Knowledge-Based Reinforcement Learning and Estimation of Distribution Algorithm for Flexible Job Shop Scheduling Problem

Yu Du, Jun-qing Li, *Member, IEEE*, Xiao-long Chen, Pei-yong Duan, and Quan-ke Pan

***Abstract*—In this study, a flexible job shop scheduling problem with time-of-use electricity price constraint is considered. The problem includes machine processing speed, setup time, idle time, and the transportation time between machines. Both maximum completion time and total electricity price are optimized simultaneously. A hybrid multi-objective optimization algorithm of estimation of distribution algorithm and deep Q-network is proposed to solve this. The processing sequence, machine assignment, and processing speed assignment are all described using a three-dimensional solution representation. Two knowledge-based initialization strategies are designed for better performance. In the estimation of distribution algorithm component, three probability matrices corresponding to solution representation are provided. In the deep Q-network component, 34 state features are selected to describe the scheduling situation, while nine knowledge-based actions are defined to refine the scheduling solution, and the reward based on the two objectives is designed. As the knowledge for initialization and optimization strategies, five properties of the considered problem are proposed. The proposed mixed integer linear programming model of the problem is validated by exact solver CPLEX. The results of the numerical testing on wide-range scale instances show that the proposed hybrid algorithm is efficient and effective at solving the integrated flexible job shop scheduling problem.**

***Index Terms*—Flexible job shop scheduling problem, deep reinforcement learning, estimation of distribution algorithm, multi-objective optimization.**

I. Introduction

FLEXIBLE job shop scheduling problem (FJSP) has been investigated and applied in various realistic industries, including semi-conductor manufacturing [1], chemical engineering [2], and mobile assembling [3]. For both economical and environmentally friendly development, the effect of time-of-use electricity price (TOUEP) should be considered in practical production; thus, electricity should be consumed during low-price periods. Furthermore, machine processing speed should be taken into account when balancing the production plan and enterprise profit, where operations with high processing speed that cost more electricity are preferred at low-price periods, and operations with low processing speed that cost less electricity can be assigned at high-price periods.

In the considered FJSP, three sub-problems should be solved. First, the order of all operations should be confirmed; then, the machine assignment of each operation should be verified; and lastly, the processing speed should be specified before production. The Canonical FJSP has been proved to be NP-hard [4]. The considered FJSP is also NP-hard because of the constraints of TOUEP, machine setup, and job transportation.

Evolutionary algorithms (EAs) and heuristics containing various reproduction operators and local search strategies have been proposed to solve the constraint FJSP. Jiang and Wang solved a multi-objective FJSP under TOUEP settings, complementing the FJSP model theories of TOUEP constraints [5]. Amiri *et al.* considered an automated FJSP with uncertain resources, where four objectives, i.e., maximum completion time (makespan), number of late jobs, total flow time, and total weighted flow time, were optimized simultaneously [6]. Wu *et al.* designed a multi-objective hybrid pigeon-inspired optimization and simulated annealing (SA) algorithm to solve an FJSP with deterioration effect [7]. Luo *et al.* solved an FJSP with variable processing speeds using grey wolf optimization (GWO) to optimize makespan and total energy consumption. Dai *et al.* developed an enhanced genetic algorithm (GA) for solving an FJSP with transportation and machine idle constraints [8]. Li *et al.* investigated an FJSP with variable processing speed constraint by an improved artifical bee colony (ABC) algorithm to minimize the makespan, total carbon emission, and machine loading simultaneously [9]. Caldeira *et al.* employed an improved backtracking search algorithm (BSA) to solve an FJSP with new job arrivals, whose objectives are makespan, energy consumption, and instability [10], and then, a discrete Jaya algorithm was utilized to optimize the makespan, total workload of machines and workload of critical machine [11].

Manuscript received 29 July 2021; revised 20 October 2021 and 6 December 2021; accepted 14 December 2021. Date of publication 8 February 2022; date of current version 24 July 2023. This work was supported in part by the National Science Foundation of China under Grants 62173216, 61773192, and 61803192, and in part by Shandong Province Natural Science Foundation under Grant ZR2018ZB0419. *(Corresponding author: Jun-qing Li.)*

Yu Du and Xiao-long Chen are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China (e-mail: dyscupse@163.com; chenxiaolong6195@163.com).

Jun-qing Li is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China and also with the School of Computer Science, Liaocheng University, Liaocheng 252059, China (e-mail: lijunqing@lcu-cs.com).

Pei-yong Duan is with the School of Mathematics and Information Sciences, Yantai University, Yantai 264005, China (e-mail: duanpeiyong@sdnu.edu.cn).

Quan-ke Pan is with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China (e-mail: panquanke@shu.edu.cn).

Digital Object Identifier 10.1109/TETCI.2022.3145706

Zhang *et al.* addressed an assembly FJSP using a distributed ant colony system, taking into account operation setup times and transition times between machines [12]. Li *et al.* used a hybrid algorithm of iterated greedy (IG) and SA to solve an FJSP with crane transportation [13]. An *et al.* developed an improved non-dominated sorting biogeography-based optimization algorithm to optimize makespan, critical machine workload, and total workload for FJSP [14]. Focused on a dynamic FJSP, Zhang *et al.* proposed a series of genetic programming approaches, reducing computation cost and achieving significantly better performance [15]–[17]. Wu *et al.* considered an FJSP with dual-resource, loading and unloading, and setup constraints using a similarity-based scheduling algorithm and an improved non-dominated sorting genetic algorithm II (NSGA-II) [18]. Gao *et al.* discussed a flexible job shop rescheduling problem for new job insertion by using discrete Jaya algorithm [19]. Li *et al.* proposed an improved artificial immune system algorithm to solve a type-2 fuzzy FJSP [20]. Du *et al.* solved the distributed FJSP with crane transportation [21].

The above literature solved the FJSP with various constraints, but the performance of these EA approaches may deteriorate drastically for large-scale problems [22]–[27]. To address these issues, reinforcement learning (RL) is being considered to improve the optimization effectiveness for FJSP, which has become a powerful tool in intelligent system design [28]. Recently, deep reinforcement learning (DRL) techniques that combine RL with deep learning (DL) have been applied in solving combinatorial optimization problems (COPs) [29]–[34], demonstrating the strong learning abilities of DRL for COPs.

In 2015, Mnih *et al.* proposed deep Q-network (DQN) based on DRL to solve class Atari 2600 games, providing the fundamental theory of DQN [35]. Then, numerous studies employed RL as the approach to optimize the parameters of EAs. Emary *et al.* utilized RL and neural network to optimize the parameter of GWO [36]. Cao *et al.* proposed a cuckoo search algorithm with RL and surrogate modeling to solve a semiconductor final testing scheduling problem, where RL was used to ensure the desired population diversification and intensification [37], and then, Cao *et al.* significantly improved the effectiveness of a parameter control scheme based on RL and the ability to escape local optima [38]. Chen *et al.* proposed a self-learning GA based on RL for FJSP, in which RL is used to adjust key parameters [39]. Although RL has been applied in parameter optimization to enhance the performance of EAs, the intelligence advantages of RL were not taken best used in the optimization process.

Recently, more studies have employed RL approaches as optimization strategy. Lin *et al.* utilized a multi-class DQN to solve job shop scheduling problem, where the makespan was optimized [40]. Park *et al.* proposed a setup change scheduling method with RL to minimize the makespan for a producing multichip products scheduling problem [41]. Luo designed a DQN algorithm for dynamic scheduling for the FJSP with new job insertions, providing a fundamental approach to solving the FJSP by RL [42]. Hu *et al.* proposed an adaptive DQN based automated guided vehicles real-time scheduling approach with mixed rules for flexible job floor to minimize the makespan and delay ratio [43]. He *et al.* solved agile satellite scheduling problem by a DQN model [44]. Park *et al.* combined graph neural network and RL to solve job shop scheduling problem, where proximal policy optimization based RL strategy were utilized to train the model [45]. Zhao *et al.* proposed a cooperative water wave optimization algorithm with RL to solve a distributed assembly no-idle flow shop scheduling problem [46]. Han and Yang constructed an end-to-end DRL framework to solve FJSP using a modified pointer network to encode the operations and a recurrent neural network to model the decoder network [47]. Kim and Lee proposed a Petri net as RL environment to solve flow shop scheduling problem, where the makespan was optimized [48]. Xu *et al.* designed an RL-based differential evolution algorithm to solve the multi-energy scheduling of an industrial integrated energy system [49]. In the aforementioned studies, COPs have been solved by different RL approaches where the intelligence of RL were utilized. However, most studies only optimized single-objective problems, and RL approaches for multi-objective problems need to be explored further. Furthermore, the advantages of both EAs and RL approaches should be considered for better optimization performance. As the intelligence factor of the model, knowledge-based strategies and heuristics can significantly improve the optimization performance [50].

Therefore, in this paper, to take advantage of both EAs and DRL, a knowledge-based hybrid algorithm of estimation of distribution algorithm (EDA) and DQN (EDA-DQN) for the FJSP is proposed. Two objectives, makespan and total electricity cost (TEC), are optimized by the Pareto-based approach simultaneously. In the EDA-DQN, EDA and DQN components are selected to strengthen the exploration and exploitation abilities, respectively. EDA has strong exploration ability, whereas DQN with knowledge-based actions can enhance the exploitation ability of the proposed algorithm. At each iteration, the EDA component can discover more solution spaces to avoid being trapping in local optima, and the DQN component can refine superior solutions for improved outputs. The DQN component, in particular, is used as a local search selector, which selects an appropriate local search strategy in various scheduling situations.

The main contributions of this work are the following: (1) an FJSP with variable processing speeds, setup time, idle time, job transportation, and TOUEP constraints is considered; (2) the mixed integer linear programming (MILP) model of the considered FJSP is proposed; (3) the makespan and TEC objectives are optimized simultaneously by a hybrid algorithm of EDA and DQN; (4) five properties based on the knowledge of the considered FJSP are constructed as the foundation of the initialization and optimization strategies; (5) 34 state features and 9 actions are designed to enhance the exploitation abilities of the DQN component; and (6) a problem-based EDA component is embedded in the algorithm to enhance the exploration abilities.

The remainder of this paper is organized as follows. Section II contains the considered FJSP and corresponding MILP model. Section III explains the details of the proposed EDA-DQN, and Section IV shows the numerical experimental analysis. Section V summarizes the conclusions and future research topics.

