation sequencing in the critical factory can increase the probability of finding nondominated solutions. Moreover, two operations are chosen randomly in the critical factory. Then, the latter operation is inserted into the front position of the former.

6) $\mathcal{N}_6$ (Randomly Factory Assignment): According to the subproblem FA of DHFJS, changing the FA can balance the workload of all factories to increase the probability of reducing the makespan. Thus, randomly select a job in the critical factory and assign it to another factory.

7) $\mathcal{N}_7$ (Ranking Factory Assignment): Previous works randomly insert a job to another factory and attempt to load balance to reduce makespan [9], [45]. However, the random rule has a low success rate. In order to increase the success rate of reducing makespan, a heuristic rule is specifically designed to solve the FA problem. Each job selects the factory with a shorter average processing time where $\bar{P}_{i,f} = \sum_{j=1}^{n_i}(\sum_{k=1}^{H_{i,j,f}} T^{O}_{i,j,f,k}/k)/n_i$. The $\bar{P}_{i,f}$ calculated in all factories and the $\bar{P}_{i,f}$ transferred to selection probability by dividing the $\sum_{f=1}^{n_f} \bar{P}_{i,f}$. Then, reselect another factory for the

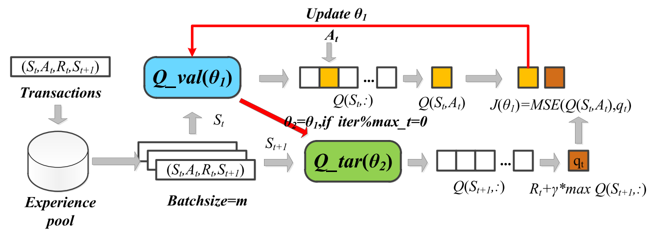


Fig. 4. Training process of DQN.

chosen job by roulette wheel selection. In the above way, the selected job is assigned to the factory with a shorter processing time, thus increasing the success rate of reducing makespan. Moreover, roulette wheel selection is adopted as a selection model to let each factory be chosen to balance the convergence and diversity.

8) $\mathcal{N}_8$ (Randomly Machine Selection): As another subproblem of DHFJS, MS also deserves attention. A critical operation is randomly selected in the critical factory. Then, another selectable machine is chosen to balance the machine load in the critical factory.

9) $\mathcal{N}_9$ (Ranking Machine Selection): Similarly, the machine with a shorter processing time has a bigger selection probability. A critical operation is randomly selected, and another machine is selected by roulette wheel selection.

G. Deep-Q-Network-Based Strategies Selection Model

Although the knowledge-driven local is powerful, how to organize them to customize the most appropriate operator for each solution is a critical question. In order to solve this problem, the DQN is applied to adaptively select a local search strategy for each solution. DQN is a powerful deep reinforcement learning algorithm that can learn the data distribution and make the right choice driven by historical experience. In DQN, a transaction comprises (S_t, A_t, R_t, S_{t+1}), including the state, chosen action, reward, and the next state of the current agent at time t. Assume that the state's set is large, limited, and cannot be exhausted. Thus, a distribution of the states of an agent must exist so that a multilayer perceptron can learn. The DQN does not require defining the states transferring path and learns the distribution of all states in the environment by a network $Q_t(\theta)$.

Fig. 4 shows the training process of DQN. First, two networks, the valuation network $Q_{\text{val}}(\theta_1)$, and target network $Q_{\text{tar}}(\theta_2)$ share the same parameter θ_0 at the beginning of training. Second, the agent randomly selects a batch of transactions in the experience pool. Next, the current state S_t is fed to $Q_{\text{val}}(\theta_1)$, and the q-values of all actions of current state $Q(S_t, :)$ are output. Similarly, the next state is input into $Q_{\text{tar}}(\theta_2)$ the q-values of the next states $Q(S_{t+1}, :)$ are obtained. Then, the target q-value is calculated by the following equation:

$$q_t = R_t + \gamma * \max Q(S_{t+1}, :). \tag{5}$$

Finally, the parameters θ_1 are updated by the following loss functions:

$$J(\theta_1) = \frac{1}{m}\sum_{1}^{m}(Q(S_t, A_t) - q_t)^2 \tag{6}$$

where m is the batch size. Finally, if the learning iteration is bigger than a threshold, the θ_2 will be replaced by θ_1.

Network Structure: In this work, the networks $Q_{\text{val}}(\theta_1)$ and $Q_{\text{tar}}(\theta_2)$ have the same structure. The network has several full connection layers, defined as follows: $fc_1(In, 128)$, $fc_2(128, 256)$, $fc_3(256, 128)$, $fc_4(128, 64)$, $fc_5(64, 32)$, are $fc_6(32, Out)$ where In equals the length of a transaction and the size of Out equals the total number of actions. The hidden layers between each fc layer are "ReLU."

State Definition: To better learn the solution distribution in DHFJS, the state is a vector merged by (OS, MS, FA). Moreover, the length of OS and MS equals the number of total operations $\sum_{i=1}^{n} n_i$, and the length of FA equals the number of jobs n.

