



Multi-agent cooperative multi-network group framework for energy-efficient distributed fuzzy flexible job shop scheduling problem

Zi-Qi Zhang^{a,b,*}, Xiao-Wei Li^{a,b,1}, Bin Qian^{a,b,*}, Huai-Ping Jin^{a,b},
Rong Hu^{a,b}, Jian-Bo Yang^c

^a School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China

^b The Higher Educational Key Laboratory for Industrial Intelligence and Systems of Yunnan Province, Kunming University of Science and Technology, Kunming 650500, China

^c Alliance Manchester Business School, The University of Manchester, Manchester M15 6PB, United Kingdom

HIGHLIGHTS

- Developed MILP model for EE-DFFJSP innovates by novel triple-MDP formulation.
- Introducing the MACMNG framework, multi-agents tackle triple-MDP with subnets.
- Decomposing EE-DFFJSP into subnets sharing experience and knowledge by DPTS.
- Balancing criteria in decision-making and network updates by MO-DQN for subnets.
- Experiments verify the superiority of MACMNG in both effectiveness and efficiency.

ARTICLE INFO

Keywords:

Multi-agent reinforcement learning
Multi-network group framework
Distributed flexible job-shop scheduling
Energy-efficient scheduling

ABSTRACT

The increasing integration of industrial intelligence and the Industrial Internet of Things (IIoT) has promoted distributed flexible manufacturing (DFM) as a fundamental component of smart manufacturing systems. However, the rising complexity in dynamic demands, production uncertainties, and the urgent need for energy efficiency pose significant challenges. To address these challenges, this study investigates the energy-efficient distributed fuzzy flexible job shop scheduling problem (EE-DFFJSP), which aims to minimize both makespan and total energy consumption (TEC) in DFM environments. To tackle fuzzy uncertainties and complex coupling characteristics inherent in EE-DFFJSP, a multi-agent cooperative multi-network group (MACMNG) framework is proposed. First, a mixed-integer linear programming (MILP) model for EE-DFFJSP is formulated, followed by an analysis of the problem's properties. A triple Markov decision process formulation adapted to the problem's characteristics is designed, enabling problem decoupling and multi-agent decision-making through specific state representations and reward functions. Next, an innovative multi-network group framework is devised, and coupled decisions are effectively handled via interaction and collaboration among independent subnets. Based on problem decomposition method, EE-DFFJSP is decomposed into a set of subproblems represented by subnets within the network group. These subnets cooperate by sharing experience and knowledge through a domain parameter transfer strategy (DPTS) to enable efficient training. Finally, MACMNG employs a multi-objective DQN (MO-DQN) integrated with a dynamic weighting mechanism, enabling subnets to effectively balance between makespan and TEC during cooperative decision-making and network parameter updating. Experimental results show that MACMNG achieves superior performance compared with three priority dispatch rules (PDRs) across various scenarios. The MACMNG outperforms seven state-of-the-art multi-objective algorithms in terms of different metrics across 69 benchmark instances. This study contributes an efficient learning-driven and multi-agent collaborative promising paradigm for the energy-efficient scheduling in DFM, providing practical insights for advancing smart manufacturing in IIoT architectures.

* Corresponding authors at: School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China.

E-mail addresses: zhangziqi@kust.edu.cn (Z.-Q. Zhang), lixiaowei@stu.kust.edu.cn (X.-W. Li), bin.qian@vip.163.com (B. Qian), ronghu@vip.163.com (R. Hu), jian-bo.yang@umist.ac.uk (J.-B. Yang).

¹ The authors contributed equally to this work.

1. Introduction

Driven by advancements in industrial intelligence, both digitization and intelligence provide real-time data and decision support for flexible manufacturing (FM), which has emerged as a cornerstone of smart manufacturing systems. With the rapid development of industrial AI and Internet of Things (IIoT)—a specialized IoT paradigm providing ubiquitous sensing, connectivity, and computing in industrial environments—IIoT technologies have been increasingly applied in domains [1], including smart homes [2], healthcare services [3], smart agriculture [4], and sensing systems [5]. Within smart manufacturing systems, scheduling has become a critical research focus aimed at optimizing resource allocation, enhancing operational efficiency and profitability, and achieving lean production, thereby strengthening market competitiveness [6]. While FM in IIoT enables enhanced production efficiency, reduced costs, shortened cycles, and improved quality and flexibility, it faces growing challenges due to the complexities of both dynamic demand and high flexibility. Recent research has emphasized that cross-regional collaborative production has greatly sparked widespread interest in distributed FM (DFM) [7]. The distributed flexible job shop scheduling (DFJS) plays a key vital in enhancing the efficiency of DFM by providing efficient strategies that enhance adaptability, availability, and productivity [8]. DFJS has been widely found with extensive applications in DFM scenarios such as semiconductor manufacturing, aerospace fabrication, and IIoT-enabled FM [9]. However, dynamic disruptions, such as machine breakdowns [10], job insertions [11], and parameter fluctuations, introduce uncertainties into DFJS; additional consideration of fuzzy processing time is essential to address these uncertainties. Furthermore, with increasing emphasis on sustainable manufacturing, it is imperative to balance the economic and environmental effects by integrating energy-efficient strategies (EESs) into DFJS [12]. Motivated by these challenges, this study investigates the energy-efficient distributed fuzzy flexible job shop scheduling problem (EE-DFFJSP), aiming to optimize manufacturing cycle and total energy consumption (TEC) from both practical and technical perspectives to ensure sustainable and efficient production practices. Regarding DFJS challenges in DFM environments, various approaches have emerged, involving exact methods (e.g., mathematical programming), heuristic methods (e.g., priority dispatch rules, PDRs), and metaheuristic-based methods. Exact methods can provide optimal solutions but suffer from computational complexity and costs [13]. PDRs produce solutions quickly, but may not ensure the quality of solutions and lack generality across scenarios. Due to the strengths of metaheuristic-based methods in emphasizing efficiency and effectiveness, many relevant studies have emerged in recent years, which have shown remarkable success in various scheduling problems. Despite their strengths in searching solution spaces, metaheuristic-based methods may face inherent limitations in maintaining exploration-exploitation balance and avoiding local optima [14]. However, problem-specific complexities involving processing uncertainty and energy-efficient requirements in DFM environments make these traditional approaches unsuitable for solving EE-DFFJSP.

Machine learning has been widely used in materials science [15], demonstrating robust capabilities in addressing complex challenges, including property prediction for ultra-high-performance concrete [16] and foam glass [17]. As a specialized machine learning paradigm, deep reinforcement learning (DRL) integrates the representation learning strengths of deep learning with the sequential decision-making framework of reinforcement learning. Recent advances in DRL-based methods have shown significant potential for scheduling problems, which are modeled as a Markov decision process (MDP) through iterative agent-environment interactions, enabling agents to make decisions on optimal state-action policies via trial-and-error feedback guided by reward signals [18]. These DRL-based methods achieve a beneficial balance between immediate and delayed rewards through adaptive exploration-exploitation trade-offs, while trial-and-error feedback enables learning-driven mechanisms for agents to learn experience from

environments, enabling to effectively balance exploration and exploitation in interactions [19]. Nevertheless, the effectiveness and efficiency of DRL-based methods are greatly affected by network architecture design and state representation [20]. As the extension of FJSPs with uncertainty in DFM, EE-DFFJSP contains coupled subproblems with multi-objective optimization, thereby significantly increasing the complexity of both high-dimensional states and complex decision spaces. To address these challenges, the development of problem-specific DRL-based methods with effective state representations and interaction mechanisms emerges as a crucial research direction.

Recent advances have witnessed increasing interest in multi-agent reinforcement learning (MARL) frameworks applied in DFJS domains. MARL-based methods utilize the advantages of sub-problem decomposition and multi-agent coordination to overcome computational bottlenecks caused by high-dimensional state-action spaces, thus making them particularly suitable for dynamic decision-making in DFM environments [21]. By synchronized training-execution coordination, the multi-agent system can effectively decompose complex challenges into tractable sub-tasks while maintaining cooperative optimization [22]. Notably, there are three principal learning-driven mechanisms for MARL-based methods, i.e., decentralized learning, communication-based strategies, and centralized training with decentralized execution (CTDE). The decentralized learning architectures, which treat peer agents as environmental components in independent policy learning, offer computational efficiency but suffer from suboptimal decisions due to maximizing local rewards instead of global optimality [23–25]. The communication-based strategies enable information exchange among agents to enhance collaborative decisions, yet introduce considerable costs such as bandwidth consumption and latency, particularly in complex scenarios. In contrast, the CTDE framework allows global information utilization during centralized policy training while making decisions that rely on local observations for decentralized execution [26, 27]. Owing to the strengths of global coordination with local interaction, CTDE has demonstrated superior capability in dealing with dynamic environments with high-dimensional state representations and complex decision spaces.

Although many applications of MARL-based methods in addressing DFJS problems (DFJSPs) have shown strengths, further investigations are still required due to unresolved theoretical and practical limitations. Notably, most existing studies have focused on simplified scheduling scenarios, often not involving complex problem decoupling and multi-objective optimization with strong constraint handling. Although these methods show superiority for such situations, they are inadequate for real-world DFM environments that require the simultaneous treatment of conflicting objectives, coupled subproblems, and complex constraints. Motivated by existing efforts and recognizing MARL's promising potential, developing efficient multi-agent cooperative frameworks to achieve multi-objective coordination and effectively address coupled subproblems of DFJSP is still an open research challenge. To the best of our authors' knowledge, research on applying MARL-based methods for EE-DFFJSP has not been reported, with end-to-end learning-driven mechanisms for tackling the challenge of EE-DFFJSP being scarce.

As an extension of DFJSPs that widely exist in DFM that comprehensively consider uncertainty and energy-efficient scheduling constraints, EE-DFFJSP requires assigning jobs to suitable factories, determining fuzzy processing times, selecting available machines, and adjusting machine speeds. In addition, the complex constraints and critical characteristics of EE-DFFJSP require multiple agents to acquire global state information during the centralized training to perform feature-learning-based joint decisions during the decentralized execution. Motivated by CTDE-based MARL frameworks, we develop an innovative framework for a multi-agent cooperative multi-network group (MACMNG) to handle the EE-DFFJSP. Unlike traditional MARL-based methods, MACMNG can decompose the EE-DFFJSP into interconnected subproblems, each modeled as a subnet in the network group.

The multi-network group dynamically optimizes multi-objective trade-offs via adaptive weight allocation across subproblems, thereby resolving the critical limitations of static and single policy networks in MARL-based methods that fail to achieve dynamic trade-offs among multiple objectives.

The main innovations and contributions are outlined as follows:

- A novel mixed-integer linear programming (MILP) model is proposed for EE-DFFJSP, and a typical MDP model for FJSPs is extended to a novel triple-MDP formulation based on problem characteristics. This triple-MDP divides states into five specific sets and designs three rewards, thus explicitly embedding optimization objectives into the state representation and the reward mechanism design. These advances facilitate the effective modeling of MDPs for subproblems of EE-DFFJSP while ensuring coordinated decision-making among the agents associated with this triple-MDP.
- To address the complex constraints and coupled characteristics of EE-DFFJSP, a multi-agent cooperative multi-network group (MACMNG) framework is introduced to handle triple-MDP. Each MDP is implemented as a single agent and each agent is configured with an independent subnet. Agents with different behaviors make independent decisions and collaborate to form a network group with specific roles to optimize two conflicting goals through problem-specific state representations and rewards, thus efficiently handling coupled decision dynamics through distributed collaborative learning-driven mechanisms.
- The application of a problem decomposition method to decompose EE-DFFJSP into a series of subproblems, each of which is represented by a subnet in the MACMNG framework. These subnets are trained cooperatively and aided by the domain parameters transfer strategy (DPTS), enabling sharing experience and transferring knowledge, significantly reducing computational overheads and efficiently producing high-quality Pareto solutions with enhanced efficiency.
- By incorporating multi-objective DQN (MO-DQN) into the framework of MACMNG, subnets can adaptively trade-off between both criteria by employing a dynamic weighting mechanism to handle rewards from each of them during cooperative decision-making and network group updating. This ensures the robust adaptation to the dynamic adjustment of the criteria as a result of changes in solving these subproblems.
- Comprehensive comparisons conducted on 69 benchmark instances demonstrate MACMNG's statistically significant superiority over state-of-the-art multi-objective algorithms across three performance metrics. These statistical results provide promising perspectives and innovative insights for the research on DFJS in DFM, highlighting MACMNG's potential to advance both economic and energy-efficient objectives in industrial applications.

The remainder of this article is organized as follows: [Section 2](#) reviews foundational advancements in energy-efficient strategies (EESs), DFJSPs, DRL-based, and MARL-based methods. [Section 3](#) formulates the MILP model of EE-DFFJSP and defines fuzzy arithmetic operations with relevant notations. [Section 4](#) presents the details of the framework and implementation of MACMNG. [Section 5](#) demonstrates experimental details and results confirming the superiority of MACMNG in solving EE-DFFJSP. Finally, [Section 6](#) discusses the implications and limitations of MACMNG and outlines future research directions.

2. Literature review

2.1. Related work on EESs

Recent advances have demonstrated the efficacy of effective energy-efficient strategies (EESs) in reducing energy consumption (EC) for energy-efficient DFJS in DFM [\[28\]](#). EESs can be categorized into three primary types: machine on/off regulation, time-of-use (TOU) electricity

pricing measure, and energy-efficient speed control. As for energy-efficient DFJS, the standby times of machines are inevitable; however, by rationally controlling of machine's states, it is possible to significantly reduce some of the unnecessary waiting time, thereby reducing TEC. The TOU electricity pricing measure can reduce the electricity cost and improves cost predictability while enhancing energy efficiency by encouraging load shifting, such as scheduling energy-intensive tasks during off-peak periods. The energy-efficient speed control, as one of the most widely applied EESs, enables machines to operate flexibly within specified speed levels, thereby reducing TEC and minimizing makespan. For machine on/off regulation, Zhang et al. [\[29\]](#) investigated the energy-efficient flexible job shop scheduling problem (EE-FJSP). Their experimental results demonstrated that reasonable control of the machine's operating states can effectively reduce TEC. Dai et al. [\[30\]](#) solved the energy-efficient flexible flow shop scheduling problem (EE-FFSP) using a genetic-simulated annealing algorithm (GSA). Their control of machines' operating states within feasible schedules significantly reduced the TEC. Meng et al. [\[31\]](#) addressed the energy-conscious dual-resource constrained FJSP (DRCFJSP) and proposed a postponing strategy and a turn-on/off strategy that effectively reduced TEC, further highlighting the critical role of machine on/off regulation in TEC reduction. However, frequent switching of machines on/off may lead to wear and potential damage, which affects the machines' longevity and reliability [\[32\]](#). For TOU electricity pricing measure, Park and Ham [\[33\]](#) established a MILP model for the EE-FJSP with dual objectives of minimizing both makespan and TEC. However, TOU electricity pricing measure that encourage additional tasks during off-peak periods may conflict with the best time for manufacturing and maintenance decisions, necessitating a trade-off between production efficiency and energy costs [\[34\]](#). Regarding energy-efficient speed control, Zhang et al. [\[35\]](#) introduced a multi-objective discrete artificial bee colony algorithm (MDABC) for solving the hybrid flow shop green scheduling problem (HFGSP), and proposed an energy-saving procedure based on energy-efficient speed control, which effectively reduced the TEC. Duan et al. [\[36\]](#) proposed an effective speed control strategy which demonstrated the effectiveness of the energy-efficient speed control in reducing TEC and makespan. They developed a multi-objective NSGA-II for EE-FJSP, significantly reducing TEC and makespan. By implementing energy-efficient speed control, the machines can operate within a flexible range of speed levels, thereby balancing EC and minimizing manufacturing cycles. Moreover, energy-efficient speed control avoids the wear and potential reliability issues associated with frequent machine on/off switching, while also avoiding the trade-off between production efficiency and energy costs resulting from the TOU electricity pricing measure. Given these advantages, this study adopts the energy-efficient speed control as the primary EES to address the EE-DFFJSP.

