An End-to-End Framework for Energy-Efficient Cascaded Dual-Shop Collaborative Scheduling With Mating Operations

Haizhu Bao, Quanke Pan, *Member, IEEE*, Chee-Meng Chew, *Senior Member, IEEE*, Ling Wang, *Senior Member, IEEE*, and Liang Gao, *Senior Member, IEEE*

***Abstract*—Due to the complexity of modern production processes and environments, most products must pass through multiple workshops from raw materials to finished goods. This article investigates a collaborative scheduling problem in a cascaded dual-shop production setting. Unlike single-shop scheduling or distributed multiworkshop scheduling, this problem emphasizes collaborative optimization between two interdependent workshops. In addition, real-world production often involves a mode where main and suborders must be integrated through mating operations. This study formulates an energy-efficient cascaded dual-shop collaborative scheduling problem with the mating operation (ECDCSP-M). The focus is on developing a mixed-integer linear programming (MILP) model for the ECDCSP-M and designing an end-to-end graph-based deep reinforcement learning (GDRL) approach. A dual-shop heterogeneous graph is constructed to capture the real-time state of the entire system, in which "job-factory" and "operation-machine" pairs are defined as agent actions. A heterogeneous graph neural network (HGNN) is then proposed, employing a three-stage embedding mechanism to model complex relationships, including mating operations. Experimental results show that the proposed method achieves strong generalization across varying problem complexities and provides robust solutions to challenging scheduling scenarios.**

Received 30 April 2025; revised 18 July 2025; accepted 25 July 2025. This work was supported in part by the National Nature Science Foundation of China under Grant 62273221; in part by the Program of Shanghai Academic/Technology Research Leader under Grant 21XD1401000; and in part by the Shanghai Key Laboratory of Power station Automation Technology. This article was recommended by Associate Editor S. Rong. *(Corresponding author: Quanke Pan.)*

Haizhu Bao is with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China, and also with the Department of Mechanical Engineering, National University of Singapore, Singapore 117575 (e-mail: banian2314@outlook.com).

Quanke Pan is with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China, and also with the School of Computer Science, Liaocheng University, Liaocheng 252000, China (e-mail: panquanke@shu.edu.cn).

Chee-Meng Chew is with the Department of Mechanical Engineering, National University of Singapore, Singapore 117575 (e-mail: chewcm@nus.edu.sg).

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

Liang Gao is with the National Center of Technology Innovation for Intelligent Design and Numerical Control, and the State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: gaoliang@mail.hust.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2025.3594063.

Digital Object Identifier 10.1109/TCYB.2025.3594063

***Index Terms*—Cascaded dual-shop, deep reinforcement learning (DRL), end-to-end framework, energy-efficient scheduling, mating operation.**

I. Introduction

IN MANUFACTURING enterprises, the production process from raw materials to finished products typically involves various stages, such as casting, machining, and surface treatment. Depending on specific production environments or requirements, these processes may be distributed across different workshops. Previous studies have classified the scheduling challenges in such production scenarios as the serial multishop collaborative scheduling problem (SMCSP) [1]. This problem is commonly observed in modern manufacturing systems, and a representative example is illustrated. As shown in Fig. S.1 of the Supplementary Material, during the production of printed circuit boards (PCBs), the main board goes through sequential phases, including surface mount technology (SMT) and dual in-line packaging (DIP). Additionally, a transportation stage exists between these production phases. The first phase comprises multiple independent production lines and includes operations, such as surface printing, high-speed chip placement, general chip placement, reflow soldering, and welding [2], [3]. This workshop can be modeled as a distributed flow-shop problem (DFSP). In the second phase, due to process constraints, each operation—such as component insertion, wave soldering, tin plating, testing, and final inspection—is performed on a set of parallel machines, characterizing this workshop as a hybrid flow-shop problem (HFSP).

In the past, production information across multiple workshops was not shared, thereby restricting production management to isolated workshop-level operations. Under such circumstances, collaborative scheduling was severely limited, compelling companies to maintain substantial inventories of work-in-progress within each workshop, which inevitably led to increased manufacturing costs. Despite significant advances in information technology, the optimization of production management—particularly shop scheduling—still largely focused on individual workshops and fail to adequately address the challenges of collaborative scheduling [1]. Therefore, optimizing collaborative scheduling in cascaded multishop environments has become imperative for improving production efficiency and reducing manufacturing costs.

Furthermore, in make-to-order production systems, it is often necessary to merge two or more operations to ensure the positioning accuracy and machining precision of critical components [4]. The corresponding orders associated with these operations are referred to as the main order and the suborder, respectively [5], [6]. For example, in the previously mentioned case, the main order produces the main control chip module, while the suborder is responsible for the power management module. These two modules must be integrated during the testing stage. In this study, operations that are required to be processed simultaneously are defined as mating operations.

This article investigates an energy-energy-efficient cascaded dual-shop collaborative scheduling problem with the mating operations (ECDCSP-Ms), which is abstracted from the aforementioned scenario. The objective is to minimize both the makespan ($C_{\max}$) and the total energy consumption (TEC). This problem is inherently NP-hard, as both the DFSP and the HFSP are already proven to be NP-hard [7], [8]. The core challenges of ECDCSP-M include: 1) developing a scheduling strategy for collaborative production across cascaded dual shops; 2) efficiently coordinating mating operations between main and suborders; and 3) simultaneously optimizing both $C_{\max}$ and TEC.

To address these challenges, the paper proposes an end-to-end graph-based deep reinforcement learning (GDRL) approach designed to learn high-quality priority dispatching rules (PDRs). First, the ECDCSP-M is formulated as a Markov decision process (MDP) grounded in PDRs. Next, a heterogeneous disjunctive graph, denoted as $\mathcal{G}^H$, is constructed to represent the MDP states of ECDCSP-M, capable of capturing complex relationships—including those induced by mating operations. Furthermore, a three-stage graph neural network (GNN) is developed to extract feature embeddings from the nodes in $\mathcal{G}^H$. These embeddings serve as the input for a policy network, which is trained to solve ECDCSP-M instances of varying scales and complexities. The main contributions of this work are summarized as follows.

1) We propose a novel scheduling framework for the ECDCSP-M that jointly considers distributed and hybrid flow-shop environments with interdependent suborders—a scenario rarely explored in existing literature.
2) We develop a unified MDP formulation that simultaneously integrates dual-shop collaborative scheduling decisions and mating constraints, overcoming the limitations of decoupled decision-making in prior studies.
3) We design a novel heterogeneous disjunctive graph structure to model the complex scheduling environment, capturing intricate relationships among operations, machines, and orders—including mating operations—while maintaining computational efficiency via a sparse graph representation.
4) We develop an end-to-end deep reinforcement learning (DRL) method that autonomously learns scheduling policies, in contrast to traditional rule-based or heuristic methods that rely on handcrafted features. The proposed approach adopts a size-agnostic architecture, enabling generalization from small- to large-scale instances.

The organization of the paper is as follows. The literature review and the problem description are given in Sections II and III, respectively. The GDRL is elaborated in Section IV, followed by the presentation and analysis of the experimental results in Section V. Finally, conclusions and potential avenues for future research are discussed in Section VI.

II. Related Works

This section presents a literature review from two perspectives. The first perspective reviews multishop scheduling problems (MSPs), which are closely related to the ECDCSP-M. The second perspective focuses on key DRL approaches relevant to scheduling problems.

A. Multishop Scheduling

In the existing literature, most research on MSP has focused on multishop assembly scheduling problems. These problems are characterized by the operational independence between the machining and assembly shops, where all manufacturing operations are completed in the machining shop before final assembly takes place in assembly shop [9]. Due to constraints imposed by the assembly process, the main challenge lies in coordinating process dependencies across different jobs. In recent years, numerous researchers have investigated production scheduling across multiple shops. Although these problems involve multiple shops, the intershop production relationships are parallel, meaning that outsourced jobs can be assigned to any shop for processing [10], [11]. Such problems are often referred to as distributed scheduling problems (DSPs) [12], [13]. In contrast, cascaded scheduling involves precedence constraints between shops, making intershop coordination the core challenge.

MSPs are common in real-world production settings; however, research on these problems remains limited. Traditionally, these problems have been addressed using heuristic rules or metaheuristic algorithms tailored to specific production scenarios. For example, Gong et al. [14] proposed a heuristic rule to solve the MSP in the steel industry. Yao et al. [15] developed a heuristic for MSPs with dynamic job arrivals in semiconductor manufacturing. Meng et al. [16] introduced a harmony search algorithm to solve MSPs in a structural metal manufacturing setting. More recently, Wang et al. [17] investigated a dual-shop scheduling problem in the context of PCB manufacturing.

As evidenced by the aforementioned literature, MSPs possess considerable practical significance. In these studies, MSPs are typically treated as integrated problems, where heuristics or metaheuristics are employed to independently address scheduling challenges within individual shops [18], [19]. However, research on collaborative optimization across multiple shops remains limited. As a result, these approaches encounter significant challenges when applied to large-scale problems.

B. DRL-Based Scheduling Methods

In recent years, scheduling methods based on DRL have achieved significant progress. The core challenge in applying DRL to scheduling problems lies in the effective representation of states [20], [21], [22]. Accordingly, this review is conducted

from two perspectives: 1) vector-based DRL methods and 2) graph-based DRL methods. Vector-based DRL methods typically represent states using manually crafted features vectors, which are input into a multilayer perceptron (MLP), producing PDRs as scheduling actions. Notable examples of vector-based DRL methods include double deep Q-networks [23], [24], [25], [26], [27], [28], pointer networks [29], and distributed DRL frameworks [24], [30], [31]. However, due to their reliance on manually crafted features, these vector-based DRL methods depend heavily on expert knowledge. Consequently, the performance of the scheduling agent is largely contingent upon the quality of the predefined PDR set [20], [32]. Moreover, restricting scheduling actions to PDRs limited the action space, thereby constraining exploration capabilities and potentially diminishing overall performance.

Graph-based DRL methods have recently attracted increasing attention for tackling complex scheduling problems, owing to their impressive performance. Their success is primarily attributed to the ability of GNNs to intricately model operations, machine, and structural dependencies within the scheduling problem, regardless of instance size [21], [33], [34]. For instance, Wang et al. [35] proposed a dual-attention network that integrates multiple operation-message blocks and machine-attention modules. Other representative works include [36], [37], [38], [39], [40]. Overall, graph-based DRL methods adopt operation-machine pair as direct scheduling actions, enabling the exploration of a substantially larger solution space and thereby enhancing their search capability. Moreover, the trained models exhibit size-invariant properties, allowing them to generalize effectively to previously unseen instance scales, thus demonstrating superior generalization performance [41], [42], [43], [44].

C. Research Gaps

Based on the preceding analysis, MSPs exhibit substantial practical significance, and graph-based DRL approaches have proven effective in addressing complex scheduling challenges. Nevertheless, several critical research gaps remain in the current literature.

1) Most existing MSP studies primarily concentrate on optimizing the scheduling of individual shops independently, with limited emphasis on effective modeling and solution strategies for collaborative scheduling across multiple workshops.
2) The scheduling challenges posed by mating operations, particularly in practical production settings involving main and suborders, remain insufficiently addressed.
3) End-to-end models for solving multiobjective scheduling problems are notably lacking.