Action and Reward Definitions: This work applies the DQN to tailor a neighborhood structure for each solution in the parasite population which can significantly improve the success rate of finding a new nondominated solution. Thus, the actions in DQN are the nine neighborhood structures mentioned in Section III-F. The reward feedback is set as follows: if the new solution replaces the old one, then the reward is five; Else if the new and old solutions cannot dominate each other, then the reward is ten; Else the reward is zero. The reward value is usually set to ten when the selected action succeeds and set to zero when it fails according to [46]. Ten is an appropriate reward. If the reward is set too large, it will increase the difficulty of DQN convergence, and the loss function will fluctuate. If the reward is set too small, it is difficult to distinguish the feedback of successful and failed actions. In order to reduce learning difficulty, no punishment is added to DQN and the reward is set to zero when the action fails. In order to guide the DQN to find more nondominated solutions, the reward is set to ten when the action finds a nondominated solution. Meanwhile, the reward is smaller when action finds a new solution, replacing the old.

Operator Selection: Before local searching, the Q networks have been initialized and constructed. First, get the current state S_t of solutions in the parasite $\mathcal{P}$. Next, input S_t into Q_θ and get the q-values of all actions. Then, the $\epsilon-$greedy strategy is executed to select an action A_t. Moreover, A_t is adopted to S_t to generate a new solution S_{t+1}. Finally, this transaction is stored in the experience pool, and Q_θ is trained when the transactions in the experience pool exceed the threshold. The detailed process of DQN-based operator selection framework is illustrated in Algorithm 1.

H. Energy-Saving Strategy

Reducing energy consumption can lower carbon and improve the profit for the enterprise. Thus, energy-saving is an essential step for green shop scheduling. Reducing idle energy consumption can lower the TEC. Thus, an energy-saving strategy based on full-active scheduling is adopted [43]. Forward scheduling is defined by traversing the operation sequencing from left to right whereas backward scheduling traverses operation sequencing from right to left. The main process of the energy-saving strategy is shown in Fig. 5. First, the operation sequencing is forward scanned to find the possible idle place,

Algorithm 1: DQN-Based Operator Selection Framework

```
1  Input: The parasite population ($\mathcal{P}$), greedy factor ($\epsilon$), experience pool size $S_E$,
   epochs, and Q network $Q_\theta$.
2  Output: Q network $Q(\theta)$ and the parasite population ($\mathcal{P}$)
3  L=size($\mathcal{P}$).
4  for l= 1 to L do
5      $S_t \leftarrow [OS, MS, FA] \leftarrow \mathcal{P}(l)$.
6      $\mathbf{q}_t \leftarrow Q_\theta(S_t)$.
7      if rand > $\epsilon$ then
8          Randomly select a $\mathcal{N}_i$ as $A_t, i \in [1, 9]$
9      else
10         $A_t$ is the $\mathcal{N}_i$ with the max $\mathbf{q}_t(i), i \in [1, 9]$.
11     $S_{t+1} \leftarrow [OS', MA', FA'] \leftarrow A_t(S_t)$.
12     if $S_{t+1}$ dominates $S_t$ then
13         $\mathcal{P}(l) \leftarrow S_{t+1}$.
14         $R_t = 5$.
15     else
16         if $S_{t+1}$ and $S_t$ do not dominate each other then
17             $\mathcal{P}(l) \leftarrow \mathcal{P}(l) \cup S_{t+1}$.
18             $R_t = 10$.
19         else
20             $R_t = 0$.
21     store transaction $\tau(S_t, A_t, R_t, S_{t+1})$ in experience pool.
22     if size of experience pool is bigger than $S_E$ then
23         for e= 1 to epochs do
24             Train $Q_\theta$ by the process in Fig. 4.
```

Algorithm 2: Produce of DQCE

```
1  Input: Crossover rate $P_c$, mutation probability $P_m$, learning rate $\alpha$, discount
   factory $\gamma$, greedy factor ($\epsilon$), batch size $bs$, experience pool size $S_E$, epochs,
   population size $ps$, and update threshold $\beta$.
2  Output: The Pareto solutions $PS$
3  $\mathcal{H} \leftarrow$ Initial host population size $ps$.
4  $\mathcal{P} \leftarrow$ Initial parasite population size zero.
5  $Q_\theta \leftarrow$ Initial Q network($\alpha, \gamma, bs, \beta, S_E, \epsilon$).
6  NEFs=0, MaxNFEs=200*$\sum_1^n n_i$
7  while NEFs<MaxNFEs do
8      $Pool \leftarrow$ Tournament Selection($\mathcal{H}$,2).
9      $C \leftarrow$ Crossover and mutation($Pool, P_c, P_m$). // $C$ size 2*$ps$
10     NEFs=NEFs+2*ps.
11     $U \leftarrow C \cup \mathcal{H}$. // $U$ size 3*$ps$
12     $\mathcal{H} \leftarrow$ Environment selection($U$). // adopt NSGA-II
13     $PF \leftarrow$ Get Pareto Front($\mathcal{H}$).
14     $\mathcal{P} \leftarrow \mathcal{P} \cup PF$. // absorb nutrition
15     $\mathcal{P} \leftarrow$ Delete Repeat solutions ($\mathcal{P}$).
16     $[\mathcal{P}, Q_\theta] \leftarrow$ DQN-based strategies selection($\mathcal{P}, Q_\theta, \epsilon, S_E$).
17     NEFs=NEFs+$|\mathcal{P}|$.
18     $\mathcal{P} \leftarrow$ Energy-saving ($\mathcal{P}$).
19     NEFs=NEFs+$|\mathcal{P}|$.
20 $PS \leftarrow$ Get Pareto Front($\mathcal{P}$).
```