2.2. Related work on DFJSPs

As an extension of FJSPs in DFM, DFJSPs have garnered significant research attention in recent years. For the DFJSP, Lin et al. [\[37\]](#) developed a genetic algorithm with a novel chromosome representation (GA_X), which adopted an incomplete chromosome structure to effectively balance the load among manufacturing resources and improve search efficiency. Xu et al. [\[38\]](#) proposed a hybrid genetic algorithm and tabu search (H-GA-TS) with three-layer encoding, which tackled the co-optimization of multiple objectives. Considering crane transportation, Du et al. [\[39\]](#) proposed an estimation of distribution algorithm (EDA) combined with variable neighborhood search (EDA-VNS) for DFJSP with crane transportations (DFJSPC). Zhang et al. [\[13\]](#) developed a Q-learning-based hyper-heuristic evolutionary algorithm (QHHEA) for DFJSPC. Considering factory transfers, Luo et al. [\[40\]](#) introduced an efficient memetic algorithm (EMA) for DFJSP with transfers (DFJSPT), aiming to minimize the makespan, maximum workload, and TEC. Despite these advancements, research on

Table 1

Summary of the literature on FJSP and related shop scheduling problems.

Author(s)	Ref.	Problem	Objective(s)	EES	Approach (es)	Description
Zhang et al. (2017)	[29]	EE-FJSP	$C_{\max}$, TEC	Machine off-on	eGEP	Efficient gene expression programming algorithm.
Dai et al. (2013)	[30]	EE-FFSP	$C_{\max}$, TEC	Machine off-on	GSA	Genetic-simulated annealing algorithm.
Meng et al. (2019)	[31]	DRCFJSP	TEC	Machine off-on	VNS	Variable neighborhood search algorithm.
Park and Ham (2022)	[33]	EE-FJSP	$C_{\max}$, TEC	TOU electricity prices	ILP/CP	Integer linear programming and constraint programming.
Zhang et al. (2019)	[35]	HFGSP	$C_{\max}$, TEC	Speed control	MDABC	Multi-objective discrete artificial bee colony algorithm.
Duan et al. (2021)	[36]	EE-FJSP	$C_{\max}$, TEC	Speed control	NSGA-II	Heuristic multi-objective non-dominated ranking genetic algorithm.
Lin et al. (2020)	[37]	DFJSP	$C_{\max}$	None	GA_X	Genetic algorithm with a new chromosome representation.
Xu et al. (2021)	[38]	DFJSP	$C_{\max}$, TEC , $costs$, $quality$	None	H-GA-TS	Hybrid genetic algorithm and tabu search with three-layer encoding.
Du et al. (2021)	[39]	DFJSPC	$C_{\max}$, TEC	None	EDA-VNS	Hybrid algorithm consisting of EDA and VNS.
Zhang et al. (2023)	[13]	DFJSPC	$C_{\max}$, TEC	None	QHHEA	Q-learning-based hyper-heuristic evolutionary algorithm.
Luo et al. (2020)	[40]	DFJSPT	$C_{\max}$, TEC , $workload$	None	EMA	Efficient memetic algorithm.
Wang et al. (2020)	[41]	EE-DFSP	$C_{\max}$, TEC	Speed control	KCA	Knowledge-based cooperative algorithm.
Shao, et al. (2022)	[42]	EE-DFSP	$C_{\max}$, TEC	Speed control	MMMA	Multi-neighborhood-based multi-objective memetic algorithm.
Zhang et al. (2025)	[43]	EE-DFSP	$C_{\max}$, TEC	Speed control	MEDHEA	Multidimensional estimation of distribution based hyper-heuristic evolutionary algorithm.
Meng et al. (2020)	[44]	EE-DFJSP	$C_{\max}$, TEC	Machine off-on	HSFLA	Efficient hybrid shuffled frog-leaping algorithm.
Yu et al. (2024)	[45]	EE-DFJSP	$C_{\max}$, TEC	Speed control	KBEA	Knowledge-guided bi-population evolutionary algorithm.
Zhang et al. (2024)	[46]	EE-DFJSP	$C_{\max}$, TEC	Speed control	MPMEA	Multidimensional probabilistic model-based evolutionary algorithm.
Li et al. (2022)	[47]	DGT2FJSP	$C_{\max}$, TEC	Speed control	TSKEA	Two-stage knowledge-driven evolutionary algorithm.
Lei et al. (2022)	[48]	FJSP	$C_{\max}$	None	Multi-PPO	Multi-proximal policy optimization.
Song et al. (2023)	[49]	FJSP	$C_{\max}$	None	PPO	Proximal policy optimization.
Yuan et al. (2024)	[50]	FJSP	$C_{\max}$	None	PPO	Proximal policy optimization.
Jing et al. (2024)	[22]	FJSP	$C_{\max}$	None	GMAS	Graph-based multi-agent system.
Liu et al. (2022)	[56]	FJSP	TCT	None	DDQN	Double deep Q-Network algorithm.
Wan et al. (2025)	[57]	FJSP	$C_{\max}$	None	AEA-MAPPO	Automatic entropy adjustment multi-agent proximal policy optimization algorithm.
Huang et al. (2024)	[58]	DFJSP	$C_{\max}$	None	PPO	Proximal policy optimization.

EE-DFFJSP remains sparse. For energy-efficient distributed flow shop scheduling problem (EE-DFSP), Wang et al. [41] proposed a knowledge-based cooperative algorithm (KCA), integrating a multi-operator cooperative search strategy that adapts to solution characteristics to balance the trade-off between two optimization objectives. Shao et al. [42] focused on the EE-DFSP with variable machine speeds and developed a multi-neighborhood-based multi-objective memetic algorithm (MMMA). In consideration of no-wait constraints, Zhang et al. [43] developed a multidimensional estimation of distribution-based hyper-heuristic evolutionary algorithm (MEDHEA) to tackle the EE-DFSP. To address the energy-efficient DFJSP (EE-DFJSP), Meng et al. [44] formulated an MILP model and designed a hybrid shuffled frog-leaping algorithm (HSFLA). Yu et al. [45] proposed a knowledge-guided bi-population evolutionary algorithm (KBEA) to simultaneously minimize makespan and TEC. Zhang et al. [46] introduced a multidimensional probabilistic model-based evolutionary algorithm (MPMEA), offering a new paradigm for solving EE-DFJSP. Considering type-2 fuzzy constraints, Li et al. [47] introduced the two-stage knowledge-driven evolutionary algorithm (TSKEA) for distributed green job shop scheduling with type-2 fuzzy processing times (DGT2FJSP), focusing on minimizing makespan and TEC. Previous

research works have used exact methods, heuristic methods, and metaheuristic-based methods with promising results in solving DFJSPs, but all suffer from limitations. Specifically, exact methods and metaheuristic-based methods are less efficient in handling high-dimensional dynamic states and complex decision spaces, as computational costs rise with the increase in problem complexity.

2.3. Related work on DRL and MARL

Recent advances in DRL-based and MARL-based methods have motivated research to investigate their applications in solving various scheduling problems. These methods demonstrate exceptional efficacy in modeling and optimizing problems characterized by high-dimensional dynamic states and decision spaces. For DRL-based methods, Lei et al. [48] developed a DRL-based method for the FJSP, which used a graph neural network (GNN) for feature representation and adopted a multi-pointer graph network (MPGN) and a multi-proximal policy optimization algorithm (multi-PPO) to efficiently learn action policies, achieving superior performance over both heuristic methods and metaheuristic-based methods. Song et al. [49] proposed a DRL-based method for solving the FJSP, which combined

Table 2

Notation and definition used in the MILP model of EE-DFFJSP.

Notation	Description
Indices	
f, f'	Indices for factories, $f, f' = 1, 2, \dots, l$.
i, i'	Indices for jobs, $i, i' = 1, 2, \dots, n$.
j, j'	Indices for operations of the job, $j, j' = 1, 2, \dots, n_i$.
k	Index for machines, $k = 1, 2, \dots, m$.
v	Index for speed levels, $v = 1, 2, \dots, s$.
Sets	
I	Set of jobs, $I = \{I_1, I_2, \dots, I_n\}$.
J_i	Set of operations for job I_i , $J_i = \{O_{i,1}, O_{i,2}, \dots, O_{i,n_i}\}$.
F	Set of factories, $F = \{F_1, F_2, \dots, F_l\}$.
V	Set of speed levels, $V = \{V_1, V_2, \dots, V_s\}$.
G_f	Set of machines in factory F_f , $G_f = \{M_1^f, M_2^f, \dots, M_m^f\}$.
UM_{ij}	Set of available machine indices for operation O_{ij} .
SM	Set of speed coefficient, $SM = \{sm_1, sm_2, \dots, sm_s\}$.
EM	Set of EC coefficient, $EM = \{em_1, em_2, \dots, em_s\}$.
Parameters	
F_f	The f -th factory, $F_f \in F$.
M_k^f	The k -th machine in factory F_f , $M_k^f \in G_f$.
I_i	The i -th job, $I_i \in I$.
O_{ij}	The j -th operation of job I_i , $O_{ij} \in J_i$.
V_v	The v -th speed level, $V_v \in V$.
sm_v	The v -th speed coefficient, $sm_v \in SM$.
em_v	The v -th EC coefficient, $em_v \in EM$.
l	Total number of factories.
n	Total number of jobs.
n_i	Total number of operations for job I_i .
m	Total number of machines in each factory.
s	Total number of speed levels.
$\tilde{t}_{k,i,j}$	The standard fuzzy processing time of O_{ij} on the k -th machine in each factory.
$\tilde{p}_{k,i,j}$	The actual fuzzy processing time of O_{ij} on the k -th machine in each factory.
SP	The standby unit EC.
PP	The standard unit EC.
$PE\tilde{C}_{f,k}$	The total fuzzy processing EC of machine M_k^f .
$SE\tilde{C}_{f,k}$	The total fuzzy standby EC of machine M_k^f .
$\tilde{C}_{f,k,i,j}$	The fuzzy completion time of operation O_{ij} on machine M_k^f .
$\tilde{C}_f$	The fuzzy completion time of all operation in factory F_f .
$\mathcal{Q}$	A sufficiently large positive number.
Objective functions	
$\tilde{C}_{\max}$	The maximum fuzzy completion time.
$\tilde{TEC}$	Total fuzzy EC for all machines.
Binary decision variables	
$X_{f,k,i,j}$	$\begin{cases} 1, \text{ if } O_{ij} \text{ is processed on } M_k^f, \\ 0, \text{ otherwise.} \end{cases}$
$Y_{i,j,j'}$	$\begin{cases} 1, \text{ if } O_{i,j'} \text{ is processed immediately after } O_{i,j}, \\ -1, \text{ if } O_{i,j} \text{ is processed immediately after } O_{i,j'}, \\ 0, \text{ otherwise.} \end{cases}$
$Z_{f,k,v,i,j}$	$\begin{cases} 1, \text{ if } O_{ij} \text{ is processed on machine } M_k^f \text{ with speed level } V_v, \\ 0, \text{ otherwise.} \end{cases}$

operation selection and machine assignment with efficient decision-making using the heterogeneous GNN (HGNN). Yuan et al. [50] introduced a DRL-based method which simplified state embedding using a lightweight multi-layer perceptron (MLP). MARL-based methods have achieved significant success across diverse domains such as collaborative control of unmanned aerial vehicles (UAVs) [51], multi-robot systems [52], and gaming applications [53–55]. Recent research has concentrated on multi-agent cooperative strategies and interaction mechanisms to effectively handle collaboration and competition in complex environments. For MARL-based methods, Jing et al. [22] integrated graph convolutional networks (GCNs) with the MARL-based method to propose a graph-based multi-agent system (GMAS) which modeled the FJSP as a directed acyclic graph and refined strategies by predicting edge connections. Liu et al. [56] presented a MARL-based method to address dynamic constraints for the FJSP, aimed at minimizing the total cumulative tardiness (TCT) of all jobs. They introduced a novel state and decision representation employing the double deep

Q-Network algorithm (DDQN) to train scheduling agents and capture the relationship between production information and scheduling objectives, capable of managing dynamic FJSPs with different scales and scenarios. Wan et al. [57] formulated the FJSP as two MDPs using different agents for operation sequencing and machine allocation. They developed an automatic entropy-adjusted multi-agent PPO (AEA-MAPPO) algorithm, which effectively trained the operation and machine policy networks to optimize operation sequencing and machine assignment strategies simultaneously. Huang et al. [58] proposed a multi-action MDP for modeling the dynamic distributed JSP with a hierarchical action space that considers operations and factories, where the reward function design is based on idle times.

As summarized in Table 1, recent advances reveal that although metaheuristic-based, DRL-based, and MARL-based methods have been successfully used for FJSPs, each type of method suffers from fundamental limitations. Metaheuristic-based methods in existing efforts still suffer shortcomings in their reliance on problem feature modeling and search behavior understanding, and demonstrate poor scalability in high-dimensional spaces. Although DRL-based methods can adaptively learn optimal policies from feature states, their effectiveness is limited by decision spaces and state representations. Up to now, the applications of these methods for problem solving by a single agent are mainly limited to simple scheduling situations involving low-coupling features with single-objective optimization. While MARL-based methods effectively reduce the computational cost of high-dimensional decision and state spaces through the collaborative decisions of distributed agents, their framework for end-to-end training and generation of scheduling schemes poses inherent limitations. The policy network maps any given state to a specific distribution of actions by means of a deterministic mapping function. The policy's deterministic mapping behavior enforces end-to-end framework to output only solutions with specific preferences when dealing with multi-objective problems (MOPs), failing to dynamically generate multi-preference solutions with dispersion and diversity along the Pareto front.

To bridge this gap, inspired by the CTDE framework in MARL-based methods, this study develops an innovative framework for a multi-agent cooperative multi-network group (MACMNG) to address the limitations of existing MARL-based methods. Unlike the aforementioned methods, MACMNG's strength lies in decomposing the EE-DFFJSP into multiple decoupling subproblems, each modeled as a subnetwork within a network group. Each subnet generates solutions for specific subproblems through cooperative training with shared parameters. The collective outputs of all subnetworks form a diverse solution set for the entire network group, thereby effectively overcoming the limitations of single solution generation inherent in end-to-end models and frameworks. The MACMNG produces solution sets that cover the Pareto front in complex constrained multi-objective optimization scenarios, effectively guaranteeing the quality and diversity of scheduling solutions generated by MARL-based methods and addressing the limitations of traditional end-to-end models, providing a new perspective on the design of learning-driven and multi-agent collaborative promising paradigms for EE-DFJSPs.

3. Problem statement

3.1. EE-DFFJSP

Under the transformation of intelligent manufacturing driven by industrial intelligence, the study of DFJS in IIoT-enabled DFM holds significant theoretical and practical importance. As a common challenge encountered in real-world DFM, the EE-DFFJSP is described as follows, with the relevant notations defined in Table 2. Consider a set of factories $F = \{F_1, F_2, \dots, F_l\}$, where each factory F_f has a flexible job shop with m heterogeneous machines $G_f = \{M_1^f, M_2^f, \dots, M_m^f\}$. There are n jobs, denoted as $I = \{I_1, I_2, \dots, I_n\}$, allocated across l factories. Each job I_i

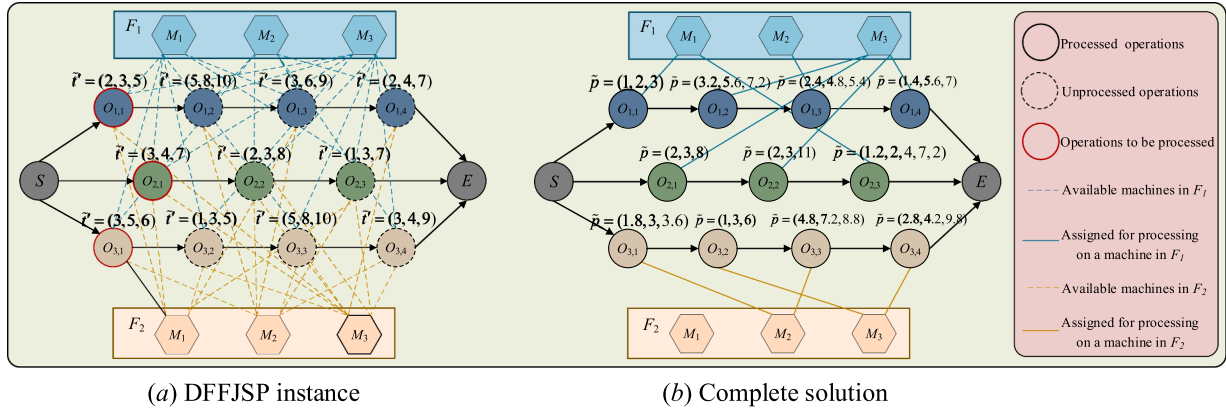


Fig. 1. Disjunctive graph representation of DFFJSP.

