Energy-efficient Distributed Heterogeneous Hybrid Flow-shop Scheduling Using Graph Neural Network and Deep Reinforcement Learning

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

***Abstract*—With growing environmental awareness and increasing energy demands, sustainable manufacturing has become a focal point in the industry. Meanwhile, globalization has propelled distributed manufacturing systems as a dominant trend. This paper tackles the energy-efficient distributed heterogeneous hybrid flow-shop scheduling problem (EDHHFSP), aiming to minimize both makespan and total energy consumption. We first formulate a mixed-integer linear programming (MILP) model to provide a benchmark for small instances. More importantly, we propose a novel end-to-end deep reinforcement learning framework based on a heterogeneous graph neural network, which models the scheduling problem as a distributed decision-making process. A key innovation lies in the design of an action space composed of "job–factory" and "operation–machine" pairs, enabling fine-grained, decentralized scheduling decisions. Our approach starts with a novel heterogeneous graph representation of scheduling states, capturing complex interactions among jobs, factories, and machines. A three-stage embedding mechanism is developed to encode real-time scheduling environments. The agent then learns a parameterized policy using the proximal policy optimization (PPO) algorithm, guided by a reward function that balances makespan and energy efficiency. Experimental results demonstrate that our method generalizes well across different problem scales and significantly outperforms traditional heuristics and learning-based baselines in terms of both scheduling quality and energy savings.**

***Note to Practitioners*—Hybrid flow-shop scheduling in heterogeneous production environments is a common challenge faced by equipment manufacturers, particularly in the electronics manufacturing industry. These manufacturers are distributed across various locations, equipped with devices of differing performance, which often have limited buffer capacities and require varying setup times. To address this challenge, a machine learning-based method is used to schedule daily production tasks. This approach designs a novel heterogeneous graph structure to dynamically capture the complex relationships among factories, operations, and machines during scheduling. It employs an HGNN with a three-stage embedding mechanism to efficiently extract state features, which are then fed into a decision network to enable agents to take optimal actions. The method, trained on small-scale instances and directly applied to large-scale ones, improves both the computational time and performance of traditional optimization algorithms. The proposed method exhibits self-learning and adaptive capabilities, with its effectiveness validated through experiments on 1,600 test instances. Its practical application has been demonstrated in production scenarios at a printed circuit board assembly company. In the future, this method can be applied to address more complex distributed production scenarios.**

***Index Terms*—Energy-efficient scheduling, distributed heterogeneous hybrid flow-shop, graph neural network, deep reinforcement learning, multi-objective optimization.**

This work is supported by the National Nature Science Foundation of China 62273221, Program of Shanghai Academic/Technology Research Leader 21XD1401000, and Shanghai Key Laboratory of Power station Automation Technology. (*Corresponding authors: Quanke Pan*).

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

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

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 Department of Automation, Tsinghua University, Beijing 100084, P. R. China (e-mail: wangling@tsinghua.edu.cn).

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

I. Introduction

THE HYBRID flow-shop scheduling problem (HFSP), also known as the flexible flow-shop scheduling problem, is an extension of the classical flow-shop scheduling problem (FSP). In HFSP, multiple machines can operate simultaneously in at least one stage of the process, which minimizes bottlenecks and enhances both flexibility and reliability [1]. This unique parallel machine structure makes HFSP highly relevant in the manufacturing industry, drawing considerable interest from researchers. Despite this, practical investigations indicate that budget constraints or specific process limitations often prevent the immediate replacement of these parallel machines with new ones [2]. Consequently, the machines in HFSP vary in technology levels and are not interchangeable, with newer machines typically offering higher processing speeds than older ones [3]. The advanced processing capabilities of these newer machines incentivize production managers to allocate as many tasks as possible to them [4], [5], [6]. However, these machines typically require special nonproductive operations—such as skilled technicians, tools, and industrial robots—before they can begin processing. These operations, often referred to as setup time, are essential for efficient job processing and are sequence-dependent based on the tasks being processed. Therefore, these setups are termed sequence-dependent setup times (SDSTs) [7].

Moreover, the traditional centralized production model exhibits limitations within the context of HFSP. Transitioning to a distributed production model can boost production capacity while mitigating associated risks. With globalization, an increasing number of enterprises are establishing factories across various regions, cities, and countries. These widespread factories, often constrained by financial and other factors, typically feature heterogeneous machinery, leading to the development of heterogeneous factories [5], [8]. This heterogeneity introduces new challenges in efficiently assigning and scheduling tasks across multiple factories. While distributed production systems and advanced machines improve overall productivity, they also raise concerns about energy consumption (EC). However, it is worth noting that many modern machines incorporate energy-saving technologies, which can potentially reduce energy consumption compared to older equipment [7]. Given the growing emphasis on sustainability and the rising importance of energy costs in manufacturing, reducing EC has become a priority for researchers aiming to support green manufacturing. Therefore, balancing production efficiency with energy sustainability has become a critical focus in the design of distributed scheduling systems. In response, this paper addresses the energy-efficient distributed heterogeneous hybrid flow-shop scheduling problem (EDHHFSP) with SDSTs aiming to minimize both makespan (C_{max}) and total energy consumption (TEC). Minimizing C_{max} enhances equipment utilization, thereby improving production efficiency, while reducing TEC improves energy efficiency and lowers, the carbon footprint [9], [10], [11].

The energy-efficient HFSP (EHFSP) has garnered substantial attention in recent years. However, scenarios involving real-world production constraints, such as heterogeneous environments, have been less explored, especially those that combine a heterogeneous environment with SDSTs. As illustrated in Fig. S-1, the printed circuit board assembly (PCBA) process exemplifies a typical HFSP. A PCBA company in Shandong, China, operates two geographically separated factories, where the core production processes include dual in-line package (DIP) steps such as horizontal insertion, vertical insertion, wave soldering, tin plating, testing, and final inspection. Each DIP operation utilizes multiple parallel machines, which often necessitate a setup phase before each production cycle. Furthermore, to minimize production costs, the storage space between consecutive stages is usually limited. As described, the PCBA process can be effectively modeled as a blocking energy-efficient distributed heterogeneous HFSP (EDHHFSP) with SDSTs. Compared to traditional EHFSPs, this scenario introduces two additional complexities: the need to allocate jobs across separate factories and the challenge of managing semi-finished products stranded on machines at various stages, which significantly complicate the solution process.

The EDHHFSP is recognized as an NP-hard problem, presenting a more significant computational challenge than the already NP-hard HFSP. This heightened complexity stems from the intricate decision-making involved in optimizing job sequencing and efficiently assigning jobs to factories and machines [12]. Solutions to such problems are generally divided into two main types: exact methods and approximate methods. Exact methods are seldom used in practice due to their extensive solving times, making them impractical for large-scale problems. On the other hand, while metaheuristic approaches—an approximation method—offer a favorable trade-off between solving time and scheduling quality, they often suffer from poor generalization performance due to the complexity and specificity of their search mechanisms.

In contrast, heuristic methods, such as priority dispatch rules (PDRs), are frequently employed for large-scale scheduling problems. PDRs are popular in practical scheduling contexts with multiple constraints due to their rapid solving speed and straightforward algorithmic structure. However, because PDRs are typically derived from human expertise and often lack a comprehensive global perspective, their effectiveness can be suboptimal [13]. Moreover, PDRs rely solely on distinct priority indicators for decision-making, leading to only a partial utilization of the available scheduling information. In practice, scheduling decisions often need to consider multiple attributes simultaneously. Thus, there is a need for a method capable of efficiently integrating various attributes into the scheduling process to guide more effective decision-making.

In recent years, machine learning algorithms, particularly reinforcement learning (RL) [14], [15], have gained traction in scheduling problems due to their ability to glean valuable insights from extensive historical data [16], [17]. Deep reinforcement learning (DRL), a subset of RL, combines deep neural networks (DNNs) with RL, allowing for autonomous learning of feature representations from data. Various DRL methods have been successfully applied to job-shop scheduling problems (JSPs) and their extensions [18], [19], including approaches like deep Q-network [20] and proximal policy optimization (PPO) [13], [21], [22]. These methods leverage DRL to generate PDRs for scheduling tasks in an end-to-end manner. By modeling decision-making in PDRs as a Markov decision process (MDP), these techniques develop scheduling policies that can anticipate future outcomes, thus mitigating the short-sightedness often associated with traditional PDRs. Furthermore, the reinforcement training process is guided by the accumulation of rewards over time, which autonomously drives performance optimization.

Despite these promising results, most existing DRL approaches have primarily concentrated on JSPs and classical non-flexible FSPs. There remains limited research focused on multi-objective scheduling in distributed production environments, which presents unique challenges. The flexibility and presence of multiple factories introduce three significant hurdles in developing effective learning mechanisms. First, decision-making in the EDHHFSP is more intricate, requiring not only optimal job sequencing but also precise assignments of factories and machines. Second, the many-to-one relationships between factories and jobs, as well as operations and machines, create substantial challenges when encoding scheduling states via neural networks. Third, for multi-objective optimization problems, crafting a well-designed reward function that effectively guides the algorithm toward optimal solutions is essential to achieving desired performance outcomes.

To address the challenges identified, this paper presents a novel DRL approach aimed at learning PDRs for EDHHFSP. To tackle the first challenge, we formulate the EDHHFSP as an MDP based on PDRs, where actions are defined as job-factory and operation-machine pairs. For the second challenge, we construct a heterogeneous graph composed of machine nodes, factory nodes, and operation nodes to represent the MDP state and capture the complex relationships among factories, operations, and machines. A three-stage graph neural network (GNN) is then introduced to extract feature embeddings for nodes within this heterogeneous graph. These embeddings are used to develop a policy network, which is trained via the PPO algorithm. For the third challenge, we design a reward function that aligns with both scheduling objectives, ensuring that maximizing cumulative rewards leads to simultaneous optimization of both. The paper highlights several key contributions:

1) Introduction of an end-to-end DRL approach for the EDHHFSP that autonomously learns PDRs, featuring a size-agnostic architecture that allows training on smaller instances with applicability to larger ones.
2) Formulation of an MDP that integrates decision-making for operation selection, factory allocation, and machine assignment within a unified framework.
3) Extension of Song et al.'s heterogeneous graph structure [13] to a multi-factory setting, enabling an integrated representation of factory, operation, and machine information, while optimizing graph density to ensure scalable and efficient learning.

These contributions collectively advance the capabilities of DRL in addressing the complexities of the EDHHFSP, offering a robust solution to the scheduling challenges posed by modern manufacturing environments. The remainder of this paper is organized as follows. Section II presents a review of closely related literature. Section III introduces and defines the EDHHFSP. Section IV outlines the proposed DRL approach. Section V reports the experimental results and analysis. Finally, Section VI concludes with directions for further research.

II. Literature Review and Motivation

This section examines previous research on the HFSP and its extensions, alongside the application of DRL in related scheduling challenges.

A. Conventional Methods for HFSP and Its Variants

Conventional approaches to the HFSP can be broadly categorized into exact and approximate methods. Typical exact methods include branch and bound [23], integer programming [24], and branch and cut [25], all of which provide theoretical guarantees for achieving optimal solutions. However, due to the NP-hard nature of HFSPs, these methods often become computationally intractable as problem size increases, resulting in impractical computation times [26]. To address this, various approximation methods, such as heuristics and metaheuristics, have been developed to tackle increasingly complex scheduling challenges. Heuristics, which are based on expert knowledge, do not guarantee optimal solutions but are notably faster than exact methods. PDR serves as a representative heuristic method [27], and Kahraman et al. [28] developed a parallel greedy heuristic for HFSP. Metaheuristics, another widely used approximation approach, include trajectory-based and population-based methods. Trajectory-based methods start with a set of solutions and explore the solution space's neighborhood until stopping criteria are met, but they often focus more on exploitation rather than exploration. Examples include genetic algorithm [29], ant colony optimization [30], and particle swarm optimization [31]. In contrast, population-based methods start with a single solution, iteratively refining it towards optimality, but they can struggle with escaping local optima [32].