III. Problem Description

The ECDCSP-M is formulated as a collaborative production process comprising two cascading flow shops. As illustrated in Fig. 1, the first phase (Phase 1) constitutes a DFSP, while the second phase (Phase 2) corresponds to a heterogeneous hybrid flow-shop scheduling problem with mating operations (HHSP-M). In Phase 1, δ identical factories are considered,

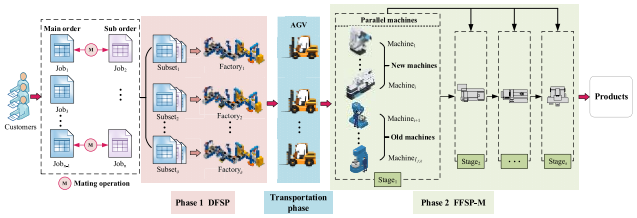


Fig. 1. Schematic view of the considered ECDCSP-M.

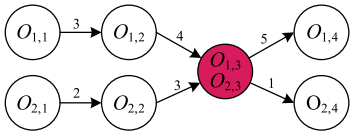


Fig. 2. Example of the precedence constraints in jobs modeled as DG.

each equipped with m machines. Each job can be assigned to any factory for processing. Within each factory, all jobs follow flow-shop permutation constraints, i.e., they must be processed in the same sequence across machines (from $M_{f,1}$ to $M_{f,m}$). Machine setup times are sequence-dependent but independent of processing durations. Upon completion in Phase 1, all n jobs are transported to Phase 2 for subsequent processing.

In Phase 2, there are s production stages, along with all jobs follow a common production route, starting from stage K_1, and proceeding sequentially to K_s. Each stage comprises r_k unrelated parallel machines, including both new and legacy equipment, which differ in processing speeds $v_{k,l}$ and unit-time energy consumption (EC) $\beta^2_{k,l}$. Each job can be scheduled on any of the parallel machines at a given stage. To illustrate the mating operation more effectively, we introduce a directed graph (DG) model. As shown in Fig. 2, the DAG comprises Jobs 1 and 2, in which operations $O_{1,3}$ and $O_{2,3}$ must be executed simultaneously. In addition, the numbers on the edge represent the sequence-dependent setup times.

To maintain generality, it is assumed that each machine can process at most one job at a time, and each job can be assigned to only one machine during processing. Furthermore, job processing is nonpreemptive, meaning that once started, a job cannot be interrupted until completion. All machines are assumed to be available at time 0 and capable of continuously processing jobs without failure. Each machine requires a setup time before processing a job. Consequently, the scheduling problem involves two interrelated subproblems: assignment and sequencing. The assignment subproblem concerns the allocation of jobs to factories in Phase 1 and to parallel machines in Phase 2. The sequencing subproblem involves determining the processing order of jobs within each factory and the operation sequence on parallel machines. The objective is to simultaneously minimize $C_{\max}$ and total EC (TEC) across all jobs.

A. MILP Model of ECDCSP-M

$$\textbf{Objectives: } Min\{C_{\max}, TEC\} \tag{1}$$

$$\textbf{s. t. } x_{0,f} = 1 \quad \forall f \tag{2}$$

$$\sum_{f=1}^{\delta} x_{j,f} = 1 \quad \forall j \notin \{0\} \tag{3}$$

$$z_{j,j',f} \le x_{j,f} \quad \forall j', j \notin \{0\}, j \neq j', f \tag{4}$$

$$z_{j,j',f} \le x_{j',f} \quad \forall j, j' \notin \{0\}, j \neq j', f \tag{5}$$

$$\sum_{j=0, j\neq j'}^{n} z_{j,j',f} = x_{j',f} \quad \forall j', f \tag{6}$$

$$\sum_{j'=0, j\neq j'}^{n} z_{j,j',f} = x_{j,f} \quad \forall j, f \tag{7}$$

$$\sum_{J_j \in \Psi} \sum_{J_{j'} \in \Psi} z_{j,j',f} \le |\Psi| - 1 \quad \forall \Psi \subseteq J, j \neq j' \tag{8}$$

$$d^1_{j,i} \ge 0 \quad \forall i, j \tag{9}$$

$$d^1_{j,i} \ge d^1_{j,i-1} + p^1_{j,i} \quad \forall i \notin \{0\}, j \notin \{0\} \tag{10}$$

$$d^1_{j',i} \ge d^1_{j,i} + p^1_{j',i} + \mathrm{st}^1_{j,j',i} + \left(z_{j,j',f} - 1\right) \cdot h \quad \forall i \notin \{0\}, j \notin \{0\}, j' \notin \{0\}, j \neq j', f \tag{11}$$

$$d^1_{j,i} \ge \mathrm{st}^1_{0,j,i} + p^1_{j,i} + \left(z_{0,j,f} - 1\right) \cdot h \quad \forall i, j, f \tag{12}$$

$$\sum_{l=1}^{r_k} y_{j,l,k} = 1 \quad \forall j, k \tag{13}$$

$$y_{0,l,k} = 1 \quad \forall l, k \tag{14}$$

$$w_{j,j',l,k} \le y_{j,l,k} \quad \forall j', j \notin \{0\}, j \neq j', k \notin \{0\}, l \tag{15}$$

$$w_{j,j',l,k} \le y_{j',l,k} \quad \forall j, j' \notin \{0\}, j \neq j', k \notin \{0\}, l \tag{16}$$

$$\sum_{j=0, j\neq j'}^{n} w_{j,j',l,k} = y_{j',l,k} \quad \forall j', k \notin \{0\}, l \tag{17}$$

$$\sum_{j'=0, j\neq j'}^{n} w_{j,j',l,k} = y_{j,l,k} \quad \forall j, k \notin \{0\}, l \tag{18}$$

$$\sum_{J_j \in \Psi} \sum_{J_{j'} \in \Psi} w_{j,j',l,k} \le |\Psi| - 1 \quad \forall \Psi \subseteq J, j, j' \notin \{0\}, j \neq j', k \notin \{0\}, l \tag{19}$$

$$d^2_{j,k} \ge 0 \quad \forall j, k \tag{20}$$

$$d^2_{j,0} \ge d^1_{j,m} + tp_{j,f} + \left(x_{j,f} - 1\right) \cdot h \quad \forall j \notin \{0\}, f \tag{21}$$

$$d^2_{j,k} \ge d^2_{j,k-1} + \frac{p^2_{j,k}}{v_{k,l}} + \left(y_{j,l,k} - 1\right) \cdot h \quad \forall j \notin \{0\}, k \notin \{0\}, l \tag{22}$$

$$t_j \ge \begin{cases} d^2_{j,k-1} + \frac{p^2_{j',k}}{v_{k,l}} + \frac{p^2_{j,k}}{v_{k,l}} + \left(y_{j,l,k} - 1\right) \cdot h \\ d^2_{j',k-1} + \frac{p^2_{j',k}}{v_{k,l}} + \frac{p^2_{j,k}}{v_{k,l}} + \left(y_{j',l,k} - 1\right) \cdot h \end{cases} \quad \forall O_{j,k}, O_{j',k} \in Y_{j,j',k}, l \tag{23}$$

$$t_j \le \begin{cases} d^2_{j,k-1} + \frac{p^2_{j',k}}{v_{k,l}} + \frac{p^2_{j,k}}{v_{k,l}} + \left(1 - y_{j,l,k}\right) \cdot h + \left(1 - \lambda_j\right) \cdot M \\ d^2_{j',k-1} + \frac{p^2_{j',k}}{v_{k,l}} + \frac{p^2_{j,k}}{v_{k,l}} + \left(1 - y_{j',l,k}\right) \cdot h + \left(1 - \lambda_{j'}\right) \cdot M \end{cases} \quad \forall O_{j,k}, O_{j',k} \in Y_{j,j',k}, l \tag{24}$$

$$\lambda_j + \lambda_{j'} \ge 1 \quad \forall O_{j,k}, O_{j',k} \in Y_{j,j',k}, l \tag{25}$$

$$d^2_{j,k} \ge t_j \quad \forall O_{j,k} \in Y_{j,j',k} \tag{26}$$

$$d^2_{j',k} \ge t_j \quad \forall O_{j',k} \in Y_{j,j',k} \tag{27}$$

$$d^2_{j',k} \ge d^2_{j,k} + \frac{p^2_{j',k}}{v_{k,l}} + \mathrm{st}^2_{j,j',l,k} + \left(w_{j,j',l,k} - 1\right) \cdot h \quad \forall j, j' \notin \{0\}, j \neq j', k \notin \{0\}, l \tag{28}$$

$$\mathrm{PEC}^1_{j,f,i} \ge \beta^1_i \cdot p^1_{j,i} + \left(x_{j,f} - 1\right) \cdot h \quad \forall j, f, i \tag{29}$$

$$\mathrm{SEC}^1_{j,f,i} \ge \gamma \cdot \mathrm{st}^1_{j,j',i} + \left(z_{j,j',f} - 1\right) \cdot h \quad \forall j, j' \notin \{0\}, j \neq j', f, i \tag{30}$$

$$\mathrm{IEC}^1_{j,f,i} \ge \theta \cdot \left(d^1_{j,i-1} - d^1_{j',i} - \mathrm{st}^1_{j,j',i}\right) + \left(z_{j,j',f} - 1\right) \cdot h \quad \forall j, j' \notin \{0\}, j \neq j', f, i \tag{31}$$

$$\mathrm{PEC}^2_{j,l,k} \ge \beta^2_{k,l} \cdot \frac{p^2_{j,k}}{v_{k,l}} + \left(y_{j,l,k} - 1\right) \cdot h \quad \forall j, l, k \tag{32}$$

$$\mathrm{SEC}^2_{j,l,k} \ge \gamma \cdot \mathrm{st}^2_{j,j',l,k} + \left(w_{j,j',l,k} - 1\right) \cdot h \quad \forall j, j' \notin \{0\}, j \neq j', l, k \tag{33}$$

$$\mathrm{IEC}^2_{j,l,k} \ge \theta \cdot \left(d^2_{j',k-1} - d^2_{j,k} - \mathrm{st}^2_{j,j',l,k}\right) + \left(w_{j,j',l,k} - 1\right) \cdot h \quad \forall j, j' \notin \{0\}, j \neq j', l, k \tag{34}$$

$$\mathrm{REC}_{j,l} \ge \mu \cdot tp_{f,l} + \left(x_{j,f} + y_{j,l,1} - 2\right) \cdot h \quad \forall j, l \tag{35}$$

$$\mathrm{TEC} \ge \sum_{f=1}^{\delta} \sum_{i=1}^{m} \sum_{j=0}^{n} \left(\mathrm{PEC}^1_{j,f,i} + \mathrm{SEC}^1_{j',f,i} + \mathrm{IEC}^1_{j,f,i}\right) + \sum_{k=1}^{s} \sum_{l=1}^{r_k} \sum_{j=0}^{n} \left(\mathrm{PEC}^2_{j,l,k} + \mathrm{SEC}^2_{j,l,k} + \mathrm{IEC}^2_{j,l,k}\right) + \sum_{l=1}^{r_1} \sum_{j=1}^{n} \mathrm{REC}_{j,l} \tag{36}$$

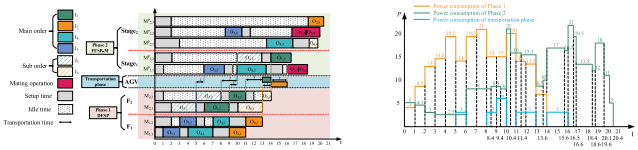


Fig. 3. Gantt chart and power consumption curves of a schedule.

$$C_{\max} \ge d_{j,s} \quad \forall j. \tag{37}$$

Refer to Section B of the Supplementary Materials for detailed definitions of all symbols and descriptions of the mixed-integer linear programming (MILP) model. The code of the MILP model is published on GitHub (https://github.com/banian2314/Model-of-ECDCSP-M.git).