Table 3

Details of processing speed levels.

V	1	2	3	4	5
SM	0.6	0.8	1	1.2	1.4
EM	8	5	3	2	1

Table 4

Details of fuzzy processing times.

Jobs	Operations	Available machines and standard fuzzy processing times.		
		M_1	M_2	M_3
J_1	$O_{1,1}$	(1,2,3)	-	(3,4,5)
	$O_{1,2}$	-	(6,9,11)	(4,7,9)
	$O_{1,3}$	(2,4,9)	(4,8,9)	-
	$O_{1,4}$	-	(3,4,9)	(1,4,5)
J_2	$O_{2,1}$	(4,5,6)	-	(2,3,8)
	$O_{2,2}$	-	(2,3,5)	(2,3,11)
	$O_{2,3}$	(1,2,6)	-	(1,4,8)
J_3	$O_{3,1}$	(3,5,6)	(3,5,6)	-
	$O_{3,2}$	(1,3,4)	-	(1,3,6)
	$O_{3,3}$	-	(6,9,11)	(4,7,9)
	$O_{3,4}$	-	(4,5,11)	(2,3,7)

consists of n_i operations, i.e., $J_i = \{O_{i,1}, O_{i,2}, \dots, O_{i,n_i}\}$. Each operation $O_{i,j}$ must be processed on at least one machine in its assigned factory. The fuzzy processing time $\tilde{t}_{k,i,j}$ for operation $O_{i,j}$ on machine M_k^f is determined via standard operating conditions. Each machine has s adjustable speed levels, and the actual processing time $\tilde{p}_{k,i,j}$ is modulated by the speed coefficient sm_v associated with the selected speed level V_v . Each machine has a standard unit of fuzzy EC during processing, denoted as PP . When no operations are processed, the machine remains in standby state with a nominal EC denoted as SP . Both PP and SP are inherent constants for machines. The actual EC of each machine is determined by its processing time, standby time, and selected speed level. To address uncertainties inherent in real-world processing times, both fuzzy processing times and fuzzy EC are represented as triangular fuzzy numbers (TFNs), i.e., (g_1, g_2, g_3) . The assumptions of EE-DFFJSP are given below:

- All factories are homogeneous flexible job shops with identical machine layouts, quantities, types, and capacities. All operations must be processed within the same factory. Cross-factory processing or dynamic reassignment of operations is strictly prohibited.
- Once the machine's speed level is set for an operation, it remains unaltered for the duration of the process. No adjustments to the speed levels are allowed during processing.

- Once an operation starts processing on a machine, it is completed without pause, rescheduling, or interruption, thus ensuring that processing processes are predictable and uninterrupted.
- Only consider the time cost and EC directly associated with machines; auxiliary factors, such as material handling delays or external energy expenditures, are excluded from the scope of this study.

Following the pioneering work of Brandimarte [59], disjunctive graphs $\mathcal{G} = (\mathcal{C}, \mathcal{E}, \mathcal{D})$ are used to represent scheduling states for FJSPs, and this representation can be extended to the EE-DFFJSP. Specifically, $\mathcal{C} = \{O_{i,j} | \forall i, j\} \cup \{S, E\}$ denotes the set of job nodes, comprising all operation nodes and two virtual nodes (S : start, E : end) with zero processing time. $\mathcal{E}$ is the set of conjunctive arcs, which represent sequential processing constraints for each job J_i from S to E . $\mathcal{D} = \bigcup_k \mathcal{D}_k$ represents a set of undirected disjunctive arcs, where $\mathcal{D}_k$ denotes a subset that connects operations eligible for the k -th machine. However, the distributed factories and coupled subproblems inherent to DFFJSP render the conventional disjunctive graphs $\mathcal{G}$ for FJSPs insufficient in representing states. To address this limitation, we have extended graph $\mathcal{G}$ to $\overline{\mathcal{G}}$, defined as $\overline{\mathcal{G}} = (\mathcal{C}, \mathcal{E}, \mathcal{V}_1, \dots, \mathcal{V}_f, \dots, \mathcal{V}_l)$ where for each factory F_f : $\mathcal{V}_f = \mathcal{D} \cup \{M_1^f, M_2^f, \dots, M_m^f\}$. Here, $\mathcal{V}_f$ integrates disjunctive arcs with factory-specific machine nodes, while operation nodes are connected to their eligible machines via directed edges.

To provide an intuitive understanding of the extended graph $\overline{\mathcal{G}}$, Fig. 1 presents an exemplar case with two factories, three jobs, and three machines. In Fig. 1(a), solid black lines denote the completed operations, solid red lines indicate schedulable jobs, and black dashed lines represent the unprocessed operations. The color lines connections between job nodes and machine nodes signify factory-specific allocations, where solid endpoints on these lines denote the assigned machines and dashed endpoints show potential machine candidates. The actual fuzzy processing time $\tilde{p}_{k,i,j}$ is marked for completed operations $O_{i,j}$, and unprocessed operations show average processing times across eligible machines at specific processing speed levels. Fig. 1(b) illustrates a view of the complete scheduling scheme. Tables 3 and 4 provide the detailed dataset of the speed levels V and standard fuzzy processing times $\tilde{t}_{k,i,j}$ for each operation $O_{i,j}$, where the symbol "-" indicates machine-process incompatibility. Following the work of Zhang et al. [43], the standard and standby EC units are set as $PP = 3$ and $SP = 1$, respectively. Fig. 2 illustrates a scheduling scheme for an instance given in Table 3, along with processing speed levels of operations on their corresponding machines. The feasible solution is represented as $\pi = (\psi_F, \psi_J, \psi_M, \psi_V)$ where $\psi_F = [1, 1, 2]$ represents the factory assignment for jobs, $\psi_J = [1, 2, 3, 1, 2, 2, 1, 3, 3, 3]$ is the order of operations, $\psi_M = [1, 3, 3, 2, 2, 3, 1, 3, 3, 2, 3]$ is the order of machines allocated to operations, and $\psi_V = [3, 2, 3, 1, 1, 3, 4, 5, 3, 2, 5]$ is the order of processing speed levels for machines. For operation $O_{1,1}$, the fuzzy completion time is calculated by $\tilde{C}_{1,1,1}(\pi) =$

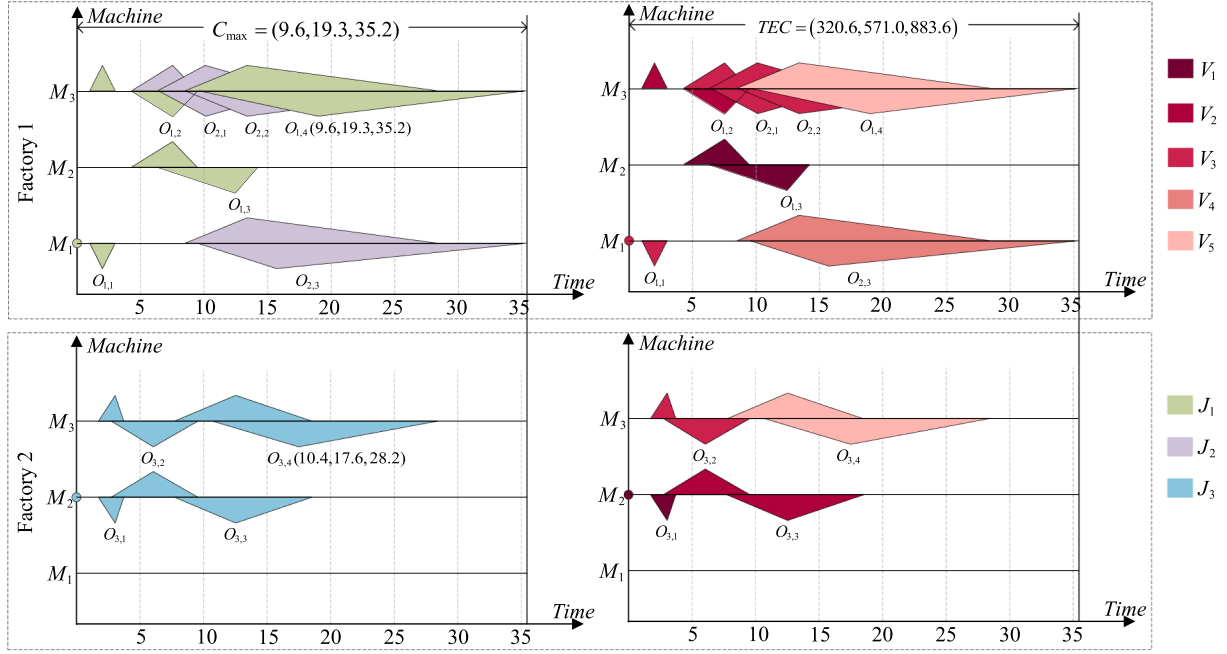


Fig. 2. Fuzzy Gantt charts and processing speed.

$\tilde{t}_{k,i,j} \times sm_v = \tilde{t}_{1,1,1} \times sm_3 = (1,2,3)$. Similarly, the fuzzy completion times for all other operations can be obtained using the same formula. For factory F_1 , the fuzzy completion time is $\tilde{C}_1(\pi) = (9.6, 19.3, 35.2)$, and for factory F_2 , it is $\tilde{C}_2 = (10.4, 17.6, 28.2)$, resulting in the maximum fuzzy completion time across both factories $\tilde{C}_{\max}(\pi) = \max(\tilde{C}_1(\pi), \tilde{C}_2(\pi)) = (9.6, 19.3, 35.2)$. The machine M_1^1 is active during $(4.2, 7.6, 9.2)$ to $(6.6, 12.4, 14.6)$, thus the fuzzy processing EC is $\widetilde{PEC}_{f,k}(\pi) = (2.4, 4.8, 5.4) \times em_1 \times PP = (2.4, 4.8, 5.4) \times 8 \times 3 = (57.6, 115.2, 129.6)$. During standby periods, machine M_1^1 remains in the standby mode, with its fuzzy standby time determined by $\tilde{C}_1 - \sum_{i=1}^n \sum_{j=1}^{n_i} \tilde{p}_{1,i,j} \times \sum_{i=1}^n \sum_{j=1}^{n_i} X_{1,1,i,j} = (7.2, 14.5, 29.8)$, and the corresponding fuzzy standby EC is $\widetilde{SEC}_{f,k}(\pi) = (C_f - \sum_{i=1}^n \sum_{j=1}^{n_i} \tilde{p}_{k,i,j} \times \sum_{i=1}^n \sum_{j=1}^{n_i} X_{f,k,i,j}) \times SP = (7.2, 14.5, 29.8) \times 1 = (7.2, 14.5, 29.8)$. The total fuzzy EC $\widetilde{TEC}$ is derived as: $\widetilde{TEC}(\pi) = (320.6, 571.0, 883.6)$ by Eq. (15).

The EE-DFFJSP aims to optimize both $\tilde{C}_{\max}$ and $\widetilde{TEC}$ as objectives and corresponding formulas in the MILP model are defined as follows:

$$f = \min_{\pi \in \Pi} \{f_1, f_2 | f_1 = \tilde{C}_{\max}(\pi), f_2 = \widetilde{TEC}(\pi)\} \quad (1)$$

$$\tilde{C}_{\max} = \max\{\tilde{C}_{f,k,i,j}\}, \forall f, k, i, j, \quad (2)$$

$$\sum_{k=1}^m \sum_{j=1}^{n_i} X_{f,k,i,j} = n_i, \forall f, i \quad (3)$$

$$\sum_{v=1}^s Z_{k,v,i,j} = 1, \forall k, i, j \quad (4)$$

$$\sum_{j=1}^{n_i} m_{i,j} > 1, \forall i \quad (5)$$

$$\tilde{p}_{k,i,j} = \tilde{t}_{k,i,j} \times \left(\sum_{v=1}^s Z_{f,k,v,i,j} \times sm_v \right), \forall f, k, i, j \quad (6)$$

$$\tilde{C}_{f,k,i,j-1} < \tilde{C}_{f,k,i,j} - \tilde{p}_{k,i,j}, \forall f, k, i, j \neq 0 \quad (7)$$

$$\tilde{C}_{f,k,i,j} - \tilde{p}_{k,i,j} \geq 0, \forall f, k, i, j \quad (8)$$

$$\tilde{C}_{f,k,i,j} > \tilde{C}_{f,k,i,j} - \tilde{p}_{k,i,j}, \forall f, k, i, j \quad (9)$$

$$\tilde{C}_{f,k,i,j} \leq \tilde{C}_{f,k,i',j'} - \tilde{p}_{k,i',j'} + \mathcal{L} \times (1 - Y_{ij,i',j'}), \forall f, k, i, i', j, j' \quad (10)$$

$$\tilde{C}_{f,k,i,j} \leq \tilde{C}_{f,k,i,j+1} - \tilde{p}_{k,i,j+1} + \mathcal{L} \times (1 - Y_{ij,i,j+1}), \forall f, k, i, i', j \neq n_i, j' \quad (11)$$

$$\sum_{f=1}^l \sum_{k=1}^m X_{f,k,i,j} = 1, \forall i, j \quad (12)$$

$$\widetilde{PEC}_{f,k} = \sum_{i=1}^n \sum_{j=1}^{n_i} (X_{f,k,i,j} \times \tilde{p}_{k,i,j} \times \sum_{v=1}^s Z_{k,v,i,j} \times em_v) \times PP \quad (13)$$

$$\widetilde{SEC}_{f,k} = \left(C_f - \sum_{i=1}^n \sum_{j=1}^{n_i} \tilde{p}_{k,i,j} \times \sum_{i=1}^n \sum_{j=1}^{n_i} X_{f,k,i,j} \right) \times SP \quad (14)$$

$$\widetilde{TEC} = \sum_{f=1}^l \sum_{k=1}^m (\widetilde{PEC}_{f,k} + \widetilde{SEC}_{f,k}) \quad (15)$$

where Eq. (1) aims to minimize both $\tilde{C}_{\max}$ and $\widetilde{TEC}$. Eq. (2) provides the formula for calculating $\tilde{C}_{\max}$. Eq. (3) and Eq. (4) restrict all operations for the same job must be processed in the same factory, and processing speed levels cannot be changed once selected. Eq. (5) ensures operational feasibility by requiring each operation to have at least one eligible machine in its assigned factory F_f . Eq. (6) computes actual processing time based on standard processing time and machine speed levels. Eq. (7) enforces the precedence relationship among operations of the same job. Eq. (8) indicates that machine availability at time zero for all operations in factory F_f . Eq. (9) suggests that the completion time of each operation cannot be earlier than its start time. Eqs. (10)–(12) demonstrate the same machine in a factory F_f can process only one operation at any time. Eqs. (13)–(15) define the calculation formulas for $\widetilde{PEC}_{f,k}$, $\widetilde{SEC}_{f,k}$, and $\widetilde{TEC}$, respectively.

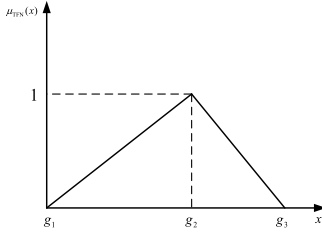


Fig. 3. Membership function of the triangular fuzzy number.

3.2. Fuzzy arithmetic principles

Due to the numerous uncertainties frequently found in DFM, the processing time for each job can only be determined within approximate ranges. The TFN is typically used to represent this processing time TFN = (g_1, g_2, g_3) .