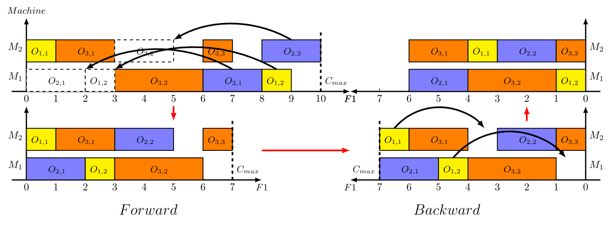


Fig. 5. Energy-saving strategy.

and current operation $O_{i,j}$ is inserted into it to decrease the idle time. Next, the improved solution is backward traversed to find the possible place to insert the current operation to reduce idle time. Finally, the TEC is reduced, and the makespan is decreased. It is worth noting that the energy-saving strategy is also adopted in the parasite population to increase the diversity of the host population.

I. Computational Complexity of DQCE

By analyzing Algorithm 2, the time complexity of our approach DQCE mainly depends on five parts, including generating offspring, updating population, strategies selection, DQN training, and energy-saving. The time complexity of the strategies selection for the inference procedure of the Q network is $O(N^2)$, where N is the number of decision variables, representing the scales of layers. The time complexity of the crossover step is $O(N * ps)$. The time complexity of population update is $O(ps^2)$ for environment selection. The time complexity of DQN training is $O(N^2)$ for a backpropagation procedure. The time complexity of energy-saving is $O(2 \times MN^2)$ for forward and backward insertion, where M is the size of the parasite population. In summary, the total time complexity of DQCE in one iteration is $O(2(M + 1)N^2)$.

IV. Experimental Results

The procedure of DQCE[1] has been introduced in detail in Section III. Parameter calibration, ablation, and comparison experiments are designed to test the performance of DQCE. All algorithms without DQN are coded in MATLAB2020, and DQCE is coded in Python 3.6 with CUDA 11.3 and Pytorch. The running environment is on an Intel Xeon Gold 6246R CPU @ 3.4 GHz with 384 G RAM, and NVIDIA GeForce RTX 3090 GPU.

A. Instances and Metrics

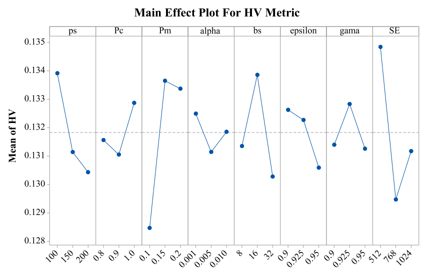
Since DHFJS is not studied before, various scales of instances are generated to evaluate the performance of DQCE. The number of jobs is $n \in \{10, 20, 30, 40, 50\,100\,150, 200\}$, and the number of factories ranges from $n_f \in \{2, 3, 4, 5, 6, 7\}$. The number of machines in each heterogeneous factory is $m = 5$, and the number of operations of each job is $n_i = 5$. Every operation's processing time in each factory is $P_{f,i,j,k} \in [5, 20]$. The processing and idle powers are chosen as $W^O_{i,j,f,k} = 4.0$ kWh and $W_I = 1.0$ kWh, respectively. Finally, 20 instances with various scales are generated. The stop criterion is MaxNEFs $= 200 \times \sum_1^n n_i$.

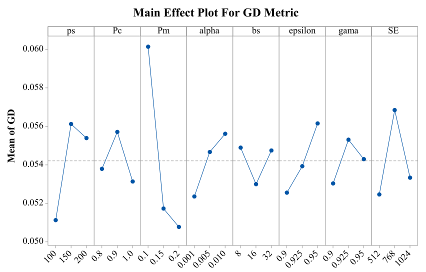
Hypervolume (HV) [47], generation distance (GD), and Spread [41] are utilized to evaluate comprehensive performance, convergence, and diversity of multiobjective evolutionary algorithms (MOEAs). Furthermore, when the GD or Spread values are lower, the convergence and diversity of an algorithm are better. Whereas, If the HV value is bigger, the comprehensive performance of an algorithm is better.

B. Parameters Calibration

It is essential to confirm the best parameter setting of DQCE because several parameters influence its performance. DQCE includes eight parameters which are crossover rate P_c, population size ps, mutation rate P_m, learning rate α, batch

[1]The code is shared on https://wewnyin.github.io/wenyingong/Software/DQCE-code.zip.