II. Problem Descriptions

In the considered FJSP, I jobs need to be operated by K machines. Each job i has OP_i operations, and each operation $O_{i,j}$ can be performed by a set of available machines $K_{i,j}$. All operations from one job should be performed in sequence. Each operation can be performed at different processing speed, with higher processing speeds costing more electricity energy but taking less time to complete. Machines cannot be shut down before scheduling is completed, and machines consume energy when idle. If two consecutive operations on the same machine are from two different jobs, machine setup condition is needed, which requires both energy and time. Transportation of jobs between machines also costs energy and time. The TOUEP constraint is considered in TEC calculation.

A. Notations

Indices:

- i: index number of jobs;
- j: index number of operations;
- k: index number of machines;
- r: index number of process priorities on each machine;
- p: index number of electricity price intervals;
- s: index number of processing speeds on each machine.

Parameters:

- I: number of jobs;
- OP_i: number of operations of job i;
- ΣOP_i: number of operations of all jobs;
- OP_{std}: standard deviation of operation numbers assigned to all machines;
- K: number of machines;
- S: number of processing speed levels;
- $O_{i,j}$: jth operation of job i;
- $k_{i,j}$: assigned machine of $O_{i,j}$;
- $K_{i,j}$: set of machines that can perform $O_{i,j}$;
- $s_{i,j}$: processing speed level of $O_{i,j}$;
- $s^*_{i,j}$: processing speed of $O_{i,j}$;
- $O_{\pi i,\pi j}$: πth operation in a critical path;
- $C_{\max}, C_{\max,\mathrm{ave}}, C_{\max,\mathrm{std}}$: makespan and average and standard deviation of makespan of all machines, respectively;
- TEC, TEC_{ave}, TEC_{std}: TEC and average and standard deviation of TEC on all machines, respectively;
- t_{MP}, $t_{MP,\mathrm{ave}}$, $t_{MP,\mathrm{std}}$: total machine processing time and average and standard deviation of machine processing time of all machines, respectively;
- TEC_{MP}, $TEC_{MP,\mathrm{ave}}$, $TEC_{MP,\mathrm{std}}$: total machine processing TEC and average and standard deviation of machine processing TEC of all machines, respectively;
- t_{MI}, $t_{MI,\mathrm{ave}}$, $t_{MI,\mathrm{std}}$: total machine idle time and average and standard deviation of machine idle time of all machines, respectively;
- TEC_{MI}, $TEC_{MI,\mathrm{ave}}$, $TEC_{MI,\mathrm{std}}$: total machine idle TEC and average and standard deviation of machine idle TEC of all machines, respectively;
- t_{MS}, $t_{MS,\mathrm{ave}}$, $t_{MS,\mathrm{std}}$: total machine setup time and average and standard deviation of machine setup time of all machines, respectively;
- $TEC_{MS}, TEC_{MS,\mathrm{ave}}, TEC_{MS,\mathrm{std}}$: total machine setup TEC and average and standard deviation of machine setup TEC of all machines, respectively;
- t_{MT}, $t_{MT,\mathrm{ave}}$, $t_{MT,\mathrm{std}}$: total transportation time and average and standard deviation of transportation time of all machines, respectively;
- TEC_{MT}, $TEC_{MT,\mathrm{ave}}$, $TEC_{MT,\mathrm{std}}$: total transportation TEC and average and standard deviation of transportation TEC of all machines, respectively;
- $pt^*_{i,j,k}$: machine processing time of $O_{i,j}$ on machine k;
- $pt_{i,j,k}$: real machine processing time of $O_{i,j}$ on machine k;
- $st_{i1,i2,k}$: machine setup time between jobs $i1$ and $i2$ on machine k;
- $it_{i,j}$: machine idle time of $O_{i,j}$;
- $tt_{k1,k2}$: job transportation time from machine $k1$ to $k2$;
- $sp_{k,s}$: sth level processing speed of machine k;
- $P_{k,s}$: operation power of machine k at sth level processing speed;
- Pti, Pts, Ptt: operation power in machine idle time, setup time and transportation time, respectively;
- $PS_{i,j}, PC_{i,j}$: starting and completion times of $O_{i,j}$, respectively;
- $MS_{k,r}$, $MC_{k,r}$: starting and completion times of the rth assigned operation on machine k, respectively;
- $SS_{i,j}$, $SC_{i,j}$: setup starting and completion times of $O_{i,j}$, respectively;
- $IS_{i,j}$, $IC_{i,j}$: idle starting and completion times of $O_{i,j}$, respectively;
- $TS_{i,j}$, $TC_{i,j}$: transportation starting and completion times of $O_{i,j}$, respectively;
- Lp_p: time of the pth price interval;
- $Ltp_{i,j,k,p,s}$: machine processing time of $O_{i,j}$ on machine k at pth electricity price interval in the sth processing speed level;
- $Lts_{i,j,k,p}$, $Lti_{i,j,k,p}$, $Ltt_{i,j,k,p}$: machine processing time, setup time, idle time, and transportation time of $O_{i,j}$ on machine k at pth price interval, respectively.

Decision variables:

- $Y_{i,j,k,r}$: a binary value that is set to 1 if $O_{i,j}$ is assigned at the rth performed operation on machine k; otherwise, $Y_{i,j,k,r}$ is set to 0;
- $S_{i,j,s}$: a binary value that is set to 1 if $O_{i,j}$ is operated at the sth level processing speed; otherwise, $S_{i,j,s}$ is set to 0;
- $\Psi_{i1,j1,i2,j2,k}$: a binary value that is set to 1 if $O_{i2,j2}$ is the successor operation of $O_{i1,j1}$ on machine k; otherwise, $\Psi_{i1,j1,i2,j2,k}$ is set to 0.

B. Assumptions

- For generality, the starting time of scheduling is 0, and at time 0, all jobs are released and all machines are available;
- Machine processing speed cannot be changed during processing conditions;
- Transportation devices are abundant; if the destination machine is not idle, the job should wait;

- If the operation machine and its same-job predecessor operation machine are identical, no transportation is needed;
- At one time, each machine can only perform one operation, and each operation can only be performed by one machine;
- Overlapping on machine processing and setup time is not permitted;
- Preemption is not permitted;
- No disruption events are considered.

C. Machine Conditions and Transportation

In this study, machine conditions include machine process, machine setup, and machine idle.

At machine process, each machine has four processing speed levels. The real machine processing time $pt_{i,j,k}$ can be calculated by corresponding processing time $pt^*_{i,j,k}$ and processing speed $s^*_{i,j}$ as follows. In this study, $1 \le s^*_{i,j} \le 4$.

$$pt_{i,j,k} = \frac{pt^*_{i,j,k}}{s^*_{i,j}}. \tag{1}$$

The electricity cost of machine processing can be calculated as (2):

$$E_{i,j} = pt_{i,j,k} \times P_{k,l} = \frac{pt^*_{i,j,k}}{s^*_{i,j}} \times P_{k,l}. \tag{2}$$

Similar to Refs [51] and [52], a higher processing speed can shorten the processing time but requires more electricity consumption, which can be expressed in (3).

$$\frac{pt^*_{i,j,k}}{s^{l1}_{i,j}} \times P_{k,l1} > \frac{pt^*_{i,j,k}}{s^{l2}_{i,j}} \times P_{k,l2}, \quad \forall s^{l1}_{i,j} > s^{l2}_{i,j}. \tag{3}$$

According to (3), the optimization of the considered FJSP should find an appropriate processing speed to balance the makespan and TEC.

If two consecutive operations from different jobs are performed on the same machine, setup time is required to adjust machine operation conditions for actual demand. The setup time of all machines is determined by the input data. A machine cannot be turned off until all assigned operations have been completed, When a machine is turned on, the machine is idle if the machine is not in machine processing or setup. When a machine is in idle time, energy can still be consumed but at a low level. When two consecutive operations of one job are not assigned to one machine, the later operation needs transportation after the former operation is finished. The transportation time is provided from the input data.

D. TOUEP

Same as Ref [53], TOUEP is formulated as (4), where n is a positive integer and the unit of time t is the hour.

$$f(t) = \begin{cases} 5, & t \in [24n-3, 24n+7) \\ 8, & t \in [24n+7, 24n+11) \cup [24n+17, 24n+21) \\ 10, & t \in [24n+11, 24n+17) \end{cases}. \tag{4}$$

It can be observed from (4) that the TOUEP keeps the same regulation every day. The electricity price in a day time can be illustrated as Fig. 1. The TEC of a solution is the sum electricity cost of all price intervals according to the TOUEP setting.

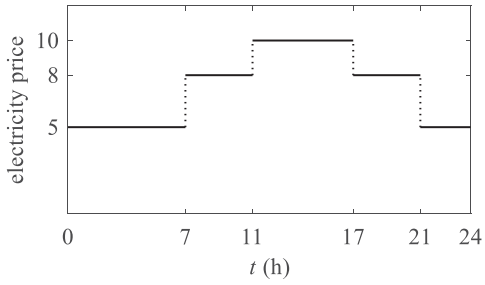


Fig. 1. TOUEP in a day.