B. Numerical Example

A numerical example with $n = 6$, $\delta = 2$, $m = 2$, $s = 2$, $r_1 = 2$, and $r_2 = 3$ is investigated to further clarify the considered problem. Fig. 3 illustrates a feasible schedule along with the corresponding power consumption profiles over time. The detailed process is provided in Section C of Supplementary Material.

IV. Proposed Algorithm

This article proposes a method for solving ECDCSP-M based on sequential decision-making. This approach iteratively performs scheduling actions, assigning each job to a factory in Phase 1 and then allocating each operation to a compatible machine in Phase 2. This iterative process continues until all jobs have been scheduled. The overall workflow of the proposed GDRL framework is illustrated in Fig. 4. In each iteration, the current scheduling state is first transformed into a heterogeneous graph $\mathcal{G}^H$ (Section IV-B). Next, a heterogeneous GNN (HGNN) applies a three-stage embedding procedure to extract feature representations for factories, operations, and machines (Section IV-C). These representations are then utilized by the decision-making network (Section IV-D). Finally, an action is selected by sampling from the learned probability distribution (Section IV-E).

A. MDP Formulation

In the proposed GDRL framework, the ECDCSP-M is formulated as an MDP. At each timestep t (i.e., at time 0 or immediately after the completion of any operation), the agent observes the current system state s_t and selects an action a_t. The action corresponds to one of two scenarios: assigning an unscheduled job to a factory in Phase 1, or allocating an unscheduled operation to a compatible machine in Phase 2, initiating it at the current time $T(t)$. The environment subsequently transitions to the next decision step $t+1$. This process repeats until all operations have been scheduled.

1) State Representation: This article jointly models the state characteristics of the MDP for ECDCSP-M by constructing a graph structure and using vector representations for the nodes. Prior research that employed GNNs for scheduling

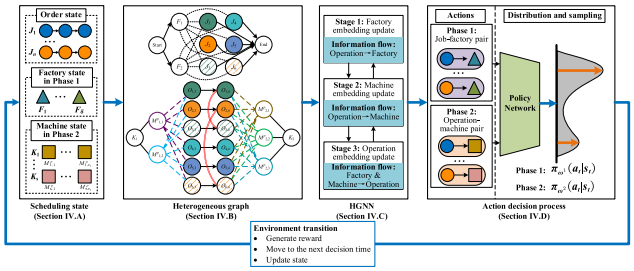


Fig. 4. Procedure of the proposed methodology.

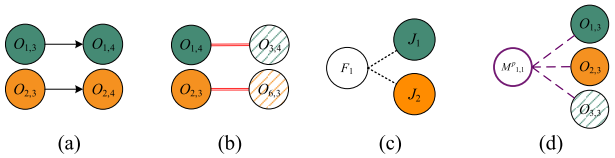


Fig. 5. Four types of arcs. (a) Conjunctive arcs represent operation dependencies. (b) Conjunctive arcs represent mating operations. (c) Disjunctive arcs represent the allocation of factories to jobs. (d) Disjunctive arcs represent the allocation of machines to operations.

problems typically represented MDP states using instance-specific disjunctive graph structures. However, the traditional disjunctive graph structure fails to comprehensively capture the state features due to the unique characteristics of collaborative scheduling and mating operations in ECDCSP-M. To address this, Section IV-B introduced a novel heterogeneous graph $\mathcal{G}^H$, specifically tailored for ECDCSP-M. $\mathcal{G}^H$ comprises a set of factory nodes (Phase 1), machine nodes (Phase 2), operation nodes, and four types of arcs, as illustrated in Fig. 5. For a detailed description of the nodes and arcs in $\mathcal{G}^H$, please refer to the GitHub link in Section III-B.

2) Action Definition: In ECDCSP-M, jobs are assigned to appropriate factories and sequenced in Phase 1, while operations are allocated to compatible machines and sequenced in Phase 2. This article proposed an integrated approach to solving ECDCSP-M. Specifically, we define job-factory pairs (J_j, F_f) and operation-machine pairs $(O_{j,k}, M^p_{k,l})$ as scheduling actions for Phase 1 and Phase 2, respectively, thereby enabling both factory assignment and machine allocation. Furthermore, the processing sequence of operations within the factories and on machines is implicitly determined by the order in which they are assigned, thereby addressing the operation sequencing problem.

3) State Transition: In Phase 1, owing to the characteristics of the DFSP, all machines within the same factory share an identical processing sequence. As a result, a decision step for action selection is triggered whenever the first machine in any factory completes an operation. In Phase 2, a decision step is triggered each time a machine completes an operation. Notably, if multiple machines complete operations simultaneously, the agent concurrently performs multiple scheduling decisions. Following each decision-making process, the decision step advances from t to $t+1$, and the environment transitions to a new state s_{t+1}, which is determined by the current state s_t and the action a_t taken by the agent.

4) Reward: We define the reward $r(s_t, a_t, s_{t+1})$ in the GDRL based on the weight quality difference between the solutions corresponding to states s_t and s_{t+1}, as shown in

$$r(s_t, a_t, s_{t+1}) = \frac{w(C_{\max}(s_t) - C_{\max}(s_{t+1}))}{C_{\max}(s_0)} + \frac{(1-w)(TEC(s_t) - TEC(s_{t+1}))}{\mathrm{TEC}(s_0)}. \quad (38)$$

Due to the significant disparity in the magnitudes of the involved indicators, we normalize the scheduling objective values using the values at the initial time step $t = 0$. Notably, since the state s_t may include unscheduled operations, the values of $C_{\max}(\cdot)$ and $\mathrm{TEC}(\cdot)$ are estimations, i.e., $C_{\max}(s_t) = \max_{\forall j} \overline{d}^2_{j,s}$ and $\mathrm{TEC}(s_t) = \sum_{j=1}^{n}\sum_{i=1}^{m} p^1_{j,i}\cdot\beta^1_i + \sum_{j=1}^{n}\sum_{l=1}^{r_k}\overline{\mathrm{REC}}_{j,l} + \sum_{j=1}^{n}\sum_{k=1}^{s}\overline{\mathrm{PEC}}^2_{j,k}$. If operation $O_{j,k}$ has been scheduled, then $\overline{d}^2_{j,k} = d^2_{j,k}$; otherwise, the estimated completion time for unscheduled operations is recursively computed as $\overline{d}^2_{j,k} = \overline{d}^2_{j,k-1} + [\sum_{l=1}^{r_k}(p^2_{j,k}/v_{k,l})]/r_k$, with the base case $\overline{d}^2_{j,0} = \sum_{i=1}^{m} p^1_{j,i} + \sum_{f=1}^{\delta} tp_{j,f}/\delta$, assuming without loss of generality. For the REC estimation in Phase 1, if $O_{j,1}$ has been scheduled, we set $\overline{\mathrm{REC}}_{j,l} = \mathrm{REC}_{j,l}$; otherwise, we estimate it as $\overline{\mathrm{REC}}_{j,l} = (\sum_{f=1}^{\delta}\sum_{l=1}^{r_k}\mu\cdot tp_{f,l})/\delta\cdot r_k$. In Phase 2, if $O_{j,k}$ has been scheduled, $\overline{\mathrm{PEC}}^2_{j,k} = \mathrm{PEC}^2_{j,l,k}$; otherwise, $\overline{\mathrm{PEC}}^2_{j,k} = \sum_{l=1}^{r_k}\beta^2_{k,l}\cdot(p^2_{j,k}/v_{k,l})/r_k$. When the discount factor $\gamma = 1$, the cumulative reward G can be calculated using

$$\begin{aligned} G &= \sum_{t=0}^{|O|} r(s_t, a_t, s_{t+1}) \\ &= \frac{w\cdot(C_{\max}(s_0) - C_{\max}(s_1) + C_{\max}(s_1) - C_{\max}(s_2) + \cdots + C_{\max}(s_{|O|-1}) - C_{\max}(s_{|O|}))}{C_{\max}(s_0)} \\ &+ \frac{(1-w)\cdot(\mathrm{TEC}(s_0) - TEC(s_1) + TEC(s_1) - TEC(s_2) + \cdots + TEC(s_{|O|-1}) - TEC(s_{|O|}))}{\mathrm{TEC}(s_0)} \\ &= \frac{w\cdot(C_{\max}(s_0) - C_{\max}(s_{|O|}))}{C_{\max}(s_0)} + \frac{(1-w)\cdot(\mathrm{TEC}(s_0) - TEC(s_{|O|}))}{\mathrm{TEC}(s_0)} \\ &= \frac{w\cdot(C_{\max}(s_0) - C_{\max})}{C_{\max}(s_0)} + \frac{(1-w)\cdot(\mathrm{TEC}(s_0) - TEC)}{\mathrm{TEC}(s_0)} \end{aligned} \quad (39)$$

where $C_{\max}(s_0)$ and $\mathrm{TEC}(s_0)$ are constants representing the estimated values of $C_{\max}$ and TEC at the initial state s_0. Therefore, maximizing the cumulative reward G is equivalent to simultaneously minimizing both $C_{\max}$ and TEC.

5) Policy: In Section IV-C, the GDRL framework represents the stochastic policy $\pi(a_t|s_t)$ using an HGNN parameterized by ω, denoted as $\pi_\omega(a_t|s_t)$. This design enables the learning of effective dispatching rules and supports generalization across different problem sizes.

B. Heterogeneous Graph

Previous studies [21], [33], [34], [35], [37], [38], [39], [40] have primarily focused on the design of disjunctive graphs for single-shop scheduling problems, aiming to represent the system state. To date, no dedicated graph structures have been proposed for the cascaded dual-shop collaborative scheduling problem. Traditional disjunctive graphs for nonflexible problems, such as job-shop scheduling problems (JSPs) and flow-shop scheduling problems (FSPs), are typically formulated as $\mathcal{G} = (\mathcal{O}, \mathcal{C}, \mathcal{D})$, where $\mathcal{O}$ represents operation nodes, $\mathcal{C}$ denotes directed arcs between operations of the same job,

and $\mathcal{D}$ indicates machine-operation relationships. However, in flexible manufacturing environments, each operation can be assigned to multiple compatible machines, which significantly increases the density of conventional disjunctive graphs. To address this issue, researchers have developed an alternative disjunctive graph $\mathcal{H} = (\mathcal{O}, \mathcal{M}, \mathcal{C}, \mathcal{E})$ for flexible JSPs [21] and HFSPs [39]. In this structure, $\mathcal{O}$ and $\mathcal{C}$ are defined as in the traditional model, while $\mathcal{M}$ denotes machine nodes, and $\mathcal{E}$ includes arcs connecting compatible machines to corresponding operations. The introduction of machine nodes effectively alleviates the excessive graph density observed in traditional disjunctive graphs for flexible scheduling problems. Despite these advancements, both graph structures encounter the following limitations when applied to ECDCSP-M. First, factory-level information is not represented in Phase 1. Second, Phase 2 cannot represent the production constraints imposed by mating operations. Third, in Phase 2, each operation can be assigned to multiple machines with heterogeneous performance. This results in variations in processing time and EC depending on the selected machine. Fourth, it remains difficult to represent the coordination between the two cascading shop schedules.