(a)


(b)

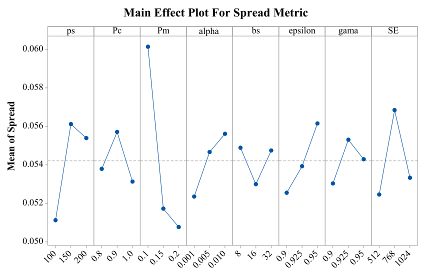

(c)

Fig. 6. Three metrics' main effect plots. (a) HV. (b) GD. (c) Spread.

size bs, greedy factor ϵ, discount factor γ, and experience pool size S_E. A Taguchi approach of design of experiment (DOE) [48] is utilized to attain the best configuration. The variant parameters' settings are given as follows: $P_c = \{0.8, 0.9, 1.0\}$; $ps = \{100, 150, 200\}$; $P_m = \{0.1, 0.15, 0.2\}$; $\alpha = \{0.001, 0.005, 0.01\}$; $\epsilon = \{0.9, 0.925, 0.95\}$; $\gamma = \{0.9, 0.925, 0.95\}$; and $S_E = \{512, 768, 1024\}$. An orthogonal table $L_{27}(8^3)$ is designed for the parameter experiment. For a fair comparison, each algorithm with a different parameter setting runs 20 times with the same stop criterion (MaxNFEs $= 200 \times \sum_1^n n_i$). The average values of all metrics of each algorithm running on each instance are collected. Fig. 6 shows three main effects plots of eight parameters for all metrics. Moreover, if the HV values are higher, the performance of a parameter is better. However, the convergence and diversity are better when GD and Spread are lower. According to the plots of all metrics, the optimal parameter combination of DQCE is $P_c = 1.0$, $ps = 100$, $P_m = 0.2$, $\alpha = 0.001$, $bs = 16$, $\epsilon = 0.9$, $\gamma = 0.9$, and $S_E = 512$.

C. Ablation Experiment

In order to evaluate the effectiveness of each improvement part in DQCE, four variant algorithms are designed as follows: 1) the memetic algorithm (MA) with the proposed nine neighborhood structures and energy-saving strategy is set to evaluate the effectiveness of PCE; 2) to verify the effectiveness of the knowledge-driven neighborhood structures, PCE without neighborhood structures called (PCE-NS) is designed; 3) PCE is generated without energy-saving method called (PCE-ES) to demonstrate its effectiveness; and 4) PCE assisted by DQN named (DQCE) is set to evaluate the improvement of DQN. In order to compare fairly, every algorithm independently executes 20 times on whole instances with the same stop criterion (MaxNFEs $= 200 \times \sum_1^n n_i$). The DQCE is coded by Python, and others are coded by MATLAB. PCE, PCE-NS, PCE-ES, and MA are improved by random neighborhood structure selection.

Table S-I, in the supplementary material records all metrics' statistical results of every variant algorithm, where the symbols "$-$" and "$+$" mean significantly worse and better than DQCE. The symbol "$=$" means no significant difference between variant algorithms and DQCE. Furthermore, the best values of each metric are marked in *bold*. Table I shows the Friedman rank-and-sum test results, where the confidence level is $\alpha = 0.05$. Based on the experiment results, the following conclusions can be drawn: 1) the p-value < 0.05 indicates the superiority of DQCE to all variants; 2) the effectiveness of the designed PCE can be evaluated by comparing PCE and MA; 3) comparing PCE with PCE-NS and PCE-ES can evaluate the effectiveness of knowledge-driven local search and energy-saving operators; and 4) comparing PCE and DQCE can ensure the effectiveness of the DQN-based strategies selection model.

TABLE I
RANK RESULTS OF THE FRIEDMAN RANK-AND-SUM TEST OF ALL VARIANT ALGORITHMS (SIGNIFICANT LEVEL $\alpha = 0.05$)

MOEAs	HV		GD		Spread	
	rank	p-value	rank	p-value	rank	p-value
MA	5.00		5.00		3.75	
PCE-NS	3.45		3.75		3.75	
PCE-ES	2.95	1.49E-29	2.75	3.34E-29	2.20	6.80E-04
PCE	2.60		2.55		3.10	
DQCE	**1.00**		**1.00**		**2.20**	

D. Comparison and Discussions

In order to evaluate the performance of DQCE in solving the problem, DQCE is compared with two mainstream MOEAs, MOEA/D [49], and NSGA-II [41]. Moreover, recently proposed MOEAs, namely, TS-NSGA-II [50] are also included. Furthermore, a state-of-the-art algorithm for DFJSP named HSLFA [45] is compared. Some reinforcement learning-based algorithms are chosen for comparison, such as LRVMA [8] and MOEA/D-DQN [46]. The parameters of comparison algorithms are set based on their references for all algorithms, which are stated as follows: crossover probability $P_c = 1.0$, population size $ps = 100$, and mutation probability $P_m = 0.2$. The amount of neighborhood is $T = 10$ for MOEA/D-DQN and MOEA/D. The reinforcement learning parameters of LRVMA and MOEA/D-DQN are the same as DQCE. All MOEAs are added with knowledge-driven local search and energy-saving strategies for a fair comparison. Meanwhile, all comparison algorithms share the same stop criterion (MaxNFEs $= 200 * \sum_1^n n_i$). Due to the complexity of DHFJS, all algorithms independently run 20 times in all instances.