The TFN membership function is shown in Fig. 3. g_1 represents the ideal processing time, g_2 is the most probable processing time with a membership degree of 1, and g_3 is the most conservative processing time. The membership function expression for TFN is as follows [60]:

$$\mu_{TFN}(x) = \begin{cases} 0, & x \leq g_1, \\ \frac{x - g_1}{g_2 - g_1}, & g_1 < x \leq g_2, \\ \frac{g_3 - x}{g_3 - g_2}, & g_2 < x \leq g_3, \\ 0, & x \geq g_3. \end{cases} \quad (16)$$

The following special cases exist: When $g_1 = g_2$, $\mu_{TFN}(x)$ can be formulated as:

$$\mu_{TFN}(x) = \begin{cases} 0, & x < g_1 = g_2, \\ \frac{g_3 - x}{g_3 - g_2}, & g_1 = g_2 \leq x < g_3, \\ 0, & x \geq g_3. \end{cases} \quad (17)$$

When $g_2 = g_3$, $\mu_{TFN}(x)$ can be formulated as:

$$\mu_{TFN}(x) = \begin{cases} 0, & x \leq g_1, \\ \frac{x - g_1}{g_2 - g_1}, & g_1 < x \leq g_2 = g_3, \\ 0, & x > g_2. \end{cases} \quad (18)$$

When $g_1 = g_2 = g_3$, $\mu_{TFN}(x)$ can be formulated as:

$$\mu_{TFN}(x) = \begin{cases} 1, & x = g_1 = g_2 = g_3, \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

To ensure the effective optimization of $\tilde{C}_{\max}$ and $\tilde{TEC}$ while maintaining scheduling feasibility, the addition, max, and ranking methods for TFN are explicitly defined. These methods are crucial for addressing the inherent uncertainties in DFFJSP [61].

1) Additional method. Let the fuzzy start time and fuzzy processing time of operation O_{ij} on machine M_k^f be $\tilde{t}_{k,ij}^S = (g_1, g_2, g_3)$ and $\tilde{t}_{k,ij} = (g'_1, g'_2, g'_3)$, respectively. The end time $\tilde{t}_{k,ij}^E$ of operation O_{ij} on machine M_k^f can be calculated by:

$$\tilde{t}_{k,ij}^E = \tilde{t}_{k,ij}^S + \tilde{t}_{k,ij} = (g_1 + g'_1, g_2 + g'_2, g_3 + g'_3). \quad (20)$$

2) Max method. Let $\tilde{t} = (g_1, g_2, g_3)$ denote the fuzzy completion time of preceding operation O_{ij-1} , and $\tilde{v} = (g'_1, g'_2, g'_3)$ represent the fuzzy completion time of the prior operation on the same machine. The

maximum fuzzy time of $\tilde{t}$ and $\tilde{v}$ can be derived from fuzzy arithmetic principles [62]:

$$\tilde{t} \vee \tilde{v} = \begin{cases} \tilde{t}, & \text{if } \tilde{t} \geq \tilde{v}, \\ \tilde{v}, & \text{otherwise.} \end{cases} \quad (21)$$

where the membership function $\mu_{\tilde{t} \vee \tilde{v}}(z)$ of $\tilde{t} \vee \tilde{v}$ is defined as:

$$\mu_{\tilde{t} \vee \tilde{v}}(z) = \sup_{z=x \vee y} \min(\mu_{\tilde{t}}(x), \mu_{\tilde{v}}(y)) \quad (22)$$

3) Ranking method. To evaluate fuzzy completion times, the ranking criterion for comparing TFNs proposed by Sakawa and Kubota [63] is employed as follows:

$$Z_1(g) = \frac{g_1 + 2g_2 + g_3}{4}. \quad (23)$$

In Eq. (23), Z_1 serves as the primary ranking criterion. When Z_1 values are equal, the secondary ranking criterion $Z_2(g) = g_2$ is utilized. If both Z_1 and Z_2 are equal, the tertiary ranking criterion $Z_3(g) = g_3 - g_1$ resolves ties.

3.3. Problem property analysis

The EE-DFFJSP aims to simultaneously optimize the maximum fuzzy completion time ($\tilde{C}_{\max}$) and reduce total fuzzy EC ($\tilde{TEC}$) to meet low-carbon manufacturing requirements. The interdependence of objectives and coupling of subproblems with uncertainty handling significantly expand the solution space, causing challenges in efficiently exploring search scopes to identify superior solutions within limited computational time. To address this, specific characteristics and properties of the EE-DFFJSP are analyzed in Section 3.3 to provide foundational insights for developing problem-specific EESs.

Property 1. For any feasible solution π , the critical factory is identified through decoding, where the critical path directly determines $\tilde{C}_{\max}(\pi)$. Reducing the speed level $V_v \in \mathbf{V}$ for operations on non-critical paths will lengthen their actual fuzzy processing time $\tilde{p}_{k,ij}$. However, if the extended $\tilde{p}_{k,ij}$ does not change the critical factory, critical path, or fuzzy completion time $\tilde{C}_{f,k,ij}$ on the critical path, $\tilde{C}_{\max}(\pi)$ remains unchanged, while $\tilde{TEC}(\pi)$ is reduced.

Proof. The critical path is defined as a series of operations without idle time that determines the maximum fuzzy completion time $\tilde{C}_{\max}(\pi)$. Any increase in processing time on critical paths directly results in an increase in $\tilde{C}_{\max}(\pi)$. However, adjusting processing speeds for operation O_{ij} on non-critical paths does not affect $\tilde{C}_{\max}(\pi)$ provided idle time intervals exist. Within these intervals, delays merely reduce speed levels V_v , thereby decreasing $\tilde{TEC}(\pi)$. The immediate precedence operations or machine-precedence operations of the critical operations on the critical path are defined as related operations, with adjustable fuzzy time margin AT_{ij} , while all other operations are categorized as unrelated operations, with adjustable fuzzy time margin defined as UAT_{ij} , which can be calculated by Eq. (24) and Eq. (25), respectively. Hence, when reducing the speed level V_v of non-critical path operations, as long as the adjusted processing time does not exceed the allowable time margins, the critical path and $\tilde{C}_{\max}(\pi)$ remain unaffected, and $\tilde{TEC}(\pi)$ is reduced.

$$AT_{ij} = \min(MS_{ij}, JS_{ij}) - \tilde{C}_{ij} \quad (24)$$

$$UAT_{ij} = \tilde{C}_{\max} - \tilde{C}_{ij} \quad (25)$$

Here, MS_{ij} represents the fuzzy start time of the operation immediately following operation O_{ij} on the machine M_k^f , where machine M_k^f is the processing machine for operation O_{ij} , JS_{ij} denotes the fuzzy start

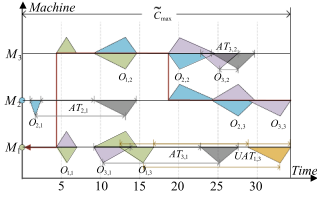


Fig. 4. Adjustable time margins for all the non-critical operations.

time of the operation immediately following operation O_{ij} in the operation sequence ψ_J , and $\tilde{C}_{ij}$ represents the fuzzy completion time of the operation O_{ij} . Fig. 4 illustrates the adjustable time margins for all non-critical paths in the fuzzy Gantt chart. In this case, $(O_{1,1}, O_{1,2}, O_{2,2}, O_{2,3}, O_{3,3})$ are operations on the critical path, with related operations being $O_{3,1}, O_{3,2}$ and $O_{2,1}$, and the unrelated operations being $O_{1,3}$.

4. MACMNG for EE-DFFJSP

In this section, MACMNG is introduced to address the challenges associated with the EE-DFFJSP, as depicted in Fig. 5. Based on the decomposition of subproblems, the EE-DFFJSP can be modeled as a triple-MDP to reflect the complex and coupled decision dynamics. Specifically, MACMNG comprises three distinct agents: a job agent $Agent_J$, a machine agent $Agent_M$, and a factory agent $Agent_F$. Both $Agent_J$ and $Agent_M$ are equipped with network groups. $Agent_J$ is embedded with the network group $Net^J = (SubNet_1^J, \dots, SubNet_h^J, \dots, SubNet_e^J)$, and $Agent_M$ is embedded with $Net^M = (SubNet_1^M, \dots, SubNet_h^M, \dots, SubNet_e^M)$. The $Agent_J$

is designed to handle operation sequences for jobs, while $Agent_M$ is responsible for machine assignment and speed selection. The $Agent_F$ serves to allocate jobs to a suitable factory for processing.

4.1. Subproblem decomposition method

The EE-DFFJSP as a complex MOP contains two conflicting objectives: $\tilde{C}_{\max}$ and $\widetilde{TEC}$. To cooperatively optimize both objectives, we employ an effective decomposition method that has been widely applied for decomposing MOPs across various domains [64,65]. By decomposing the EE-DFFJSP into a set of weighted objective subproblems $PB = (SubPB_1, \dots, SubPB_h, \dots, SubPB_e)$ and explicitly partitioning them based on predefined weightings, each subproblem can be independently resolved within the specific subspace. Each subproblem's feasible solution typically corresponds to one solution in Pareto solution set. Once all

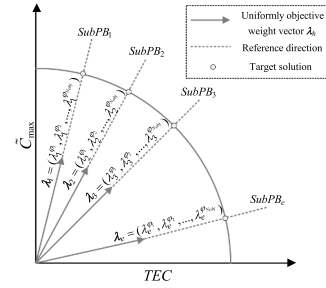


Fig. 6. Subproblem decomposition method.

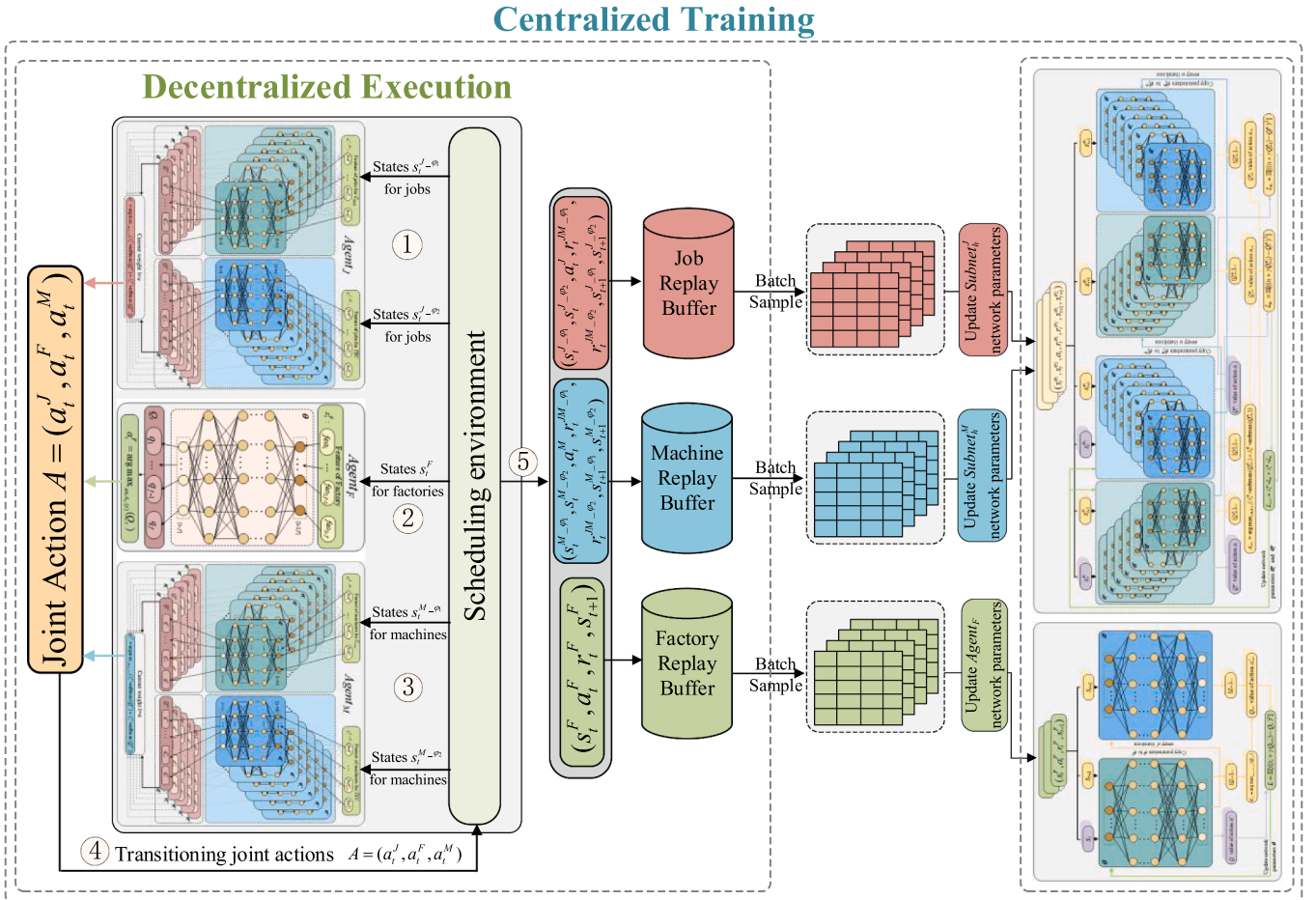


Fig. 5. The framework of MACMNG.

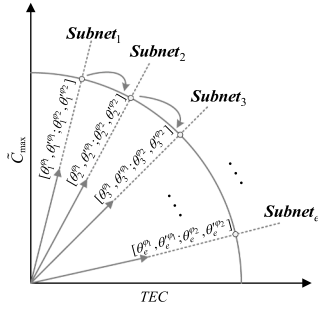


Fig. 7. Domain parameters transfer strategy.

subproblems are addressed, a set of Pareto solutions can be obtained, thereby revealing the balance and inherent conflicts between the optimization objectives $\tilde{C}_{\max}$ and $\tilde{TEC}$.

These subproblems $\mathbf{PB} = (\text{SubPB}_1, \dots, \text{SubPB}_h, \dots, \text{SubPB}_e)$ are partitioned through the weighted sum method [66], as shown in Fig. 6. This method produces a set of uniformly distributed objective weight vectors $\lambda = (\lambda_1, \dots, \lambda_h, \dots, \lambda_e)$ to decompose MOPs, where $\lambda_h = (\lambda_h^{\phi_1}, \lambda_h^{\phi_2}, \dots, \lambda_h^{\phi_{N_{\text{obj}}}})$. Here, N_o represents the number of objectives, and e denotes the number of decomposed subproblems. For the EE-DFFJSP, the optimization objectives are $\tilde{C}_{\max}$ and $\tilde{TEC}$ ($N_{\text{obj}} = 2$). Specifically, superscript ϕ_1 indicates that the symbol is associated with the objective $\tilde{C}_{\max}$, while superscript ϕ_2 denotes relevance to the objective $\tilde{TEC}$. The objective function f' for the h -th subproblem SubPB_h is as Eq. (26).

$$f' = \sum_{i=1}^{N_{\text{obj}}} \lambda_h^{\phi_i} f_i = \lambda_h^{\phi_1} f_1 + \lambda_h^{\phi_2} f_2, \forall h. \quad (26)$$

4.2. Domain parameter transfer strategy

In EE-DFFJSP, each subproblem SubPB_h is modeled as a corresponding subnet SubNet_h . These subnets constitute an integrated network group $\mathbf{Net} = (\text{SubNet}_1, \dots, \text{SubNet}_h, \dots, \text{SubNet}_e)$. With an increasing number of subproblems, the corresponding number of subnets also expands. In addition, training each subnet as an independent network makes the training of the network group $\mathbf{Net}$ difficult to complete within a feasible timeframe. Without sharing learned experience and knowledge among subnets, the network group $\mathbf{Net}$ training becomes significantly inefficient. To address this issue, the domain parameter transfer strategy (DPTS) is implemented to significantly reduce the training time for each neighboring subnet. The adjacent subproblems have similar weight configurations, which implies a structural similarity between subproblems SubPB_h and SubPB_{h-1} . By transferring the network parameters of subnet SubNet_{h-1} to its neighboring subnet SubNet_h for initialization, the time cost of retraining subnet SubNet_h is avoided. The DPTS not only shortens training time but also significantly improves the overall efficiency of the MACMNG in tackling the EE-DFFJSP.