In addressing the extended challenges of HFSP, our focus is on energy-aware strategies and distributed scheduling. These energy-saving strategies primarily involve three techniques: shutting down machines during idle periods [33], implementing speed scaling [8], and selecting low-power machines [5]. Mouzon et al. [33] highlighted significant energy savings by powering off non-bottleneck machines during idle times, and Dai et al. [34] further reduced energy consumption (EC) in HFSP with unrelated parallel machines by employing this strategy. Meng et al. [35] investigated an energy-conscious HFSP using an on/off strategy, while Wang et al. [36] explored HFSP with machine eligibility constraints, applying an improved ε-constraint to determine the Pareto optimal solutions. However, frequent power cycling can reduce machine lifespan, prompting the adoption of speed-scaling strategies, where increased machine speeds reduce processing time but elevate power consumption. Shao et al. [8] examined energy-saving distributed HFSP using a multi-neighborhood multi-objective memetic algorithm (MMMA). The final strategy, selecting low-power machines, is particularly effective in heterogeneous parallel machine environments with varying speeds and power needs. Effective scheduling strategies aim to allocate operations to suitable machines while developing comprehensive schedules that balance cost efficiency with energy savings. Bruzzone et al. [37] introduced an energy-conscious scheduling strategy focused on strategically planning energy savings within predefined schedules and Li et al. [38] devised a multi-objective optimization approach for machine assignment and job sequencing in HFSP. Gong et al. [39] addressed HFSP with labor flexibility, using a hybrid evolutionary algorithm to assign machines and laborers for each operation, minimizing EC. Hasani and Hosseini [6] proposed an HFSP with machine-dependent processing stages, utilizing a non-dominated sorting genetic algorithm (NSGA-II) to optimize production costs and EC simultaneously.

Distributed HFSPs have attracted growing attention due to their complexity and practical relevance [40]. A summary of existing research on distributed HFSP is provided in Table S-I of the Supplementary Material. Shao et al. [41] studied a distributed no-wait HFSP and designed a constructive heuristic-based VND algorithm. A multi-objective distributed HFSP was tackled in [42] via a multi-objective evolutionary algorithm with multiple neighborhoods. Li et al. [43] considered variable-speed constraints and proposed a knowledge-guided multi-objective evolutionary algorithm. Lu et al. [44] introduced energy-aware modeling and developed a hybrid multi-objective iterated greedy (IG) algorithm. Yu et al. [45] addressed assembly and dual-resource constraints with a knowledge-based IG approach. Liu et al. [46] incorporated blocking constraints and proposed a tri-individual IG algorithm. Wang et al. [5] integrated energy constraints and RL-enhanced memetic search. Shao et al. [47] applied meta-Q-learning to a fuzzy HFSP. Cui et al. [29] investigated heterogeneity and developed a multi-population genetic algorithm with inter-factory neighborhoods. Lei et al. [48] focused on SDSTs using a multi-class teaching–learning-based optimization algorithm. Qin et al. [49] addressed blocking with a collaborative IG algorithm. Shao et al. [50] examined lot-streaming and employed heuristic construction with iterated local search algorithm. Xuan et al. [51] considered transport times and SDSTs, solved by an artificial immune differential evolution algorithm. Lu et al. [52] integrated energy efficiency using a Pareto-based hybrid IG framework. Zhao et al. [53] formulated an energy-aware distributed heterogeneous HFSP, and proposed a DRL-guided co-evolutionary algorithm combining adversarial learning.

B. DRL-Based Scheduling Methods

A key challenge in DRL-based scheduling is the extraction of state features. In the context of the JSPs and their extensions, some researchers have manually curated and designed state vectors as inputs for deep neural networks (DNNs), employing DNNs and DRL to extract state features [54]. This approach is constrained by fixed vector dimensions, limiting its adaptability to varying instance sizes and relying on human expertise for scheduling rule design. Unlike JSP, the HFSP requires maintaining a consistent processing order for jobs across different stages due to process constraints, presenting a distinct problem structure. Kwon et al. [55] addressed the dimensional limitations of HFSP using a variable-sized structure for state vectors, partially mitigating this issue, although the model remains bound to a fixed number of HFSP instances. Ni et al. [56] employed DRL to derive heuristics for HFSP, introducing a graph convolutional network that uses a multi-graph structure to represent each search step and tackles dimensionality constraints with attention-based pooling across multiple stages. Pan et al. [57] developed a recurrent neural network architecture to manage permutation FSPs of varying scales but focused solely on a single objective and did not account for distributed factory settings.

Fortunately, Graph Neural Networks (GNNs), with their capability to manage graphs of varying sizes, present a promising solution to the scheduling challenges previously outlined. Zhang et al. [58] successfully integrated DRL and GNN to develop effective PDRs, modeling states as disjunctive graphs with diverse connections and encoding these using GNNs. Similarly, Park et al. [59] employed this approach to address the JSP, highlighting GNNs' ability to capture complex node relationships during message passing. Song et al. [13] proposed an end-to-end method that integrates a heterogeneous graph neural network with deep reinforcement learning to address the FJSP. In their model, operations and machines are represented as different types of nodes, while edges capture various relationships such as precedence and resource constraints. This heterogeneous graph structure enables more expressive state representations under flexible scheduling conditions, facilitating effective information propagation across entities. The proposed framework demonstrates strong performance on large-scale instances by overcoming limitations in conventional state modeling. Wang et al. [60] proposed an end-to-end DRL framework by GNNs, achieving superior performance on multi-objective FJSP. Moon et al. [61] developed the heterogeneous Graph scheduler, which models FJSP with transportation as a heterogeneous graph and employs attention-based encoders to achieve scale-invariant scheduling solutions. Wan et al. [62] introduced a meta-path-based heterogeneous GNN embedded in a dual-task DRL framework that jointly optimizes operation selection and machine allocation for scalable and efficient FJSP solutions. Lei et al. [63] proposed a hierarchical reinforcement learning framework with a GNN-based lower layer for real-time operation sequencing in large-scale dynamic FJSP, enabling near real-time scheduling. Zhang et al. [64] addressed DFJSP with limited transportation resources by integrating DRL with a heterogeneous GNN to learn high-quality priority dispatching rules in an end-to-end fashion. Wang et al. [65] presented a dual-attention network combining DRL and self-attention mechanisms to model intricate operation-machine relations, yielding results comparable to exact methods. Song et al. [16] proposed a self-attention-enhanced end-to-end DRL method for the stochastic economic lot scheduling problem.

C. Research Gaps and Motivation

In researching the HFSP and its extended problems, most studies concentrate on single-factory scenarios [21], [57], [66] and make simplified assumptions, such as assuming identical equipment across all factories [5], [8], ignoring setup times [47], [56], allowing unlimited buffer space [66], and permitting flexible machine speed adjustments [7], [8] and frequent machine on/off switching [6]. While these assumptions simplify the complexities involved, they overlook critical factors necessary for maintaining stable and efficient operations in real-world production scheduling.

Most traditional methods for solving scheduling problems face a trade-off between solution quality and computational efficiency. Exact methods, while providing high-quality solutions, incur prohibitive time costs for large-scale problems [23], [24], [25]. Heuristic methods, which depend heavily on expert knowledge, often demonstrate limited generalization capabilities [27], [28]. Metaheuristic approaches, on the other hand, require significant time to develop problem-specific neighborhood structures [8]. Popular DRL methods predominantly focus on JSP [67] and their extensions [13], with a limited exploration of HFSP. Moreover, existing DRL-based scheduling frameworks are rarely extended to distributed multi-factory environments, largely due to the absence of explicit modeling for factory-level information [19], [59]. In particular, the lack of factory node embeddings in current graph structures limits the capability of these methods to handle factory assignment decisions while accounting for inter-factory heterogeneity. Additionally, some DRL methods utilize problem-dependent PDRs [67], which perform well in specific contexts but lack generalization, as real-world production environments rarely align precisely with modeled scenarios. This paper introduces a novel heterogeneous graph network structure and a three-stage node embedding mechanism that converts graph states into embeddings, enabling real-time environmental awareness. We also formulated job-factory and operation-machine pairings as actions to achieve end-to-end optimization. Notably, limited research exists on multi-objective problems that simultaneously optimize energy and production efficiencies, which this work seeks to address.

III. Problem Description

The EDHHFSP comprises δ heterogeneous factories, each characterized by distinct processing capacities. Each factory operates as a flexible flow-shop with m sequential stages. Given the diversity in flexibility across different factories, each factory F_f is equipped with $l_{f,k}$ unrelated parallel machines possessing different technology levels at stage K_k. Denoted as $M_i^{f,k}$, the ith machine at stage K_k in F_f has a processing speed $v_{f,k,i}$. A set of n jobs must be allocated to one of the δ factories and subsequently executed on any of the UPMs across m stages sequentially within a specific factory. Each job J_j consists of m operations $(O_{j,1}, O_{j,2}, \dots, O_{j,m})$, where the standard time for $O_{j,k}$ is $p_{j,k}$. The actual processing time of $O_{j,k}$ on $M_i^{f,k}$ is $p_{j,k}/v_{f,k,i}$, with an EC per unit time of $\beta_{f,i,k}$. Due to the lack of intermediate buffers between parallel machines at different stages, the current machine will retain the job until a machine in the next stage is idle, resulting in a blocking effect on the current machine. Additionally, in alignment with practical production scenarios, setup times are distinct from the processing times and depend on the job sequence. Thus, each machine $M_i^{f,k}$ can operate under four distinct EC modes, namely processing mode EC ($PEC_{j,i,f,k}$), setup mode EC ($SEC_{j,i,f,k}$), blocking mode EC ($BEC_{j,i,f,k}$), and idle mode EC ($IEC_{j,i,f,k}$). The EDHHFSP seeks to minimize C_{max} and TEC, while also incorporating other typical flow-shop constraints. (1) All jobs become available at time zero; (2) each job is assigned to a single machine at any given time; (3) each machine handles only one job at a time; (4) preemption is strictly prohibited.