To address these issues, this article proposes a novel disjunctive graph $\mathcal{G}^H = (\mathcal{F}, \mathcal{M}, \mathcal{O}, \mathcal{C}, \mathcal{H}, \mathcal{K}, \mathcal{E})$, where $\mathcal{F}$ denotes additional factory nodes, $\mathcal{K}$ is the set of arcs connecting jobs and factories in Phase 1, and $\mathcal{H}$ represents arcs for mating operations in Phase 2. The remaining symbols are consistent with those used in previously discussed graphs [21]. As illustrated in Fig. 6, the traditional arcs $\mathcal{D}$ are replaced by the operation-machine pairs (O, M) arcs set $\mathcal{E}$, where each element $E_{j,k,l} \in \mathcal{E}$ is an undirected arc linking an operation node $O_{j,k}$ to a compatible machine node $M^p_{k,l}$ in Phase 2. Furthermore, factory nodes $\mathcal{F}$ and an undirected arc set $\mathcal{K}$, representing job-factory pairs (J, F), are incorporated. Each element $K_{j,f} \in \mathcal{K}$ is an undirected arc linking a job node J_j to a factory node F_f in Phase 1. As the scheduling process unfolds, the arcs in $\mathcal{E}$ and $\mathcal{K}$ are visually distinguished as dashed lines for unscheduled operations and solid lines for scheduled ones. Notably, job nodes J_j are not explicitly represented in $\mathcal{G}^H$. This design choice stems from the constraint that, once the initial operation of a job is assigned to a specific factory in Phase 1, all subsequent operations must be allocated to the same factory.

Compared to traditional disjunctive graphs, the proposed heterogeneous graph offers several notable advantages. First, it leads to a significant reduction in graph density. According to the definition of traditional disjunctive graphs $\mathcal{G} = (\mathcal{O}, \mathcal{C}, \mathcal{D})$, and assuming each factory can process n jobs, the set of disjunctive arcs in Phase 1 is $|\mathcal{D}^1| = \delta \cdot \binom{n}{2} = ([\delta \cdot n \cdot (n-1)]/2)$. In Phase 2, assuming each machine can process n operations, then $|\mathcal{D}^2| = \sum_{k=1}^{s} (r_k \cdot \binom{n}{2}) = \sum_{k=1}^{s} ([r_k \cdot n \cdot (n-1)]/2)$. Ultimately, $|\mathcal{D}| = |\mathcal{D}^1| + |\mathcal{D}^2|$. However, for the heterogeneous graph $\mathcal{G}^H$ designed in this article, $|\mathcal{K}| = \delta \cdot n$ and $|\mathcal{E}| = \sum_{k=1}^{s} r_k \cdot n$. It is easy to prove that $|\mathcal{D}| > |\mathcal{K}| + |\mathcal{E}|$ when $n > 3$. For large-scale problems, $\mathcal{G}^H$ involves significantly fewer $\mathcal{K}$ and $\mathcal{E}$ arcs than the disjunctive arcs $\mathcal{D}$ in $\mathcal{G}$. Second, the inclusion of factory nodes and machine nodes in $\mathcal{G}^H$ enables the embedding of machine-specific information and enable the extraction of valuable features to distinguish between factories and machines within a given state. Moreover, after Phase 1 is completed, transportation time information for transferring from different factories to Phase 2 can be conveniently attached as a feature to the (J, F) arc $K_{j,f}$. Likewise, in Phase 2, processing time and EC can be embedded as features to the (O, M) arc $E_{j,k,l}$.

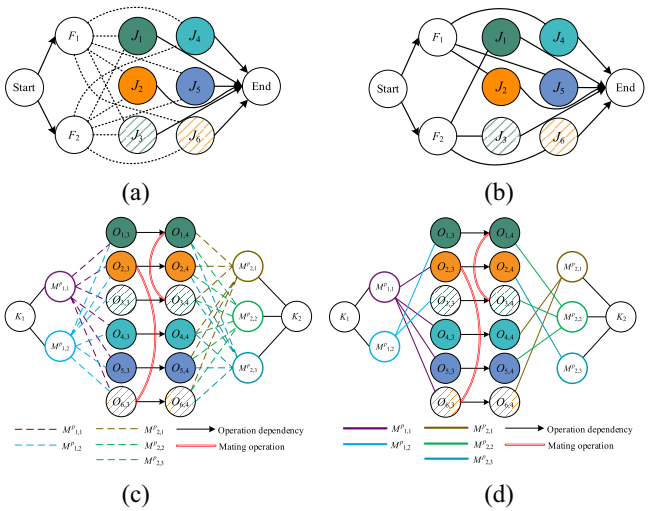


Fig. 6. Heterogeneous graph of ECDCSP-M. The dashed line means processable, while the solid line means scheduled (a) Instance of job assigning in Phase 1 (b) Instance of machine assigning in Phase 2. (c) Solution of job assigning in Phase 1. (d) Solution of machine assigning in Phase 2.

Based on the aforementioned definition of $\mathcal{G}^H$, each state s_t is represented as a heterogeneous graph $\mathcal{G}^H_t = (\mathcal{F}, \mathcal{M}, \mathcal{O}, \mathcal{C}, \mathcal{H}, \mathcal{K}_t, \mathcal{E}_t)$, where $\mathcal{K}_t$ and $\mathcal{E}_t$ evolve dynamically during the solving process. Specifically, when an action a_t is executed at step s_t, only $K_{j,f}$ or $E_{j,k,l}$ is retained, while the remaining job-factory arcs of J_j and operation-machine arcs of $O_{j,k}$ are removed to construct $\mathcal{G}^H_{t+1}$. At each step t, let $\mathcal{N}^t(F_f)$, $\mathcal{N}^t(M^p_{k,l})$, and $\mathcal{N}^t(O_{j,k})$ denote neighboring jobs of the factory F_f, the neighboring operations of machine $M^p_{k,l}$, and the neighboring machines of operation $O_{j,k}$, respectively. For each factory, machine, operation, (J, F) arc, and (O, M) arc, the raw feature vectors $v_f \in \mathbb{R}^2$, $\varpi_{k,l} \in \mathbb{R}^4$, $\mu_{j,k} \in \mathbb{R}^8$, $\lambda_{j,f} \in \mathbb{R}$, and $\xi_{j,k,l} \in \mathbb{R}^2$ are defined to characterize their states at step t. The detailed definitions of these feature vectors are provided in the GitHub repository, accessible via the link in Section III-B.

C. Heterogeneous Graph Neural Network

Given the varying sizes and scales of ECDCSP-M instances, the neural architecture must be adaptable to accommodate state graphs of varying dimensions [39]. Achieving such size-agnostic capability necessitates the use of GNNs. However, most existing studies [38], [40] have focused on homogeneous graphs, which are not well-suited for representing $\mathcal{G}^H$. Although recent studies [21], [34], [39] have started exploring HGNNs, they have yet to consider the unique characteristics of cascaded dual-shop collaborative scheduling. First, the three

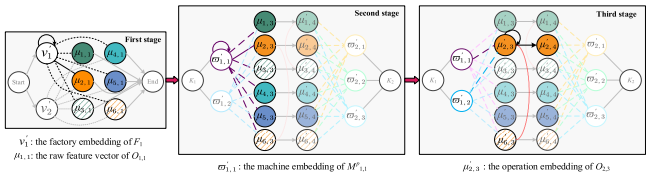


Fig. 7. Three-stage embedding scheme. Update of factory embedding v'_1, machine embedding $\varpi'_{1,1}$, and operation embedding $\mu'_{2,3}$ are highlighted for illustration.

types of nodes in $\mathcal{G}^H$ exhibit distinct connectivity patterns: factory nodes are only connected to jobs via undirected arcs, whereas operation nodes can connect to other operations through directed arcs to machines through undirected arcs. Second, the (J, F) arcs in Phase 1 and the (O, M) arcs in Phase 2 play a crucial role in solving the scheduling problem. Nevertheless, existing HGNNs typically focus on node-level representations while neglecting the rich semantics embedded in arc featuresand.

This article introduces a tailored HGNN architecture for ECDCSP-M, aiming to fully exploit the expressive power of heterogeneous graph structures through efficient encoding in $\mathcal{G}^H$. As illustrated in Fig. 7, the proposed HGNN employs a three-stage embedding process that integrates graph topology with numerical attributes (raw features) to transform the nodes in $\mathcal{G}^H$ into d-dimensional representations. During this process, relevant information is progressively aggregated to refine the embeddings of factories, machines, and operations.

1) Factory Node Embedding: In $\mathcal{G}_t^H$, the neighbors of a factory F_f are defined as the set of associated jobs $\mathcal{N}^t(F_f)$. Owing to the singular type of connection between factory and job nodes (i.e., allocation relationship), multiple MLPs are employed to separately process information from each source, including the features of F_f itself. The outputs are then concatenated and projected back into the d-dimensional space to obtain the embedding of F_f. Accordingly, the raw feature vector of each $O_{j,1} \in \mathcal{N}^t(F_f)$ is augmented by concatenating its intrinsic features with those of the corresponding (J, F) arc, yielding $\zeta_{j,f} = [\mu_{j,1}||\lambda_{j,f}] \in \mathbb{R}^9$. The corresponding embedding computation for F_f at step t is formulated in (40). Here, MLP_{ϑ_0}, MLP_{ϑ_1}, and MLP_{ϑ_2} denote three MLPs, each generating d-dimensional outputs. These MLPs consist of two hidden layers of size d_ϑ, and employ the ELU activation function

$$v'_f = \text{MLP}_{\vartheta_0}\big(\text{ELU}\big[\text{MLP}_{\vartheta_1}(v_f)||\text{MLP}_{\vartheta_2}(\zeta_{j,f})\big]\big). \tag{40}$$

2) Machine Node Embedding: In contrast to factory nodes, machine nodes in Phase 2 have many neighboring nodes—primarily operation nodes—connected exclusively through allocation relationships. In practical scheduling, operations with earlier expected start times are considered more critical than those scheduled later. This consideration motivates the adoption of a graph attention network (GAT) [45], which employs attention mechanisms to automatically learn the relative importance of neighboring nodes. While traditional GATs that ignore edge features in homogeneous graphs have been extensively explored, limited research has addressed the integration of edge features in heterogeneous graphs [35]. To address this gap, this article proposes an attention network tailored for heterogeneous graphs to enhance node embeddings.