Table S-II, in the supplementary material lists each algorithm's statistical results, including average values and standard deviation values, for HV, GD, and Spread metrics.

TABLE II
RANK RESULTS OF THE FRIEDMAN RANK-AND-SUM TEST OF ALL COMPARISON ALGORITHMS (SIGNIFICANT LEVEL $\alpha = 0.05$)

MOEAs	HV		GD		Spread	
	rank	*p*-value	rank	*p*-value	rank	*p*-value
NSGA-II	5.75	2.10E-19	5.25	9.14E-19	3.50	8.26E-08
MOEA/D	2.80		3.00		5.40	
TS-NSGA-II	4.30		3.80		3.25	
HSLFA	4.55		5.00		4.65	
LRVMA	2.65		2.95		5.35	
MOEA/D-DQN	6.95		7.00		3.95	
DQCE	**1.00**		**1.00**		**1.90**	

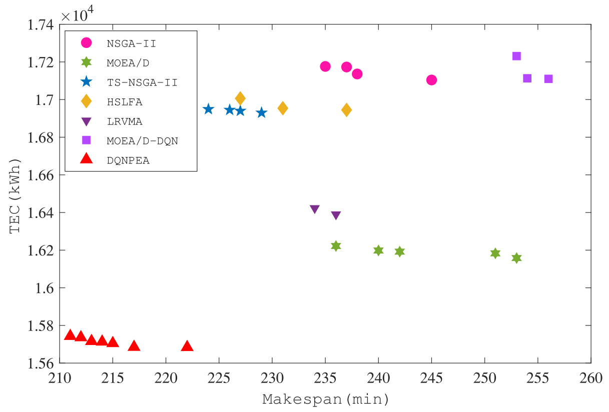


Fig. 7. Approximate Pareto Front found by each algorithm on 100J4F.

At the last row of each subtable, the symbol "−/+" indicates the compared algorithm is significantly worse or better than DQCE, and "+" means no significant difference. Meanwhile, the optimal value of each metric is marked in *bold*. Based on the results of Table S-II, in the Supplementary material, DQCE is significantly superior to all comparison algorithms for HV and GD metrics. Nevertheless, as for the Spread metric, there is no significant distinction between all algorithms because all MOEAs apply our knowledge-driven local search strategies. Table II presents the Friedman rank-and-sum test results of all algorithms on whole instances, while the confidence level is $\alpha = 0.05$. DQCE ranks first for all metrics and $p-$value < 0.05, indicating the superiority of DQCE to comparison algorithms.

The success and great performance of DQCE rely on its design. First, the designed PCE can improve exploration and exploitation. Second, nine local search strategies are proposed based on problem features. Then, a DQN-based operator selection model is designed to reduce the randomness of local search. Finally, a problem-specific heuristic strategy is implemented to reduce the makespan and TEC efficiently.

Several figures are performed to verify the performance of all comparison algorithms. Fig. 7 shows the Pareto front of all comparison algorithms with the best HV metric among 20 independent runs. By comparing the diversity and convergence of each PF, DQCE can find Pareto solutions with better objectives than all comparison algorithms, indicating that DQCE can get closer approximations toward practical PF. Furthermore, as shown in Fig. 7, makespan and TEC have a conflicting relationship. Fig. 8 shows the strategies selection ratio by DQN during solving 100J4F. The most efficient strategy is different for different generations. Besides, Fig. 9 shows

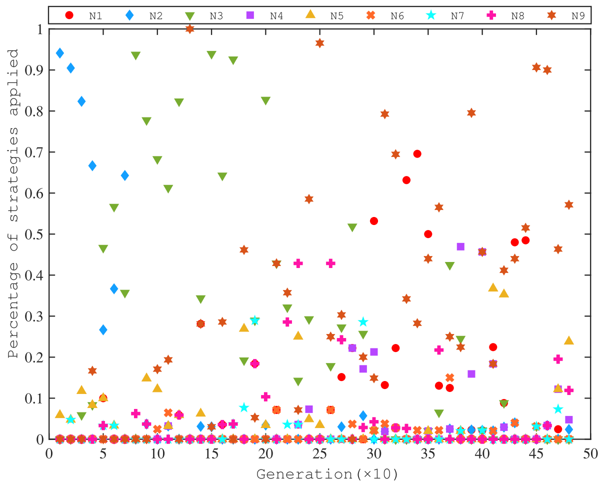


Fig. 8. Ratio of strategies selected by DQCE during the optimization process of solving 100J4F.