A. MILP Model of EDHHFSP

The notations are enumerated in Section S-IV of the Supplementary Material.

$$\textbf{Objectives:}\quad \text{Minimize}\begin{cases} f_1 = C_{max} \\ f_2 = TEC \end{cases} \tag{1}$$

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

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

$$\sum_{i=1}^{l_{f,k}} y_{j,i,f,k} = x_{j,f}, \forall j \notin \{0\}, f, k \notin \{0\} \tag{4}$$

$$y_{0,i,f,k} = 1, \forall i, f, k \tag{5}$$

$$z_{j,j',i,f,k} \le y_{j,i,f,k}, \forall j', j \notin \{0\}, j \ne j', k \notin \{0\}, i, f \tag{6}$$

$$z_{j,j',i,f,k} \le y_{j',i,f,k}, \forall j, j' \notin \{0\}, j \ne j', k \notin \{0\}, i, f \tag{7}$$

$$\sum_{j=0, j\ne j'}^{n} z_{j,j',i,f,k} = y_{j',i,f,k}, \forall j', k \notin \{0\}, i, f \tag{8}$$

$$\sum_{j'=0, j\ne j'}^{n} z_{j,j',i,f,k} = y_{j,i,f,k}, \forall j, k \notin \{0\}, i, f \tag{9}$$

$$\sum_{J_j \in \Psi} \sum_{J_{j'} \in \Psi} z_{j,j',i,f,k} \le |\Psi| - 1, \forall \Psi \subseteq J, j, j' \notin \{0\}, j \ne j', k \notin \{0\}, i, f \tag{10}$$

$$d_{0,k} = 0, \forall k \tag{11}$$

$$d_{j,k} \ge 0, \forall j, k \tag{12}$$

$$d_{j,k} \ge d_{j,k-1} + \frac{p_{j,k}}{v_{f,k,i}} + (x_{j,f} - 1) \cdot h, \forall j \notin \{0\}, f, k \notin \{0\}, i \tag{13}$$

$$d_{j',k} \ge d_{j,k+1} + st_{j,j',i,k+1} + (z_{j,j',i,f,k+1} - 1) \cdot h, \forall j, j' \notin \{0\}, j \ne j', k \notin \{m\}, i, f \tag{14}$$

$$d_{j',k} \ge d_{j,k} + \frac{p_{j',k}}{v_{f,k,i}} + st_{j,j',i,k} + (z_{j,j',i,f,k} - 1) \cdot h, \forall j, j' \notin \{0\}, j \ne j', k, i, f \tag{15}$$

$$d_{j,m} \le d_{j,m-1} + \frac{p_{j,m}}{v_{f,m,i}} - (y_{j,i,f,m} - 1) \cdot h, \forall j, i, f \tag{16}$$

$$d_{j,k} \ge st_{0,j,i,k} + \frac{p_{j,k}}{v_{f,k,i}} + (z_{0,j,i,f,k} - 1) \cdot h, \forall j \notin \{0\}, k \notin \{0\}, i, f \tag{17}$$

$$PEC_{j,i,f,k} \ge \beta_{f,i,k} \cdot \frac{p_{j,k}}{v_{f,k,i}} + (y_{j,i,f,k} - 1) \cdot h, \forall j, i, f, k \notin \{0\} \tag{18}$$

$$SEC_{j,i,f,k} \ge \gamma \cdot st_{j,j',i,k} + (z_{j,j',i,f,k} - 1) \cdot h, \forall j \ne j', k \notin \{0\}, i, f \tag{19}$$

$$BEC_{j,i,f,k} \ge \omega \cdot \left(d_{j,k} - d_{j,k-1} - \frac{p_{j,k}}{v_{f,k,i}} \right) + (y_{j,i,f,k} - 1) \cdot h, \forall j, k \notin \{0, m\}, i, f \tag{20}$$

$$IEC_{j,i,f,k} \ge \theta \cdot (d_{j',k-1} - d_{j,k} - st_{j,j',i,k}) + (z_{j,j',i,f,k} - 1) \cdot h, \forall j, j' \notin \{0\}, j \ne j', k \notin \{0\}, i, f \tag{21}$$

$$TEC \ge \sum_{f=1}^{\delta} \sum_{k=1}^{m} \sum_{i=1}^{l_{f,k}} \sum_{j=0}^{n} (PEC_{j,i,f,k} + SEC_{j,i,f,k} + BEC_{j,i,f,k} + IEC_{j,i,f,k}) \tag{22}$$

$$C_{max} \ge d_{j,m}, \forall j \tag{23}$$

The primary objectives (1) aim to minimize both C_{max} and TEC. Constraint (2) ensures that each factory contains a designated dummy job. Constraint (3) guarantees that each job is assigned exclusively to one factory. Constraint (4) mandates that each job is processed by one machine at each stage. Constraint (5) implies assigning the dummy job to all machines. Constraints (6) and (7) ensure that if a job is not assigned to a particular machine, no preceding or subsequent job is assigned to that machine. Constraints (8)-(10) prevent a subset of jobs assigned to any machine from creating cyclic dependencies. Constraint (11) specifies that all machines are initially available at the start of the scheduling horizon. Constraint (12) enforces a non-negative completion time for each job. Constraint (13) ensures that a job's operation cannot begin until the prior stage's operation concludes. Constraints (14) and (15) define departure timing constraints for two consecutive jobs on any machine. Constraint (16) ensures the prompt release of each job upon the completion of its final stage. Constraint (17) defines the departure time of the first job. The EC of each machine in processing, setup, blocked, and idle modes is detailed in Constraints (18)-(21), respectively. Equations (22) and (23) formally define C_{max} and TEC, respectively. The MILP code is accessible on GitHub (https://github.com/banian2314/Model-of-EDHFSP.git).

B. A Numerical Example

We examine a numerical example involving 2 factories, 4 jobs, and 3 stages to illustrate the problem in detail. The considered problem clarified in Section S-V of Supplementary Material. The optimized results are $C_{max} = 18$ and $TEC = 379.9$. Fig. S-2 presents a feasible schedule along with corresponding power consumption profiles over time.

C. Conflicting Relationship Between C_{max} and TEC

An example illustrates the correlation between C_{max} and TEC in the EDHHFSP. Resource configurations are detailed in Section S-VI of Supplementary Material. We explore all possible solutions of the MILP model for this example using CPLEX, and the experimental findings are shown in Fig. S-3. Two key observations emerge from the figure: (1) a single C_{max} value corresponds to multiple TEC values, indicating multiple optimal solutions in the EDHHFSP solution space; (2) C_{max} and TEC represent two conflicting objectives. A decrease in C_{max} results in an increase in TEC, and vice versa. This trade-off relationship has also been observed in previous study [68], and our example further visualizes this phenomenon in the context of EDHHFSP, thereby helping to illustrate the multi-objective nature of EDHHFSP for readers.

IV. Solution Methodology

This section presents a comprehensive analysis of the proposed methodology, which addresses the EDHHFSP as a sequential decision-making problem. The overall workflow is depicted in Fig. 1. To keep the main text concise, detailed discussions of the challenges and methodological innovations in adapting GNN-RL to multi-factory scheduling are provided in the Supplementary Material (see Section S-VII).

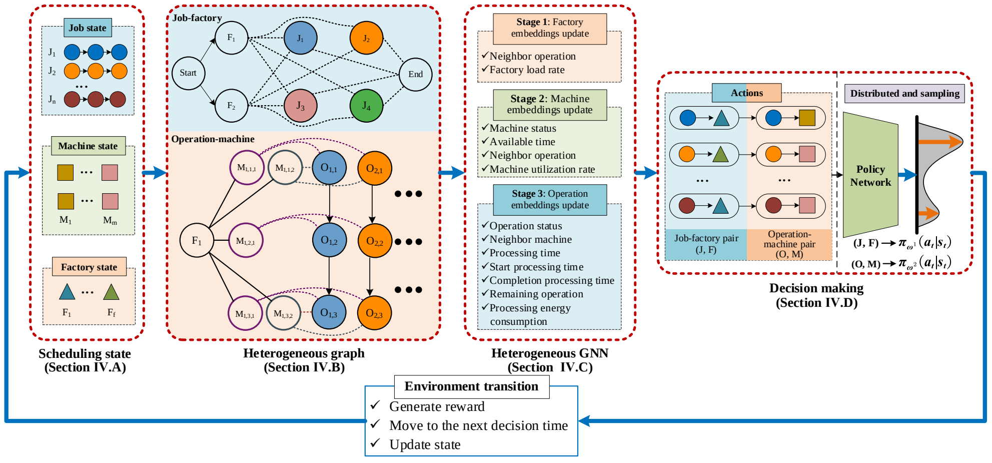

Job state
Machine state
Factory state
Scheduling state (Section IV.A)
Job-factory
Operation-machine
Heterogeneous graph (Section IV.B)
Stage 1: Factory embeddings update
✓Neighbor operation
✓Factory load rate
Stage 2: Machine embeddings update
✓Machine status
✓Available time
✓Neighbor operation
✓Machine utilization rate
Stage 3: Operation embeddings update
✓Operation status
✓Neighbor machine
✓Processing time
✓Start processing time
✓Completion processing time
✓Remaining operation
✓Processing energy consumption
Heterogeneous GNN (Section IV.C)
Actions
Job-factory pair (J, F)
Operation-machine pair (O, M)
Distributed and sampling
Policy Network
Decision making (Section IV.D)
Environment transition
✓ Generate reward
✓ Move to the next decision time
✓ Update state

Fig. 1. Workflow of the proposed methodology for EDHHFSP.

A. MDP Formulation

The scheduling workflow for EDHHFSP is structured as follows: at each decision step t (either at the initial time or upon the completion of an operation), the agent observes the system state s_t and selects an action a_t. This action involves assigning an unscheduled operation to an appropriate machine at a compatible factory and initiating it at the current time, $T(t)$. The environment then transitions to the next decision step $t+1$. This process iteratives until all operations are scheduled. The corresponding MDP is defined below.

1) State representation

This paper captures the state features of the MDP for the EDHHFSP through two main approaches: constructing a graph structure and vectorizing node representations. Previous research utilizing GNNs for scheduling problems has typically depicted MDP states via disjunctive graph structures tailored to specific instances [13], [59], [67]. However, due to the unique complexities of EDHHFSP, accurately representing it using traditional disjunctive graphs poses challenges. To address this, we propose a novel heterogeneous graph (HDG) structure specifically designed for EDHHFSP, detailed in Section IV-B. The HDG, denoted as $\mathcal{H}^D = (\mathcal{F}, \mathcal{O}, \mathcal{M}, \mathcal{C}, \mathcal{E}, \mathcal{K})$, consists of factory nodes ($\mathcal{F}$), operation nodes ($\mathcal{O}$), machine nodes ($\mathcal{M}$), and three distinct types of arcs. These arcs include conjunctive arcs to enforce processing sequence constraints ($\mathcal{C}$), disjunctive arcs to allocate operations to machines ($\mathcal{E}$), and additional disjunctive arcs to define allocation relationships between jobs and factories ($\mathcal{K}$), as illustrated in Fig. 2.

Defining the raw feature vectors for each node type within the HDG is crucial for enabling the scheduling algorithm to derive an optimal policy. Let the embedding vector of the factory node F_f be $v_f^t = \{|\mathcal{N}_f^t|, Lr_f^t\}$, where $|\mathcal{N}_f^t|$ is the number of neighbor operation nodes of F_f. Lr_f^t denotes the factory load rate of F_f at time t, calculated as $Lr_f^t = \sum_{j \in J} x_{j,f}/n$. Let the embedding vector of the operation node $O_{j,k}$ be $\mu_{j,k}^t = \{I_{j,k}^t, |\mathcal{N}_{j,k}^t|, pt_{j,k}, ST_{j,k}^t, CT_j, m_j, ec_{j,k}\}$, where $I_{j,k}^t$ is a binary variable that takes value 1 if $O_{j,k}$ is scheduled at time t; otherwise, it is 0. $|\mathcal{N}_{j,k}^t|$ represents the number of neighbor machine nodes of $O_{j,k}$ at time t. $pt_{j,k}$ denotes the processing time of $O_{j,k}$; if scheduled at time t, it is the actual value $p_{j,k}/v_{f,k,i}$; otherwise, it is the average value $\bar{p}_{j,k} = (\sum_{M_{f,k,i} \in \mathcal{M}_{j,k}} p_{j,k}/v_{f,k,i})/|\mathcal{M}_{j,k}|$. The values of f and i in $v_{f,k,i}$ are determined according to the machine index $M_{f,k,i} \in \mathcal{M}_{j,k}$, where $\mathcal{M}_{j,k}$ denotes the set of machines capable of processing operation $O_{j,k}$. $ST_{j,k}^t$ indicates the actual or estimated start time of $O_{j,k}$; if scheduled at t, it is the actual $ST_{j,k}$; otherwise, it is an estimate based on precedence constraints. If its direct predecessor $O_{j,k-1}$ is assigned to $M_{f,k-1,i}$, then $ST_{j,k}^t =$

$ST_{j,k-1} + pt_{j,k}$; otherwise, $ST^t_{j,k} = ST^t_{j,k-1} + \bar{p}_{j,k-1}$. The final completion time of the job J_j, denoted CT_j, is $CT_j = ST^t_{j,m} + pt_{j,m}$. m_j is the number of remaining operations. $ec_{j,k}$ represents processing energy consumption of $O_{j,k}$; if scheduled at t, it is $(p_{j,k} \cdot \beta_{f,i,k})/v_{f,k,i}$; otherwise, it is the average $\bar{e}_{j,k} = (\sum_{M_{f,k,i} \in \mathcal{M}_{j,k}} p_{j,k} \cdot \beta_{f,i,k}/v_{f,k,i})/|\mathcal{M}_{j,k}|$.