E. Formulation of Considered FJSP Model

This section provides the FJSP model. The objectives of the FJSP model are defined as (5–7).

$$\min \{C_{\max}, TEC\}, \tag{5}$$

$$C_{\max} = \max C_{i,j}, \ \forall i,j, \tag{6}$$

$$TEC = TEC_{MP} + TEC_{MS} + TEC_{MI} + TEC_{MT}, \tag{7}$$

$s.t.$

$$PS_{i,j} \ge PC_{i,j-1}, \ \forall i, j = 2, \ldots, OP_i, \tag{8}$$

$$PC_{i,j} \ge PS_{i,j}, \ \forall i,j, \tag{9}$$

$$MS_{k,r} \le MC_{k,r}, \ \forall k,r, \tag{10}$$

$$PS_{i,j} = MS_{k,r}, \ \forall i,j,k,r, Y_{i,j,k,r} = 1, \tag{11}$$

$$PC_{i,j} = MC_{k,r}, \ \forall i,j,k,r, Y_{i,j,k,r} = 1, \tag{12}$$

$$PS_{i,j} \ge 0, \ \forall i,j, \tag{13}$$

$$\sum_{k=1}^{K} \sum_{r=1}^{R} Y_{i,j,k,r} = 1, \ \forall i,j, \tag{14}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{OP_i} Y_{i,j,k,r} \le 1, \ \forall k,r, \tag{15}$$

$$\sum_{i1=1}^{I} Y_{i1,j1,k,r-1} \ge \sum_{i2=1}^{I} Y_{i2,j2,k,r}, \ \forall j1, j2, k, r = 2, \ldots, R, \tag{16}$$

$$PC_{i,j} - PS_{i,j} = pt^*_{i,j,k}/sp_s, \ \forall i,j,k,r,s, \ Y_{i,j,k,r} = 1 \wedge S_{i,j,s} = 1, \tag{17}$$

$$MS_{k,r} \ge MC_{k,r-1}, \ \forall k, r = 2, \ldots, R, \tag{18}$$

$$SC_{k,r} - SS_{k,r} = st_{k,i1,i2}, \ \forall i1, i2, j1, j2, k, \ r = 2, \ldots, R,$$

$$Y_{i1,j1,k,r-1} = 1 \wedge Y_{i2,j2,k,r} = 1, \tag{19}$$

$$SS_{k,r} = MC_{k,r-1}, \ \forall k, r = 2, \ldots, R, \tag{20}$$

$$IS_{k,r} = SC_{k,r}, \ \forall k,r, \tag{21}$$

$$SS_{k,r} \ge 0, \ \forall k,r, \tag{22}$$

$$SC_{k,1} - SS_{k,1} = 0, \ \forall k, \tag{23}$$

$$IS_{k,1} = 0, \ \forall k, \tag{24}$$

$$TC_{i,1} - TS_{i,1} = 0, \ \forall i, \tag{25}$$

$$SC_{k,r} \geq SS_{k,r},\ \forall k, r, \tag{26}$$

$$IC_{k,r} = MS_{k,r},\ \forall k, r, \tag{27}$$

$$IC_{k,r} \geq IS_{k,r},\ \forall k, r, \tag{28}$$

$$TC_{i,j} - TS_{i,j} = tt_{k1,k2},\ \forall i, j = 2, \ldots, OP_i, k1, k2, r1, r2,$$

$$Y_{i,j-1,k1,r1} = 1 \wedge Y_{i,j,k2,r2} = 1, \tag{29}$$

$$TS_{i,j} \geq PC_{i,j-1},\ \forall i,\ j = 2, \ldots, OP_i, \tag{30}$$

$$PS_{i,j} \geq TC_{i,j},\ \forall i,\ j, \tag{31}$$

$$Ltp_{i,j,k,p-1,s} \geq Ltp_{i,j,k,p,s},\ \forall i,\ j,\ k,\ s, p = 2, \ldots, P, \tag{32}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{OP_i} \sum_{s=1}^{S} Ltp_{i,j,k,p,s} \leq Lp_p,\ \forall k,\ p, \tag{33}$$

$$\sum_{p=1}^{P} Ltp_{i,j,k,p} = \frac{PC_{i,j} - PS_{i,j}}{sp_{s,k}},\ \forall i, j, k, r, s, Y_{i,j,k,r}$$

$$= 1 \wedge S_{i,j,s} = 1, \tag{34}$$

$$\sum_{p=1}^{P} Lts_{i,j,k,p} = SC_{k,r} - SS_{k,r},\ \forall i, j, k, r, Y_{i,j,k,r} = 1, \tag{35}$$

$$\sum_{p=1}^{P} Lti_{i,j,k,p} = IC_{k,r} - IS_{k,r},\ \forall i, j, k, r, Y_{i,j,k,r} = 1, \tag{36}$$

$$\sum_{p=1}^{P} Ltt_{i,j,k,p} = TC_{i,j} - TS_{i,j},\ \forall i, j, k, r, Y_{i,j,k,r} = 1, \tag{37}$$

$$Ltp_{i,j,k,p,s} \geq 0,\ \forall i, j, k, p, s, \tag{38}$$

$$Lts_{i,j,k,p} \geq 0,\ \forall i, j, k, p, \tag{39}$$

$$Lti_{i,j,k,p} \geq 0,\ \forall i, j, k, p, \tag{40}$$

$$Ltt_{i,j,k,p} \geq 0,\ \forall i, j, k, p, \tag{41}$$

$$\sum_{s=1}^{S} S_{i,j,s} = 1,\ \forall i, j, \tag{42}$$

$$TEC_{MP} = \sum_{i=1}^{I} \sum_{j=1}^{OP_i} \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{s=1}^{S} (Pt_{k,s} \cdot Ltp_{i,j,k,p} \cdot EP_p), \tag{43}$$

$$TEC_{MS} = \sum_{i=1}^{I} \sum_{j=1}^{OP_i} \sum_{k=1}^{K} \sum_{p=1}^{P} (Lts_{i,j,k,p} \cdot Pts \cdot EP_s), \tag{44}$$

$$TEC_{MI} = \sum_{i=1}^{I} \sum_{j=1}^{OP_i} \sum_{k=1}^{K} \sum_{p=1}^{P} (Lti_{i,j,k,p} \cdot Pti \cdot EP_i), \tag{45}$$

$$TEC_{MT} = \sum_{i=1}^{I} \sum_{j=1}^{OP_i} \sum_{k=1}^{K} \sum_{p=1}^{P} (Ltt_{i,j,k,p} \cdot Ptt \cdot EP_t), \tag{46}$$

Constraint (8) guarantees that all operations of each job should be operated in sequence. Constraints (9) and (10) ensure that the completion time of each operation should be no earlier than its starting time. Constraints (11) and (12) are the links between job processing time and machine perspective, respectively. Constraint (13) indicates that the processing time of each operation is reasonable. Constraint (14) specifies that each operation should be operated by only one machine. Constraint (15) states that only one operation is performed at each processing priority of each machine. Constraint (16) ensures that operations are performed in priority order on each machine. Constraint (17) rules the processing time of each operation. Constraint (18) makes sure that no time overlapping is permitted in FJSP scheduling. Constraint (19) rules the setup time of each operation. Constraint (20) describes that the setup starting time is the completion time of the same-machine predecessor operation. Constraint (21) shows that the setup completion time is the idle starting time of each operation. Constraint (22) denotes that the setup time of each operation is reasonable. Constraints (23)–(25) indicate that the first performed operation of each machine does not need machine setup, idle time, and transportation time, respectively. Constraint (26) ensures that the setup completion time is no earlier than its setup starting time. Constraint (27) describes that the idle completion time is the starting time of each operation. Constraint (28) rules that the idle completion time is no earlier than its idle starting time. Constraint (29) rules the transportation time of each operation. Constraint (30) states that transportation begins when the same-job predecessor operation is completed. Constraint (31) guarantees that the job should be performed after transportation stage. Constraint (32) ensures that processing time is distributed in price intervals order. Constraint (33) ensures that the times in different price intervals are not overlapping. Constraints (34–37) calculate varying time stages under various electricity price intervals. All of the constraints on electricity price intervals are similar to those in Ref [5]. Constraints (38–41) ensure that all time is feasible under different electricity price intervals. Constraint (42) denotes that each operation only has one processing speed. Constraints (43–46) calculate the TEC of machine processing, machine setup, machine idle, and transportation, respectively.

F. Critical Path and Problem-Specific Properties

The critical path $\Pi = \{O^1_{\pi i,\pi j}, O^2_{\pi i,\pi j}, \ldots, O^P_{\pi i,\pi j}\}$ in the considered FJSP is defined as follows: Π starts from the last performed operation on the machine with the maximum makespan; then, if the operation has the machine idle condition, switch to its same-job predecessor operation; and if not, the next operation in Π is the pre-performed operation on the same machine. Operations in Π decide the makespan of scheduling solution. Five problem-specific properties are proposed in this section as the knowledge of the considered FJSP.

Property 1: If $O_{i,j}$ is performed right after $O_{i,j-1}$ ($j \geq 2$) on the same machine k, the operation $O_{i,j}$ does not need transportation, machine setup, and machine idle conditions.

Proof: The two consecutive operations $O_{i,j-1}$ and $O_{i,j}$ are assigned on the same machine, so $O_{i,j}$ can be directly performed on the machine k without unnecessary transportation. For the machine, $O_{i,j-1}$ and $O_{i,j}$ are from one job, and machine conditions should be kept the same, so the machine setup is not

TABLE I
GENERAL CONCLUSION OF OPERATION MOVE [a]

Situations	$PC'_{i,j} < PC_{i,j}$	$PC_{i,j} \le PC'_{i,j} < PC_{i,j+1}$	$PC_{i,j+1} \le PC'_{i,j} < C_{\max}$	$PC'_{i,j} \ge C_{\max}$
$j = OP_i$	$\surd$ (Property 4)	not exist	not exist	$\times$
$\forall O_{i,j'} \notin \Pi, j' > j$	$\surd$ (Property 4)	$*$	$*$	$\times$
$\exists O_{i,j'} \in \Pi, j' > j$	$\surd$ (Property 5)	$*$	$\times$	$\times$

[a] **** or $\times$ means the makespan can be decreased or increased if the corresponding move are realized; * means if no new critical path is generated, the makespan can be decreased.

needed. As for the machine idle condition, $O_{i,j}$ can be performed immediately after the completion of $O_{i,j-1}$, so machine idle is not needed.

Property 2: When solution A only contains one Π, $C_{\max}(A)$ can be decreased if the processing speed of operation $O_{\pi i,\pi j}$ in Π is increased.

Proof: According to (3), when processing speed increases, the processing time of $O_{\pi i,\pi j}$ decreases, and the following operations in Π will move forward to fill the time gap. Eventually, the last operation in Π moves forward, indicating that $C_{\max}(A)$ can be decreased.

Property 3: If solution A has the last performed operation $O_{i,j}$ on any machine that satisfies $O_{i,j} \notin \Pi \wedge s_{i,j} > 1$, and solution B is A with decreased $s_{i,j}$, then the *TEC* of B is smaller than A. Moreover, if the makespan of A and B are the same, B dominates A.

Proof: Considering $s_{i,j}(B) < s_{i,j}(A)$, according to (3), the machine processing energy of $O_{i,j}$ in B is smaller than that in A, and the scheduling elements of other operations remain the same. Therefore, the *TEC* of B is smaller than that of A. If the makespan of the solution is not changed, $C_{\max}(A) = C_{\max}(B)$ and $TEC(B) < TEC(A)$ are satisfied, so B dominates A.

Property 4: For solution A, move $O_{i,j}$ out of Π, the completion time of the moving operation is $PC'_{i,j}$, and if $\forall O_{i,j'}(j' > j) \notin \Pi$, $C_{\max}(A)$ can be decreased.