As illustrated in Fig. 6, each machine $M_{k,l}^p$ is connected to its neighboring operation via a single operation–machine pair $(O_{j,k}, M_{k,l}^p)$ arc. Thus, the raw feature vector for each $O_{j,k} \in \mathcal{N}^t(M_{k,l}^p)$ is enriched by concatenating its own features with those of the corresponding (O, M) arc, resulting in $\mu_{j,k,l} = [\mu_{j,k}||\xi_{j,k,l}] \in \mathbb{R}^{10}$. Next, two distinct linear transformations are applied: $\mathbf{W^M} \in \mathbb{R}^{d\times 4}$ for machine nodes and $\mathbf{W^O} \in \mathbb{R}^{d\times 10}$ for operation nodes. For each machine $M_{k,l}^p$, the attention coefficient $e_{j,k,l}$, which quantifies the relevance of each neighboring operation $O_{j,k} \in \mathcal{N}^t(M_{k,l}^p)$, is computed as

$$e_{j,k,l} = \text{LeakyReLU}\Big(a^T\Big[\mathbf{W^M}\varpi_{k,l}||\mathbf{W^O}\mu_{j,k,l}\Big]\Big). \tag{41}$$

Self-loops of machine nodes are also incorporated. The corresponding attention coefficient is calculated as follows:

$$e_{k,l,k,l} = \text{LeakyReLU}\Big(a^T\Big[\mathbf{W^M}\varpi_{k,l}||\mathbf{W^M}\varpi_{k,l}\Big]\Big). \tag{42}$$

These coefficients are subsequently normalized via (43) to yield the normalized attention coefficients $\alpha_{j,k,l}$ and $\alpha_{k,l,k,l}$

$$\alpha_{j,k,l} = \frac{\exp\big(e_{j,k,l}\big)}{\sum_{u\in\mathcal{N}\left(M_{k,l}^p\right)}\exp\big(e_{j,k,u}\big)} \quad \forall O_{j,k}\in\mathcal{N}\Big(M_{k,l}^p\Big). \tag{43}$$

Finally, the machine embedding $\varpi'_{k,l}$ is calculated by integrating features from both neighboring operations and the machine node itself. Considering the critical role of processing time and EC, the extended raw feature vector $\mu_{j,k,l}$ is employed for each neighboring operation $O_{j,k}$. The aggregation function used to compute $\varpi'_{k,l}$ is defined as follows:

$$\varpi'_{k,l} = \sigma\left(\alpha_{k,l,k,l}\mathbf{W^M}\varpi_{k,l} + \sum_{O_{j,k}\in\mathcal{N}^t(M_{k,l}^p)}\alpha_{j,k,l}\mathbf{W^O}\mu_{j,k,l}\right). \tag{44}$$

3) Operation Node Embedding: Unlike factory and machine nodes, operation nodes in $\mathcal{G}_t^H$ possess fewer neighbors but exhibit a more diverse range of connection types. For a given operation node $O_{j,k}$, its neighboring nodes include its predecessor operation nodes $PO_{j,k}$, successor operation node $O_{j,k+1}$, the associated machine nodes $\mathcal{M}_{j,k} \in \mathcal{N}^t(O_{j,k})$, and the node itself. Additionally, if $O_{j,k}$ is a mating operation, it may have multiple immediate predecessors. Operation nodes are connected via directed arcs, while links to machine nodes are undirected. Due to the heterogeneity in information sources and arc types, the attention mechanism proves less effective for embedding operation nodes. Consequently, multiple MLPs are employed to separately process information from different node and arc types. The outputs of these MLPs are concatenated and projected into a d-dimensional space to create the final embedding of $O_{j,k}$.

Building on the preceding analysis, the embedding of operation nodes is derived using five distinct MLPs $(\text{MLP}_{\eta_0}, \ldots, \text{MLP}_{\eta_4})$. Each MLP outputs a d-dimensional vector and comprises two hidden layers of width d_η, with the

ELU activation function applied. These MLPs are responsible for processing information from various sources, including the set of predecessor operations $PO_{j,k}$, the successor $O_{j,k+1}$, the associated machine nodes $\mathcal{M}_{j,k} \in \mathcal{N}^t(O_{j,k})$, and the operation node itself $O_{j,k}$. The final embedding of $O_{j,k}$ is obtained as follows:

$$\mu'_{j,k} = \text{MLP}_{\eta_0}(\text{ELU}[\text{MLP}_{\eta_1}(\overline{\mu}_{j,k})||\text{MLP}_{\eta_2}(\mu_{j,k+1})|| \text{MLP}_{\eta_3}(\overline{\varpi}'_{j,k})||\text{MLP}_{\eta_4}(\mu_{j,k})]) \quad (45)$$

$$\overline{\mu}_{j,k} = \sum_{O_{j',k'} \in PO_{j,k}} \mu_{j',k'} \quad (46)$$

$$\overline{\varpi}'_{j,k} = \sum_{\mathcal{M}_{j,k} \in \mathcal{N}^t(O_{j,k})} \varpi'_{k,l}. \quad (47)$$

Since both $PO_{j,k}$ and $\mathcal{N}^t(O_{j,k})$ may contain multiple elements, their contributions are individually aggregated using (46) and (47), respectively.

4) Stacking and Pooling: After obtaining the embedding v'_f, $\varpi'_{k,l}$, and $\mu'_{j,k}$ of F_f, $M^p_{k,l}$, and $O_{j,k}$, respectively, from $\mathcal{G}^H_t$ using the above method, the standard mean pooling method is applied to compute the average of all node embeddings prior concatenation. These three embeddings collectively constitute one layer of the HGNN. To enhance feature extraction capabilities, we stack L HGNN layers, each sharing an identical structure but independently trainable parameters, to derive the final embeddings $v'^{(L)}_f$, $\varpi'^{(L)}_{k,l}$, and $\mu'^{(L)}_{j,k}$. Notably, the raw features v'_f, $\varpi'_{k,l}$, and $\mu'_{j,k}$ are used exclusively in the first layer, while the raw features $\lambda_{j,f}$ of the (J, F) arcs and $\xi_{j,k,l}$ of (O, M) arcs are propagated through all L layers. The resulting three d-dimensional vectors are then concatenated to form the final embedding $h_t \in \mathbb{R}^{3d}$ of the heterogeneous graph state $\mathcal{G}^H_t$ as follows:

$$h_t = \left[\frac{1}{|\mathcal{F}|} \sum_{F_f \in \mathcal{F}} v'^{(L)}_f || \frac{1}{|\mathcal{M}|} \sum_{M^p_{k,l} \in \mathcal{M}} \varpi'^{(L)}_{k,l} || \frac{1}{|\mathcal{O}|} \sum_{O_{j,k} \in \mathcal{O}} \mu'^{(L)}_{j,k} \right]. \quad (48)$$

Through this process, the five original feature vectors of varying dimensions in $\mathcal{G}^H_t$ are ultimately transformed into a fixed-size embedding. Let θ denote the set of parameters in the HGNN.

D. Decision Making

At each decision step t, the policy $\pi(a_t|s_t)$ selects an action a_t conditioned on the current state s_t. As described in Section IV-A, under the MDP framework of ECDCSP-M, an action at time step t corresponds to either a job-factory pair (J_j, F_f) or an operation-machine pair $(O_{j,k}, M^p_{k,l})$. Leveraging the aforementioned heterogeneous graph and HGNN, the extracted embeddings are transformed into a probability distribution over actions, denoted by the policy $\pi_\omega(a_t|s_t)$. Specifically, for each feasible action $a_t \in A_t = (J_j, F_f) \cup (O_{j,k}, M^p_{k,l})$ at time step t, if the system is in Phase 1 of ECDCSP-M, the embeddings of the corresponding job, factory, and current state are concatenated. Otherwise, in Phase 2 of ECFJSP-M, the embeddings of the corresponding operation, machine, and current state are concatenated. The resulting concatenated vectors are then passed through an MLP to compute a priority score for each action at state s_t, as follows:

$$\Pr(a_t|s_t) = \begin{cases} \text{MLP}_{\omega^1}\left[\mu'^{(L)}_{j,1}||v'^{(L)}_f||h_t\right], a_t \in (J_j, F_f) \\ \text{MLP}_{\omega^2}\left[\mu'^{(L)}_{j,k}||\varpi'^{(L)}_{k,l}||h_t\right], a_t \in \left(O_{j,k}, M^p_{k,l}\right). \end{cases} \quad (49)$$

Here, MLP_{ω^1} and MLP_{ω^2} refer to two separate MLPs, each comprising two hidden layers with d_ω neurons and employing the tanh activation function. The softmax function is subsequently applied to derive a probability distribution over the action space at each time step. This probability distribution is then used to sample actions during the training process, as described in

$$\pi_\omega(a_t|s_t) = \frac{\exp(Pr(a_t|s_t))}{\sum_{a'_t \in A_t} \exp\left(Pr(a'_t|s_t)\right)} \quad (50)$$

$$a_t \leftarrow \text{argmax}_{\forall a'_t \in A_t} \pi_\omega\left(a'_t|s_t\right). \quad (51)$$

During training, actions are sampled from the policy $\pi_\omega(a_t|s_t)$ to enhance the exploration capability of the algorithm. In the testing phase, actions are selected greedily by choosing those with the highest probabilities, as defined by (51). The inherent randomness in the sampling strategy causes variability in the results across different runs, thereby increasing the opportunity to identify superior solutions. Additionally, leveraging the parallel computation capabilities of GPUs enables multiple simultaneous runs on specific test instances. In each parallel run, actions are sampled probabilistically, resulting in multiple candidate solutions. Ultimately, the best scheduling solution among these candidates is selected as the final output.

E. Learning Algorithm

Considering the continuous state space and the high-dimensional features resulting from the large-scale nature of ECDCSP-M, policy-based methods are more suitable for DRL algorithms than value-based approaches. Therefore, we adopt the PPO algorithm [46], which is renowned for its robust performance. PPO employs a policy gradient approach within an actor–critic framework to effectively manage the training process. It consists of both actor and critic networks, enabling efficient parameters updates after each timestep, which accelerates convergence and improves solution speed [21].

The actor and critic are represented by the policy network π_ω and the value network υ_ϕ, which, respectively, model the action-selection policy and the value estimation of the state s_t. Specifically, π_ω is implemented as an MLP, as described in (50), and is responsible for learning the stochastic policy $\pi_\omega(a_t|s_t)$, which defines a probability distribution over possible actions. Similarly, υ_ϕ is designed as an MLP, as shown in (52), which takes the state embedding h_t generated by the HGNN as input to estimate the state value function $\upsilon_\phi(s_t)$. MLP_ϕ shares the same architecture as MLP_{ω^1} and MLP_{ω^2}, consisting of two hidden layers with d_ϕ neurons and employing the tanh activation function, but it is parameterized by a different set of weights ϕ

$$\upsilon_\phi(s_t) = \text{MLP}_\phi(h_t). \quad (52)$$

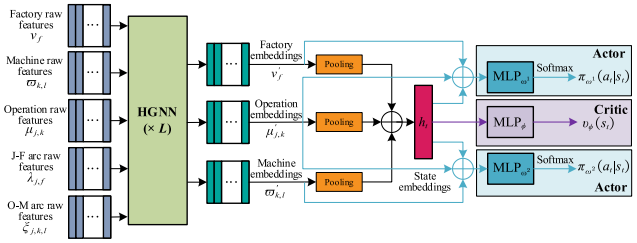


Fig. 8. Network architecture.