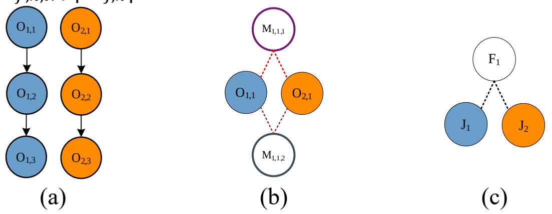


Fig. 2. Illustration of arc types in the scheduling model: (a) Conjunctive arcs: representing fixed sequential dependencies between operations, (b) disjunctive arcs between operations and machines: highlighting alternative paths for operation-to-machine assignments, (c) disjunctive arcs between jobs and factories: showing flexible allocation options for assigning jobs to factories.

The representation vector of the machine node $M_{f,k,i}$ is defined as $\varpi^t_{f,k,i} = \{I^t_{f,k,i}, AT^t_{f,k,i}, |\mathcal{N}^t_{f,k,i}|, U^t_{f,k,i}\}$, where $I^t_{f,k,i}$ is a binary variable that takes value 1 if $M_{f,k,i}$ is assigned a process at time t; otherwise, it is 0. $AT^t_{f,k,i}$ indicates the available time when $M_{f,k,i}$ finishes processing the assigned operation and enters the idle state. $|\mathcal{N}^t_{f,k,i}|$ denotes the number of neighbor operation nodes of $M_{f,k,i}$ at time t. $U^t_{f,k,i}$ represents the utilization rate of $M_{f,k,i}$ at time t, calculated as $U^t_{f,k,i} = \sum_0^t NIT_{f,k,i}/t$, where $\sum_0^t NIT_{f,k,i}$ represents the total non-idle time of $M_{f,k,i}$ up to time t.

Among the three types of arcs, the conjunctive arc $\mathcal{C}$, representing the processing sequence constraint, is a static, directed arc. The first type of disjunctive arc $\mathcal{E}$ connects the operation node $O_{j,k}$ to each compatible machine node $M_{f,k,i}$. This disjunctive arc is characterized by eigenvalue information jointly determined by the machine and the operation nodes, enabling effective embedding between the two types of nodes in Section IV-C. Let the representation vector of the disjunctive arc $E_{j,f,k,i} \in \mathcal{E}$ be $\lambda_{j,f,k,i} = \{pt_{j,f,k,i}, ec_{j,f,k,i}\}$, where $pt_{j,f,k,i} = p_{j,k}/v_{f,k,i}$ and $ec_{j,f,k,i} = (p_{j,k}/v_{f,k,i}) \cdot \beta_{f,i,k}$ denote the processing time and EC of $O_{j,k}$ on $M_{f,k,i}$, respectively. The second type of disjunctive arc $\mathcal{K}$ is an undirected arc connecting the job node associated with the operation node $O_{j,k}$ to each factory node F_f. Let the representation vector of the disjunctive arc $K_{j,f,k,i} \in \mathcal{K}$ be $\xi_{j,k,f} = \{tpt_{j,f,k}, tec_{j,f,k}\}$, where $tpt_{j,f,k} = \sum_{i \in M_{f,k}} pt_{j,f,k,i}$ and $tec_{j,f,k} = \sum_{i \in M_{f,k}} ec_{j,f,k,i}$ denote the total processing time and total processing EC of job j in F_f, respectively. Notably, due to the unique characteristics of the FSP, the HDG does not explicitly represent the connection between job nodes and operation nodes, as each job contains an equal number of operations.

2) Action definition

Traditional DRL methods typically observe the environment, select an action from the action space, and execute that action to transition to the next state. In handling EDHHFSP, the agent needs to make two types of decisions: selecting a job to assign to a compatible factory and selecting an operation to assign to a compatible machine. To achieve this, we design job-factory pairs (J, F) and operation-machine pairs (O, M) to manage these two types of decisions separately. Specifically, an action $a_t \in A_t$ is expressed as a valid job-factory pair (J_j, F_f) or an operation-machine pair $(O_{j,k}, M_{f,k,i})$ at step t, where both $O_{j,k}$ and $M_{f,k,i}$ both are eligible; that is, the immediate predecessor $O_{j,k-1}$ of $O_{j,k}$ is completed and $M_{f,k,i}$ is idle. $M_{f,k,i}$ then immediately enters the setup phase for $O_{j,k}$ and proceeds to process $O_{j,k}$ (i.e., $ST_{j,k} = T(t) + st_{j,j',i,k}$). The action set A_t depends on the current step, encompassing all viable pairs. Given that, each job can have only one operation ready at a time and $|M_{f,k,i}| \leq \sum_{f \in F} l_{f,k}$, it follows that $|A_t| \leq n \cdot \sum_{f \in F} l_{f,k}$. Notably, after the initial operation of a job is assigned to a factory, all subsequent operations of that job must be assigned to the same factory. Consequently, for non-initial operations of each job, the action set A_t will be smaller.

3) State transition

Each decision step t is triggered by the completion of an operation by a machine. If multiple machines complete their operations simultaneously, the agent must make multiple scheduling decisions. After each decision, the step advances from t to $t+1$. The environment then transitions to a new state s_{t+1} based on the previous state s_t and the action a_t selected by the agent. Notably, in the initial state s_0, where all jobs and machines are available, the agent initially schedules $\sum_{f \in F} \sum_{k \in K} l_{f,k}$ operations until no further (O, M) or (J, F) pairs are available.

4) Reward function

The reward function integrates both the action space and the state representation, guiding the optimization of the strategy [13], [69]. The reward function is defined as Eq. (24), where w_1 and w_2 are weights, ensuring that the magnitudes of the first and second terms are comparable. When the discount factor $\gamma = 1$, the cumulative reward is calculated as

$$r(s_t, a_t, s_{t+1}) = w_1(C_{max}(s_t) - C_{max}(s_{t+1})) + w_2(TEC(s_t) - TEC(s_{t+1})) \tag{24}$$

$$G = \sum_{t=0}^{|O|} r(s_t, a_t, s_{t+1}) = w_1(C_{max}(s_0) - C_{max}) + w_2(TEC(s_0) - TEC). \tag{25}$$

For a given problem instance, $C_{max}(s_0)$ and $TEC(s_0)$ represent the C_{max} and TEC values of the environment in the initial state s_0, respectively, and they are both estimated constants. The method details are as follows: $C_{max}(s_0) = \max_{j \in J} CT_j$, $TEC(s_0) = \sum_{j \in J, k \in K} ec_{j,k}$. Therefore, maximizing the cumulative reward G aligns with minimizing C_{max} and TEC, which constitutes the primary objective of this paper.

5) Policy

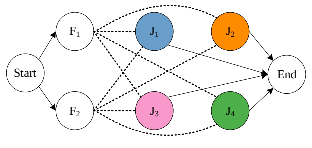
For state s_t, a stochastic policy $\pi(a_t|s_t)$ provides a distribution of the actions in A_t. In Section IV-E, we design a DRL algorithm in which π is parameterized as a neural network, and it is optimized to maximize the anticipated cumulative reward. This approach facilitates the learning of effective dispatching rules and allows for size-agnostic generalization.

B. Heterogeneous Graph

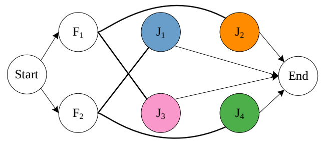
Previous studies used disjunctive graphs $\mathcal{G} = (\mathcal{O}, \mathcal{C}, \mathcal{D})$ to represent the state of JSP [58], [59] and FSP [21], where $\mathcal{O}$ is the operation nodes, $\mathcal{C}$ denotes priority constraints between operation nodes, and $\mathcal{D}$ specifies the precedence relationships between different operations processed on the same machine. For FJSP and HFSP, since each operation must be allocated to one machine from a set of parallel machines, introducing flexibility, several studies [13] have extended the conventional disjunctive graph by adding machine nodes, represented as $\mathcal{H} = (\mathcal{O}, \mathcal{M}, \mathcal{C}, \mathcal{E})$, where $\mathcal{M}$ is the collection of all machine nodes, and $\mathcal{E}$ represents the disjunctive arcs that denote assignment relationships between operations and machines. However, compared to JSP and FSP, each operation in EDHHFSP can be processed by multiple machines, leading to an expanded set of disjunctive arcs. Additionally, unlike FJSP and HFSP, each job in EDHHFSP can be assigned to any factory, which introduces heterogeneity and further increases the set of disjunctive arcs. Such a dense graph is challenging to handle efficiently. Furthermore, the processing time and EC of an operation vary significantly across compatible machines, creating additional challenges in representation.

The extended model is defined as $\mathcal{H}^D = (\mathcal{F}, \mathcal{O}, \mathcal{M}, \mathcal{C}, \mathcal{E}, \mathcal{K})$, where $\mathcal{F}$ represents the collection of factory nodes, $\mathcal{K}$ denotes the disjunctive arcs defining allocation relationships between jobs and factories. The definitions of other symbols remain consistent with those in the prior model. As shown in Fig. 3, the disjunctive arc set $\mathcal{D}$ of $\mathcal{G}$ in JSP and FSP is replaced by an operation-machine pair (O, M) arc set $\mathcal{E}$, in which each element $E_{j,f,k,i} \in \mathcal{E}$ represents an undirected arc that connects an operation node $O_{j,k}$ with a compatible machine node $M_{f,k,i}$. $\mathcal{F}$ and $\mathcal{K}$ denote the factory nodes and the arcs that define connections to jobs, respectively. Each element $K_{j,f} \in \mathcal{K}$ represents a job-factory pair (J, F). During the scheduling process, $\mathcal{E}$ and $\mathcal{K}$ are divided into two categories: dashed arcs representing unscheduled operations and solid arcs representing scheduled operations.

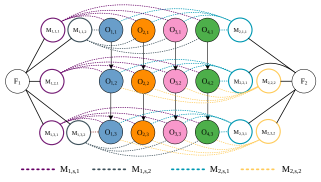
$$v_f' = \sigma(\alpha_{f,f}\mathbf{W^F}v_f + \textstyle\sum_{O_{j,k}\in\mathcal{N}_f^t} \alpha_{j,k,f}\mathbf{W^J}\mu_{j,k,f}). \tag{30}$$



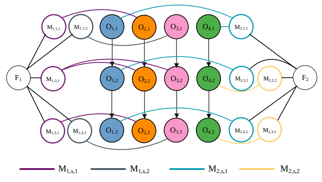
(a) An instance of factory assigning.



(c) A solution of factory assigning.



(b) An instance of machine assigning.



(d) A solution of machine assigning.

Fig. 3. Heterogeneous graph representation of EDHHFSP. Dashed lines represent processable relationships, indicating potential operations or assignments, while solid lines denote scheduled connections, reflecting confirmed operations or allocations within the scheduling framework.