The network parameters for the subnet SubNet_h are denoted as $\text{SubNet}_h = [\theta_h^{\phi_1}, \theta_h^{\phi_2}; \theta_h^{\phi_2}, \theta_h^{\phi_2}]$, where $\theta_h^{\phi_1}$ and $\theta_h^{\phi_2}$ correspond to the on-line-Q network parameters and target-Q network parameters optimized for $\tilde{C}_{\max}$, while $\theta_h^{\phi_2}$ and $\theta_h^{\phi_2}$ are for $\tilde{TEC}$, respectively. Assuming subnet SubNet_{h-1} has been fully trained and is near optimal, the subnet SubNet_h can share the parameters of SubNet_{h-1} as its initial training parameters. The subnet parameters $\text{SubNet}_{h-1} = [\theta_{h-1}^{\phi_1}, \theta_{h-1}^{\phi_2}; \theta_{h-1}^{\phi_2}, \theta_{h-1}^{\phi_2}]$ are then sequentially passed to the subsequent subnet SubNet_h for further training, as depicted in Fig. 7 and detailed in Algorithm 1. By solving each subproblem in Fig. 6 (which corresponds to each subnet in Fig. 7) to obtain the target solution, all feasible solutions in specific reference directions form the Pareto solution set. By systematically solving all subproblems, the MACMNG framework can ultimately construct an approximate Pareto front, thereby providing an efficient learning-driven and multi-agent collaborative promising paradigm for solving complex MOPs.

Algorithm 1. : Initialization network parameters for SubNet_h

Input: SubNet_h

```

1: for  $i = 1 : N_{\text{obj}}$  do
2:   if  $h = 1$  then
3:     Random initialize  $\theta_h^{\phi_i}$ . Set  $\theta_h^{\phi_i} = \theta_h^{\phi_i}$ .
4:   else
5:     Set  $\theta_h^{\phi_i} = \theta_{h-1}^{\phi_i}, \theta_h^{\phi_i} = \theta_{h-1}^{\phi_i}$ .
6:   end if
7: end for

```

Output: SubNet_h .

4.3. Triple Markov decision process formulation

According to the work of Song et al. (2023), addressing the FJSP can be formulated as the MDP, denoted as $\mathcal{M}(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$, where $\mathcal{S}$ represents the set of states, $\mathcal{A}$ denotes a set of actions, $\mathcal{T}$ is the state transition function, R refers to the reward function, and γ is the discount factor. However, the traditional single MDP is inadequate for modeling the complex characteristics of the EE-DFFJSP. To address this limitation, the triple-MDP formulation is introduced, denote as $(\mathcal{M}_J, \mathcal{M}_F, \mathcal{M}_M)$. Each MDP is defined as $\mathcal{M}_F = (\mathcal{S}_F, \mathcal{A}_F, \mathcal{T}, R_F, \gamma)$, and $\mathcal{M}_M = (\mathcal{S}_M^{\phi_1}, \mathcal{S}_M^{\phi_2}, \mathcal{A}_M, \mathcal{T}, R_{JM}^{\phi_1}, R_{JM}^{\phi_2}, \gamma)$. Unlike traditional MDPs, consisting of only one set of states and one reward function, the triple-MDP incorporates five sets of states ($\mathcal{S}_F^{\phi_1}, \mathcal{S}_F^{\phi_2}, \mathcal{S}_F, \mathcal{S}_M^{\phi_1}, \mathcal{S}_M^{\phi_2}$) and three distinct reward functions ($R_{JM}^{\phi_1}, R_{JM}^{\phi_2}, R_F$). Despite this, the triple-MDP share a common state transition function ($\mathcal{T}$). The state transition in triple-MDP is contingent upon the receipt of the joint action $\mathbf{A} = (a_t^J \in \mathcal{A}_J, a_t^F \in \mathcal{A}_F, a_t^M \in \mathcal{A}_M)$, which emerges from the cooperative decision-making of three agents (Agent_J , Agent_F , and Agent_M). Fig. 8 provides an illustrative example of the state transition within the triple-MDP. At the decision time step $t = 3$, the agents collectively determine a joint action $\mathbf{A}$, upon which the environment

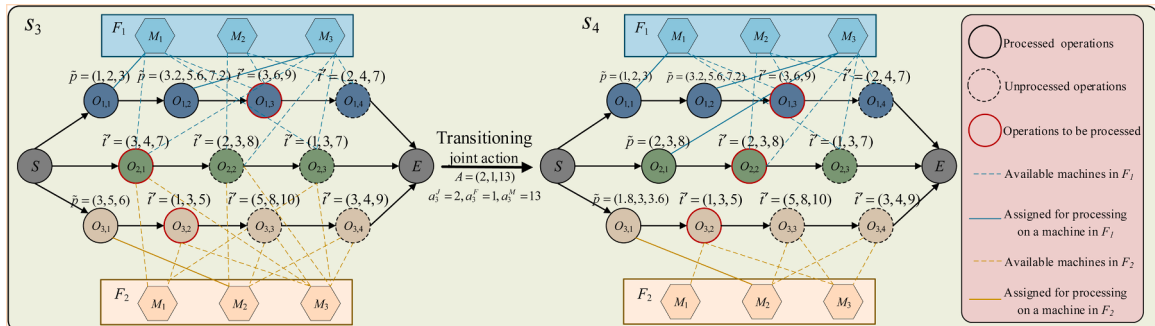


Fig. 8. Illustration of the state transition for triple-MDP.

Table 5Details of $\mathcal{A}_J(3)$ with feature mappings.

Features	$\mathcal{A}_J(3)$		
	Action 1 [#]	Action 2 [#]	Action 3 [#]
$CP_i^{\theta_1}(3)$	0	0	0
$TPT_i^{\theta_1}(3)$	(3,6,9)	(3,4,7)	(1,3,5)
$OCT_i^{\theta_1}(3)$	(5,9,12)	(0,0,0)	(3,5,6)
$PD_i^{\theta_1}(3)$	(5,9,12)	(0,0,0)	(3,5,6)
$CP_i^{\theta_2}(3)$	0	0	0
$TPT_i^{\theta_2}(3)$	(9,18,27)	(9,12,21)	(3,9,15)
$OCT_i^{\theta_2}(3)$	(5,9,12)	(0,0,0)	(3,5,6)
$PD_i^{\theta_2}(3)$	(15,27,36)	(0,0,0)	(9,15,18)

Table 6Details of $\mathcal{A}_F(3)$ with feature mappings.

Features	$\mathcal{A}_F(3)$	
	Action 1 [#]	Action 2 [#]
$FPL_f(3)$	(5,9,12)	(3,5,6)
$SPD_f(3)$	(4,2,10,6,15,2)	(4,6,19)

Table 7Details of $\mathcal{A}_M(3)$ with feature mappings at speed level V_3 .

Features	$\mathcal{A}_M(3)$		
	Action 1 [#]	Action 2 [#]	Action 3 [#]
$UR_k^{\theta_1}(3)$	100	-1	72
$MCT_k^{\theta_1}(3)$	(1,2,3)	-1	(4,2,7,6,10,2)
$MPL_k^{\theta_1}(3)$	(1,2,3)	-1	(3,2,5,6,7,2)
$PT_k^{\theta_1}(3)$	(4,5,6)	-1	(2,3,8)
$UR_k^{\theta_2}(3)$	100	-1	72
$MCT_k^{\theta_2}(3)$	(1,2,3)	-1	(4,2,7,6,10,2)
$MPL_k^{\theta_2}(3)$	(3,6,9)	-1	(9,6,16,8,21,6)
$PT_k^{\theta_2}(3)$	(36,45,54)	-1	(18,27,24)

transitions to the next decision time step $t = 4$. This transition process is the foundation for the subsequent description of the triple-MDP. The detailed formulation of the triple-MDP is presented as follows:

4.3.1. Action sets

The action sets in MARL are crucial as they delineate the scope of decisions agents can execute, directly impacting their ability to explore environments, learn knowledge, and optimize policies in complex environments. These agents $Agent_J$, $Agent_M$ and $Agent_F$ have independent action set at decision time step t : $\mathcal{A}_J(t)$, $\mathcal{A}_M(t)$, and $\mathcal{A}_F(t)$, respectively. At the initial decision time step $t = 0$, the first operation of each job is ready for processing. The action set $\mathcal{A}_J(t)$ involved the dispatching of n jobs, and as the decision-making progresses, the operations of the jobs are gradually scheduled, leading to a reduction of the action set $\mathcal{A}_J(t)$. When the action set $\mathcal{A}_J(t)$ is empty, it indicates that all jobs have been fully processed and signifies the end of scheduling. The action set $\mathcal{A}_F(t)$ involved allocating of available factories, allowing each job to be assigned to the most suitable factory for processing, and the action set $\mathcal{A}_M(t)$ determines the set of eligible machines for the execution of the operation $O_{i,j}$ of the job I_i determined by the decision action $a_t^J \in \mathcal{A}_J(t)$ of $Agent_J$. Here, $O_{i,j}$ refers to the operation of job I_i that is available for processing at the decision time step t . Meanwhile, the action set $\mathcal{A}_M(t)$ is further expanded by incorporating the relevant actions for selecting each speed level $V_v \in \mathbf{V}$ of machines and adapting the relevant features to the speed level V_v . For the states at the decision time step $t = 3$ in Fig. 8, the action sets $\mathcal{A}_J(3)$, $\mathcal{A}_F(3)$, and $\mathcal{A}_M(3)$ with their corresponding feature mappings are listed in Tables 5–7, where details of the action set $\mathcal{A}_M(3)$ with speed level V_3 are given in Table 7.

4.3.2. State representation

The state representation is crucial in MARL, and capturing critical characteristics and dynamics from the triple-MDP allows agents to learn policies efficiently and suitably coordinate their behaviors, thus determining how agents interpret and respond to the potential actions with others. At decision time step t , the state s_t can be represented as $s_t = (s_t^{J-\theta_1}, s_t^{J-\theta_2}, s_t^{M-\theta_1}, s_t^{M-\theta_2}, s_t^F)$. $s_t^{J-\theta_1} = [CP_1^{\theta_1}(t), TPT_1^{\theta_1}(t), OCT_1^{\theta_1}(t), PD_1^{\theta_1}(t), \dots, CP_n^{\theta_1}(t), TPT_n^{\theta_1}(t), OCT_n^{\theta_1}(t), PD_n^{\theta_1}(t)] \in \mathcal{S}_J^{\theta_1}$ and $s_t^{J-\theta_2} = [CP_1^{\theta_2}(t), TPT_1^{\theta_2}(t), OCT_1^{\theta_2}(t), PD_1^{\theta_2}(t), \dots, CP_n^{\theta_2}(t), TPT_n^{\theta_2}(t), OCT_n^{\theta_2}(t), PD_n^{\theta_2}(t)] \in \mathcal{S}_J^{\theta_2}$ represent the feature mappings of each action a_t^J in the action set $\mathcal{A}_J(t)$, which consist of the following features:

(1) $CP_i^{\theta_1}(t)$ denotes the completion states of all operations of job I_i at the decision time step t .

$$CP_i^{\theta_1}(t) = CP_i^{\theta_1}(t) = \begin{cases} 1, & \text{if job } I_i \text{ is done,} \\ 0, & \text{otherwise.} \end{cases} \quad (27)$$

(2) $TPT_i^{\theta_1}(t)$ represents the average fuzzy processing time of operation $O_{i,j}$ for all available machines, where operation $O_{i,j}$ is processable operation for job I_i at the decision time step t . $TPT_i^{\theta_1}$ indicates the average fuzzy EC for processing operation $O_{i,j}$ on available machines, calculated as:

$$TPT_i^{\theta_1}(t) = \frac{1}{|\mathbf{UM}_{i,j}|} \sum_{k \in \mathbf{UM}_{i,j}} \tilde{p}_{k,i,j}, \quad (28)$$

$$TPT_i^{\theta_2}(t) = TPT_i^{\theta_1}(t) \times PP. \quad (29)$$

(3) $OCT_i^{\theta_1}(t)$ denotes the fuzzy completion time of the predecessor operation $O_{i,j-1}$ for operation $O_{i,j}$ at the decision time step t . If $O_{i,j}$ has no predecessor, then $OCT_i^{\theta_1}(t) = 0$. Similarly, $OCT_i^{\theta_2}(t) = OCT_i^{\theta_1}(t)$.

(4) $PD_i^{\theta_1}(t)$ denotes the fuzzy processed time of job I_i at the decision time step t and $PD_i^{\theta_2}(t)$ is calculated by $PD_i^{\theta_2}(t) = PD_i^{\theta_1}(t) \times PP$.

The $s_t^{M-\theta_1} = [UR_1^{\theta_1}(t), MCT_1^{\theta_1}(t), MPL_1^{\theta_1}(t), PT_1^{\theta_1}(t), \dots, UR_{s \times m}^{\theta_1}(t), MCT_{s \times m}^{\theta_1}(t), MPL_{s \times m}^{\theta_1}(t), PT_{s \times m}^{\theta_1}(t)] \in \mathcal{S}_M^{\theta_1}$ and $s_t^{M-\theta_2} = [UR_1^{\theta_2}(t), MCT_1^{\theta_2}(t), MPL_1^{\theta_2}(t), PT_1^{\theta_2}(t), \dots, UR_{s \times m}^{\theta_2}(t), MCT_{s \times m}^{\theta_2}(t), MPL_{s \times m}^{\theta_2}(t), PT_{s \times m}^{\theta_2}(t)] \in \mathcal{S}_M^{\theta_2}$ represent the feature mappings of each action a_t^M in the action set $\mathcal{A}_M(t)$, which contain the following features:

- (1) $UR_k^{\theta_1}(t)$ indicates the utilization rate of machine M_k^f with speed level V_v at the decision time step t , where the factory F_f is the factory to which job I_i belongs. The job I_i is selected by $Agent_J$ via the decision action $a_t^J \in \mathcal{A}_J(t)$, and $UR_k^{\theta_2}(t) = UR_k^{\theta_1}(t)$.
- (2) $MCT_k^{\theta_1}(t)$ represents the fuzzy ready time of machine M_k^f with speed level V_v at the decision time step t , and $MCT_k^{\theta_2}(t) = MCT_k^{\theta_1}(t)$.
- (3) $MPL_k^{\theta_1}(t)$ denotes the fuzzy load time of machine M_k^f with speed level V_v at the decision time step t , and $MPL_k^{\theta_2}(t) = MPL_k^{\theta_1}(t) \times PP$.
- (4) $PT_k^{\theta_1}(t)$ refers to the fuzzy processing time of operation $O_{i,j}$ of job I_i selected by the decision action $a_t^J \in \mathcal{A}_J(t)$ of $Agent_J$ on the machine M_k^f with speed level V_v at the decision time step t , and $MPL_k^{\theta_2}(t) = MPL_k^{\theta_1}(t) \times em_v \times PP$.

The $s_t^F = [FPL_1(t), SPD_1(t), \dots, FPL_l(t), SPD_l(t)] \in \mathcal{S}_F$ represents the feature mappings of each action a_t^F in the action set $\mathcal{A}_F(t)$, which consists of two types of features:

- (1) $FPL_f(t)$ denotes the fuzzy processing load of factory F_f at the decision time step t .

$$FPL_f(t) = \sum_{k=1}^m MPL_k(t). \quad (30)$$

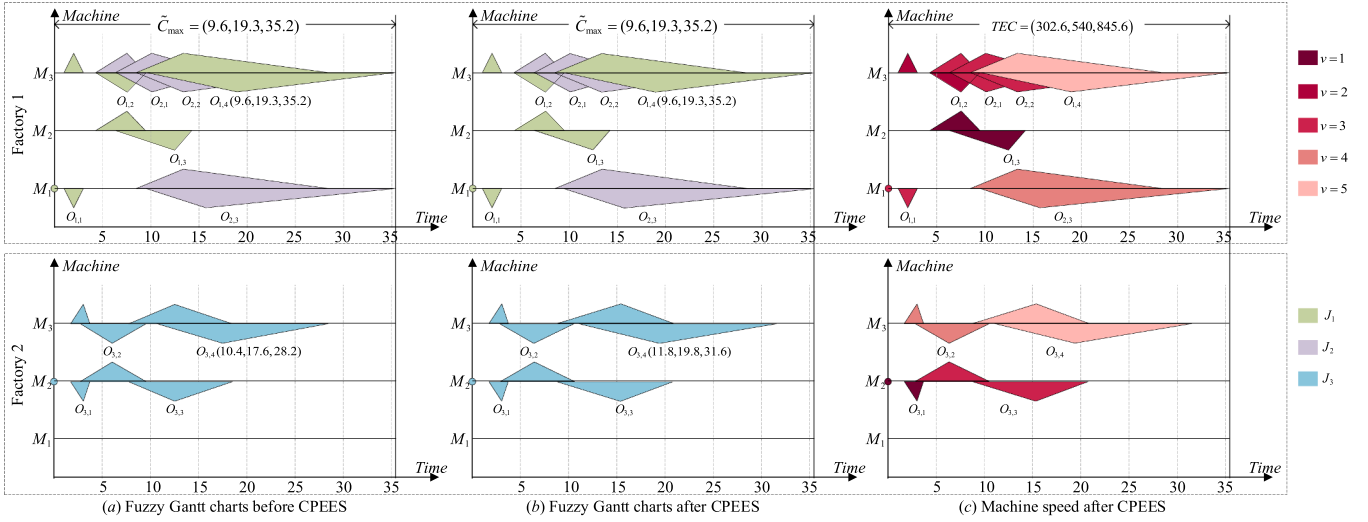


Fig. 9. An example of the implementation of CPEES.