The critic network serves as an advantage function estimator and is primarily responsible for computing the state value function $\upsilon_\phi(s_t)$. It subsequently computes an estimator $\widehat{A}_t$ of the variance-reduced advantage function based on (53), where γ is the discount factor, t' indicates the current step, and $r_{t'}$ is the corresponding reward

$$\widehat{A}_t = -\upsilon_\phi(s_{t'}) + \sum_{t=t'}^{T} \gamma^{t'} r_{t'}. \tag{53}$$

This article adopts the widely used PPO-clip algorithm, which constrains policy updates through a clipping mechanism. The corresponding loss function is expressed as follows:

$$L_{\omega,\phi} = c_p L^{\mathrm{CLIP}}(\omega) - c_v L^{\mathrm{VF}}(\phi) + c_e L^{S}(\omega) \tag{54}$$

$$L^{\mathrm{CLIP}}(\omega) = \min(r_t(\omega), clip(r_t(\omega), 1-\epsilon, 1+\epsilon))\widehat{A}_t, \quad r_t(\omega) = \frac{\pi_\omega(a_t|s_t)}{\pi_{\omega\mathrm{old}}(a_t|s_t)} \tag{55}$$

$$L^{\mathrm{VF}}(\phi) = \left(\upsilon_\phi(s_t) - \upsilon_t\widehat{A}_t\right)^2 \tag{56}$$

$$L^{S}(\omega) = \sum_{t=0}^{T} S(\pi_\omega(a_t|s_t)) \tag{57}$$

where c_p, c_v, and c_e denote the coefficient for policy loss, value function loss, and entropy loss, respectively. The terms clip and ϵ refer to the clipping function and clipping ratio, respectively.

The overall network architecture is shown in Fig. 8, while the entire training procedure is outlined in Algorithm 1. The training process spans T iterations, each comprising B rollout episodes (lines 4–17) and R update epochs (lines 18-22). To ensure sample diversity and reduce the risk of overfitting, a batch of B instances is solved in parallel by the DRL agent during training, and this batch is refreshed every 20 iterations. Moreover, to facilitate algorithm convergence, the policy is evaluated on an independent validation set every ten iterations (Lines 23–25).

Algorithm 1: Training Procedure for the HGNN Scheduler

Input: Problem size: $(n, m, \delta, s, r_k, J_M, J_S, \Psi, v_{k,l})$;
Distribution: $p^1_{j,i}, p^2_{j,k} \sim U(low_1, high_1)$;
$st^1_{j,j',i}, st^2_{j,j',l,k} \sim U(low_2, high_2)$; $tp_{j,f} \sim U(low_3, high_3)$;
Network: $\theta, \omega^1, \omega^2, \phi$;
Output: optimized policy network; optimized critic network;

1 Generate ECFJSP − M instances;
2 Sample a batch of B ECFJSP-M instances;
3 **for** *iter* $= 1$ *to* T **do**
4 **for** episode $b = 1$ *to* B **do**
5 Initialize state s_t based on instance b;
6 **while** the episode b is not terminated **do**
7 Extract embeddings ($v'^{(L)}_f, \varpi'^{(L)}_{k,l}, \mu'^{(L)}_{j,k}, h_t$) by HGNN;
8 **if** the timestep t is in Phase 1 **then** // $a_t \in (J_j, F_f)$
9 .
10 **else** // the timestep t is in Phase 2, $a_t \in (O_{j,k}, M^p_{k,l})$
11 Sample $a_t \sim \pi_{\omega^1}(\cdot \mid s_t)$;
12 **end if**
13 Sample $a_t \sim \pi_{\omega^2}(\cdot \mid s_t)$;
14 Calculate reward r_t and next state s_{t+1};
15 State transition;
16 **end while**
17 Calculate the advantages estimator $\widehat{A}_t$ by Eq. (53);
18 **end for**
19 **for** *ep* $= 1$ *to* R **do**
20 Calculate the loss $L_{\omega,\phi}$ by Eq. (54);
21 Optimize the parameters θ, ω, and ϕ;
22 Update all parameters;
23 **end for**
24 **if** *iter* mod $10 = 0$ **then** // $a_t \in (J_j, F_f)$
25 Validate the policy;
26 **end if**
27 **if** *iter* mod $20 = 0$ **then** // $a_t \in (J_j, F_f)$
28 Break;
29 **end if**
30 **end for**
31 **Return**

V. Experiments and Analysis

A. Experimental Setup

1) Training and Testing Instances: The GDRL is trained and evaluated on synthetic instances of varying scales. The instance generation process follows a methodology similar to that employed in the referenced study. The parameters used for generating these instances are specified as follows, $n \in \{15, 20, 40, 50\}$, $\delta \in \{2, 3, 4, 5\}$, $m \in \{5, 10\}$, $s \in \{5, 10\}$, $r_k \sim U[2, 5]$, $p^1_{j,i} \sim U[10, 30]$, $p^2_{j,k} \sim U[10, 50]$, $\mathrm{st}^1_{j,j',i} \sim U[10, 20]$ $\mathrm{st}^2_{j,j',l,k} \sim U[10, 20]$, $tp_{j,f} \sim U[2, 5]$, $\beta^1_i \sim U[4, 8]$, $\beta^1_{k,l} \sim U[5, 10]$ $v_{k,l} \in \{1, 1.1, 1.2, 1.3, 1.4\}$, $\gamma = 1, \theta = 0.5, \mu = 3$, $|JM| = 0.75n$, and $|JS| = 0.25n$, where U denotes the uniform distribution, and all other symbols are thoroughly defined in Section III-A. Notably, 64 groups of instances are generated, covering all combinations of jobs, factories, machines, and stages. Each group consists 40 individual instances.

The proposed GDRL, initially trained on smaller-scale instances ($n = 15$), is subsequently evaluated on larger, previously unseen synthetic instances ($n = 20, 40, 50$) to assess its generalization capability across varying problem sizes. The training data is generated dynamically and includes 40 validation instances. During testing, 40 instances are sampled for each problem size to evaluate the performance.

2) Hyperparameters Settings: In this article, the number of HGNN iterations is set to $L = 2$, and the embedding dimensions for factory, machine, and operation nodes are set to $d = 128$. The hidden dimensions for the MLPs are set to $d_\vartheta = d_\eta = 128$ and $d_\omega = d_\phi = 64$. The training process involves $T = 1000$ iterations with a batch size $B = 20$. An ablation analysis was conducted to evaluate the impact

of different reward function weights. The detailed results and discussions are provided in the Supplementary Material. Based on the experimental findings, the reward function weight is defined as $w = 0.8$. The coefficients for policy loss (with a clipping ratio of $\epsilon = 0.2$), value loss, and entropy term are set to $c_p = 1$, $c_v = 0.5$, and $c_e = 0.01$, respectively. The PPO algorithm is performed over $R = 3$ epochs using a discount factor $\gamma = 1.0$. The network parameters are optimized using the Adam optimizer with a learning rate $l_r = 2 \times 10^{-4}$.

The proposed method is implemented in PyTorch. The PC hardware configuration includes a machine with Windows 11 64-bit OS, Intel Xeon Silver 4210R CPU @2.40GHz, 128 GBytes of RAM, and NVIDIA GeForce GTX TITAN X GPU.

3) Baselines: As the ECDCSP-M problem remains largely unexplored, as discussed in Section II, we have selected a diverse set of baseline algorithms for comparison, encompassing DRL-based and RL-based approaches, metaheuristics, and multiobjective optimization methods:

1) *DRL-Based Method:* DRL-FJSP [21].
2) *RL-Based Method:* population-based iterative greedy algorithm (PBIGA) [47].
3) *Metaheuristic Algorithms:* hybrid scheduling algorithm (HAS) [19], multineighborhood-based multiobjective memetic algorithm (MMMA) [10].
4) *Multiobjective Optimization Method:* NSGA-II [18].

The reasons for choosing these algorithms is as follows: HAS and MMMA are recent and efficient methods for solving HFSPs; PBIGA is a high-performance algorithm for distributed scheduling; NSGA-II is a representative nondominated sorting-based multiobjective optimization method, with improved versions that have been successfully applied to HFSPs with notable results. For fairness in comparison, the termination criteria for HAS, MMMA, NSGA-II, and PBIGA were all set to a maximum elapsed CPU time ($c \times n \times m \times s$) milliseconds, with c is set to 20 [48]. Additionally, we include a recent end-to-end DRL approach [21] specifically designed for FJSP. To account for the stochastic nature of these algorithms, each baseline is executed 30 times.

4) Performance Evaluation Metrics: Convergence and diversity are essential metrics for evaluating multiobjective optimization methods. Hypervolume (HV) [49] metric is a comprehensive indicator that simultaneously reflects both aspects. To further clarify the performance of GDRL, we introduce the Gap metric, which measures the relative difference in HV compared to GDRL. A positive Gap value indicates that the corresponding method outperforms GDRL, whereas a negative value implies inferior performance

$$\text{Gap} = \frac{HV(\text{other}) - HV(\text{GDRL})}{HV(\text{GDRL})}. \tag{58}$$

B. Performance on Training

1) The Convergence Performance Analysis of GDRL: To analyze the convergence of GDRL, we present the training results for instances with $= 15, \delta = 3, m = 5, s = 5, |JM| = 11$, and $|JS| = 4$. Fig. S.9 in the Supplementary Material shows the average reward and loss obtained from parallel cases at each iteration. By examining the smoothed curve generated using the Savitzky–Golay method [34], it is clear that the average reward value steadily increases as training progresses. This trend indicates that the model effectively benefits from the proposed reward function. During the rollout phase (lines 4-17 in Algorithm 1), the agent samples actions based on the policy's output probabilities, accounting for the observed variance in rewards. Additionally, as the number of iterations increases, the training loss consistently decreases, suggesting that the model is progressively learning to select appropriate scheduling actions across diverse scenarios.

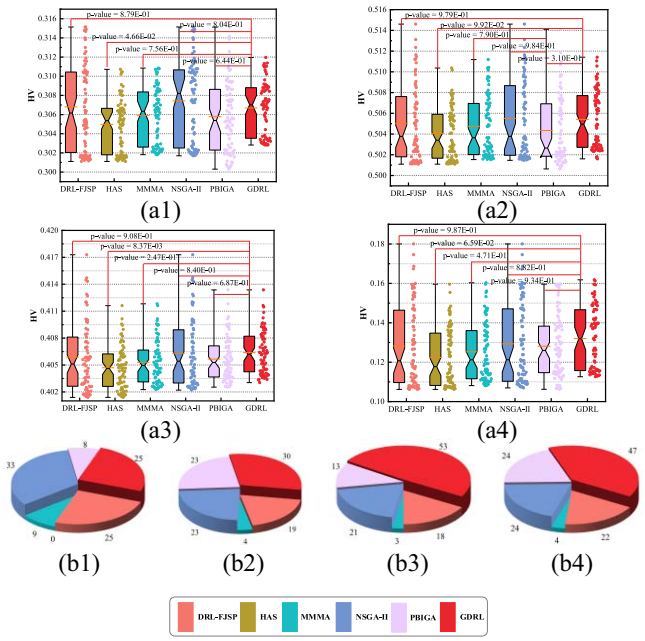


Fig. 9. Result of training size and peer comparison, (a) Boxplots of HV; (b) Win rate %. (a.1) $n = 15, m = 5, s = 5$ (a.2) $n = 15, m = 5, s = 10$ (a.3) $n = 15, m = 10, s = 5$ (a.4) $n = 15, m = 10, s = 10$ (b.1) $n = 15, m = 5, s = 5$ (b.2) $n = 15, m = 5, s = 10$ (b.3) $n = 15, m = 10, s = 5$ (b.4) $= 15, m = 10, s = 10$.