The proposed HDG offers several advantages over the original disjunctive graph. Firstly, it exhibits a significantly lower graph density. Assuming each machine $M_{f,k,i}$ has $n_{f,k,i}$ processable operations, where $|\mathcal{D}| = \sum_{f\in F,k\in K,i\in M} C^2_{n_{f,k,i}} = \sum_{f\in F,k\in K,i\in M} n_{f,k,i}(n_{f,k,i} - 1)/2$, and $|\mathcal{E}| = \sum_{f\in F,k\in K,i\in M} n_{f,k,i}$. It can be demonstrated that when $n_{f,k,i} > 3$ for all $M_{f,k,i}$, then $|\mathcal{D}| > |\mathcal{E}|$. Assuming each factory F_f has n_f processable jobs, where $|\mathcal{K}| = \sum_{f\in F} n_f = f \cdot n$. For large-scale problems, the heterogeneous graph $\mathcal{H}^D$ requires significantly fewer elements in $\mathcal{E}$ and $\mathcal{K}$ compare to $\mathcal{D}$ in $\mathcal{G}$. Secondly, the factory nodes and machine nodes in $\mathcal{H}^D$ provide a straightforward and effective approach for embedding factory and machine information and extracting relevant features. This approach strongly facilitates EDHHFSP in solving both the factory assignment and machine assignment subproblems. Finally, the processing time $pt_{j,f,k,i}$ and EC $ec_{j,f,k,i}$ can be effectively incorporated as features in the disjunctive arcs $E_{j,f,k,i} \in \mathcal{E}$.

The HDG's dynamic evolution throughout the solution process ultimately yields a complete solution. Each state s_t is represented as an HDG $\mathcal{H}_t^D = (\mathcal{F}, \mathcal{O}, \mathcal{M}, \mathcal{C}, \mathcal{E}_t, \mathcal{K}_t)$, where $\mathcal{E}_t$ and $\mathcal{K}_t$ continuously evolve during the solution process. For $\mathcal{E}_t$, following an action, i.e. operation-machine pair $(O_{j,k}, M_{f,k,i})$, only $E_{j,f,k,i}$ is retained, while all other arcs connected to $O_{j,k}$ are removed. Notably, for $\mathcal{K}_t$, at time step s_1, only $K_{j,f}$ is retained, and all other factory nodes connected to the job node corresponding to the operation node $O_{j,k}$ are removed. This process continues until all operations are scheduled.

C. Heterogeneous Graph Neural Network

1) Factory node embedding

In $\mathcal{H}_t^D$, the neighbors of a factory F_f are represented by the set of jobs $\mathcal{N}_f^t$. Although these nodes are numerous, their connections are characterized by a single type (allocation relationship). In the actual scheduling process, jobs that enter the factory earlier for processing are typically prioritized over later entries due to the SDSTs in EDHHFSP. This motivates us to explore graph attention networks (GAT), which utilize an attention mechanism to determine the relative significance of different nodes. Research on traditional GATs is well-established [70]. However, as previously discussed, traditional GNNs are primarily designed for homogeneous graphs and do not incorporate edge features. To enable the computation of importance for a neighboring job to a factory, we modified the traditional GNN approach. For each factory F_f, there is only one (J, F) arc connecting it to its neighboring job, and each job is connected to its corresponding operation nodes via directed arcs. Consequently, the raw feature vector of each $O_{j,k} \in \mathcal{N}_f^t$ is augmented by combining its intrinsic raw features with those of the corresponding (J, F) arc as $\mu_{j,k,f} = [\mu_{j,k}||\xi_{j,k,f}] \in \mathbb{R}^9$. In contrast to the shared linear transformation commonly applied in traditional GNNs, we employ two separate linear transformations ($\mathbf{W^F} \in \mathbb{R}^{d\times 2}$ and $\mathbf{W^J} \in \mathbb{R}^{d\times 9}$) for factory nodes and operation nodes, respectively. Ultimately, for a factory F_f, the attention coefficient is given by:

$$e_{j,k,f} = \text{LeakyReLU}(\mathrm{a}^{\mathrm{T}}[\mathbf{W^F}v_f||\mathbf{W^J}\mu_{j,k,f}]) \tag{26}$$

where $\mathrm{a} \in \mathbb{R}^{2d}$, d denotes the embedding dimensionality. We use '$||$' to represent the concatenation of feature vectors. In this manner, information from the factory nodes and (J, F) arcs is effectively integrated into the attention computation.

The attention coefficient of the factory F_f to itself is omitted in Eq. (26). As factory nodes lack self-loop arc information, the attention coefficient for F_f to itself is calculated as follows:

$$e_{f,f} = \text{LeakyReLU}(\mathrm{a}^{\mathrm{T}}[\mathbf{W^F}v_f||\mathbf{W^F}v_f]). \tag{27}$$

Therefore, the attention coefficients for all neighboring nodes of F_f along the connected arcs (including self-loops) can be calculated. These coefficients are then normalized according to Eq. (27), yielding the normalized coefficients $\alpha_{j,k,f}$ and $\alpha_{f,f}$.

$$\alpha_{j,k,f} = \frac{\exp(e_{j,k,f})}{\sum_{u\in\mathcal{N}_p}\exp(e_{j,k,f})}. \tag{28}$$

$$\alpha_{f,f} = \frac{\exp(e_{f,f})}{\sum_{u\in\mathcal{N}_p}\exp(e_{f,f})}. \tag{29}$$

Finally, the factory embedding v_f' is calculated by aggregating its own features and those of neighboring operations.

2) Machine node embedding

Similar to factory nodes, machine nodes are characterized by a single connection type that generally involves multiple neighboring operations nodes linked by processing relationships. In actual scheduling processes, operations nearing their start times hold higher scheduling priority than those scheduled for later times. Therefore, we apply a machine node embedding approach analogous to the factory node embedding. The raw feature vector of each $O_{j,k} \in \mathcal{N}_{f,k,i}^t$ is enhanced by integrating its primary attributes with those of the corresponding (O, M) arc as $\mu_{j,f,k,i} = [\mu_{j,k}||\lambda_{j,f,k,i}] \in \mathbb{R}^9$. Next, two separate linear transformations $\mathbf{W^M} \in \mathbb{R}^{d\times 4}$ and $\mathbf{W^O} \in \mathbb{R}^{d\times 9}$ are employed for machine nodes and operation nodes, respectively. For a machine $M_{f,k,i}$, the attention coefficients $e_{j,f,k,i}$, which denotes the importance of each neighboring operation $O_{j,k} \in \mathcal{N}_{f,k,i}^t$, can be computed as:

$$e_{j,f,k,i} = \text{LeakyReLU}(\mathrm{a}^{\mathrm{T}}[\mathbf{W^M}\varpi_{f,k,i}||\mathbf{W^O}\lambda_{j,f,k,i}]). \tag{31}$$

We also consider the self-loops of machine nodes.

$$e_{f,k,i,f,k,i} = \text{LeakyReLU}(\mathrm{a}^{\mathrm{T}}[\mathbf{W^M}\varpi_{f,k,i}||\mathbf{W^M}\varpi_{f,k,i}]). \tag{32}$$

These coefficients are then normalized through the procedures in Eq. (28) and Eq. (29) to obtain the normalized coefficients $\alpha_{j,f,k,i}$ and $\alpha_{f,k,i,f,k,i}$.

Ultimately, the machine embedding $\varpi_{f,k,i}'$ is calculated by aggregating features from neighboring operations and its own features. Given the significance of processing time and EC, the augmented raw feature vector $\lambda_{j,f,k,i}$ is employed for each neighboring operation node $O_{j,k}$. The aggregation function employed to obtain $\varpi'_{f,k,i}$ is as follows:

$$\varpi'_{f,k,i} = \sigma(\alpha_{f,k,i,f,k,i}\mathbf{W}^{\boldsymbol{M}}\varpi_{f,k,i} + \textstyle\sum_{O_{j,k}\in\mathcal{N}^t_{f,k,i}} \alpha_{j,f,k,i}\mathbf{W}^{\boldsymbol{O}}\lambda_{j,f,k,i}). \quad (33)$$

3) Operation node embedding

For an operation node $O_{j,k}$, its neighbors include a predecessor operation node $O_{j,k-1}$, a successor operation node $O_{j,k+1}$, machine nodes $\mathcal{M}_{j,k} \in \mathcal{N}^t_{j,k}$, and the node itself. Operation nodes are connected via directed arcs, while they are connected to machine nodes through undirected arcs. Due to the diverse sources of information and types of arcs, the attention mechanism proves less effective in embedding operation nodes. Therefore, we employed multiple distinct multilayer perceptrons (MLPs) to process information from various node types and connection types. The results are then concatenated and projected into d-dimensional space to form the embedding for $O_{j,k}$. The embedding of operation nodes is computed using five distinct MLPs $(\mathrm{MLP}_{\eta_0}, \ldots, \mathrm{MLP}_{\eta_4})$ with d-dimensional outputs and two hidden layers of size d_h. Each of these MLPs process information from specific node types, including the predecessor operation nodes $O_{j,k-1}$, successor operation nodes $O_{j,k+1}$, machine nodes $\mathcal{M}_{j,k} \in \mathcal{N}^t_{j,k}$, and the operation node itself $O_{j,k}$. The calculation of the embedding for $O_{j,k}$ is calculated as follows:

$$\begin{gathered}\mu'_{j,k} = \mathrm{MLP}_{\eta_0}(\mathrm{ELU}[\mathrm{MLP}_{\eta_1}(\mu_{j,k-1})||\mathrm{MLP}_{\eta_2}(\mu_{j,k+1}) \\ ||\mathrm{MLP}_{\eta_3}(\bar{\varpi}'_{j,k})||\mathrm{MLP}_{\eta_4}(\mu_{j,k})])\end{gathered} \quad (34)$$

$$\bar{\varpi}'_{j,k} = \sum_{f\in F,\mathcal{M}_{j,k}\in\mathcal{N}^t_{j,k}} \varpi'_{f,k,i} \quad (35)$$

Since $\mathcal{N}^t_{j,k}$ may include multiple machines, the summation is performed over all available machine nodes in $\mathcal{N}^t_{j,k}$ by Eq. (35).

4) Stacking and pooling

The embedding v'_f of F_f, the embedding $\varpi'_{f,k,i}$ of $M_{f,k,i}$, and the embedding $\mu'_{j,k}$ of $O_{j,k}$ are considered to represent a single HGNN layer. Then, the standard mean pooling method is employed to compute the average value of all node embeddings before concatenating. The resulting embedding $h^D_t \in \mathbb{R}^{3d}$ of the HDG state $\mathcal{H}^D_t$, composed of three d-dimensional vectors, is then combined as follows:

$$h^D_t = \left[\frac{1}{|\mathcal{F}|}\sum_{F_f\in\mathcal{F}} v'^L_f \,||\, \frac{1}{|\mathcal{M}|}\sum_{M_{f,k,i}\in\mathcal{M}} \varpi'^L_{f,k,i} \,||\, \frac{1}{|\mathcal{O}|}\sum_{O_{j,k}\in\mathcal{O}} \mu'^L_{j,k}\right]. \quad (36)$$

Through the above process, the five feature vectors of varying sizes in $\mathcal{H}^D_t$ are ultimately transformed into a fixed-dimension embedding.

D. Decision-making Process

In the decision-making process at step t, $\pi_{\omega^1}(a_t|s_t)$ and $\pi_{\omega^2}(a_t|s_t)$ are designed to evaluate and score the two types of candidate actions. The factory node embedding v'_f is concatenated with the operation node embedding $\mu'_{j,k}$ to form a vector representing the job-factory pair. Similarly, the machine node embedding $\varpi'^L_{f,k,i}$ is concatenated with the operation node embedding $\mu'_{j,k}$ to represent the operation-machine pair. These concatenated vectors are then input into two MLPs to calculate the priority $Pr(a_t|s_t)$ of each action, as shown in Eq. (37),

$$\mathrm{Pr}(a_t|s_t) = \begin{cases} \mathrm{MLP}_{\omega^1}[\mu'^L_{j,k}||v'^L_f||h^D_t], a_t \in (J_j, F_f) \\ \mathrm{MLP}_{\omega^2}[\mu'^L_{j,k}||\varpi'^L_{f,k,i}||h^D_t], a_t \in (O_{j,k}, M_{f,k,i}) \end{cases} \quad (37)$$

where MLP_{ω^1} and MLP_{ω^2} are structured with two hidden layers, each containing d_{ω^i} neurons and employing the tanh activation function. Further, the softmax function is employed to calculate the probability of selecting each a_t, thereby guiding the action sampling process during training, as depicted in Eq. (39).

$$\pi_{\omega^i}(a_t|s_t) = \frac{\exp(\mathrm{Pr}(a_t|s_t))}{\sum_{a'_t\in A_t}\exp(\mathrm{Pr}(a'_t|s_t))}, i \in \{1,2\} \quad (38)$$

$$a_t \leftarrow \mathrm{argmax}_{\forall a'_t\in A_t}\pi_{\omega^i}(a'_t|s_t), i \in \{1,2\} \quad (39)$$

During the training process, actions are sampled according to the policy's probability distribution $\pi_{\omega^i}(a_t|s_t)$ to improve the algorithm's exploration. During testing, actions are selected greedily by choosing the action with the highest probability as calculated by Eq. (39).