Proof: As Property 3, when $O_{i,j}$ is moved out of Π, a time gap occurs in Π, and all of the following operations in Π would move forward to fill the gap if possible. $\forall O_{i,j'}(j' > j) \notin \Pi$ means that the completion time of moved $O'_{i,j}$ cannot influence other operations when A is changed while Π is shortened. Therefore, $C_{\max}(A)$ can be decreased.

Property 5: For solution A, move $O_{i,j}$ out of Π, the completion time of the moved operation is $PC'_{i,j}$, and if $\exists O_{i,j'}(j' > j) \in \Pi \wedge PC'_{i,j} < PC_{i,j}$, $C_{\max}(A)$ can be decreased.

Proof: As Property 4, if the successor operation $O_{i,j'}$ of $O_{i,j}$ exists in Π, the completion time change of $O_{i,j}$ should be considered. If $PC_{i,j} \le PC'_{i,j} < PC_{i,j+1}$, the machine assignment change of $O_{i,j}$ will not influence the successor same-job operation in Π. Moreover, if $PC'_{i,j} \ge PC_{i,j+1}$, $C_{\max}(A)$ will increase for new tardiness. To sum up, if $\exists O_{i,j'}(j' > j) \in \Pi \wedge PC'_{i,j} < PC_{i,j}$, $C_{\max}(A)$ can be decreased.

On the basis of Properties 4 and 5, conclusions of all operation moves are summarized in Table I.

III. METHODOLOGY

A. Algorithm Framework

The algorithm framework is illustrated in Fig. 2. In the proposed EDA-DQN, the DQN component should be prepared by network training with target network $\hat{Q}$ and online network Q. Then, all solutions are managed in a population. Following population initialization, the solutions are optimized by the EDA and DQN components in each iteration, where Pareto-based superior solutions are retained by non-dominated sorting and crowding distance sorting. Finally, the Pareto-based solutions set can be obtained in a given CPU runtime.

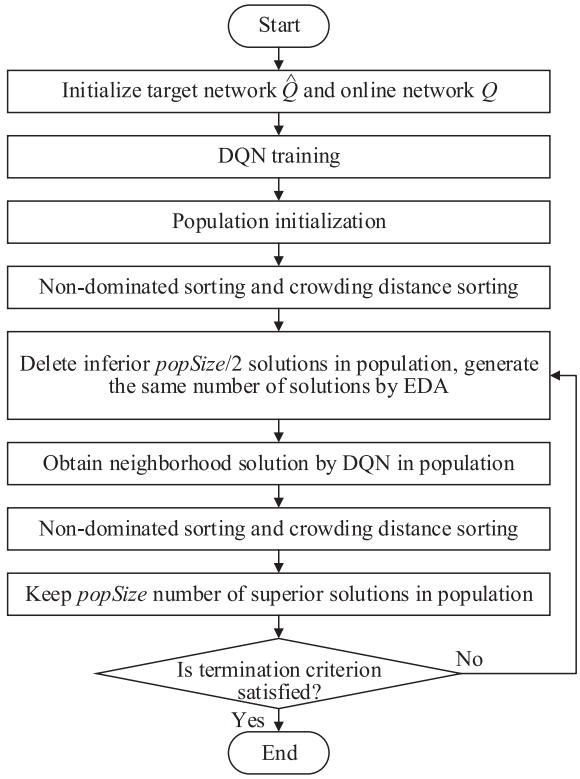


Fig. 2. Algorithm framework of EDA-DQN.

B. Encoding and Decoding

For the solution representation of the considered FJSP, a three-dimensional encoding strategy is designed as follows. First, the scheduling sequence of all operations is managed by scheduling vector (SV), where each element is a job number of the corresponding operation, and the jth occurrence of the job number means the jth operation of the job. The machine assignment of all operations is then recorded in machine assignment vector (MAV), with the machine number of each operation sequenced in the same order as that of SV. Finally, the processing speed level of each operation is listed in processing speed vector (PSV), with the processing speed level of each operation is expressed as the

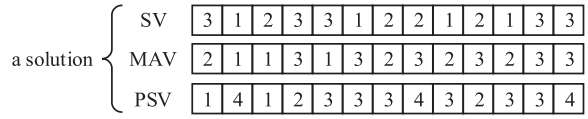


Fig. 3. Solution representation of a three-job and three-machine FJSP.

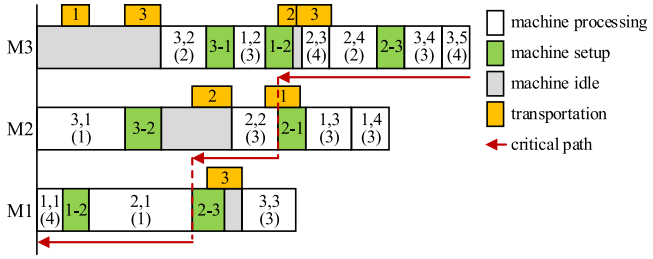


Fig. 4. Gantt chart of a scheduling solution.

corresponding position in MAV. Fig. 3 shows an example of a three-job and three-machine FJSP solution.

In decoding, the processing order of all operations is provided by SV and the starting and completion times of each operation are decided by the completion time of the same-job predecessor operation, machine setup, idle, and transportation time. In the calculation of objectives, makespan is the completion time of the last completed operation, and TEC is the summation of all electricity cost based on TOUEP setting, as shown in (6) and (7), respectively. Fig. 4 depicts a Gantt chart of the decoding results shown in Fig. 3.

C. Population Initialization

Reasonable initialization can cultivate appropriate circumstances in improving exploration and exploitation abilities. In this study, three different initialization strategies are designed based on SV, MAV, and PSV for better outputs. To obtain a diversely initialized population, one-third of the population is initialized based on initialization strategy 1, one-third of the population is initialized based on initialization strategy 2, and the remaining population is initialized at random. In all initialization strategies, all operations are assigned to an available machine, guaranteeing the feasibility of the solutions.

According to Property 1, the next assigned operation in SV from the last uncompleted jobs is preferred for initialization strategy 1 to reduce time and electricity cost of transportation, machine setup, and idle stages. The corresponding machine assignment is the same as predecessor operation. To reduce the TEC, a low processing speed is preferred, so all elements in PSV are chosen at random from $\{1, 2\}$. Initialization strategy 1 is described in Algorithm 1.

Initialization strategy 2 is identical to initialization strategy 1, except for processing speed. To reduce the makespan, a high speed level is preferred. All elements in PSV are randomly selected from $\{3, 4\}$.

Algorithm 1: Initialization strategy 1.

Input: $pt^*_{i,j,k}$, $st_{i1,i2,k}$, $tt_{k1,k2}$ data
Output: initialized solution
for *each position in SV* **do**
  if *the last assigned job i has been completed* **then**
    Randomly choose an uncompleted job i' as the next element in SV
    Choose an available assignment machine k as $\arg\min_{k \in M_{i',j}}(\max(CT_k, CT_{i',j-1}))$ in MAV, if more than one machines have minimum completion time, randomly choose one
  else
    Choose job i as the next element in SV
    if $k_{i,j-1} \in K_{i,j}$ **then**
      Choose the assignment machine as $k_{i,j-1}$ in MAV
    else
      Choose an available assignment machine k as $\arg\min_{k \in M_{i,j}}(\max(CT_k, CT_{i,j-1}))$ in MAV, if more than one machines have minimum completion time, randomly choose one
    end
  end
  The processing speed of the corresponding operation is randomly selected from $\{1, 2\}$
end

D. Estimation of Distribution Algorithm

In the EDA component, three probability matrices, namely, A, B, and C of the considered FJSP are designed to describe production sequence, machine assignment, and processing speed, respectively. In this section, the initialization, update, and solution generation of the probability matrices are described.

1) Probability Matrices Initialization and Update: A_{mi} is the probability that the operation of job i is assigned at the mth position of SV. The initialization of A_{mi} is expressed in (47). The update of A_{mi} in one iteration is ruled in (48), where the indicator I_{mi} is defined in (49), SI is the number of all superior solutions, and the update rate α is 0.5. In this study, SI is a half of all solution numbers.

$$A_{mi}(Gen = 0) = 1/I, \tag{47}$$

$$A_{mi}(Gen + 1) = (1 - \alpha) \cdot A_{mi}(Gen) + \alpha \cdot \frac{1}{SI} \sum_{n=1}^{SI} I^n_{mi}, \tag{48}$$

$$I_{mi} = \begin{cases} 1, & \text{if job } i \text{ is assigned at the } m\text{th position in SV} \\ 0, & \text{otherwise} \end{cases}. \tag{49}$$

B_{ijk} is the probability that $O_{i,j}$ is performed by machine k. The initialization of B_{ijk} is expressed in (50). The update of B_{ijk} in one iteration is ruled in (51), where the indicator I_{ijk} is defined

in (52), and the update rate β is 0.5.

$$B_{ijk}(Gen=0)=\begin{cases}1/K_{av}, & \text{if machine } k \text{ can perform } O_{i,j}\\ 0, & \text{otherwise}\end{cases}, \tag{50}$$

$$B_{ijk}(Gen+1)=(1-\beta)\cdot B_{ijk}(Gen)+\beta\cdot\frac{1}{SI}\sum_{n=1}^{SI}I_{ijk}^{n}, \tag{51}$$

$$I_{ijk}=\begin{cases}1, & \text{if } O_{i,j} \text{ is assigned at machine } k \text{ in MAV}\\ 0, & \text{otherwise}\end{cases}. \tag{52}$$

C_{ijs} is the probability that $O_{i,j}$ is assigned at the sth processing speed level. The initialization of C_{ijs} is expressed in (53). The update of C_{ijs} in one iteration is ruled in (54), where the indicator I_{ijs} is defined in (55), and update rate γ is 0.5.

$$C_{ijs}(Gen=0)=1/S, \tag{53}$$

$$C_{ijs}(Gen+1)=(1-\gamma)\cdot C_{ijs}(Gen)+\gamma\cdot\frac{1}{SI}\sum_{n=1}^{SI}I_{ijs}^{n}, \tag{54}$$

$$I_{ijs}=\begin{cases}1, & \text{if } O_{i,j} \text{ at the } s\text{th speed level in PSV}\\ 0, & \text{otherwise}\end{cases}. \tag{55}$$

2) Solution Generation: According to the three probability matrices, the SV, MAV, and PSV of a solution can be generated. Specifically, for A_{mi}, if all operations of one job have been assigned, the probability of the job assigned at the corresponding position is 0; for B_{ijk}, if machine k cannot perform $O_{i,j}$, B_{ijk} is always 0; and for C_{ijs}, the speed assignment is followed by the probability distribution.