- (2) $SPD_f(t)$ represents the predicted $\tilde{C}_{\max}$ obtained by performing a complete pre-scheduling in factory F_f for job I_i selected by the $Agent_f$ via the decision action $a_t^f \in \mathcal{A}_f(t)$. The job I_i is pre-scheduled in factory F_f by selecting the machine with the shortest fuzzy processing time for each operation.

4.3.3. Reward function

The design of MARL's rewards involves considering agents' interactions and making trade-offs between objectives to ensure that local behavior is consistent with the global goal. The two agents $Agent_f$ and $Agent_M$ share the same global goal of minimizing $\tilde{C}_{\max}$ and $\widetilde{TEC}$. The operations to be processed are virtually scheduled, using the average value of available fuzzy processing times as placeholders until all operations are pre-scheduled, thus yielding the predicted solution π at decision time step t . The reward r_t^{JM} comprises two parts $r_t^{JM-\varphi_1}$ and $r_t^{JM-\varphi_2}$, $r_t^{JM-\varphi_1} = \tilde{C}_{\max}(\pi) - \tilde{C}_{\max}(\pi')$ and $r_t^{JM-\varphi_2} = \widetilde{TEC}(\pi) - \widetilde{TEC}(\pi')$, where π' is the expected solution at the decision time step $t+1$. Since $Agent_f$ handles the appropriate allocation of jobs to the factories, the reward r_t^f must keep a balance between the factory's processing capacity and efficiency. After $Agent_f$ assigning the job I_i selected by $Agent_f$ via $a_t^f \in \mathcal{A}_f(t)$ to factory F_f via $a_t^f \in \mathcal{A}_f(t)$, all unprocessed operations are scheduled using the designed PDR, i.e., the earliest available operation with the lowest load machine (EAO-LLM). The reward r_t^f can be calculated as $r_t^f = - \sum_{1 \leq f \leq F} |\tilde{C}_f(\pi) - \tilde{C}_f(\pi')|$.

State $s_3 = (s_3^{J-\varphi_1}, s_3^{J-\varphi_2}, s_3^{M-\varphi_1}, s_3^{M-\varphi_2}, s_3^F)$ illustrated in Fig. 8 as an example. The unprocessed operations $O_{1,3}, O_{1,4}, O_{2,1}, O_{2,2}, O_{2,3}, O_{3,2}, O_{3,3}$ and $O_{3,4}$ are pre-scheduled at the decision time step $t=3$, resulting in a feasible solution π with $\tilde{C}_{\max}(\pi) = (15.2, 27.6, 48.2)$. The triple-MDP transition from state s_3 to state s_4 based on the joint action $A = (a_3^J = 2, a_3^F = 1, a_3^M = 13)$, where the current processable operation $O_{2,1}$ of the job I_2 (job index i is $i = a_3^J = 2$) is processed on the machine M_3^J (machine index k is $k = (a_3^M/s) + 1 = (13/5) + 1 = 3$, and factory index f is $f = a_3^F = 1$) with the speed level V_3 (speed level index v is $v = a_3^M \bmod s = 13 \bmod 5 = 3$), resulting in the next state s_4 . At the decision time step $t=4$, the unprocessed operations are $O_{1,3}, O_{1,4}, O_{2,2}, O_{2,3}, O_{3,2}, O_{3,3}$ and $O_{3,4}$. Through virtual scheduling, a feasible solution π' is yielded, resulting in $\tilde{C}_{\max}(\pi') = (14.2, 26.6, 47.2)$. The reward $r_3^{JM-\varphi_1} = \tilde{C}_{\max}(\pi') - \tilde{C}_{\max}(\pi) = (1, 1, 1)$ and the computation of reward $r_3^{JM-\varphi_2}$ follows a similar method to that of reward $r_3^{JM-\varphi_1}$. In the process of

calculating reward r_3^F , based on joint action $A = (a_3^J = 2, a_3^F = 1, a_3^M = 13)$, job I_2 is assigned to factory F_2 . Then, the EAO-LLM is used to all unprocessed operations, resulting in a feasible solution π'' . The reward r_3^F is calculated as $r_3^F = -|\tilde{C}_2(\pi'') - \tilde{C}_1(\pi'')| = -|(6, 9, 11) - (7.2, 14.6, 23.2)| = -(1.2, 5.6, 12.2)$.

4.4. Critical-path-based energy-efficient strategy

After decomposing the EE-FFJSP into a set of weighted objective subproblems using the subproblem decomposition method, the subproblem $SubPB_h$ can independently generate a solution, denoted as solution π_h . Collectively, the solutions obtained from all subproblems can constitute the Pareto solution set. Although an appropriate balance already exists between $\tilde{C}_{\max}(\pi_h)$ and $\widetilde{TEC}(\pi_h)$, it is necessary to adjust the speed levels of operations on non-critical paths for machines M_k^f to further reduce $\widetilde{TEC}(\pi_h)$ (see Section 3.3). This section proposes an effective critical-path-based-EES (CPEES), building upon Property 1 in Section 3.3. The following outlines the detailed steps:

Step 1: Critical factories are identified, followed by the determination of critical paths within critical factories. In cases where multiple critical paths exist, one path is randomly selected as the critical path.

Step 2: Determine the conditions for speed reduction by traversing each operation O_{ij} on non-critical paths to verify whether the speed-down criteria are satisfied: (1) The fuzzy completion time of operation O_{ij} not exceeds the fuzzy start time of operation O_{ij+1} . (2) The fuzzy completion time of operation O_{ij} not exceeds the fuzzy start time of the next operation on the same machine. (3) The processing machine M_k^f for operation O_{ij} operates at a speed level $V_v \in \mathbf{V}$ that is higher than V_s .

Step 3: Calculate adjustable fuzzy time margins AT_{ij} and UAT_{ij} by Eq. (24) and Eq. (25), respectively.

Step 4: Gradually reduce the processing speed level $V_v \in \mathbf{V}$ of operation O_{ij} on machine M_k^f to a level $V_v \in \mathbf{V}$, and continue the speed level reduction until $\tilde{t}_{k,ij} \times sm_v - \tilde{t}_{k,ij} \times sm_v > AT_{ij}(\text{or } UAT_{ij})$ is satisfied. After executing speed level adjustment, determine whether the critical path has changed; If a change is detected, revert the speed level back to V_v and return to Step 2. Repeat this process until all operations on the non-critical paths have been thoroughly checked.

As depicted in Fig. 9, this schematic illustrates the proposed CPEES, based on the example in Fig. 2. Fig. 9(b) shows the Gantt chart after CPEES implementation, while Fig. 9(c) displays the processing speed level settings of each machine for every operation after executing CPEES. In Fig. 9(a), factory F_1 is designated as the critical factory, with

the critical path being $(O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{1,4})$. The related operation is $O_{1,3}$ and the unrelated operations are $O_{2,3}$, $O_{3,1}$, $O_{3,2}$, $O_{3,3}$, and $O_{3,4}$. However, only operations $O_{3,2}$ and $O_{3,3}$ meet the conditions in Step 2 for speed adjustment. Following the implementation of CPEES, the speed level vector $\psi_V = [3, 2, 3, 1, 1, 3, 4, 5, 3, 2, 5]$ in $\pi = (\psi_F, \psi_J, \psi_M, \psi_V)$ is transformed into $\psi'_V = [3, 2, 3, 1, 1, 3, 4, 5, 4, 3, 5]$ through the speed level reduction of $O_{3,2}$ and $O_{3,3}$. $\widetilde{TEC}(\pi')$ was reduced from $\widetilde{TEC}(\pi) = (320.6, 571.0, 883.6)$ to $(302.6, 540, 845.6)$. The pseudocode for the CPEES is given in Algorithm 2.

Algorithm 2. Critical-path-based energy-efficient strategy.

Input: Feasible solution π_h .

- 1: Decode the feasible solution π_h (see Section 3.1). Set $NOs = \emptyset$.
- 2: Identify the critical factory as well as the critical path within critical factory.
- 3: Sort all operations on non-critical path in ascending order of completion time.
- 4: Save all operations on non-critical path to NOs .
- 5: **While** $NOs \neq \emptyset$ **do**
- 6: Select the first non-critical operation $O_{i,j}$ from NOs .
- 7: **if** $O_{i,j}$ meets the speed down conditions **then**
- 8: **if** $O_{i,j}$ is related operation **then**
- 9: Calculate the $AT_{i,j}$ of $O_{i,j}$ by Eq. (24).
- 10: **for** $v' = v + 1$ **to** s **do**
- 11: **if** $\tilde{t}_{k,i,j} \times sm_{v'} - \tilde{t}_{k,i,j} \times sm_v > AT_{i,j}$ **then**
- 12: $v = v' - 1$.
- 13: **end if**
- 14: **end for**
- 15: **end if**
- 16: **if** $O_{i,j}$ is unrelated operation **then**
- 17: Calculate the $UAT_{i,j}$ of $O_{i,j}$ by Eq. (25).
- 18: **for** $v' = v + 1$ **to** s **do**
- 19: **if** $\tilde{t}_{k,i,j} \times sm_{v'} - \tilde{t}_{k,i,j} \times sm_v > UAT_{i,j}$ **then**
- 20: $v = v' - 1$.
- 21: **end if**
- 22: **end for**
- 23: **end if**
- 24: **end if**
- 25: Remove the current operation $O_{i,j}$ from NOs .
- 26: **end while**

Output: Updated feasible solution π_h .

4.5. Multi-objective DQN algorithm

For EE-DFFJSP, two conflicting objectives $\widetilde{TEC}$ and $\widetilde{C}_{max}$ need to be optimized cooperatively. Traditional DQN-based method select optimal actions by estimating Q-values [67]. However, the single Q-value update mechanism is inadequate for balancing two objectives, which makes traditional DQN-based method difficult to find suitable solutions. This limitation reduces the diversity of obtainable Pareto solutions and hinders adaptability to dynamic changes in optimization objectives during subproblem solving. To address this limitation, MACMNG employs an MO-DQN algorithm during the training phase. In MO-DQN, the Q-values for optimization objectives are aggregated using uniformly objective weight vectors λ (as detailed in Section 4.1), enabling the cooperative

optimization of multiple conflicting objectives and adapting to dynamic objective weight changes. The MO-DQN algorithm effectively resolves conflicts between competing objectives while yielding a diverse set of Pareto solutions, thereby addressing the limitation of traditional DQN-based method. The implementation of MO-DQN is elaborated on in two key aspects: decision-making for actions and updating parameters for networks.

According to subproblem decomposition method (see Section 4.1), the EE-DFFJSP is decomposed into multiple subproblems ($SubPB_1, \dots, SubPB_h, \dots, SubPB_e$). Each subproblem is aimed at optimizing two objectives of $\widetilde{C}_{max}$ and $\widetilde{TEC}$. Fig. 10 provides an illustrative depiction for the

decision-making of MO-DQN. The agent $Agent_I$ and $Agent_M$ each contain a corresponding network group, namely Net^I and Net^M , respectively. Each network group consists of multiple subnets, referred to as $(SubNet_1^I, \dots, SubNet_h^I, \dots, SubNet_e^I)$ and $(SubNet_1^M, \dots, SubNet_h^M, \dots, SubNet_e^M)$, with the number of subnets corresponding to the total number of subproblems. Each subnet $SubNet_h$ is responsible for handling a specific subproblem $SubPB_h$. While all subnets share the same structural design, they have independent network parameters, enabling the learning of distinct policies corresponding to their respective subproblems. At decision time step t , the scheduling environment first extracts the states $s_t^{I-\varphi_1}$ and $s_t^{I-\varphi_2}$. These state features are input into the subnet $SubNet_h^I$ corresponding to the current subproblem $SubPB_h$ within the job agent's network group Net^I . After calculating the Q-values $Q_t^{I-\varphi_1}$ and $Q_t^{I-\varphi_2}$ for the two optimization objectives $\widetilde{C}_{max}$ and $\widetilde{TEC}$, the action a_t^I

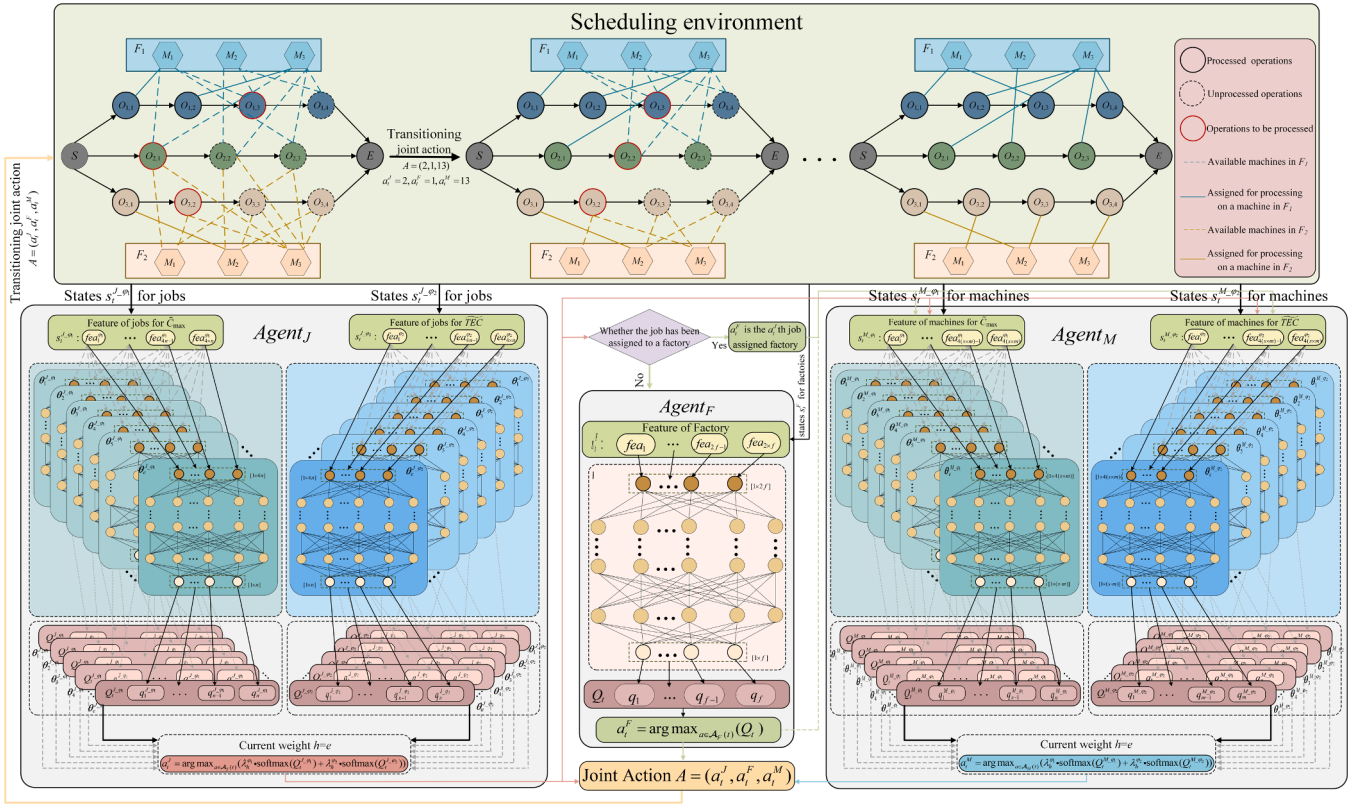


Fig. 10. Schematic diagram for the decision-making of MO-DQN.