2) Evaluation of Instances of Training Size: The average HV and Gap values across all instances for each training size are reported in Table S.V of the Supplementary Material. Among the compared methods, NSGA-II achieved the highest HV values for instances with $m = 5, s = 5$ and $m = 5, \ s = 10$. However, for instances with $m = 10, s = 5$ and $= 10, s = 10$, the proposed GDRL attained the highest HV values. This pattern may be attributed to the fact that NSGA-II can continually update the Pareto front in very small-scale instances. As the problem size grows, particularly due to the increase in the number of operations in Phase 1 of ECDCSP-M, NSGA-II requires more CPU time to compute nondominated solutions. Furthermore, GDRL outperforms three recently published baselines—HAS, MMMA, and PBIGA—across all scenarios. This superiority likely stems from the fact that these metaheuristic algorithms do not fully consider the unique characteristics of ECDCSP-M, especially the constraints related to mating operations.

Another deep learning approach, DRL-FJSP, demonstrates performance close to that of GDRL across all scenarios, with

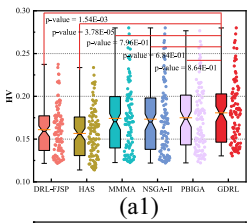
comparable computational time. However, as the problem size increases, the performance of DRL-FJSP gradually deteriorates. This decline is likely due to the significant influence of the environment on DRL-based methods. The lack of environmental information in DRL-FJSP for ECDCSP-M often results in "short-sighted" actions. Furthermore, as shown in Fig. 9, the pie chart provides a clear visual representation indicating that GDRL achieves a higher number of maximum HV values compared to other algorithms, except in the scenario where $m = 5$ and $s = 5$. The win rate denotes the percentage of runs in which an algorithm attains the highest HV value. The Tukey HSD test confirms that the HV values of GDRL are statistically comparable to those of other methods across all training scales.

C. Comparison of Other Algorithms

Next, we conducted a more in-depth analysis of the proposed method's capability to handle larger instances. Specifically, we applied the strategy learned from instances with $n = 15$, $m = 10$, and $s = 10$ directly to instances with $n = 20$, 40, and 50.

1) Generalization Performance on Large-Sized Instances: Table S.VI of the Supplementary Material presents the HV values, Gap metrics, and run times obtained by GDRL and five comparison algorithms across different instance groups. The results reveal that GDRL consistently achieves the highest HV values across all large-scale instances, leading to the largest corresponding Gap values. Furthermore, except for the time required by DRL-FJSP on the (20,5,5) and (20,5,10) groups, the proposed algorithm's computation time is only slightly higher than that of DRL-FJSP in other instance groups. A detailed comparison and analysis of the runtime between these two DRL-based methods is provided in Section V-C. Moreover, the advantage of GDRL becomes more pronounced with an increasing number of jobs. To further investigate this trend, a one-way ANOVA was conducted across different job sizes, followed by Tukey HSD tests, with the results illustrated in Fig. 10. Fig. 10 shows that the median value of GDRL (represented by the red solid line in the box plot) is consistently higher than those of the other five algorithms, indicating that the Pareto front identified by GDRL generally dominates the others. The mean HV value of GDRL (represented by the black solid line in the box plot) is also the highest, demonstrating both optimal performance and stability. Compared to the other four baseline algorithms (excluding PBIGA), the p-values obtained for the instance groups with $n = 40$ and $n = 50$ are all less than 0.05, indicating that GDRL achieves statistically significant performance improvements. Although GDRL does not exhibit a statistically significant advantage over PBIGA, as shown in Fig. 10(a), it still outperforms PBIGA in terms of solution quality.

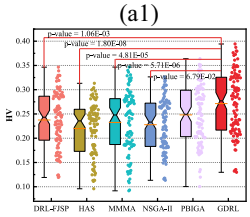
Furthermore, we conducted pairwise comparisons of all algorithms to assess the presence of significant differences among them. As shown in Fig. 10(b), the Tukey HSD test identified several homogeneous subsets, suggesting that the means within each subset do not differ significantly from one another, as indicated by p-values greater than 0.05. The



(a1)

Algorithms	Subsets $\alpha = 0.05$		
	1	2	3
HAS	0.1566	/	/
DRL-FJSP	0.1610	0.1610	/
NSGA-II	/	0.1733	0.1733
MMMA	/	0.1742	0.1742
PBIGA	/	0.1749	0.1749
GDRL	/	/	0.1807
p-value	9.57E-01	6.72E-02	6.84E-01

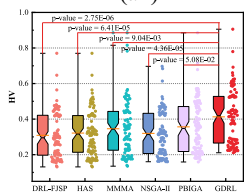
(b1)



(a2)

Algorithms	Subsets $\alpha = 0.05$		
	1	2	3
HAS	0.2190	/	/
NSGA-II	0.2279	0.2279	/
MMMA	0.2316	0.2316	/
DRL-FJSP	0.2376	0.2376	/
PBIGA	/	0.2481	0.2481
GDRL	/	/	0.2714
p-value	2.43E-01	1.62E-01	6.79E-02

(b2)



(a3)

Algorithms	Subsets $\alpha = 0.05$	
	1	2
DRL-FJSP	0.3089	/
NSGA-II	0.3199	/
HAS	0.3215	/
MMMA	0.3458	/
PBIGA	0.3565	0.3565
GDRL	/	0.4126
p-value	1.50E-01	5.08E-02

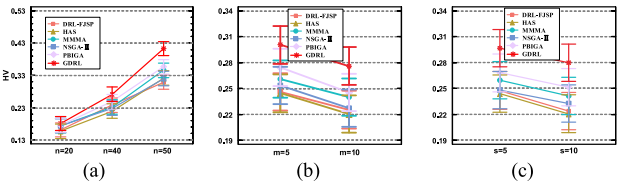
(b3)

Fig. 10. Results on large-sized instances and 95% Tukey HSD intervals in the ANOVA, (a) Boxplots of HV; (b) Homogeneous subsets of Tukey HSD (a.1) $n = 20$. (a.2) $n = 40$. (a.3) $n = 50$. (b.1) $n = 20$. (b.2) $n = 40$. (b.3) $n = 50$.

distribution of these subsets reveals that for $n = 20$, GDRL, PBIGA, MMMA, and NSGA-II belong to the same homogeneous subset, indicating no significant difference among these algorithms. Similarly, no significant difference was observed among PBIGA, MMMA, NSGA-II, and DRL-FJSP when $n = 20$, or between HAS and DRL-FJSP. However, GDRL exhibits a significant difference compared to both DRL-FJSP and HAS. For the instance groups with $n = 40$ and $n = 50$, no statistically significant difference was observed between GDRL and PBIGA. GDRL exhibits significant performance differences compared to the other algorithms. Notably, when $n = 50$, no statistically significant difference was observed among the five algorithms, suggesting the strong generalization capability of GDRL across different problem scales.

To further explore the relationship between algorithm performance and the scale characteristics of ECDCSP-M, we present interaction plots under varying jobs, machines, and stages, as shown in Fig. 11. The results reveal that the advantages of GDRL become increasingly prominent with larger job sizes. However, as the number of machines in Phase 1 of ECDCSP-M increases, the performance gap between GDRL and the other four algorithms remains relatively stable. A similar pattern is observed with respect to the number of stages in Phase 2 of ECDCSP-M.

Finally, to more precisely examine performance differences among all algorithms across varying instance scales, the Friedman test was conducted on 12 distinct configurations. The results are shown in Table S.VII. and Fig. S.10 of the Supplementary Materials. The results clearly show that for all large-scale instances, GDRL consistently ranks highest among the evaluated algorithms. All p-values are significantly below 0.05, confirming statistical significance at the 95%



(a) (b) (c)

Fig. 11. Interactions and 95% Tukey HSD intervals in ANOVA. (a) Job. (b) Machine in Phase 1. (c) Stage in Phase 2.

and 90% confidence levels. In Fig. S.10, the solid and dashed lines represent the critical difference (CD) at 90% and 95% confidence levels, respectively. The results indicate that, in most cases, GDRL exhibits a significant performance advantage, as its bar lies on the opposite side of the CD line from the other four algorithms. Moreover, for the configuration with $n = 20$, $m = 5$, $s = 10$, both MMMA and GDRL are positioned above the CD line, indicating no significant difference between them. However, GDRL is ranked higher than MMMA, suggesting that in most scenarios, its solutions tend to dominate those of MMMA. Additionally, as the problem size increases, the performance of the other DRL-based method gradually deteriorates, especially for instances with $n = 50$, $m = 10$. This degradation may be attributed to the increased number of mating operations as the job size grows, coupled with the absence of embedded mating operation information in the DRL-FJSP network structure.

2) Run Time Analysis: The average runtime of GDRL and FJSP-DRL across the test instance groups is illustrated in Fig. S.11 of the Supplementary Materials. It is worth noting that Table S.VI reports the average runtimes of all algorithms, with HAS, MMMA, and NSGA-II being executed under a maximum runtime cutoff criterion. As their runtimes were significantly longer than those of the two DRL-based algorithms, these three methods were excluded from the comparison shown in Fig. S.11. The results reveal the following findings: 1) in most instances, the runtime of GDRL is slightly longer than that of FJSP-DRL, except for the $n = 20$ instance group; and 2) as the job size n increases, a noticeable rise in runtime is observed for both algorithms, which can be attributed to the expansion of the scheduling action space. Nevertheless, considering that the training process is performed offline, this level of computational overhead remains acceptable.

VI. Conclusion and Future Work

This article investigates a cascaded dual-shop collaborative scheduling problem with mating operations, focusing on both time and EC. The study achieves cascaded dual-shop collaborative optimization by leveraging an efficient end-to-end DRL approach. During the problem modeling phase, an integrated method is employed within the underlying MDP to unify the decision-making processes of both phases in ECFJSP-M. The scheduling state is represented using a heterogeneous graph that incorporates features of both production processes and EC activities. The reward is formulated using a normalized function for the two objectives, which serves as a crucial component for balancing production efficiency and EC. In the solution phase, a novel HGNN-based scheduler is introduced. This tool employs a three-stage node embedding mechanism to transform high-dimensional features in the graph into low-dimensional embeddings. Based on this foundation, the widely adopted PPO algorithm is used to design and train the policy and value networks. To further validate the effectiveness of the proposed method, extensive experiments were conducted on a large set of test instances. The method was evaluated from multiple perspectives, including training efficiency, robustness, generalization capability, and runtime performance. The results indicate that the proposed approach consistently achieves superior performance in all these aspects.

Several areas merit further investigation. First, real-world production scheduling often involves more complexities beyond mating operations, such as uncertainties like machine failures and urgent order insertions. Incorporating these elements could enhance the robustness and adaptability of the scheduling model. Additionally, the end-to-end DRL approach could benefit from further improvements in feature extraction of the scheduling state, as well as the integration of advanced search mechanisms, to boost training efficiency and overall performance.