E. Deep Reinforcement Learning

1) State Features: State features describe the solution state at the decision point. Therefore, the design of the state features should reflect the influences of actions, assisting the prediction of action selection.

In this study, 34 state features are adopted in the DQN component. The state features can be divided into three groups. The first group includes problem scale state features such as the number of jobs, machines, operations of all jobs, and standard deviation of operation numbers on all machines OP_{std}. The second group includes objective features such as makespan, average, and standard deviation of the makespan on all machines, TEC, and average and standard deviation of TEC on all machines. The third group includes state features concerning scheduling compositions, where the total time and electricity cost of machine processing, machine idle, machine setup, and transportation are considered. Meanwhile, the corresponding average and standard deviation of the time and electricity cost are adopted. All state features are listed in Fig. 5.

2) Actions: In EDA-DQN, nine actions are designed to improve the quality of generated solutions based on problem-specific properties. Thus, the actions can effectively improve the quality of generated solutions. Furthermore, three random actions are included to improve the exploration ability of the DQN.

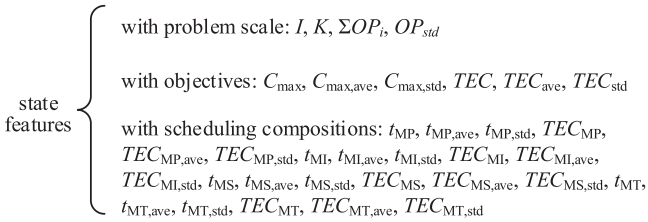


Fig. 5. State features in EDA-DQN.

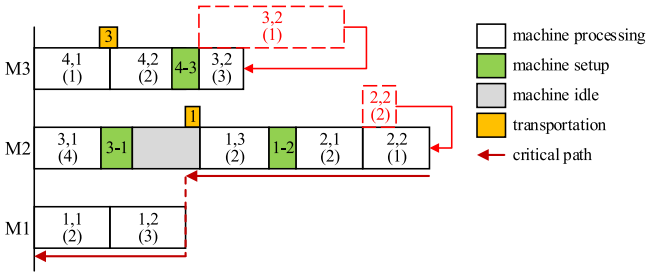


Fig. 6. First action a_1.

Algorithm 2: First action a_1.

Input: initial solution A
Output: improved solution B
for *each operation* $O_{\pi i,\pi j}$ *in critical path* Π *of* A **do**
  if $\exists s_{\pi i,\pi j} < S$ **then**
    Increase $s_{\pi i,\pi j}$
  end
  Randomly select the last performed operation $O_{i,j}$ of each machine
  if $\exists O_{i,j} \notin \Pi \wedge s_{i,j} > 1$ **then**
    Decrease $s_{i,j}$
  end
  return B
end

The first action a_1 is designed based on Properties 2 and 3, optimizing the makespan and TEC simultaneously. a_1 is described in Algorithm 2. It can be seen that, in critical path, if the saved TEC can make up for the extra TEC cost, the solution B generated by a_1 can dominate initial solution A. Fig. 6 illustrates an example of a_1. In Fig. 6, the processing speed of $O_{2,2}$ on the critical path is increased to shorten the makespan, and the speed of $O_{3,2}$ is decreased to make up for the extra electricity cost.

Based on Property 4, the second action a_2 changes the sequence of the operations on one machine from a critical path, whose main purpose is to decrease the machine idle time and electricity cost. a_2 is described in Algorithm 3, and Fig. 7 illustrates an example of a_2. In Fig. 7, $O_{2,1}$ in the critical path is moved in front of $O_{1,3}$ to fill the machine idle time, shortening the makespan without increasing TEC.

The third action a_3 is constructed based on Properties 4 and 5, and it changes the machine assignment of an operation on the critical path. a_3 is described in Algorithm 4, and Fig. 8 illustrates an example of a_3. In Fig. 8, $O_{2,2}$ on the critical path is moved to machine 3 to decrease the makespan.

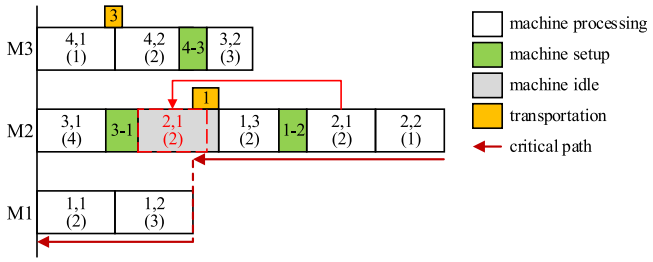


Fig. 7. Second action a_2.

Algorithm 3: Second action a_2.

Input: initial solution A
Output: improved solution B
for *each operation* $O_{\pi i,\pi j}$ *in critical path* Π *of* A **do**
  for *each operation* $O_{i,j}$ *before* $O_{\pi i,\pi j}$ *on its machine* k **do**
    if $(IC_{\pi i,\pi j} - IS_{\pi i,\pi j}) > pt_{\pi i,\pi j,k}$ **then**
      Move $O_{\pi i,\pi j}$ right before $O_{i,j}$
      return B
    end
  end
end

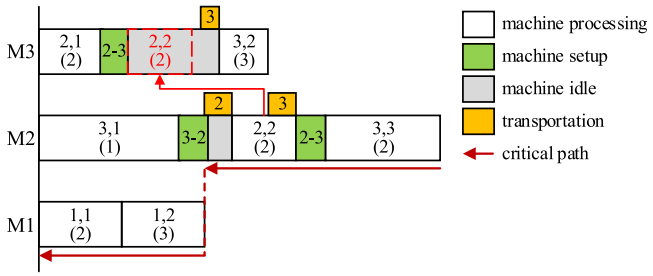


Fig. 8. Third action a_3.

Actions a_4, a_5, and a_6 are referred from canonical mutation operators. a_4 swaps the two operations from different jobs in SV while keeping all machine assignments and processing speed of all operations unchanged. a_5 swaps two pairs of operations as a_4. a_6 randomly selects a pair of operations, inserts the later one right before the former one, and maintains the machine assignment and processing speed of all operations.

To enhance the exploration ability, actions a_7, a_8, and a_9 are designed as random strategies. a_7 adjusts the SV by changing the position of a randomly selected operation, and the assigned machine and processing speed level of all operations remain the same. a_8 only changes the machine assignment of a random operation, and the newly assigned machine is available for operation. a_9 only changes the processing speed level of a random operation.

3) Reward: At decision point t, the reward of the DQN component is defined in Algorithm 5. In reward calculation, all objective values should be normalized based on the objective values at decision point $t-1$. The normalized objective values in decision point t and $t-1$ are $(C_{\max,t}, TEC_t)$ and $(C_{\max,t-1}, TEC_{t-1})$, respectively. The reference point $(C_{\max,R}, TEC_R)$

Algorithm 4: Third action a_3.

Input: initial solution A
Output: improved solution B
for *each operation* $O_{\pi i,\pi j}$ *in critical path* Π *of* A **do**
  if $O_{\pi i,\pi j+1}$ *is on* Π **then**
    if $PC_{\pi i,\pi j-1} < PS_{\pi i,\pi j} \wedge (IC_{i,j} - IS_{i,j}) > 0$ *when* $\Psi_{\pi i,\pi j-1,i,j,k1} = 1$ **then**
      $k_{\pi i,\pi j} = k1$
      return B
    end
  else
    if $(IC_{i,j} - IS_{i,j}) > 0$ *when* $\Psi_{\pi i,\pi j,i,j,k2} = 1$ **then**
      $k_{\pi i,\pi j} = k2$
      return B
    end
  end
end

Algorithm 5: Reward definition.

Input: normalized objective values $C_{\max,t-1}$, $C_{\max,t}$, $C_{\max,R}$, TEC_{t-1}, TEC_t, TEC_R
Output: r_t
$\Delta C_{\max} = C_{\max,t} - C_{\max,t-1}$
$\Delta TEC = TEC_t - TEC_{t-1}$
if $(\Delta C_{\max} < 0 \wedge \Delta TEC < 0) \vee (\Delta C_{\max} > 0 \wedge \Delta TEC > 0)$ **then**
  $r_t = (C_{\max,R} - C_{\max,t-1}) \cdot (TEC_R - TEC_{t-1}) - (C_{\max,R} - C_{\max,t}) \cdot (TEC_R - TEC_t)$
else
  $r_t = 0$
end

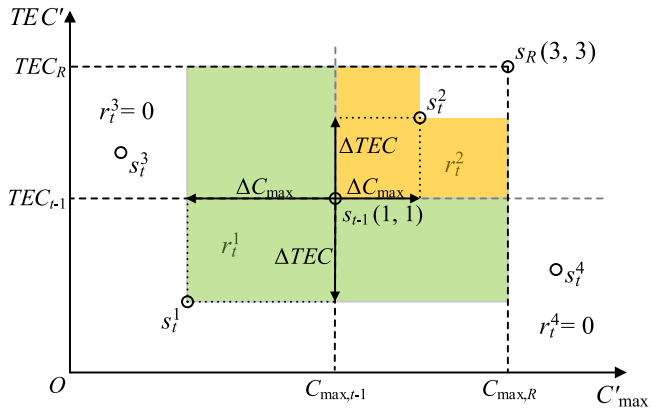


Fig. 9. Rewards of all situations.

is set as (3,3), which is identical to the reference point in hypervolume (HV) calculation.