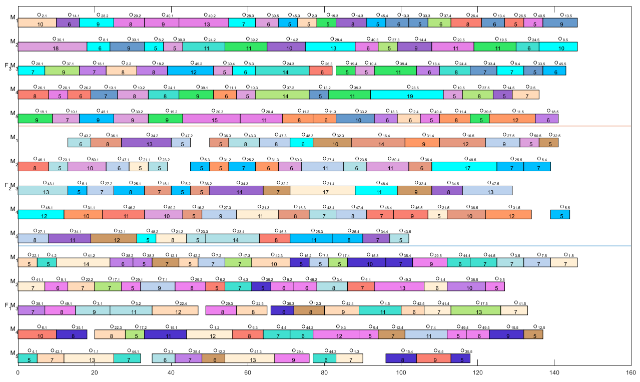


Fig. 9. Gantt chart of solution A with the best $C_{\max}$ on 50J3F.

a Gantt chart of the solution A with the minimum makespan on 50JF3 instance ($C_{\max} = 146$ h, TEC $= 7954$ kWh), and Fig. 10 displays a Gantt chart of the solution B with the best TEC on 50JF3 instance ($C_{\max} = 154$ h, TEC $= 7885$ kWh). The experimental results indicate that our approach DQCE can optimize DHFJS better.

E. Comparisons on the Real-World Case

In order to better evaluate the effectiveness of DQCE, the comparison algorithms are evaluated on a real-world case.[2] This case describes the model blanking shop provided by a Chinese factory of a large equipment enterprise. The blanking system transforms the raw material (steel) into different sizes of parts for subsequent machining shop processing. A batch of orders are regarded as a basic processing job in the blanking system. There are three steps to process a batch: 1) cutting; 2) clamping; and 3) beveling. However, due to commercial privacy, the processing data of each stage is not given, and each job only contains one integer operation named blanking. There are 55 jobs in the real-world case, and each job contains two features (weight and difficulty factor). Besides, there are two factories in this case, and each factory has four small groups

[2]The real-world case can be obtained from https://cuglirui.github.io/Dataset/DHFJSP.rar.

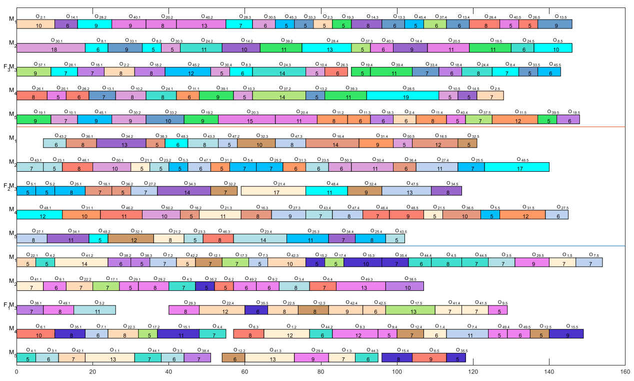


Fig. 10. Gantt chart of solution A with the best TEC on 50J3F.

TABLE III
STATISTICAL RESULTS AMONG THREE METRICS OF ALL COMPARISON ALGORITHMS IN REAL-WORLD CASE

MOEAs	HV		GD		Spread	
	mean	std	mean	std	mean	std
NSGA-II	0.167102	0.000574	0.001878	0.000917	0.943469	0.110437
MOEA/D	0.165206	0.000996	0.002287	0.00066	1.019607	**0.101734**
TS-NSGA-II	0.16741	0.000433	0.001892	0.000725	0.973863	0.109112
HSLFA	0.164324	0.001245	0.003141	0.001154	1.112472	0.309525
LRVMA	0.164192	0.001149	0.003522	0.00168	1.150408	0.327005
MOEA/D-DQN	0.166126	0.000549	0.001716	0.000726	1.090645	0.111967
DQCE	**0.167491**	**0.000287**	**0.001497**	**0.000401**	**0.929345**	0.10818
p-value	8.60E-19		1.62E-07		5.78E-05	

of workers. Each group operates a machine, and different groups have different capacities (tons of steel processed per hour weight). Each job's processing time can be calculated by $weight \times factor \times 1000/capacity$. Thus, the processing time of every operation is different for each group. Besides, each group of workers can process all jobs. The objective is to minimize makespan and TEC in this real-world case. Thus, this case is constructed as the DHFJS. The detailed information is shown in Tables S-III and S-IV, in the supplementary material.

The algorithms presented in Section IV-D are compared with DQCE, while the stop criterion and parameter setting are the same in the last section. Each algorithm independently runs 20 times. The statistical results of all algorithmic metrics are shown in Table III. The best value is marked in *bold*. As shown in Table III, our approach DQCE has the best mean of all metrics, and the std is the best in HV and GD, indicating that DQCE has the best performance, including convergence, diversity, and comprehensive performance in a real-world case. Meanwhile, the performance is more stable than others. The p-value < 0.05 states the significantly superior to the comparison algorithms because DQCE designs the local search operators based on the problem extraction knowledge and employs DQN to enhance the decision ability of the algorithm.

V. CONCLUSION

This article proposed a deep reinforcement learning-based co-evolutionary algorithm for the energy-aware DHFJS problem. First, a novel PCE was proposed to solve the problem. Second, nine knowledge-driven local search strategies were proposed to optimize three subproblems of the problem. Then, a deep reinforcement learning algorithm DQN was adopted to learn the distribution of solutions and select the best local search strategy for them to enhance convergence and diversity. Besides, a full-active scheduling-based energy-saving strategy was adopted to reduce TEC efficiently. Finally, the numerical experimental results performed on 20 instances and a real-world case demonstrate that DQCE performs the best on DHFJS.