E. Algorithm for Training

We employ the PPO algorithm, which has proven effective in training tasks due to its actor-critic structure within the policy gradient method. The actor refers to the policy network π_ω, responsible for determining actions, while the critic, represented by the value network v_ϕ, evaluates the state s_t to predict its value. As shown in Eq. (40), the critic is an MLP structure with a parameter ϕ. It takes the state embedding h^D_t derived from the HGNN as input to derive the state value function $v_\phi(s_t)$. Further, the critic computes an estimator $\widehat{A_t}$ of the advantage function with reduced variance, as shown in Eq. (41), where γ represents a discount factor, t' is the current step, and $r_{t'}$ is the reward at time step t'.

$$v_\phi(s_t) = \mathrm{MLP}_\omega(h^D_t) \quad (40)$$

$$\widehat{A_t} = -v_\phi(s_{t'}) + \sum_{t=t'}^{T}\gamma^{t'}r_{t'} \quad (41)$$

The loss function of the algorithm is given as follows:

$$L_{\omega^i,\phi} = c_pL^{CLIP}(\omega^i) - c_vL^{VF}(\phi) + c_eL^S(\omega^i), i \in \{1,2\} \quad (42)$$

$$\begin{gathered} L^{CLIP}(\omega^i) = \min(r_t(\omega^i), clip(r_t(\omega^i), 1-\epsilon, 1+\epsilon)) \\ \widehat{A_t}, r_t(\omega^i) = \frac{\pi_{\omega^i}(a_t|s_t)}{\pi_{\omega^i{}_{old}}(a_t|s_t)}, i \in \{1,2\} \end{gathered} \quad (43)$$

$$L^{VF}(\phi) = (v_\phi(s_t) - v_t\widehat{A_t})^2 \quad (44)$$

$$L^S(\omega^i) = \sum_{t=0}^{T} S(\pi_{\omega^i}(a_t|s_t)), i \in \{1,2\} \quad (45)$$

where c_p, c_v, and c_e are policy loss coefficient, value function loss coefficient, and entropy loss coefficient, respectively. $clip$ and ϵ are clipping function and clipping ratio, respectively.

Algorithm 1: Training process with PPO

Input: Problem size: $(n, m, \delta, l_{f,k})$;
Distribution: $p_{j,k} \sim U(low_1, high_1)$; $st_{j,j',i,k} \sim U(low_2, high_2)$;
Network: HGNN network η, policy network ω, and critic network ϕ;
Output: optimized policy network; optimized critic network;

```
1  Generate EDHHFSP instances;
2  Sample a batch of $B$ EDHHFSP instances;
3  for $iter = 1$ to $IT$ 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 $(v'^L_f, \varpi'^L_{f,k,i}, \mu'^L_{j,k}, h^D_t)$;
8              if the timestep $t$ is in $K_1$ then // $a_t \in (J_j, F_f)$
9                  Sample $a_t \sim \pi_{\omega^1}(\cdot|s_t)$;
10             else // $a_t \in (O_{j,k}, M_{f,k,i})$
11                 Sample $a_t \sim \pi_{\omega^2}(\cdot|s_t)$;
12             endif
13             Calculate reward $r_t$ and next state $s_{t+1}$;
14             State transition;
15         end while
16         Calculate the advantages estimator $\widehat{A_t}$ by Eq. (41);
17     end for
18     for $ep = 1$ to $R$ do
19         Calculate the loss $L_{\omega,\phi}$ by Eq. (42);
20         Optimize the parameters $\eta$, $\omega$, and $\phi$;
21         Update all parameters;
22     end for
23     if $iter$ mod $10 = 0$ then
24         Validate the policy;
25     end if
26     if $iter$ mod $20 = 0$ then
27         Break;
28     end if
29 end for
Return
```

The network structure is depicted in Fig. 4. Algorithm 1 outlines the complete training procedure. The training continues for IT iterations. Importantly, to maintain sample diversity and mitigate overfitting, the DRL agent processes batches of B instances in parallel during training, with instances being replaced every 20 iterations (Lines 4-17). Moreover, to enhance the convergence of the algorithm, the policy is evaluated on a separate set of validation instances every ten iterations (Lines 23-25).

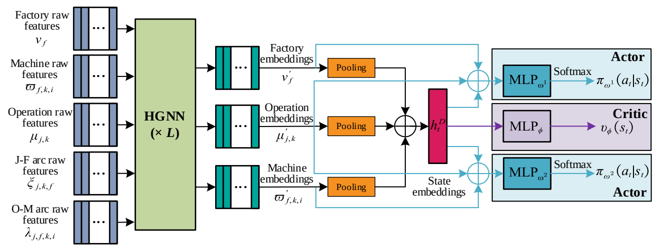

Factory raw features v_f
Machine raw features $\varpi_{f,k,i}$
Operation raw features $\mu_{j,k}$
J-F arc raw features $\xi_{j,k,f}$
O-M arc raw features $\lambda_{j,f,k,i}$
HGNN (× L)
Factory embeddings v'_f
Operation embeddings $\mu'_{j,k}$
Machine embeddings $\varpi'_{f,k,i}$
Pooling
Pooling
Pooling
h_t^D
State embeddings
Actor
MLP$_{\omega^1}$
Softmax
$\pi_{\omega^1}(a_t|s_t)$
Critic
MLP$_\phi$
$v_\phi(s_t)$
MLP$_{\omega^2}$
Softmax
$\pi_{\omega^2}(a_t|s_t)$
Actor

Fig. 4. The network architecture.

V. Experiments and Analysis

A. Experimental Setup

1) Evaluation instances

As with many related studies, this paper generates synthetic instances of the EDHHFSP for both training and testing purposes. The parameters used for generating these instances are detailed in Table S-VI. In this study, training is conducted on the 20 smaller-sized groups, encompassing all instances with $n = 15$ and $n = 20$, to develop effective scheduling policies. To evaluate the generalization capabilities of these trained policies, the largest 20 groups, with sizes $n = 40$ and $n = 60$, are used. Training instances are dynamically generated and include 40 validation instances to ensure robustness, while testing involves sampling 40 instances for each size to thoroughly assess performance across different scales.

2) Hyperparameters settings

We conducted training on instances with parameters $\delta = 2$, $n = 15$, and $m = 5$ to fine-tune and optimize algorithm parameters for enhanced performance. These parameters are divided into two categories: those related to the HGNN and those pertaining to PPO. Detailed parameter specifications can be found in Table S-VII. An experimental analysis was conducted on different weights in the reward function. For detailed results and discussion, please refer to Section S-VIII in the Supplementary Material. Based on the experimental findings, the reward function weight is defined as $w_1 = 0.8, w_2 = 0.2$.The training was executed on a system equipped with Windows 11 64-bit OS, an Intel(R) Xeon(R) Silver 4210R CPU @2.40GHz, 128 GBytes of RAM, and an NVIDIA GeForce GTX TITAN X GPU, providing substantial computational power for efficient processing and model training.

3) Comparison methods

We compare the proposed method with several existing approaches:

- PDRs: longest processing time (LPT), shortest processing time (SPT);
- Metaheuristic algorithms: HAS [30], MMMA [8], QABC [2];
- Multi-objective optimization method: NSGA-II [6];
- DRL-based method: DRL-FJSP [13], DRLCEA [53].

We have redesigned these comparative algorithms to adapt to EDHHFSP. Please refer to the supplementary materials for more information. For fair comparisons, the termination criteria of HAS, MMMA, NSGA-II, QABC, and DRLCEA are based on the maximum elapsed CPU time ($c \times n \times m \times \delta$) milliseconds, with the c set to 20 [7]. Hypervolume (HV) and inverse generational distance (IGD) metrics are considered in this paper.

B. Performance on Training

1) The convergence performance analysis of DRL

To analyze the convergence of the proposed DRL, we present the training outcomes for $n = 20, m = 5, \delta = 4$. We track the rewards and losses across each iteration. Fig. S-7 illustrates the reward and loss curves for these training results. By analyzing the smoothed curve, generated using the Savitzky-Golay method, it is evident that a marked increase during the initial 400 iterations and then levels off, demonstrating effective learning based on the proposed reward function. Moreover, the training loss steadily decreases with more iterations, indicating consistent model learning across iterations.

2) Evaluation of instances of training size

For each of the training sizes, Table S-IX provides the average HV and IGD based on 40 test instances sampled from the training data distribution. As CPLEX is an exact solver for combinatorial optimization problems, its runtime is capped at 1800 seconds. Only a subset of the smaller instances is successfully resolved within this time limit, highlighting the intricate nature of the EDHHFSP. The other instances with $n = 20$ are challenges for exact solvers due to the NP-hardness of EDHHFSP. Furthermore, DRL approach consistently outperforms two PDRs (LPT and SPT) across the 20 training sizes. This observation corroborates the point made in Section II-A that PDRs are inherently tied to the objective function, focusing solely on minimizing C_{max} and overlooking another objective. When compared with other commonly used methods, DRL remains effective in addressing the EDHHFSP. In certain cases, the two DRL-based methods and the QABC algorithm achieved comparable performance to our proposed DRL approach on the training instances, which may be attributed to their scheduling mechanisms that better accommodate the specific complexities of these instances.

Fig. S-8 of the Supplementary Material presents violin plots depicting the average IGD across training instances under different numbers of factories. These plots not only illustrate central tendencies but also visualize the variance and distribution spread of each algorithm's performance. Among the eight methods tested on small-scale EDHHFSP instances with varying numbers of factories, both the DRL-based approach and the exact solver CPLEX consistently yield lower IGD values, indicating their superior convergence toward the true Pareto front. Interestingly, DRL achieves slightly better IGD values than CPLEX in cases where $\delta = 3$, $\delta = 5$, and $\delta = 6$. This suggests that DRL is capable of learning generalizable scheduling policies that not only approximate optimal solutions but can sometimes surpass the performance of exact solvers, especially when the problem structure favors learned heuristics over exhaustive search. Two PDRs exhibit the weakest performance across most test cases, with IGD values significantly higher than those of DRL and other metaheuristics. This performance gap highlights the inherent limitations of rule-based methods in handling multi-objective optimization, particularly when objectives are conflicting. Since PDRs often follow greedy, one-shot decision-making heuristics, they fail to explore the solution space effectively. The relatively strong performance of metaheuristic algorithms such as QABC and NSGA-II can be attributed to their iterative refinement processes and efficient local search strategies, which help to continuously update and expand the Pareto front. These strategies allow the algorithms to escape local optima and promote better distribution and diversity of solutions. The performance of DRL across different δ values demonstrate its robustness and scalability to changes in factory configuration. The consistent improvement across various configurations implies that DRL can adaptively learn decision policies that generalize across different scheduling environments. This generalizability is a key advantage over traditional methods, which often require manual tuning or redesign when problem parameters change.