The design of the rewards is shown in Fig. 9. The solution before the action is s_{t-1}. If the newly generated solution s_t^1 dominates s_{t-1}, which means that the solution is improved, the reward r_t (the green area in Fig. 9) is a positive value. On the contrary, if the newly generated solution s_t^2 is dominated by s_{t-1}, indicating that the solution is worsened by the action, the reward r_t (the yellow ara in Fig. 9) is a negative value. If the newly generated solutions s_t^3 (or s_t^4) and s_{t-1} are not dominated

TABLE II
PARAMETERS IN DQN TRAINING

Parameters	Values
Probability ϵ in ϵ-greedy policy	0.5
Number of training episodes L	10
Number of optimization iteration T	20
Replay memory capacity N	1000
Discount factor γ	0.9
Minibatch size of samples to perform gradient descent	32
τ in soft target update strategy	0.01

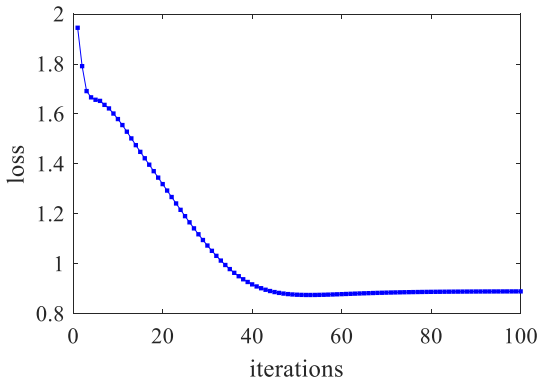


Fig. 10. Loss trend in DQN training (instance 10J5M).

by each other, the reward is 0. This reward design guarantees that the DQN simultaneously optimizes the makespan and TEC in HV perspective.

F. DQN Topology and Training

In DQN, the numbers of input and output nodes are the same as the numbers of state features and actions, respectively. A fully connected neural network is built using two hidden layers, each of which has 20 nodes. The activation function of the input and hidden layers is "ReLU", and "purelin" for output layer. Algorithm 6 describes the DQN training method. Similar to Ref [42], the training parameters are depicted in Table II. The loss trend in DQN training of instance includes 10 jobs and 5 machines (10J5M), which is illustrated in Fig. 10.

G. Non-Dominated Sorting and Crowding Distance Sorting

For two solutions x and y, if $(C_{\max,x} \leq C_{\max,y}) \wedge (TEC_x < TEC_y)$ or $(C_{\max,x} < C_{\max,y}) \wedge (TEC_x \leq TEC_y)$, x dominates y; else, x and y are non-dominated with each other. Comparison of solutions is realized by non-dominated sorting and crowding distance sorting in [54]. The two techniques are utilized in each iteration, and the population structure in an iteration of EDA-DQN is illustrated in Fig. 11.

Algorithm 6: DQN training.

Input: initialized online network $Q(s,a;\theta)$ and target network $\hat{Q}(s,a;\theta^-)$ with the same random weights $\theta = \theta^-$, void replay memory D
Output: trained target network $\hat{Q}$

Initialize a population with L solutions, each solution experiences a whole episode for optimization
for *each episode* **do**
  Observe the initial state s_1 of the solution
  for *each optimization point* $t = 1 : T$ **do**
    Generate a random value X in [0,1]
    if $X < \epsilon$ **then**
      Select a random action a_t
    else
      Select $a_t = \arg\max_a Q(s_t, a; \theta)$
    end
    Execute a_t, observe the next state s_{t+1}, and calculate reward $r_{t,m}$, where $r_{t,m}$ is the reward value r_t when the mth action is executed
    Store combination $\{s_t, a_t, r_{t,m}, s_{t+1}\}$ in D
    Sample random minibatch of combinations $\{s_j, a_j, r_{j,m}, s_{j+1}\}$ from D
    for *each combination of the minibatch* **do**
      Set $y_{j,m} =$
      $\begin{cases} r_{j,m}, \text{if episode terminates at step } j+1 \\ r_{j,m} + \gamma \hat{Q}(s_{t+1}, \arg\max_{a'} Q; \theta^-), \text{otherwise} \end{cases}$
      Perform a gradient descent step on $\sum_{m=1}^{|A|} \left(y_{j,m} - Q(s_j, a_j; \theta)\right)^2$ with respect to the parameters θ of online network Q
      Update target network $\hat{Q} = \tau Q + (1-\tau)\hat{Q}$
    end
  end
end

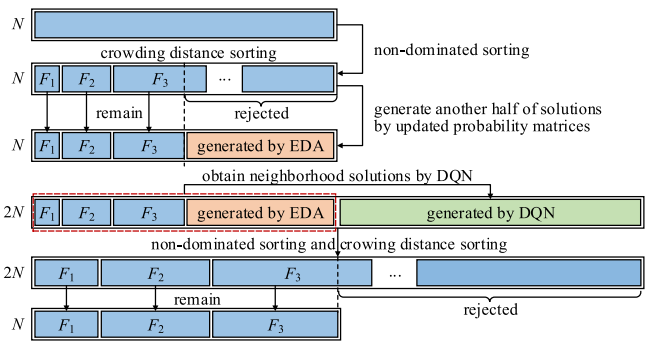


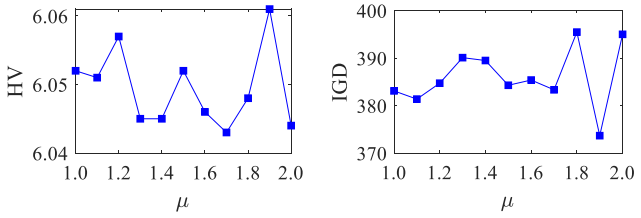
Fig. 11. Population structure in an iteration.

IV. EXPERIMENTAL ANALYSIS

A. Test Instances

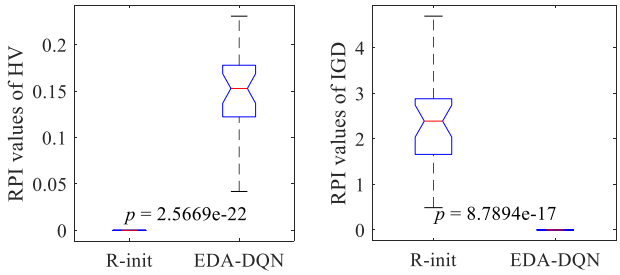
In this study, three groups of instances are generated by code. The first group includes 30 instances, where the job range is $\{10, 20, 30, 40, 50, 60, 80, 100, 150, 200\}$, and the machine range is $\{5, 8, 10\}$. The second group includes 12 instances, where the job range is $\{4, 6, 8, 10\}$, and the machine range is $\{2, 3, 5\}$. The third group includes 10 instances that are from the first group, where the machine number is 8. The instance named "10J5M" means that the instance includes 10 jobs and 5 machines.

The first group instances are used to compare the performance of various algorithms in Sections IV-D to IV-G. The second set of instances is for CPLEX validation, as shown in Section IV-H.



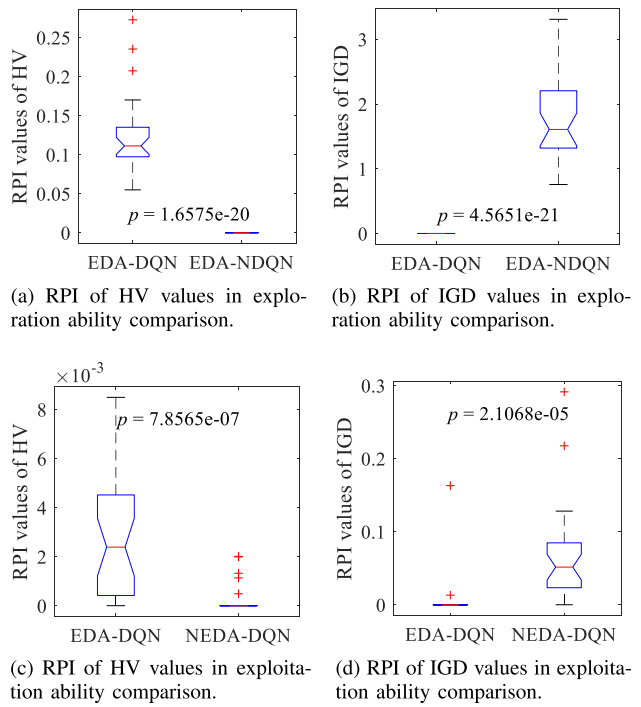
(a) HV trends. (b) IGD trends.

Fig. 12. Criteria values under different levels of μ.



(a) RPI of HV values. (b) RPI of IGD values.

Fig. 13. ANOVA results of initialization strategy comparison.



(a) RPI of HV values in exploration ability comparison.

(b) RPI of IGD values in exploration ability comparison.

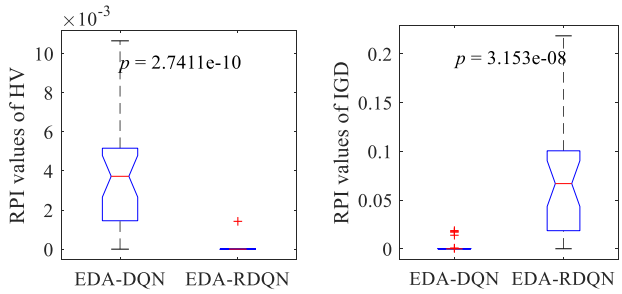
(c) RPI of HV values in exploitation ability comparison.

(d) RPI of IGD values in exploitation ability comparison.

Fig. 14. ANOVA results of exploration and exploitation abilities comparison.

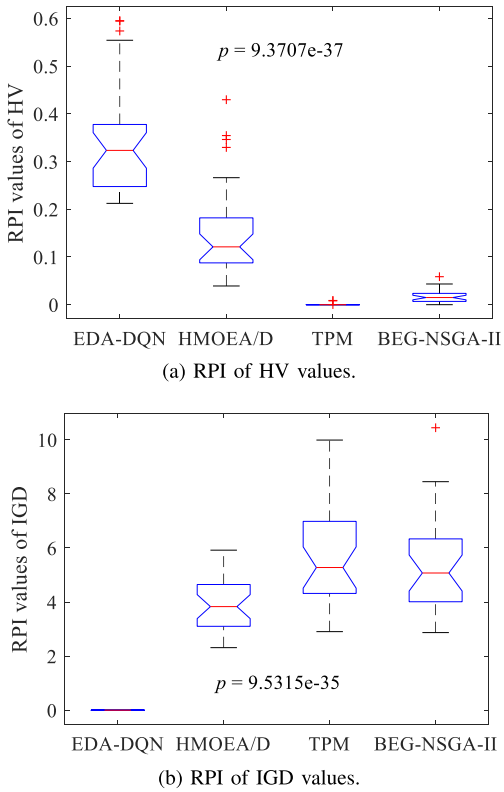
The third group instance is for parameter calibration, which is explained in Section IV-C.

In this study, every runtime in different algorithms for an instance was the same for fairness in comparison. The CPU runtime of each instance was ΣOP_i s.



(a) RPI of HV values. (b) RPI of IGD values.

Fig. 15. ANOVA results of random actions selection comparison.



(a) RPI of HV values.

(b) RPI of IGD values.

Fig. 16. ANOVA results of efficient algorithms comparison.