The further research directions are: 1) enhancing the explanation of the reasoning process of DQN; 2) considering end-to-end network for DHFJS; and 3) considering the differential distributed heterogeneous FJSP with different machine numbers, machine failure, or maintenance.

REFERENCES

[1] J.-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.

[2] K. Peng, Q.-K. Pan, L. Gao, B. Zhang, and X. Pang, "An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process," *Comput. Ind. Eng.*, vol. 122, pp. 235–250, Aug. 2018.

[3] K. Peng, Q.-K. Pan, L. Gao, X. Li, S. Das, and B. Zhang, "A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling," *Swarm Evol. Comput.*, vol. 45, pp. 92–112, Mar. 2019.

[4] C. Lu, Y. Huang, L. Meng, L. Gao, B. Zhang, and J. Zhou, "A pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers," *Robot. Comput. Integr. Manuf.*, vol. 74, Apr. 2022, Art. no. 102277.

[5] G. Zhang, L. Wang, and K. Xing, "Dual-space co-evolutionary memetic algorithm for scheduling hybrid differentiation flowshop with limited buffer constraints," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 11, pp. 6822–6836, Nov. 2022.

[6] Z. Pan, D. Lei, and L. Wang, "A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5295–5307, Aug. 2022.

[7] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Comput. Ind. Eng.*, vol. 142, Apr. 2020, Art. no. 106347.

[8] R. Li, W. Gong, C. Lu, and L. Wang, "A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time," *IEEE Trans. Evol. Comput.*, vol. 27, no. 3, pp. 610–620, Jun. 2023.

[9] R. Li, W. Gong, L. Wang, C. Lu, and S. Jiang, "Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time," *Swarm Evol. Comput.*, vol. 74, Oct. 2022, Art. no. 101139.

[10] X. Wu and X. Liu, "An improved differential evolution algorithm for solving a distributed flexible job shop scheduling problem," in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng. (CASE)*, 2018, pp. 968–973.

[11] W. Xu, Y. Hu, W. Luo, L. Wang, and R. Wu, "A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107318.

[12] B. Marzouki, O. B. Driss, and K. Ghédira, "Solving distributed and flexible job shop scheduling problem using a chemical reaction optimization metaheuristic," *Procedia Comput. Sci.*, vol. 126, pp. 1424–1433, Jan. 2018.

[13] Y. Du, J.-Q. Li, C. Luo, and L.-L. Meng, "A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100861.

[14] 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.*, vol. 17, no. 10, pp. 6687–6696, Oct. 2021.

[15] G. Wang, X. Li, L. Gao, and P. Li, "Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified MOEA/D," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100858.

[16] W. Shao, Z. Shao, and D. Pi, "An ant colony optimization behavior-based MOEA/D for distributed heterogeneous hybrid flow shop scheduling problem under nonidentical time-of-use electricity tariffs," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3379–3394, Oct. 2022.