of the job agent $Agent_J$ is determined according to the action selecting function defined in Eq. (31). Subsequently, it is determined whether the selected job has already been assigned to a factory. If it has, the factory agent $Agent_F$ directly outputs its action a_t^F , pointing to the assigned factory. If not, the factory state features s_t^F are extracted, and the factory agent $Agent_F$ computes the Q -values Q_t through its network to determine action a_t^F using Eq. (32). Based on actions a_t^J and a_t^F , the machine states $s_t^{M-\varphi_1}$ and $s_t^{M-\varphi_2}$ at the decision time step t are then extracted. These state features are processed through the relevant subnet $SubNet_h^M$ in the machine agent's network group Net^M to obtain Q -values $Q_t^{M-\varphi_1}$ and $Q_t^{M-\varphi_2}$, and the machine agent's action a_t^M is selected according to Eq. (33). These three actions collectively form the joint action $A = (a_t^J, a_t^F, a_t^M)$, thereby completing the cooperative decision-making process.

$$a_t^J = \arg\max_{a \in \mathcal{A}_J(t)} \left(\lambda_h^{\varphi_1} \cdot \text{softmax} \left(Q_t^{J-\varphi_1} \left(s_t^{J-\varphi_1}, a; \theta_h^{J-\varphi_1} \right) \right) + \lambda_h^{\varphi_2} \cdot \text{softmax} \left(Q_t^{J-\varphi_2} \left(s_t^{J-\varphi_2}, a; \theta_h^{J-\varphi_2} \right) \right) \right) \quad (31)$$

$$a_t^F = \arg\max_{a \in \mathcal{A}_F(t)} Q_t(s_t^F, a; \theta) \quad (32)$$

$$a_t^M = \arg\max_{a \in \mathcal{A}_M(t)} \left(\lambda_h^{\varphi_1} \cdot \text{softmax} \left(Q_t^{M-\varphi_1} \left(s_t^{M-\varphi_1}, a; \theta_h^{M-\varphi_1} \right) \right) + \lambda_h^{\varphi_2} \cdot \text{softmax} \left(Q_t^{M-\varphi_2} \left(s_t^{M-\varphi_2}, a; \theta_h^{M-\varphi_2} \right) \right) \right) \quad (33)$$

Here, $\lambda_h^{\varphi_1}$ and $\lambda_h^{\varphi_2}$ represent the objective weights for $\tilde{C}_{\max}$ and $\tilde{TEC}$ of $SubPB_h$, respectively. The SoftMax function is used to the Q -values of each action, ensuring that the selection probability of actions is biased towards those with higher expected utility.

Since subnet $SubNet_h^J$ and subnet $SubNet_h^M$ share the same structural design and parameters update mechanism, they are collectively referred to as subnet $SubNet_h$ in the following discussion. Accordingly, their corresponding states, actions, and rewards are represented using unified notation; for instance, symbol $s_t^{\varphi_1}$ is used to denote both $s_t^{J-\varphi_1}$ and $s_t^{M-\varphi_1}$.

As illustrated in Fig. 11, ensure that $SubNet_h$ can optimize according to λ_h , the update of network parameters $[\theta_h^{\varphi_1}; \theta_h^{\varphi_2}]$ must take into account the comprehensive influence of $\tilde{C}_{\max}$ and $\tilde{TEC}$. Building upon the DQN-based method network update mechanism, a weighting mechanism is introduced to better balance the two objectives $\tilde{C}_{\max}$ and $\tilde{TEC}$. Upon completion of the decision-making processes by $Agent_J$ or $Agent_M$, an experience tuple $(s_t^{\varphi_1}, s_t^{\varphi_2}, a_t, r_t^{\varphi_1}, r_t^{\varphi_2}, s_{t+1}^{\varphi_1}, s_{t+1}^{\varphi_2})$ is generated by the scheduling environment and used to replace a corresponding sample in the replay buffer. Subsequently, a batch of experiences is sampled to compute the losses (L_{φ_1} and L_{φ_2}) associated with the two optimization objectives $\tilde{C}_{\max}$ and $\tilde{TEC}$, respectively, according to Eqs. (34)–(37). The total weighted loss L_{total} is then calculated by Eq. (38) and employed to update the online Q -network parameters $\theta_h^{\varphi_1}$ and $\theta_h^{\varphi_2}$. Every u update iterations, the values of $\theta_h^{\varphi_1}$ and $\theta_h^{\varphi_2}$ are copied to $\bar{\theta}_h^{\varphi_1}$ and $\bar{\theta}_h^{\varphi_2}$, respectively, for updating the target Q -network parameters. The corresponding pseudocode is detailed in Algorithm 3.

$$y_t^{\varphi_1} = r_t^{\varphi_1} + \gamma Q_t^{\varphi_1}(s_{t+1}^{\varphi_1}, \arg\max_{a \in \mathcal{A}(t+1)} Q_{t+1}^{\varphi_1}(s_{t+1}^{\varphi_1}, a; \bar{\theta}_h^{\varphi_1}); \theta_h^{\varphi_1}), \quad (34)$$

$$L_{\varphi_1} = \mathbb{E} \left[\left(y_t^{\varphi_1} - Q_t^{\varphi_1}(s_t^{\varphi_1}, a_t; \theta_h^{\varphi_1}) \right)^2 \right] \quad (35)$$

$$y_t^{\varphi_2} = r_t^{\varphi_2} + \gamma Q_t^{\varphi_2}(s_{t+1}^{\varphi_2}, \arg\max_{a \in \mathcal{A}(t+1)} Q_{t+1}^{\varphi_2}(s_{t+1}^{\varphi_2}, a; \bar{\theta}_h^{\varphi_2}); \theta_h^{\varphi_2}) \quad (36)$$

$$L_{\varphi_2} = \mathbb{E} \left[\left(y_t^{\varphi_2} - Q_t^{\varphi_2}(s_t^{\varphi_2}, a_t; \theta_h^{\varphi_2}) \right)^2 \right] \quad (37)$$

$$L_{\text{total}} = \lambda_h^{\varphi_1} \cdot L_{\varphi_1} + \lambda_h^{\varphi_2} \cdot L_{\varphi_2} \quad (38)$$

While performing network parameters $[\theta, \bar{\theta}]$ updates, the $Agent_F$, similar to its decision-making process, does not consider the influence of weight vectors λ_h . Fig. 12 illustrates the update process of network parameters for $Agent_F$. After the decision-making process of $Agent_F$ is completed, the scheduling environment generates an experience tuple $(s_t^F, a_t^F, r_t^F, s_{t+1}^F)$, which is used to replace a corresponding sample in the

replay buffer. A batch of experiences is then sampled from the replay buffer to compute the loss L_F according to Eq. (39) and Eq. (40), which is subsequently used to update the online Q -network parameters θ . After every u' update iterations, the values of the online Q -network parameters θ are used to update the target Q -network parameters θ' . The corresponding pseudocode is provided in Algorithm 4. The overall training process of the MACMNG is provided in Algorithm 5.

$$y_t = r_t^F + \gamma Q_t(s_{t+1}^F, \arg\max_{a \in \mathcal{A}_F(t+1)} Q_{t+1}(s_{t+1}^F, a; \theta); \theta') \quad (39)$$

$$L_F = \mathbb{E}[(y_t - Q_t(s_t^F, a_t^F; \theta))^2] \quad (40)$$

The flowchart of the EE-DFFJSP using the proposed MACMNG is illustrated in Fig. 13. First, the original EE-DFFJSP is decomposed into e

Algorithm 4. Updating network parameters of $Agent^F$

Input: Network parameters $[\theta, \theta']$; Experience tuples $(s_t^F, a_t^F, r_t^F, s_{t+1}^F)$; update frequency u' .

- 1: **if** s_{t+1}^F is terminal **then**
- 2: Calculate the $y_t = r_t^F$.
- 3: **else**
- 4: Calculate the y_t by Eq. (39).
- 5: **end if**
- 6: Perform a gradient descent with L_F by Eq. (40).
- 7: Set $\theta' = \theta$ every u' steps.

Output: Updated network parameters $[\theta, \theta']$.

Algorithm 5. Training the MACMNG via MO-DQN algorithm

Input: $Agent_J$ and $Agent_M$ Network group Net^J Net^M ; $Agent_F$; Number of episodes: H ; Uniformly objective weight vectors: $\lambda_1, \lambda_2, \dots, \lambda_e$.

- 1: Random initialize parameters $[\theta, \theta']$.
- 2: **for** $h = 1, \dots, e$ **do**
- 3: Initialize $SubNet_h^J$ and $SubNet_h^M$ by Algorithm 1.
- 4: **for** $episode = 1, \dots, H$ **do**
- 5: Generate an instance of EE-DFFJSP, denoted ED .
- 6: **while** ED not finish **do**
- 7: Collect experience $(s_t^{J-\varphi_1}, s_t^{J-\varphi_2}, a_t^J, r_t^{JM-\varphi_1}, r_t^{JM-\varphi_2}, s_{t+1}^{J-\varphi_1}, s_{t+1}^{J-\varphi_2})$.
- 8: Collect experience $(s_t^{M-\varphi_1}, s_t^{M-\varphi_2}, a_t^J, r_t^{JM-\varphi_1}, r_t^{JM-\varphi_2}, s_{t+1}^{M-\varphi_1}, s_{t+1}^{M-\varphi_2})$.
- 9: Collect experience $(s_t^F, a_t^F, r_t^F, s_{t+1}^F)$.
- 10: Update network parameters for $SubNet_h^J$ by Algorithm 3.
- 11: Update network parameters for $SubNet_h^M$ by Algorithm 3.
- 12: Update network parameters for $Agent_F$ by Algorithm 4.
- 13: **end while**
- 14: **end for**
- 15: **end for**

Output: $Agent_J$, $Agent_M$ and $Agent_F$.

subproblems based on a subproblem decomposition method (see Section 4.1). Then, according to Algorithm 5, a set of randomly generated instances is used to train the network parameters of $Agent_J$, $Agent_M$, and $Agent_F$. After the training phase is completed, the trained agents are utilized to solve the e subproblems of the original EE-DFFJSP, resulting in the final solution set denoted as Π . The CPEES is performed for each feasible solution in Π via Algorithm 2, and the Pareto solution set is updated.

Algorithm 3. Updating network parameters of subnet $SubNet_h$

Input: Network parameters $[\theta_h^{q_1}, \theta_h^{q_2}, \theta_h^{q_2}, \theta_h^{q_2}]$; Experience tuples $(s_t^{q_1}, s_t^{q_2}, a_t^J, r_t^{JM-\varphi_1}, r_t^{JM-\varphi_2}, s_{t+1}^{q_1}, s_{t+1}^{q_2})$; λ_h ; update frequency u ; γ .

- 1: **if** $s_{t+1}^{q_1}$ and $s_{t+1}^{q_2}$ is terminal **then**
- 2: Calculate $y_t^{q_1} = r_t^{q_1}$ and $y_t^{q_2} = r_t^{q_2}$.
- 3: **else**
- 4: Calculate $y_t^{q_1}$ and $y_t^{q_2}$ by Eq. (34) and Eq. (36).
- 5: **end if**
- 6: Calculate losses L_{q_1} and L_{q_2} by Eq. (35) and Eq. (37).
- 7: Perform a gradient descent with L_{total} by Eq. (38).
- 8: Set $\theta_h^{q_1} = \theta_h^{q_1}, \theta_h^{q_2} = \theta_h^{q_2}$ every u steps.

Output: Updated network parameters $[\theta_h^{q_1}, \theta_h^{q_2}, \theta_h^{q_2}, \theta_h^{q_2}]$.

5. Experimental comparisons and results analysis

This section provides a comprehensive analysis of the performance of the MACMNG for solving EE-DFFJSP. Section 5.1 introduces in detail the benchmark datasets, the experimental setup, and the hyperparameter settings in MACMNG. Section 5.2 describes the performance evaluation metrics for multi-objective algorithms. Section 5.3 presents a comparative analysis of MACMNG and composite PDRs across all benchmark instances. Section 5.4 provides an in-depth and detailed experimental analysis that compares MACMNG with state-of-the-art multi-objective algorithms and verifies the effectiveness and efficiency of MACMNG.

5.1. Experimental details

To evaluate the performance of MACMNG in solving the EE-DFFJSP, three benchmark datasets from existing literature were selected for validation:

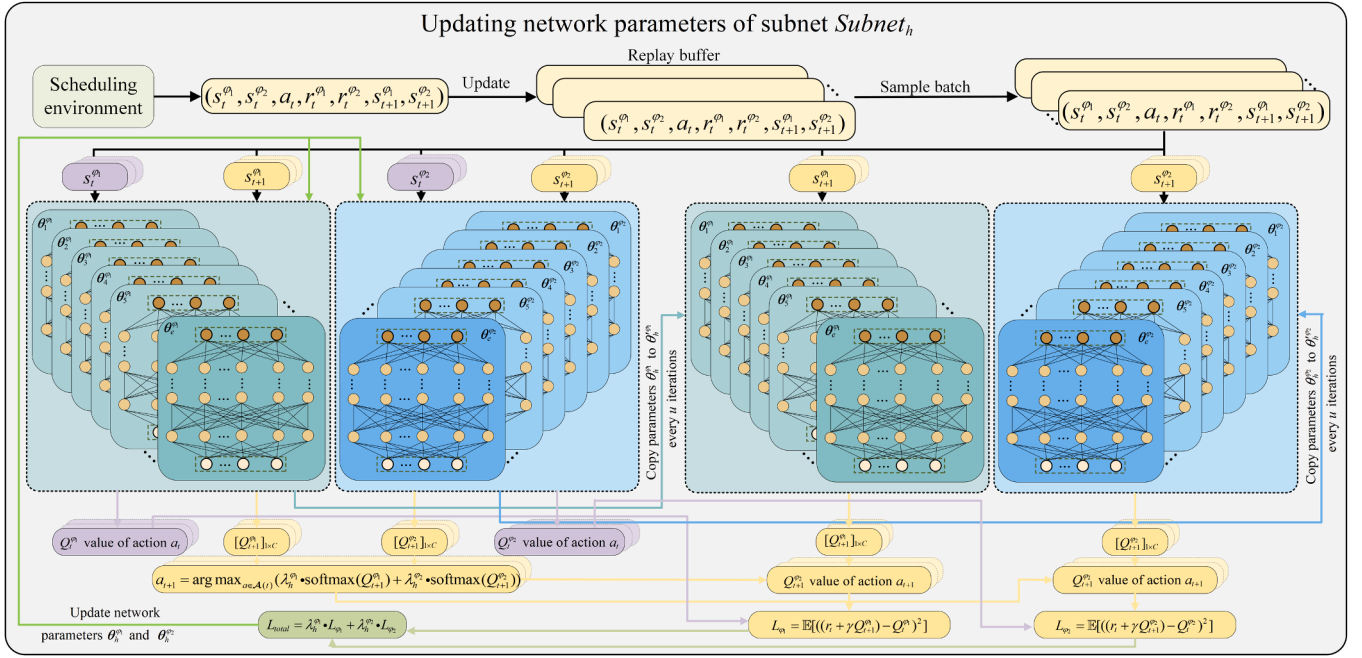


Fig. 11. Schematic diagram for updating network parameters of subnet *SubNet_h*.

- The first dataset, derived from *Lei et al.* [62,68], comprises five instances (*Lei01~Lei05*) with the number of jobs $n \in \{10, 10, 10, 10, 15\}$, the number of operations $\{40, 40, 50, 50, 80\}$, and the number of machines $m = 10$.
- The second dataset, proposed by *Gao et al.* [69], includes eight instances (*Reman01~Reman08*) with the number of jobs $n \in \{5, 8, 10, 10, 15, 15, 20, 20\}$, the number of operations $\{23, 64, 81, 100, 171, 185, 308, 355\}$, and the number of machines $m \in \{4, 8, 6, 10, 8, 10, 10, 15\}$.
- The third dataset, introduced by *Li et al.* [70], contains ten instances (*FMK01~FMK10*) with the number of jobs $n \in \{10, 10, 15, 15, 15, 10, 20, 20, 20, 20\}$, the number of operations $\{55, 58, 150, 90, 106, 150, 100, 225, 240, 240\}$, and the number of machines. $m \in \{6, 6, 8, 8, 4, 15, 5, 10, 10, 15\}$.