C. Comparison of Other Algorithms

We further investigate the generalization capability of the proposed policy by applying it to larger instances. Specifically, policies trained on instances with parameters $n = 20$, $m = 10$, and $\delta = 2,3,4,5,6$ are directly used on instances with $n = 40,60$, and $m = 5,10$. These five policies were tested on large instances with varying factory counts, ensuring that the number of factories matched those utilized during the training of each respective policy.

1) *Ablation studies*

The key components that distinguish DRL from other baselines are the action selection based on two pairs and the heterogeneous graph-based state feature embedding. In this section, we deactivate both components. Instead, we implement a classic action execution scheme and a disjunctive graph structure [13], resulting in two modified algorithms, referred to as DRL-P and DRL-G, respectively, to assess the contribution of these components to overall performance.

As shown in Fig. S-9, DRL outperforms the other two methods across all instance scales, with statistically significant differences. Additionally, the heterogeneous graph structure contributes significantly to the enhanced performance of this method, as indicated by the lower p-value for DRL-G compared to DRL-P. This improvement is due to the heterogeneous graph structure's more accurate embedding of factory and machine information under various states compared to the commonly used disjunctive graph, allowing agents to extract valuable features that distinguish factory and machine conditions. Furthermore, the actions based on two pairs design enable agents to make rapid decisions at various scheduling moments, bypassing the complexity of separately addressing assignment and routing issues as seen in prior studies.

2) *Generalization performance on large-sized instances*

The outcomes of large-sized instances are presented in Table S-X. The data shows that the proposed method maintains its performance advantage in these larger instances, indicating that patterns learned from small and medium-sized instances remain effective when applied to larger ones. Fig. S-10 presents violin plots of the average IGD on large-sized instances across all algorithms in different scenarios. Additionally, the three DRL-based methods consistently outperform metaheuristic algorithms and multi-objective optimization methods. Furthermore, this advantage becomes even more pronounced as the problem size increases. To visually assess the performance of various algorithms, the Pareto frontiers between seven algorithms for some large-sized instances are illustrated in Fig. S-11. The Pareto frontiers represent the outcomes of all runs. DRL provides a wider range of non-dominated solutions compared to other methods, thereby aiding decision-makers in balancing these conflicting objectives.

Subsequently, the Friedman test is conducted for statistical comparison. The outcomes are presented in Table I and Fig. S-12, where CN represents the count of instances. DRL achieves the highest ranking among the seven algorithms. It is evident that the p-values are all below 0.05 across all large-sized instances. This indicates that the majority of nondominated solutions generated by DRL outperform those obtained by the other methods. In Fig. S-12, the solid and dashed lines represent the critical difference (CD) at 95% and 90% confidence intervals, respectively. In most cases, DRL demonstrates significant performance advantages over the other algorithms, as the DRL bar charts do not coincide with those of the six other algorithms.

TABLE I
RESULTS ACHIEVED BY FRIEDMAN TEST

Algorithms	$n = 40, m = 5$		$n = 40, m = 10$		$n = 60, m = 5$		$n = 60, m = 10$	
	IGD	HV	IGD	HV	IGD	HV	IGD	HV
DRL	3.44	7.21	2.74	7.19	2.67	6.83	1.78	7.02
LPT	6.99	4.62	7.31	4.42	6.28	5.31	7.22	5.82
SPT	7.47	1.33	7.36	1.48	7.59	1.74	7.73	2.53
HAS	5.37	2.59	7.31	3.22	6.60	2.36	6.60	2.76
MMMA	4.05	5.84	3.36	5.35	4.24	5.34	3.95	5.32
NSGA-II	5.78	2.50	6.67	4.62	3.76	3.30	6.53	3.12
DRL-FJSP	3.54	4.60	3.28	2.18	3.86	3.37	3.25	3.80
QABC	4.83	3.04	3.68	2.21	6.63	3.00	5.28	3.71
DRLCEA	3.54	5.88	3.28	6.60	3.37	5.98	2.65	6.21
CN	200	200	200	200	200	200	200	200
p-value	5.05E-05	4.22E-05	5.08E-05	2.06E-04	4.00E-04	3.21E-04	2.07E-04	4.25E-04
Crit. Diff $\alpha = 0.05$	0.749							
Crit. Diff $\alpha = 0.1$	0.684							

Furthermore, as shown in Fig. S-12(a), three DRL-based methods generally fall below the CD lines, indicating that there is no statistically significant difference among them under most settings. However, the performance ranking shows that the proposed DRL method consistently outperforms both DRL-FJSP and DRLCEA, with p-values approaching zero. This suggests that, although the differences are not always statistically significant, DRL tends to achieve better performance in the majority of test cases. More importantly, as the problem size increases, the performance gap between DRL and the other two DRL variants becomes more pronounced, highlighting the superior capability of the proposed method in extracting rich and representative state features in complex distributed environments.

3) *Run time analysis*

Additionally, we incorporated the average running time (ART) for each instance type in Table S-X to assess the balance between solution time and result quality. It is evident that the PDRs, specifically LPT and SPT, maintain high efficiency, with only a slight increase in runtime as the problem size increases. In comparison, the runtime for DRL and DRL-FJSP is longer than that of the PDRs. This is partially due to the fact that DRL generates scheduling actions through the neural network framework utilizing an HGNN, which can be relatively time-intensive. Nonetheless, given that the training process is conducted offline, this trade-off between computation time and solution quality is deemed acceptable. Fig. S-13 presents the runtime curves of the two DRL-based methods in comparison with five other algorithms—HAS, MMMA, NSGA-II, QABC, and DRLCEA—across instances of different scales. As the number of jobs increases, both DRL variants show a noticeable increase in runtime, which is expected due to the higher computational complexity of larger instances. Nevertheless, the runtime gap between the two DRL methods remains consistently small, indicating comparable inference efficiency. In contrast, the runtime of other methods grows more steeply with the problem size, as they require extensive iterative exploration to search the solution space. These results highlight the advantage of DRL approaches in large-scale scenarios, where the use of pretrained models enables rapid inference and significantly improves computational efficiency in obtaining high-quality scheduling solutions.

D. Industrial Case Study

The empirical validation of this study is carried out in a Printed Circuit Board Assembly (PCBA) manufacturing workshop located in Shandong Province, China. The workshop operates two production lines, denoted as F_1 and F_2, which are depicted in Fig. S-1 of Section I. These lines encompass the complete Dual In-line Package (DIP) process, comprising 10 distinct operations. Detailed configurations for each production line are outlined in Section S-XI of Supplementary Material. The study focuses on four specific types of PCBA products, selected for experimental analysis, with their detailed processing information available in Table S-XI of Supplementary Material.

Using the provided production information, the proposed DRL method generates a scheduling scheme. Fig. S-14 of Supplementary Material presents the Gantt chart of this scheme, with the corresponding C_{max} and TEC values of 186 and 7539.97, respectively. In real-world production settings, creating production plans is often time-intensive and inefficient. However, the proposed DRL method proves to be highly effective. For the test cases, the scheduling plan is generated in just 1.3673 seconds, a feat impractical to achieve manually. This practical application of DRL highlights its engineering value and significantly enhances production efficiency.

E. Discussion and Analysis

In the experiments conducted, the DRL approach consistently outperforms other algorithms according to the HV and IGD metrics. Simple PDRs tend to focus on a single objective, which makes them ineffective in reducing energy consumption and leads to poorly distributed solutions. Metaheuristic algorithms, while versatile, often experience slow convergence and diminished precision due to the lack of problem-specific insights. Classical multi-objective optimization methods, although designed to optimize two objectives concurrently through non-dominated sorting, struggle with the complexity of the EDHHFSP. This, coupled with the computational demands of large-scale instances, results in considerable time consumption when calculating dominance relations. The challenges are further amplified in large-scale problems, leading to suboptimal performance.

The runtime analysis indicates that the time cost associated with the DRL solution process is well within an acceptable range. Significantly, unlike metaheuristics and multi-objective optimization methods, once DRL is trained to produce a robust scheduling policy, it can efficiently handle EDHHFSP instances of varying complexities through a single computational process. Additionally, the HGNN can be configured in a parallel setup, enabling the concurrent processing of multiple instances. As demonstrated in Section V-C, DRL does not necessitate repeated adjustments of model parameters when addressing instances of differing complexities. This adaptability presents a considerable advantage for the practical application of DRL in real-world scenarios, streamlining operations and enhancing efficiency.

VI. Conclusion and Future Work

This paper introduces an end-to-end DRL approach designed to develop high-quality scheduling policies for the EDHHFSP. At the heart of this approach is an MDP framework that seamlessly integrates factory and machine allocation within a unified decision-making structure. We introduce a heterogeneous graph structure to represent the scheduling state, comprising diverse node types interconnected by various arcs. To process this complex graph, a novel HGNN model is employed. Building on this foundation, we developed an actor-critic architecture, which is trained using the PPO algorithm. Comprehensive experiments were conducted to assess the proposed method across multiple performance dimensions, including training efficiency, robustness, generalizability, and runtime analysis. The results indicate strong performance across all these dimensions, validating the effectiveness of our approach.

We adopted a weighted sum approach to scalarize multiple objectives into a single reward function. While this method is computationally efficient and aligns with the cumulative reward maximization objective of the PPO algorithm, it may not fully capture the trade-offs between objectives or preserve the entire Pareto front, particularly in non-convex regions. In future work, we aim to explore more advanced multi-objective RL techniques to overcome this limitation.

Scheduling is a complex and challenging task in real-world applications. Future work aims to incorporate additional real-world constraints into the environment, such as transportation times of workpieces between machines, batching processes, and system uncertainties including order cancellations and emergency orders. Furthermore, the end-to-end DRL approach should be refined to better extract scheduling state features and integrate advanced search mechanisms, ultimately improving training performance and robustness.

References

[1] Y. Li, X. Li, L. Gao, and Z. Lu, "Multi-agent deep reinforcement learning for dynamic reconfigurable shop scheduling considering batch processing and worker cooperation," *Robot. Comput.-Integr. Manuf.*, vol. 91, Art. no. 102834, Feb. 2025.

[2] F. Zhao, Z. Wang, and L. Wang, "A reinforcement learning driven artificial bee colony algorithm for distributed heterogeneous no-wait flowshop scheduling problem with sequence-dependent setup times," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 4, pp. 2305–2320, Oct. 2023.

[3] L. Zhao, J. Fan, C. Zhang, W. Shen, and J. Zhuang, "A DRL-based reactive scheduling policy for flexible job shops with random job arrivals," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 2912–2923, May 2023.

[4] 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, Art. no. 102997, Feb. 2024.

[5] J. Wang and L. Wang, "A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 26, no. 3, pp. 461–475, Jun. 2022.

[6] Hasani and S. M. H. 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, Art. no. 125533, Dec. 2020.

[7] 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, Art. no. 101399, Dec. 2023.

[8] 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.

[9] F. Zhao, H. Zhang, L. Wang, T. Xu, N. Zhu, and J. Jonrinaldi, "A multi-objective discrete differential evolution algorithm for energy-efficient distributed blocking flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 62, no. 12, pp. 4226–4244, Sep. 2023.

[10] 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.

[11] X. He, Q. Pan, L. Gao, L. Wang, and P. N. Suganthan, "A greedy cooperative co-evolutionary algorithm with problem-specific knowledge for multiobjective flowshop group scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 27, no. 3, pp. 430–444, Jun. 2023.

[12] Hasani and S. M. H. Hosseini, "Auxiliary resource planning in a flexible flow shop scheduling problem considering stage skipping," *Comput. Oper. Res.*, vol. 138, Art. no. 105625, Feb. 2022.

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

[14] Li, X. Zhao, H. Cao, L. Li, and X. Chen, "A data and knowledge-driven cutting parameter adaptive optimization method considering dynamic tool wear," *Robot. Comput.-Integr. Manuf.*, vol. 81, Art. no. 102491, Jun. 2023.