B. Evaluation Criteria

In this study, to evaluate the performance of the proposed algorithm and other comparison algorithms, HV and inverted generational distance (IGD) are employed.

In HV calculation, the two objectives were normalized as (56):

$$f_i' = f_i / f_{i,\max}, \tag{56}$$

where f_i' is the ith normalized objective value, f_i is the ith objective value, and $f_{i,\max}$ is the maximum value in the ith objective values.

The reference point in HV is set as (3, 3). The HV is calculated as follows:

$$HV = \lambda(\cup_{i=1}^{NS} v_i), \tag{57}$$

TABLE III
COMPARISON RESULTS WITH EFFICIENT ALGORITHMS

Instance	RPI-HV				RPI-IGD			
	EDA-DQN	HMOEA/D	TPM	BEG-NSGA-II	EDA-DQN	HMOEA/D	TPM	BEG-NSGA-II
10J5M	**0.414**	0.211	0.000	0.011	**0.000**	3.594	7.264	6.930
10J8M	**0.573**	0.346	0.008	0.000	**0.000**	4.556	9.988	10.445
10J10M	**0.595**	0.429	0.000	0.000	**0.000**	3.028	8.491	8.407
20J5M	**0.319**	0.111	0.000	0.012	**0.000**	3.750	6.722	6.334
20J8M	**0.594**	0.329	0.000	0.029	**0.000**	3.240	7.381	6.762
20J10M	**0.554**	0.354	0.008	0.000	**0.000**	2.863	7.467	7.218
30J5M	**0.316**	0.089	0.000	0.035	**0.000**	4.441	5.176	4.427
30J8M	**0.378**	0.181	0.000	0.013	**0.000**	3.398	7.774	7.340
30J10M	**0.364**	0.216	0.000	0.021	**0.000**	3.075	9.261	8.450
40J5M	**0.327**	0.091	0.000	0.023	**0.000**	5.031	5.336	5.074
40J8M	**0.359**	0.128	0.000	0.026	**0.000**	4.673	6.983	6.074
40J10M	**0.431**	0.266	0.000	0.019	**0.000**	2.312	6.789	6.243
50J5M	**0.212**	0.062	0.000	0.007	**0.000**	4.969	5.766	5.146
50J8M	**0.356**	0.123	0.000	0.043	**0.000**	4.210	4.966	4.216
50J10M	**0.368**	0.175	0.000	0.038	**0.000**	3.345	6.216	5.358
60J5M	**0.256**	0.089	0.000	0.022	**0.000**	3.776	4.551	4.010
60J8M	**0.329**	0.137	0.000	0.036	**0.000**	3.101	4.185	3.725
60J10M	**0.377**	0.175	0.000	0.058	**0.000**	4.087	6.237	5.212
80J5M	**0.245**	0.074	0.000	0.020	**0.000**	2.850	3.708	3.207
80J8M	**0.232**	0.118	0.000	0.004	**0.000**	4.628	4.719	4.184
80J10M	**0.351**	0.143	0.000	0.022	**0.000**	2.934	3.909	3.941
100J5M	**0.221**	0.069	0.000	0.015	**0.000**	4.016	4.140	3.990
100J8M	**0.295**	0.128	0.000	0.023	**0.000**	2.928	2.904	2.869
100J10M	**0.230**	0.102	0.000	0.003	**0.000**	3.747	4.035	3.482
150J5M	**0.224**	0.039	0.000	0.006	**0.000**	5.582	5.271	5.071
150J8M	**0.250**	0.092	0.000	0.010	**0.000**	4.933	5.001	5.033
150J10M	**0.287**	0.086	0.000	0.002	**0.000**	4.882	5.279	4.962
200J5M	**0.220**	0.065	0.000	0.014	**0.000**	4.647	4.318	4.140
200J8M	**0.274**	0.087	0.000	0.003	**0.000**	5.917	5.045	5.399
200J10M	**0.247**	0.073	0.000	0.011	**0.000**	3.884	3.209	3.141

where $\lambda(\bullet)$ is the Lebesgue measure, NS is non-dominated set, and v_i is the HV of the non-dominated solution p_i.

IGD is calculated as follows:

$$IGD = \frac{1}{n}\sum_{j\in PF*}(\min_{i\in NS}|j-i|), \tag{58}$$

where $PF*$ is the true Pareto front, and $|j-i|$ is the Euclidean distance between solution j from $PF*$ and solution i from NS.

According to the definitions of HV and IGD, it can be found that the solution with bigger HV or smaller IGD means a better result.

For the comparison of criteria values from different algorithms, the relative percentage increase (RPI) is utilized, which is calculated as follows.

$$RPI = \frac{|f_c - f_{\min}|}{f_{\min}} \times 100. \tag{59}$$

In (59), f_c is the average criteria value obtained by the corresponding algorithm, and $f_{\min}$ is the best average criteria value in all algorithms.

TABLE IV
COMPARISON RESULTS WITH THE EXACT CPLEX SOLVER

Instance	HV		IGD	
	CPLEX	EDA-DQN	CPLEX	EDA-DQN
4J2M	5.1794	**6.1863**	7.9873	**0.0820**
4J3M	5.9806	**6.7014**	6.5833	**0.1035**
4J5M	5.9502	**6.4110**	2.6419	**0.1662**
6J2M	5.0513	**6.0607**	12.1929	**0.1929**
6J3M	4.8795	**6.2586**	15.2654	**0.2172**
6J5M	5.1730	**6.2623**	8.9640	**0.2153**
8J2M	5.3437	**6.1432**	14.1404	**0.2697**
8J3M	3.8761	**6.2051**	21.0945	**0.4405**
8J5M	1.5703	**6.4842**	57.3773	**1.0234**
10J2M	5.1050	**6.1502**	23.5459	**0.6413**
10J3M	3.1900	**5.9991**	45.7380	**0.9439**
10J5M	\	**5.8517**	\	**1.1056**

C. Parameter Calibration

For the optimization of EDA-DQN, the most influential factor is the μ of the softmax strategy in action selection. Specifically, large μ tends to select the action with the highest Q value but can

be easily trapped in local optima, whereas small μ tends to select the action more randomly but loses information of knowledge-based strategies. As a result, we only calibrated the parameter μ to improve algorithm performance. In parameter calibration, the third group instances are used, where each instance was run for 20 repeats, and the average HV and IGD values were collected.

Fig. 12(a) and (b) show the HV and IGD trends for various μ values from 1.0 to 2.0, respectively. According to the trends, when μ is 1.9, both the HV and IGD values achieve the best performance. As a result, μ is set to 1.9 for all following experiments.

D. Comparison With Initialization Strategies

To validate the effectiveness of the proposed initialization strategies, a comparison algorithm EDA-DQN with the random initialization strategy (R-init) is executed. The analysis of variance (ANOVA) results of the RPI values of HV and IGD are illustrated in Fig. 13(a) and (b), respectively. The results showed that the initialization strategies of the EDA-DQN perform significantly better than the random initialization strategy.

E. Comparison of Exploration and Exploitation Abilities

To compare the exploration and exploitation abilities of the EDA-DQN, three algorithms, i.e. EDA-DQN without EDA component (NEDA-DQN), EDA-DQN without DQN component (EDA-NDQN), and EDA-DQN with all components (EDA-DQN), are selected. Because the complexity of the three algorithms differs, all of the comparison algorithms in this section are iterated 1000 times for fairness.

The ANOVA results of the RPI values of HV and IGD are illustrated in Figs. 14(a)–(d), respectively. From the ANOVA results, the proposed EDA-DQN performs better than both the vanilla EDA component and vanilla DQN component, indicating that EDA-DQN combines the exploration ability of EDA and the exploitation ability of DQN.

F. Comparison With Random Actions Selection

To prove the effectiveness of the DQN selection, a comparison experiment to EDA-DQN with random action selection (EDA-RDQN) is executed. Fig. 15(a) and (b) illustrate the RPI of HV and IGD values, respectively. The results showed that the DQN component can select appropriate action in various scheduling conditions.

G. Comparison With Efficient Algorithms

To validate the effectiveness of EDA-DQN, three competitive algorithms, i.e., HMOEA/D proposed by Jiang and Wang [5], TPM proposed by Lei *et al.* [55], and BEG-NSGA-II proposed by Deng *et al.* [56], were selected as the comparison algorithms. All the three comparison algorithms are designed for solving FJSP.

The HV and IGD values of all comparison algorithms are collected, and the average RPI values are listed in Table III.

Fig. 16 illustrates the RPI of the HV and IGD values. The results showed that EDA-DQN performs better compared with the other efficient algorithms in solving the considered FJSP.

H. Comparison With the Exact CPLEX Solver

To validate the proposed MILP model, performance comparison with the exact solver IBM ILOG CPLEX 12.7 was employed. In CPLEX optimization, the weighted objective $w * C_{\max} + (1 - w) * TEC$ was utilized as the model objective. The weigh w varies in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, and nine optimization results were obtained in an instance. The running time of each weight at each instance in CPLEX is 1 h. Table IV shows the optimization results of the CPLEX solver and EDA-DQN, where the CPLEX result of instance 10J5M is not available in the given running time. The HV and IGD values of the exact CPLEX solver are determined by the Pareto front of the nine results. In Table IV, the HV and IGD values of EDA-DQN are the maximum values in 20 repeat runs.

From Table IV, conclusions can be observed that EDA-DQN can explore more superior solutions both in HV and IGD compared with the CPLEX solver in all 12 test instances.

V. Conclusion

In this study, a knowledge-based hybrid algorithm EDA-DQN is proposed to solve the constrained FJSP, where the makespan and TEC are optimized simultaneously. Five knowledge-based properties of FJSP are summarized and proved to enhance the exploration and exploitation abilities of EDA-DQN. The effectiveness of the proposed initialization strategies and MILP model are validated by numerical experiments. Moreover, the comparison experiments proved that both EDA and DQN components can improve the performance of EDA-DQN, and the EDA-DQN can obtain more satisfied solutions compared with other related competitive algorithms.

The limitations of this study are the following: (1) neural network models designed for sequence, such as pointer networks, should be considered for better outputs, and (2) specific application circumstances should be applied. As a result, works will primarily focus on the following tasks: (1) applying other effective RL approaches to solve the constrained FJSP, (2) applying other realistic constraints and objectives into the considered problem, (3) further enhancing the population diversity and convergence abilities of the proposed EDA-DQN algorithm, and (4) applying EDA-DQN to other real-world problems, such as prefabricated systems.