[17] J. Chen, L. Wang, X. He, and D. Huang, "A probability model-based memetic algorithm for distributed heterogeneous flow-shop scheduling," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2019, pp. 411–418.
[18] 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.*, vol. 51, no. 11, pp. 5291–5303, Nov. 2021.
[19] F. Zhao, S. Di, and L. Wang, "A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem," *IEEE Trans. Cybern.*, vol. 53, no. 5, pp. 3337–3350, May 2023.
[20] F. Zhao, Z. Xu, L. Wang, N. Zhu, T. Xu, and J. Jonrinaldi, "A population-based iterated greedy algorithm for distributed assembly no-wait flow-shop scheduling problem," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 6692–6705, May 2023.
[21] R. Li, W. Gong, and C. Lu, "A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling," *Expert Syst. Appl.*, vol. 203, Oct. 2022, Art. no. 117380.
[22] K. Peng, X. Deng, C. Zhang, Q.-K. Pan, L. Ren, and X. Pang, "An improved imperialist competitive algorithm for hybrid flowshop rescheduling in steelmaking-refining-continuous casting process," *Meas. Control*, vol. 53, nos. 9–10, pp. 1920–1928, Oct. 2020.
[23] 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, Jun. 2022, Art. no. 108099.
[24] X. Ma et al., "A survey on cooperative co-evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 421–441, Jun. 2019.
[25] L. Miguel Antonio and C. A. Coello Coello, "Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 851–865, Dec. 2018.
[26] X. Wang, K. Zhang, J. Wang, and Y. Jin, "An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 859–871, Oct. 2022.
[27] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3696–3708, Aug. 2020.
[28] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation—Past, present amp; future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.
[29] J. Wang, W. Ren, Z. Zhang, H. Huang, and Y. Zhou, "A hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 11, pp. 4732–4745, Nov. 2020.
[30] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236
[31] J. Chang, G. Zhu, H. Zhang, Y. Zhou, X. Luo, and L. Wu, "Reversible data hiding for color images based on adaptive 3D prediction-error expansion and double deep q-network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 8, pp. 5055–5067, Aug. 2022.
[32] K. Tan, D. Bremner, J. L. Kernec, Y. Sambo, L. Zhang, and M. A. Imran, "Intelligent handover algorithm for vehicle-to-network communications with double-deep q-learning," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7848–7862, Jul. 2022.
[33] F. Khoramnejad and M. Erol-Kantarci, "On joint offloading and resource allocation: A double deep Q-network approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1126–1141, Dec. 2021.
[34] V. Bui, A. Hussain, and H. Kim, "Double deep *Q*-learning-based distributed operation of battery energy storage system considering uncertainties," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 457–469, Jan. 2020.
[35] T.-H. S. Li et al., "Fuzzy double deep Q-network-based gait pattern controller for humanoid robots," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 1, pp. 147–161, Jan. 2022.
[36] C.-C. Lin, D.-J. Deng, Y.-L. Chih, and H.-T. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep Q network," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4276–4284, Jul. 2019.
[37] B. Yin, S. Zhang, and Y. Cheng, "Application-oriented scheduling for optimizing the age of correlated information: A deep-reinforcement-learning-based approach," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8748–8759, Sep. 2020.
[38] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, and G. Wu, "A generic markov decision process model and reinforcement learning method for scheduling agile earth observation satellites," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 3, pp. 1463–1474, Mar. 2022.
[39] S. Luo, L. Zhang, and Y. Fan, "Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning," *Comput. Ind. Eng.*, vol. 159, Sep. 2021, Art. no. 107489.
[40] Y. Du, J.-Q. Li, X.-L. Chen, P.-Y. Duan, and Q.-K. Pan, "Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1036–1050, Aug. 2023.
[41] 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.
[42] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Biomimicry of parasitic behavior in a coevolutionary particle swarm optimization algorithm for global optimization," *Appl. Soft Comput.*, vol. 32, pp. 224–240, Jul. 2015.
[43] R. Li, W. Gong, L. Wang, C. Lu, and X. Zhuang, "Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling," *IEEE Trans. Cybern.*, early access, Jun. 8, 2023, doi: 10.1109/TCYB.2023.3280175.
[44] E. Balas and A. Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Manag. Sci.*, vol. 44, no. 2, pp. 262–275, Feb. 1998.
[45] L. Meng, Y. Ren, B. Zhang, J.-Q. Li, H. Sang, and C. Zhang, "Milp modeling and optimization of energy- efficient distributed flexible job shop scheduling problem," *IEEE Access*, vol. 8, pp. 191191–191203, 2020.
[46] Y. Tian, X. Li, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1051–1064, Aug. 2023.
[47] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 29–38, Feb. 2006.
[48] R. C. Van Nostrand, "Design of experiments using the taguchi approach: 16 steps to product and process improvement," *Technometrics*, vol. 44, no. 3, p. 289, Aug. 2002.
[49] Q. Zhang and L. Hui, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pP. 712–731, Dec. 2007.
[50] F. Ming, W. Gong, and L. Wang, "A two-stage evolutionary algorithm with balanced convergence and diversity for many-objective optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 10, pp. 6222–6234, Oct. 2022.

Rui Li received the B.Eng. degree in information science and the M.Eng. degree in computer science and technology from the China University of Geosciences, Wuhan, China, in 2020 and 2023, respectively. He is currently pursuing the Ph.D. degree in control theory and control engineering with the Department of Automation, Tsinghua University, Beijing, China.

His current research interests include distributed and green scheduling with intelligent optimization and reinforcement learning.

Wenyin Gong (Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees in computer science from the China University of Geosciences, Wuhan, China, in 2004, 2007, and 2010, respectively.

He is currently a Professor with the School of Computer Science, China University of Geosciences. He has published over 80 research papers in journals and international conferences. His research interests include evolutionary algorithms, evolutionary optimization, and their applications.

Prof. Gong served as a Referee for over 30 international journals, such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, *IEEE Computational Intelligence Magazine*, *ACM Transactions on Intelligent Systems and Technology*, *Information Sciences*, *European Journal of Operational Research*, *Applied Soft Computing*, and *Journal of Power Sources*. He currently serves as an Associate Editor for *Expert Systems with Applications*, *International Journal of Bio-Inspired Computation*, and *Memetic Computing*.

Ling Wang (Member, IEEE) 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 an Editor-in-Chief of *International Journal of Automation and Control* and an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Swarm and Evolutionary Computation*, and *Expert Systems with Applications*.

Chao Lu received the Ph.D. degree in industrial engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2017.

He is currently an Associate Professor of Computer Sciences with the School of Computer Science, China University of Geosciences, Wuhan. His research interests include multiobjective evolutionary algorithm and intelligent optimization scheduling.

Chenxin Dong received the bachelor's degree in vehicle engineering from China Agricultural University, Beijing, China, in 2010, and the master's degree in engineering management from the China University of Petroleum, Dongying, China, in 2020, also studied in Kettering University, Flint, MI, USA, for one year in 2017.

He is currently a Lecturer with the Qingdao Hengxing University of Science and Technology, Qingdao, China. His current research interests include optimization and automotive engineering.