References

[1] L. Gui et al., "Optimisation framework and method for solving the serial dual-shop collaborative scheduling problem," *Int. J. Prod. Res.*, vol. 61, no. 13, pp. 4341–4357, Sep. 2022.

[2] C. Wang, Q. Pan, H. Sang, and X. Jing, "A cascaded flowshop joint scheduling problem with makespan minimization: A mathematical model and shifting iterated greedy algorithm," *Swarm. Evol. Comput.*, vol. 86, Apr. 2024, Art. no. 101489.

[3] C. Wang, Q. Pan, and H. Sang, "The cascaded flowshop joint scheduling problem: A mathematical model and population-based iterated greedy algorithm to minimize total tardiness," *Robot. Comput. Manuf.*, vol. 88, Aug. 2024, Art. no. 102747.

[4] B. Chen, J. Zhang, J. Xiong, W. Tang, and S. Jiang, "An explainable multi-layer graph attention network for product completion time prediction in aircraft final assembly lines," *J. Manuf. Syst.*, vol. 80, pp. 1053–1071, Jun. 2025.

[5] X. Tao, Q. Pan, and L. Gao, "An iterated greedy algorithm with reinforcement learning for distributed hybrid flowshop problems with job merging," *IEEE Trans. Evol. Computat.*, vol. 29, no. 3, pp. 589–600, Jun. 2025.

[6] Z. Cao, C. Lin, and M. Zhou, "A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 1, pp. 56–69, Jan. 2021.

[7] W. Shao, Z. Shao, and D. Pi, "A network memetic algorithm for energy and labor-aware distributed heterogeneous hybrid flow shop scheduling problem," *Swarm. Evol. Comput.*, vol. 75, Dec. 2022, Art. no. 101190.

[8] F. Zhao, C. Zhuang, L. Wang, and C. Dong, "An iterative greedy algorithm with Q-Learning mechanism for the multiobjective distributed no-idle permutation flowshop scheduling," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 54, no. 5, pp. 3207–3219, May 2024.

[9] 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.

[10] W. Shao, Z. Shao, and D. Pi, "A multi-neighborhood-based multi-objective memetic algorithm for the energy-efficient distributed flexible flow shop scheduling problem," *Neural Comput. Appl.*, vol. 34, no. 24, pp. 22303–22330, Aug. 2022.

[11] S. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 46, no. 1, pp. 139–149, Jan. 2016.

[12] X. He, Q. Pan, L. Gao, J. S. Neufeld, and J. N. D. Gupta, "Historical information based iterated greedy algorithm for distributed flowshop group scheduling problem with sequence-dependent setup times," *Omega-Int. J. Manage. Sci.*, vol. 123, Feb. 2024, Art. no. 102997.

[13] F. Zhao, R. Ma, and L. Wang, "A self-learning discrete Jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory System," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 12675–12686, Dec. 2022.

[14] H. Gong, L. Tang, and C. Duin, "A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times," *Comput. Oper. Res.*, vol. 37, no. 5, pp. 960–969, May 2010.

[15] F. Yao, M. Zhao, and H. Zhang, "Two-stage hybrid flow shop scheduling with dynamic job arrivals," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1701–1712, Jul. 2012.

[16] R. Meng, Y. Rao, Y. Zheng, and D. Qi, "Modelling and solving algorithm for two-stage scheduling of construction component manufacturing with machining and welding process," *Int. J. Prod. Res.*, vol. 56, no. 19, pp. 6378–6390, Jul. 2017.

[17] C. Wang, Q. Pan, and X. Jing, "An effective adaptive iterated greedy algorithm for a cascaded flowshop joint scheduling problem," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 121856.

[18] A. Hasani and S. Hosseini, "A bi-objective flexible flow shop scheduling problem with machine-dependent processing stages: Trade-off between production costs and energy consumption," *Appl. Math. Comput.*, vol. 386, Dec. 2020, Art. no. 125533.

[19] Y. Wang, Z. Jia, and X. Zhang, "A hybrid meta-heuristic for the flexible flow shop scheduling with blocking," *Swarm Evol. Comput.*, vol. 75, Dec. 2022, Art. no. 101195.

[20] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 28, no. 1, pp. 147–167, Feb. 2024.

[21] W. Song, X. Chen, Q. Li, and Z. Cao, "Flexible job-shop scheduling via graph neural network and deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1600–1610, Feb. 2023.

[22] C. Su, C. Zhang, C. Wang, W. Cen, G. Chen, and L. Xie, "Fast pareto set approximation for multi-objective flexible job shop scheduling via parallel preference-conditioned graph reinforcement learning," *Swarm. Evol. Comput.*, vol. 88, Jul. 2024, Art. no. 101605.

[23] F. Zhao, F. Yin, L. Wang, and Y. Yu, "A co-evolution algorithm with dueling reinforcement learning mechanism for the energy-aware distributed heterogeneous flexible flow-shop scheduling problem," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 55, no. 3, pp. 1794–1809, Mar. 2025.

[24] Y. Du, J. Li, C. Li, and P. Duan, "A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 5695–5709, Apr. 2024.

[25] Y. Du and J. Li, "A deep reinforcement learning based algorithm for a distributed precast concrete production scheduling," *Int. J. Prod. Econ.*, vol. 268, Feb. 2024, Art. no. 109102.

[26] R. Li, W. Gong, L. Wang, C. Lu, and C. Dong, "Co-evolution with deep reinforcement learning for energy-aware distributed heterogeneous flexible job shop scheduling," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 54, no. 1, pp. 201–211, Jan. 2024.

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

[28] R. Liu, R. Piplani, and C. Toro, "A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem," *Comput. Oper. Res.*, vol. 159, Nov. 2023, Art. no. 106294.

[29] Y. Zhao, B. Li, J. Wang, D. Jiang, and D. Li, "Integrating deep reinforcement learning with pointer networks for service request scheduling in edge computing," *Knowl.-Based Syst.*, vol. 258, Dec. 2022, Art. no. 109983.

[30] Y. Gui, D. Tang, H. Zhu, Y. Zhang, and Z. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Comput. Ind. Eng.*, vol. 180, Jun. 2023, Art. no. 109255.

[31] R. Liu, R. Piplani, and C. Toro, "Deep reinforcement learning for dynamic scheduling of a flexible job shop," *Int. J. Prod. Res.*, vol. 60, no. 13, pp. 4049–4069, Apr. 2022.

[32] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, Apr. 2021.

[33] K. Lei et al., "A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem," *Expert Syst. Appl.*, vol. 205, Nov. 2022, Art. no. 117796.

[34] Z. Rui et al., "Graph reinforcement learning for flexible job shop scheduling under industrial demand response: A production and energy nexus perspective," *Comput. Ind. Eng.*, vol. 193, Jul. 2024, Art. no. 110325.

[35] R. Wang, G. Wang, J. Sun, F. Deng, and J. Chen, "Flexible job shop scheduling via dual attention network-based reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 3091–3102, Mar. 2024.

[36] J. Huang, L. Gao, and X. Li, "A hierarchical multi-action deep reinforcement learning method for dynamic distributed job-shop scheduling problem with job arrivals," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 2501–2513, 2025.

[37] J. Huang, L. Gao, and X. Li, "An end-to-end deep reinforcement learning method based on graph neural network for distributed job-shop scheduling problem," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 121756.

[38] C. Liu and T. Huang, "Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 53, no. 11, pp. 6836–6848, Nov. 2023.

[39] Y. Zhao, X. Luo, and Y. Zhang, "The application of heterogeneous graph neural network and deep reinforcement learning in hybrid flow shop scheduling problem," *Comput. Ind. Eng.*, vol. 187, Jan. 2024, Art. no. 109802.

[40] 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, Jan. 2021.

[41] R. Zhong et al., "Solving flexible job-shop problem considering skilled workers via multi-domain graph attention network," *Int. J. Prod. Res.*, Jun. 2025, to be published.

[42] G. Shen, S. Sun, and Y. Liu, "A bidding-based deep reinforcement learning approach for multi-agent job shop scheduling problem," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 55, no. 8, pp. 5642–5654, Aug. 2025.

[43] X. Zhao, H. Qu, M. Zhang, J. Feng, L. Wang, and Q. Wu, "A flexible job shop scheduling method based on heterogeneous disjunctive graph and deep reinforcement learning," *Eng. Appl. Artif. Intell.*, vol. 158, Oct. 2025, Art. no. 111356.

[44] W. Zhang, F. Zhao, B. Feng, and X. Mei, "A novel reinforcement learning framework based on simplified graph transformer for large-scale fuzzy flexible job shop scheduling problem," *Eng. Appl. Artif. Intell.*, vol. 158, Oct. 2025, Art. no. 111295.

[45] P. Velikovi, A. Casanova, P. Lio, G. Cucurull, A. Romero, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[47] F. Zhao, Y. Du, C. Zhuang, L. Wang, and Y. Yu, "An iterative greedy algorithm for solving a multiobjective distributed assembly flexible job shop scheduling problem with fuzzy processing time," *IEEE Trans. Cybern.*, vol. 55, no. 5, pp. 2302–2315, May 2025.

[48] H. Bao, Q. Pan, R. Ruiz, and L. Gao, "A collaborative iterated greedy algorithm with reinforcement learning for energy-aware distributed blocking flow-shop scheduling," *Swarm. Evol. Comput.*, vol. 83, Dec. 2023, Art. no. 101399.

[49] 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.

Haizhu Bao received the B.Sc. and M.Sc. degrees from the Lanzhou University of Technology, Lanzhou, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with Shanghai University, Shanghai, China and also with the National University of Singapore, Singapore.

His research interests include intelligent optimization, reinforcement learning, and scheduling.

Quanke Pan (Member, IEEE) 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. He has been with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China, since 2015. He has authored two academic books and more than 300 refereed papers. His current research interests include intelligent optimization and scheduling algorithms.

Prof. Pan acts as the Editorial Board Member for several journals, including *Operations Research Perspective* and *Swarm and Evolutionary Computation*.

Chee-Meng Chew (Senior Member, IEEE) received the B.E. (Hons.) degree in mechanical engineering from the National University of Singapore (NUS), Singapore, in 1991, and the S.M. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1998 and 2000, respectively.

After completing the Ph.D. degree, he joined the Department of Mechanical Engineering, NUS, as an Assistant Professor. His main research interests are in machine learning, autonomous systems, bioinspired and biomimetic systems, novel actuation and mechanism, and assistive device.

Ling Wang (Senior Member, IEEE) received the B.Sc. degree in automation and the Ph.D. degree in control theory and control engineering from the 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 research interests include intelligent optimization and production scheduling.

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

Liang Gao (Senior Member, IEEE) received the B.Sc. degree in mechatronic engineering from Xidian University, Xi'an, China, in 1996, and the Ph.D. degree in mechatronic engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2002.

He is currently a Professor with the Department of Industrial Manufacturing System Engineering, the State Key Laboratory of Intelligent Manufacturing Equipment Technology, and the School of Mechanical Science Engineering, HUST. He has published more than 400 refereed articles. His research interests include operations research and optimization, big data, and machine learning.

Prof. Gao also serves as the Co-Editor-in-Chief for *IET Collaborative Intelligent Manufacturing* and an Associate Editor for *Swarm and Evolutionary Computation* and *Journal of Industrial and Production Engineering*.