References

[1] T. Jamrus, C.-F. Chien, M. Gen, and K. Sethanan, "Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 32–41, Feb. 2018.

[2] P. Dziurzanski, S. Zhao, S. Scholze, A. Zilverberg, K. Krone, and L. S. Indrusiak, "Process planning and scheduling optimisation with alternative recipes," *At-Automatisierungstechnik*, vol. 68, no. 2, pp. 140–147, 2020.

[3] A. Ham, "Transfer-robot task scheduling in flexible job shop," *J. Intell. Manuf.*, vol. 31, no. 7, pp. 1783–1793, 2020.

[4] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 904–916, Jul. 2019.

[5] E. D. 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.

[6] F. Amiri, B. Shirazi, and A. Tajdin, "Multi-objective simulation optimization for uncertain resource assignment and job sequence in automated flexible job shop," *Appl. Soft Comput.*, vol. 75, pp. 190–202, 2019.

[7] X. Wu, X. Shen, and C. Li, "The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously," *Comput. Ind. Eng.*, vol. 135, pp. 1004–1024, 2019.

[8] M. Dai, D. Tang, A. Giret, and M. A. 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.

[9] Y. Li, W. Huang, R. Wu, and K. Guo, "An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem," *Appl. Soft Comput.*, vol. 95, 2020, Art. no. 106544.

[10] R. H. Caldeira, A. Gnanavelbabu, and T. Vaidyanathan, "An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption," *Comput. Ind. Eng.*, vol. 149, 2020, Art. no. 106863.

[11] R. H. Caldeira and A. Gnanavelbabu, "A Pareto based discrete jaya algorithm for multi-objective flexible job shop scheduling problem," *Expert Syst. Appl.*, vol. 170, 2021, Art. no. 114567.

[12] S. Zhang, X. Li, B. Zhang, and S. Wang, "Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system," *Eur. J. Oper. Res.*, vol. 283, no. 2, pp. 441–460, 2020.

[13] J. Li *et al.*, "A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem," *IEEE Trans. Automat. Sci. Eng.*, to be published, doi: 10.1109/TASE.2021.3062979.

[14] Y. An, X. Chen, Y. Li, Y. Han, J. Zhang, and H. Shi, "An improved non-dominated sorting biogeography-based optimization algorithm for the (hybrid) multi-objective flexible job-shop scheduling problem," *Appl. Soft Comput.*, vol. 99, 2021, Art. no. 106869.

[15] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 552–566, Jun. 2021.

[16] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 1797–1811, Apr. 2021.

[17] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2021.3050141.

[18] X. Wu, J. Peng, X. Xiao, and S. Wu, "An effective approach for the dual-resource flexible job shop scheduling problem considering loading and unloading," *J. Intell. Manuf.*, vol. 32, no. 3, pp. 707–728, 2021.

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

[20] J. Li, Z. Liu, C. Li, and Z. Zheng, "Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 11, pp. 3234–3248, Nov. 2021.

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

[22] Y. Tian *et al.*, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, 2022, Art. no. 174, doi: 10.1145/3470971.

[23] R. Tanabe and H. Ishibuchi, "A review of evolutionary multimodal multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 193–200, Feb. 2020.

[24] R. Wang, S. Lai, G. Wu, L. Xing, L. Wang, and H. Ishibuchi, "Multi-clustering via evolutionary multi-objective optimization," *Inf. Sci.*, vol. 450, pp. 128–140, 2018.

[25] Y. Wang, L. Wang, Z. Peng, G. Chen, Z. Cai, and L. Xing, "A multi ant system based hybrid heuristic algorithm for vehicle routing problem with service time customization," *Swarm Evol. Computation*, vol. 50, 2019, Art. no. 100563.

[26] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Computation*, vol. 18, no. 1, pp. 82–97, Feb. 2014.

[27] A. Majumder, D. Laha, and P. N. Suganthan, "A hybrid cuckoo search algorithm in parallel batch processing machines with unequal job ready times," *Comput. Ind. Eng.*, vol. 124, pp. 65–76, 2018.

[28] M. H. Khooban and M. Gheisarnejad, "A novel deep reinforcement learning controller based type-II fuzzy system: Frequency regulation in microgrids," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 4, pp. 689–699, Aug. 2021.

[29] J. James, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3806–3817, Oct. 2019.

[30] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3103–3114, Jun. 2021.

[31] K. Li, T. Zhang, R. Wang, Y.-h. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Trans. Cybern.*, to be published, 2021, doi: 10.1109/TCYB.2021.3103811.

[32] J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," *Comput. Ind. Eng.*, vol. 110, pp. 75–82, 2017.

[33] Y. Zhao and H. Zhang, "Application of machine learning and rule scheduling in a job-shop production control system," *Int. J. Simul. Model.*, vol. 20, no. 2, pp. 410–421, 2021.

[34] B. Cunha, A. Madureira, B. Fonseca, and J. Matos, "Intelligent scheduling with reinforcement learning," *Appl. Sci.*, vol. 11, no. 8, 2021, Art. no. 3710.

[35] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] E. Emary, H. M. Zawbaa, and C. Grosan, "Experienced gray wolf optimization through reinforcement learning and neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 681–694, Mar. 2018.

[37] Z. Cao, C. Lin, M. Zhou, and R. Huang, "Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 2, pp. 825–837, Apr. 2019.

[38] 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, Jan. 2021.

[39] R. Chen, B. Yang, S. Li, and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 149, 2020, Art. no. 106778.

[40] C. Lin, D. Deng, Y. Chih, and H. 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.

[41] I.-B. Park, J. Huh, J. Kim, and J. Park, "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities," *IEEE Trans. Automat. Sci. Eng.*, vol. 17, no. 3, pp. 1420–1431, Jul. 2020.

[42] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Appl. Soft Comput.*, vol. 91, 2020, Art. no. 106208.

[43] H. Hu, X. Jia, Q. He, S. Fu, and K. Liu, "Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0," *Comput. Ind. Eng.*, vol. 149, 2020, Art. no. 106749.

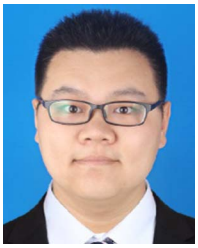
[44] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, and G. Wu, "A generic markov decision process model and reinforcement learning method for scheduling agile earth observation satellites," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, 2020, doi: 10.1109/TSMC.2020.3020732.

[45] J. Park, J. Chun, S. H. Kim, Y. Kim, and J. Park, "Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning," *Int. J. Prod. Res.*, vol. 59, no. 11, pp. 3360–3377, 2021.

[46] F. Zhao, L. Zhang, J. Cao, and J. Tang, "A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 153, 2021, Art. no. 107082.

[47] B. Han and J. Yang, "A deep reinforcement learning based solution for flexible job shop scheduling problem," *Int. J. Simul. Model.*, vol. 20, no. 2, 2021, pp. 375–386.

[48] H. Kim and J. Lee, "Scheduling of dual-gripper robotic cells with reinforcement learning," *IEEE Trans. Automat. Sci. Eng.*, early access, 2021, doi: 10.1109/TASE.2020.3047924.
[49] Z. Xu, G. Han, L. Liu, M. Martínez-García, and Z. Wang, "Multi-energy scheduling of an industrial integrated energy system by reinforcement learning based differential evolution," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1077–1090, Sep. 2021.
[50] J. Li, X. Chen, P. Duan, and J. Mou, "Kmoea: A knowledge-based multi-objective algorithm for distributed hybrid flow shop in a prefabricated system," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2021.3128405.
[51] 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.
[52] Z. Pan, D. Lei, and L. Wang, "A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2020.3026571.
[53] H. Luo, B. Du, G. Q. Huang, H. P. Chen, and X. L. Li, "Hybrid flow shop scheduling considering machine electricity consumption cost," *Int. J. Prod. Econ.*, vol. 146, no. 2, pp. 423–439, Dec. 2013.
[54] 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.
[55] 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.
[56] Q. Deng, G. Gong, X. Gong, L. Zhang, W. Liu, and Q. Ren, "A bee evolutionary guiding nondominated sorting genetic algorithm II for multiobjective flexible job-shop scheduling," *Comput. Intell. Neurosci.*, vol. 2017, 2017, Art. no. 5232518.



Yu Du received the B.S. and M.S. degrees from the School of Chemical Engineering, Sichuan University, Chengdu, China. He is currently working toward the Ph.D. degree with Shandong Normal University, Jinan, China.

His current research interests include intelligent optimization and control.

Jun-qing Li (Member, IEEE) received the master's degree in computer science and technology from Shandong Economic University, Shandong, China, in 2004, and the Ph.D. degree from Northeastern University, Shenyang, China, in 2016.

Since 2004, he has been with the School of Computer, Liaocheng University, Liaocheng, China. Since 2017, he has been with the School of Information Science and Engineering, Shandong Normal University, Jinan, China, where he became a Professor in 2017. He has authored more than 30 refereed papers. His current research interests include intelligent optimization and scheduling.

Xiao-long Chen received the B.S. degree from the School of Computer, Liaocheng University, Liaocheng, China. He is currently working toward the Ph.D. degree with Shandong Normal University, Jinan, China. His current research interests include intelligent optimization and control.

Pei-yong Duan received the B.Sc. degree from the Shandong University of Technology (merged into Shandong University in 2000), Shandong, China, in 1996, and the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 1999.

From 1999 to 2014, he was with the School of Information and Electrical Engineering, Shandong Jianzhu University, where he was appointed as an Associate Professor in 1999, and a Full Professor in 2002. Since 2017, he has been with the School of Information Science and Engineering, Shandong Normal University, Jinan, China. He has authored more than 60 refereed papers. His current research interests include discrete optimization and scheduling.

Quan-ke Pan received the B.Sc. and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1993 and 2003, respectively.

From 2003 to 2011, he was with the School of Computer Science Department, Liaocheng University, Liaocheng, China, where he became a Full Professor in 2006. From 2011 to 2014, he was with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China. From 2014 to 2015, he was with the State Key Laboratory of Digital Manufacturing and Equipment Technology, Huazhong University of Science and Technology, Wuhan, China. Since 2015, he has been with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China. He has authored one academic book and more than 200 refereed papers. His current research interests include intelligent optimization and scheduling algorithms.

Prof. Pan is an Editorial Board Member for several journals, including *Operations Research Perspective* and *Swarm and Evolutionary Computation*.