[15] X. Zhao, C. Li, Y. Tang, X. Li, and X. Chen, "Reinforcement learning-based cutting parameter dynamic decision method considering tool wear for a turning machining process," *Int. J. Precis. Eng. Manuf.-Green Tech.*, vol. 11, no. 4, pp. 1053–1070, Jan. 2024.

[16] W. Song, N. Mi, Q. Li, J. Zhuang, and Z. Cao, "Stochastic economic lot scheduling via self-attention based deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1457–1468, Apr. 2024.

[17] S. Luo, L. Zhang, and Y. Fan, "Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3020–3038, Oct. 2022.

[18] 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.

[19] 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.

[20] S. Luo, L. Zhang, and Y. Fan, "Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning," *Comput. Ind. Eng.*, vol. 159, Art. no. 107489, Sep. 2021.

[21] 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, Art. no. 109802, Jan. 2024.

[22] 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.

[23] S. A. Brah and J. L. Hunsucker, "Branch and bound algorithm for the flow shop with multiple processors," *Eur. J. Oper. Res.*, vol. 51, no. 1, pp. 88–99, Mar. 1991.

[24] B. Zhang, Q. Pan, L. Gao, L. Meng, X. Li, and K. Peng, "A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem," *IEEE Trans. Syst. Man Cybern.-Syst.*, vol. 50, no. 12, pp. 4984–4999, Dec. 2020.

[25] B. Naderi, S. Gohari, and M. Yazdani, "Hybrid flexible flowshop problems: Models and solution methods," *Appl. Math. Model.*, vol. 38, no. 24, pp. 5767–5780, Dec. 2014.

[26] Y. Pan, K. Gao, Z. Li, and N. Wu, "Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 1, pp. 361–371, Feb. 2022.

[27] J. R. Montoya-Torres, E. L. Solano-Charris, and A. Muñoz-Villamizar, "Assessing the performance of dispatching policies for hybrid flowshop manufacturing systems," *IFAC-PapersOnLine*, vol. 49, no. 31, pp. 109–113, 2016.

[28] Kahraman, O. Engin, İ. Kaya, and R. E. Öztürk, "Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach," *Appl. Soft Comput.*, vol. 10, no. 4, pp. 1293–1300, Sep. 2010.

[29] H. Cui, X. Li, and L. Gao, "An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem," *Expert Syst. Appl.*, vol. 222, Art. no. 119805, Jul. 2023.

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

[31] C. Li, X. Chen, Y. Tang, and L. Li, "Selection of optimum parameters in multi-pass face milling for maximum energy efficiency and minimum production cost," *J. Clean. Prod*., vol. 140, pp. 1805–1818, Jan. 2017.

[32] Defersha, D. Obimuyiwa, and A. Yimer, "Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem," *Comput. Ind. Eng*., vol. 171, Art. no. 108487, Sep. 2022.

[33] Mouzon, M. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *Int. J. Prod. Res*., vol. 45, no. 18, pp. 4247–4271, Sep. 2007.

[34] M. Dai, D. Tang, A. Giret, M. Salido, and W. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 5, pp. 418–429, Oct. 2013.

[35] L. Meng, C. Zhang, X. Shao, Y. Ren, and C. Ren, "Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines," *Int. J. Prod. Res.*, vol. 57, no. 4, pp. 1119–1145, Jul. 2018.

[36] S. Wang, X. Wang, F. Chu, and J. Yu, "An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production," *Int. J. Prod. Res.*, vol. 58, no. 8, pp. 2283–2314, Jun. 2019.

[37] Bruzzone, D. Anghinolfi, M. Paolucci, and F. Tonelli, "Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops," *CIRP Ann-Manuf. Technol.*, vol. 61, no. 1, pp. 459–462, Jan. 2012.

[38] J. Li, H. Sang, Y. Han, C. Wang, and K. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Clean. Prod.*, vol. 181, pp. 584–598, Apr. 2018.

[39] Gong, R. Chiong, Q. Deng, W. Han, L. Zhang, W. Lin, and K. Lin, "Energy-efficient flexible flow shop scheduling with worker flexibility," *Expert Syst. Appl.*, vol. 141, Art. no. 112902, Mar. 2020.

[40] J. Behnamian and SMTF Ghomi, "A survey of multi-factory scheduling," *J. Intell. Manuf.*, vol. 27, no. 1, pp. 231–249, Mar. 2014.

[41] W. Shao, Z. Shao, and D. Pi, "Effective constructive heuristics for distributed no-wait flexible flow shop scheduling problem," *Comput. Oper. Res*., vol. 136, Art. no. 105482, Dec. 2021.

[42] W. Shao, Z. Shao, and D. Pi, "Multi-objective evolutionary algorithm based on multiple neighborhoods local search for multi-objective distributed hybrid flow shop scheduling problem," *Expert Syst. Appl.*, vol. 183, Art. no. 115453, Nov. 2021.

[43] J. Li, X. Chen, P. Duan, and J. Mou, "KMOEA: A knowledge-based multiobjective algorithm for distributed hybrid flow shop in a prefabricated system," *IEEE Trans. Ind. Inf.*, vol. 18, no. 8, pp. 5318–5329, Aug. 2022.

[44] C. Lu, J. Zhou, L. Gao, X. Li, and J. Wang, "Modeling and multi-objective optimization for energy-aware scheduling of distributed hybrid flow-shop," *Appl. Soft Comput.*, vol. 156, Art. no. 111508, Mar. 2024.

[45] F. Yu, C. Lu, J. Zhou, and L. Yin, "Mathematical model and knowledge-based iterated greedy algorithm for distributed assembly hybrid flow shop scheduling problem with dual-resource constraints," *Expert Syst. Appl.*, vol. 239, Art. no. 122434, Apr. 2024.

[46] F. Liu, G. Li, C. Lu, L. Yin, and J. Zhou, "A tri-individual iterated greedy algorithm for the distributed hybrid flow shop with blocking," *Expert Syst. Appl.*, vol. 237, Art. no. 121667, Mar. 2024.

[47] Z. Shao, W. Shao, J. Chen, and D. Pi, "MQL-MM: A meta-q-learning-based multi-objective metaheuristic for energy-efficient distributed fuzzy hybrid blocking flow-shop scheduling problem," *IEEE Trans. Evol. Comput.*, early access, May. 10, 2024, doi:10.1109/tevc.2024.3399314.

[48] D. Lei and B. Su, "A multi-class teaching–learning-based optimization for multi-objective distributed hybrid flow shop scheduling," *Knowledge-Based Syst.*, vol. 263, Art. no. 110252, Mar. 2023.

[49] Qin, Y. Han, Y. Liu, J. Li, Q. Pan, and X. Han, "A collaborative iterative greedy algorithm for the scheduling of distributed heterogeneous hybrid flow shop with blocking constraints," *Expert Syst. Appl.*, vol. 201, Art. no. 117256, Sep. 2022.

[50] W. Shao, Z. Shao, and D. Pi, "Modelling and optimization of distributed heterogeneous hybrid flow shop lot-streaming scheduling problem," *Expert Syst. Appl.*, vol. 214, Art. no. 119151, Mar. 2023.

[51] Xuan, W. Li, and B. Li, "An artificial immune differential evolution algorithm for scheduling a distributed heterogeneous flexible flowshop," *App. Soft Comput.* vol. 145, Art. no. 110563, Jul. 2023.

[52] C. Lu, Q. Liu, B. Zhang, and L. Yin, "A Pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop," *Expert Syst. Appl.*, vol. 204, Art. no. 117555, Oct. 2022.

[53] 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.

[54] C. Lin, D. Deng, Y. Chih, and H. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep Q network," *IEEE Trans. Ind. Inf.*, vol. 15, no. 7, pp. 4276–4284, Jul. 2019.

[55] Y. Kwon, J. Choo, I. Yoon, M. Park, D. Park, and Y. Gwon, "Matrix encoding networks for neural combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5138–5149, 2021.

[56] F. Ni, J. Hao, J. Lu, X. Tong, M. Yuan, J. Duan, Y. Ma, and K. He, "A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem," *in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, New York, USA*, pp. 3441–3451, Aug. 2021.

[57] Z. Pan, L. Wang, J. Wang, and J. Lu, "Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 7, no. 4, pp. 983–994, Aug. 2023.

[58] C. Zhang, W. Song, Z. Cao, J. Zhang, P. Tan, and C. Xu, "Learning to dispatch for job shop scheduling via deep reinforcement learning," *Advances in Neural Information Processing Systems,* vol. 33, pp. 1621–1632, 2020.

[59] 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.

[60] R. Wang, Y. Jing, C. Gu, S. He, and J. Chen, "End-to-end multitarget flexible job shop scheduling with deep reinforcement learning," *IEEE Internet Things J.*, vol. 12, no. 4, pp. 4420–4434, Feb. 2025.

[61] S. Moon, S. Lee, and K.-J. Park, "Learning-enabled flexible job-shop scheduling for scalable smart manufacturing," *J. Manuf. Syst.*, vol. 77, no. 4, pp. 356–367, Oct. 2024.

[62] Wan, L. Fu, C. Li, and K. Li, "Flexible job shop scheduling via deep reinforcement learning with meta-path-based heterogeneous graph neural network," *Knowledge-Based Syst.*, vol. 296, Art. no. 111940, Jul. 2024.

[63] Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Trans. Ind. Inf.*, vol. 20, no. 1, pp. 1007–1018, Jan. 2024.

[64] Zhang, L. Wang, F. Qiu, and X. Liu, "Dynamic scheduling for flexible job shop with insufficient transportation resources via graph neural network and deep reinforcement learning," *Comput. Ind. Eng.*, vol. 186, Art. no. 109718, Dec. 2023.

[65] 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.

[66] Z. Pan, L. Wang, C. Dong, and J. Chen, "A knowledge-guided end-to-end optimization framework based on reinforcement learning for flow shop scheduling," *IEEE Trans. Ind. Inf.*, vol. 20, no. 2, pp. 1853–1861, Feb. 2024.

[67] 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, Jan. 2025.

[68] S. A. Mansouri, E. Aktas, and U. Besikci, “Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption,” *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 772–788, Feb. 2016.
[69] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, and G. Wu, “A generic Markov decision process model and reinforcement learning method for scheduling agile earth observation satellites,” *IEEE Trans. Syst. Man Cybern.-Syst.*, vol. 52, no. 3, pp. 1463–1474, Mar. 2022.
[70] B. Jiang, Y. Lu, X. Chen, X. Lu, and G. Lu, “Graph attention in attention network for image denoising,” *IEEE Trans. Syst. Man Cybern.-Syst.*, vol. 53, no. 11, pp. 7077–7088, Nov. 2023.

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. at the Shanghai University in China and also at the National University of Singapore. His research interests include intelligent optimization, reinforcement learning, and scheduling.

Quanke Pan (Member, IEEE) received the B.Sc. degree and the Ph.D. degree from Nanjing university of Aeronautics and Astronautics, Nanjing, China, in 1993 and 2003, respectively. From 2003 to 2011, he was with School of Computer Science Department, Liaocheng University, where he became a Full Professor in 2006. From 2011 to 2014, he was with State Key Laboratory of Synthetical Automation for Process Industries (Northeastern University), Shenyang, China. From 2014 to 2015, he was with State Key Laboratory of Digital Manufacturing and Equipment Technology (Huazhong University of Science and Technology). He has been with School of Mechatronic Engineering and Automation, Shanghai University since 2015. His current research interests include intelligent optimization and scheduling algorithms. He has authored two academic books and more than 300 refereed papers. He 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 his 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 5 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 Xi-dian 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, State Key Laboratory of Intelligent Manufacturing Equipment Technology, 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. He 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*.