These three types of instances are designed for fuzzy FJSP (FFJSP), were extended to adapt to the DFFJSP by incorporating the number of factories $l \in \{2, 3, 4\}$. These extended instances were renamed with appended factory identifiers (e.g., *FMK01~2* denotes the two-factory extension of *FMK01*). The machine speed configuration included five discrete levels $\mathbf{V} = \{1, 2, 3, 4, 5\}$, with corresponding speed coefficients $\mathbf{SM} = \{0.6, 0.8, 1, 1.2, 1.4\}$ and energy consumption (EC) coefficients $\mathbf{SE} = \{8, 5, 3, 2, 1\}$. The fuzzy processing times for all jobs were derived from the original benchmark instances, while actual fuzzy processing times were determined by selected speed levels. The standard processing unit EC is set at $PP = 3$ and the standby unit EC is set at $SP = 1$. For instances *Lei01* to *Lei05* [62,68], all machines are regarded as available candidate machines for processing, which implies a fully flexible production environment. In contrast, for instances *Reman01* to *Reman08* [69] and instances *FMK01* to *FMK10* [70], each operation is assigned at least one candidate machine for processing, ensuring partial flexibility in production environment. All algorithms, including MACMNG, are implemented in Python and executed on the workstation equipped with a 12th Gen Intel® Core™ i5-12400F CPU and an NVIDIA GeForce RTX 4060 GPU. All algorithms are independently executed across 10 runs for each instance, with each run having the same maximum elapsed CPU time of $l \times m \times n \times 0.1$ seconds. To validate the convergence and diversity of Pareto solutions yielded by MACMNG, average results across 10 runs are collected for comparative analysis using three performance metrics (see Section 5.2). These benchmark datasets and MACMNG's

implementation details are publicly accessible at <https://github.com/i-xw/MACMNG>. For MARL-based methods, the selection of hyperparameters not only affects the convergence speed of algorithms but also directly impacts the quality of solutions. To ensure that MACMNG achieves the best performance in addressing EE-DFFJSP, Table 8 provides a detailed explanation of the settings of each hyperparameter.

5.2. Performance metrics

To comprehensively evaluate the performance of MACMNG and comparative algorithms, three widely adopted performance metrics are employed to assess the quality and diversity of the obtained Pareto solutions:

- (1) *Hypervolume (HV)* [71]: This metric measures the volume of the objective space spanned by the Pareto front (PF) relative to a predefined reference point. A larger HV value indicates that the solution set is closer to the ideal Pareto front (PF^*) and exhibits superior distribution uniformity. Here, PF is a set of the Pareto solutions obtained by an algorithm, and PF^* is the reference set of the Pareto solutions, which is formed by combining all sets of Pareto solutions from all algorithms. The HV is calculated as:

$$HV(PF, P_{ref}) = \bigcup_{i \in PF} v(i, P_{ref}), \quad (41)$$

where $P_{ref} = (1, 1)$ is defined as the reference point and $v(i, P_{ref})$ represents the hypervolume enclosed between the reference point P_{ref} and the i -th solution in the Pareto front PF .

- (2) *Generational distance (GD)*: This metric quantifies the average minimum Euclidean distance from the obtained PF to PF^* . A smaller GD value implies that the solution set is closer to the ideal Pareto front PF^* . The GD is formulated as:

$$GD(PF, PF^*) = \sqrt{\frac{1}{|PF|} \sum_{i=1}^{|PF|} dist(i, PF^*)^2} \quad (42)$$

where $dist(i, PF^*)$ represents the Euclidean distance between the i -th solution in PF and its nearest neighbor solution in PF^* .

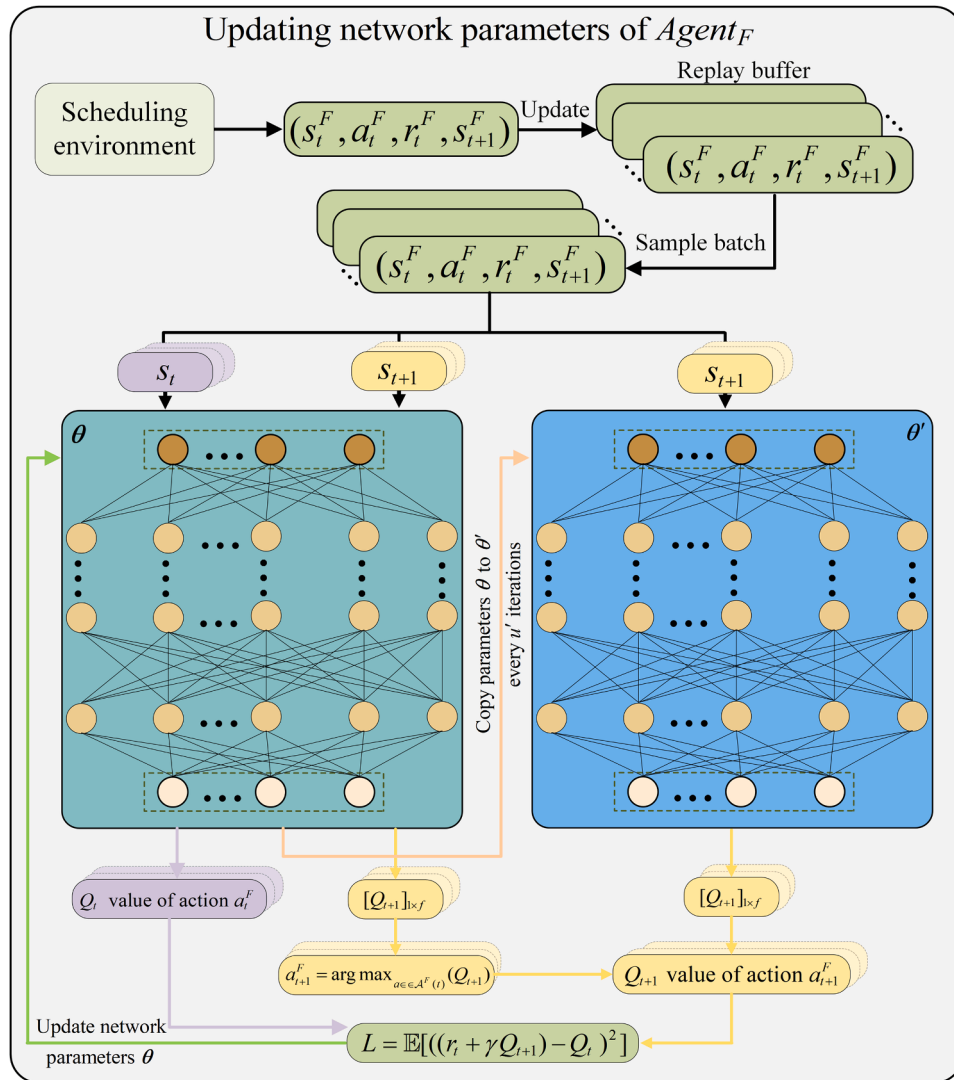


Fig. 12. Schematic diagram for updating network parameters of $Agent_F$.

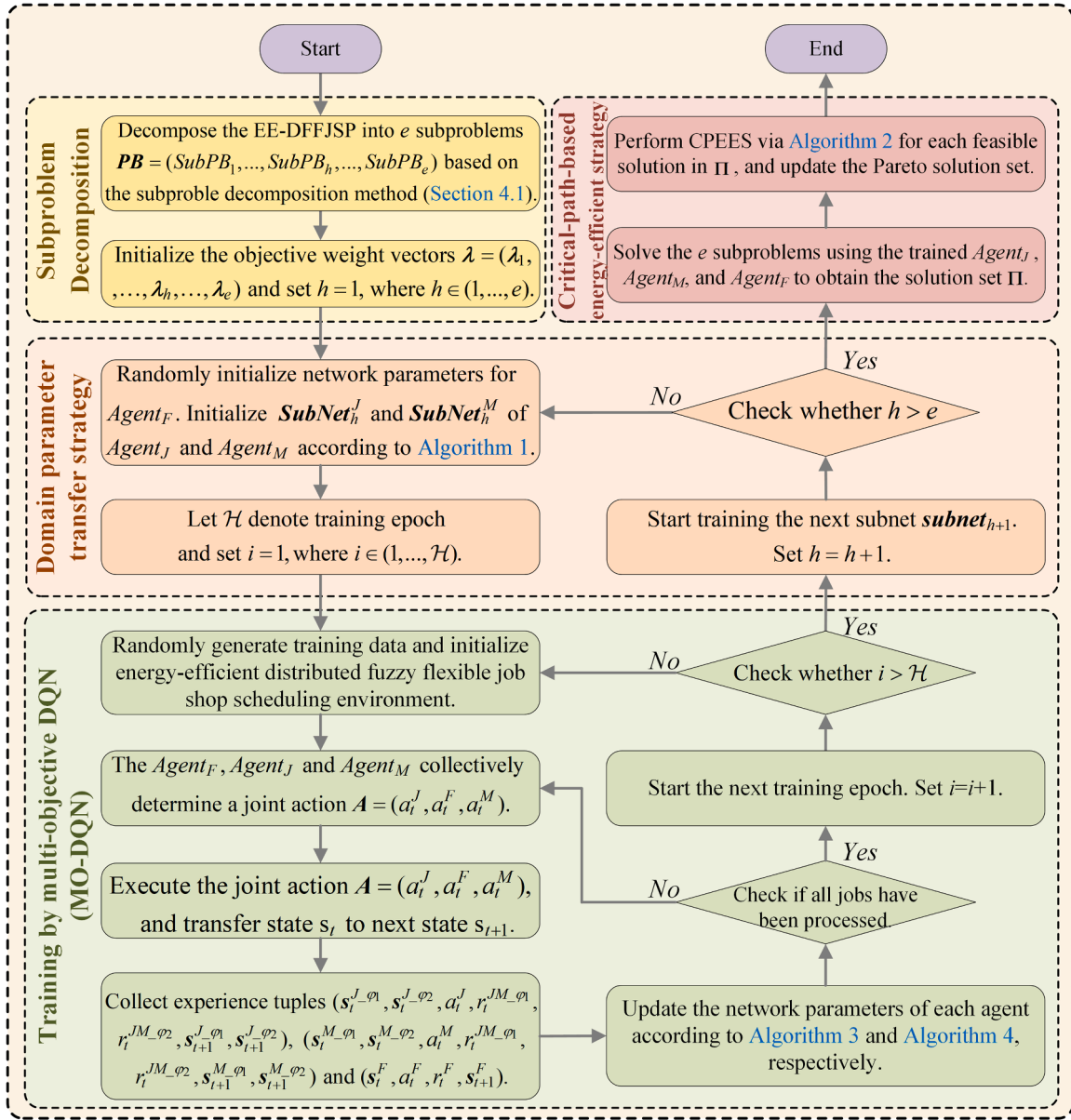


Fig. 13. The flowchart of MACMNG for EE-DFFJSP.

Table 8
Hyperparameters details.

Hyperparameter	Symbol (value)
Number of subproblems	$e = 101$
Uniformly objective weight vectors	$\lambda = ([1, 0], \dots, [0, 1])$
Number of hidden layers	2
Number of neurons in the hidden layer	512
Learning rate	$r = 0.00001$
Discount factor	$\gamma = 0.9$
Exploration rate	$\epsilon = 0.1$
Target network update frequency	$u = 10, u' = 10$
Experience replays buffer size	200
Minimum sample size	50
Batch size	32

(3) *Inverted generational distance (IGD)*: The IGD evaluates the average minimum distance from PF^* to PF , serving as a dual measure to GD . A smaller IGD value implies that PF is not only

closer to PF^* but also has comprehensive coverage. The IGD is defined as:

$$IGD(PF, PF^*) = \sqrt{\frac{1}{|PF^*|} \sum_{i=1}^{|PF^*|} \text{dist}(i, PF)^2}, \quad (43)$$

where $\text{dist}(i, PF)$ is the Euclidean distance between the i -th solution in ideal Pareto front PF^* and its closest counterpart in PF .

5.3. Comparison of MACMNG with effective PDRs

To evaluate the effectiveness of MACMNG, a comparative analysis was conducted against three state-of-the-art composite priority dispatching rules (PDRs) [50]: FIFO+EET, MWKR+EET, and MOPNR+EET. Given that these composite PDRs are designed to optimize the makespan objective ($\bar{C}_{\max}$), these comparisons focused on contrasting the Pareto solutions yielded by MACMNG (with minimal $\bar{C}_{\max}$) against those produced by PDRs that achieve the minimization of $\bar{C}_{\max}$. It is noted that the total energy consumption ($\bar{TEC}$) for all solutions

Table 9
Statistical results between MACMNG and FIFO+EET on both $\tilde{C}_{\max}$ and $\overline{TEC}$ for FMK instances.

Instances	FIFO+EET					MACMNG				
	V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	Best $\tilde{C}_{\max}$
<i>FMK01</i> – 2	$\tilde{C}_{\max}$ (13.6,21.6,29.0)	(20.8,28.8,36.0)	(26.0,36.0,45.0)	(31.2,43.2,54.0)	(36.4,50.4,63.0)	$\tilde{C}_{\max}$ (10.6,21.8,29.4)	(2363.8,3708.0,5053.0)	(1805.4,2832.8,3872.8)	(1480.4,2323.0,3181.0)	(1605.3,2590.8,3525.2)
<i>FMK02</i> – 2	$\overline{TEC}$ (3033.2,4869.0,6688.0)	(13.8,24.2,35.4)	(19.0,30.0,43.0)	(21.6,36.0,50.4)	(25.2,43.4,64.4)	$\overline{TEC}$ (10.8,19.6,29.4)	(2923.2,4825.8,6682.2)	(2147.4,3522.6,4895.0)	(1532.2,2578.8,3581.8)	(1319.6,2247.8,3217.8)
<i>FMK03</i> – 2	$\tilde{C}_{\max}$ (10.8,18.0,25.2)	(52.6,82.4,112.0)	(70.2,110.2,148.6)	(92.0,137.0,183.0)	(110.4,164.4,219.6)	$\tilde{C}_{\max}$ (62.6,96.4,126.8)	(68.0,105.0,134.2)	(104.0,135.2,175.2)	(156.0,202.8,262.8)	(182.0,236.6,306.6)
<i>FMK04</i> – 2	$\overline{TEC}$ (3507.0,5918.0,8189.4)	(38.4,59.2,84.0)	(49.0,74.0,102.0)	(57.6,88.8,126.0)	(67.2,103.6,147.0)	$\overline{TEC}$ (20.8,39.8,54.0)	(26751.2,37565.8,48833.0)	(14274.4,20267.2,26350.0)	(11552.4,16472.8,21364.6)	(7782.6,11705.0,15187.0)
<i>FMK05</i> – 2	$\tilde{C}_{\max}$ (25.2,45.0,65.4)	(3916.6,6241.6,8593.2)	(3168.2,5058.6,6928.0)	(2848.0,4576.4,6273.0)	(2521.4,4072.6,5570.6)	$\tilde{C}_{\max}$ (2528.8,4266.0,5837.6)	(5437.0,8774.6,11993.2)	(3168.2,5058.6,6928.0)	(2848.0,4576.4,6273.0)	(490.74,4.98.0)
<i>FMK06</i> – 2	$\overline{TEC}$ (52.6,82.4,112.0)	(70.2,110.2,148.6)	(92.0,137.0,183.0)	(110.4,164.4,219.6)	(128.8,191.8,256.2)	$\overline{TEC}$ (49.0,74.0,98.0)	(10986.6,16847.4,22749.4)	(8198.8,12501.2,16800.2)	(4900.8,7475.0,10037.6)	(6303.6,9366.8,12522.0)
<i>FMK07</i> – 2	$\tilde{C}_{\max}$ (33.8,54.6,77.8)	(52.0,72.8,96.8)	(56.0,94.0,122.0)	(78.0,109.2,145.2)	(91.0,127.4,169.4)	$\tilde{C}_{\max}$ (37.2,60.2,80.0)	(14317.4,21420.4,29057.8)	(8651.0,12901.4,17517.6)	(6743.4,10022.6,13579.0)	(4662.2,7420.8,9933.6)
<i>FMK08</i> – 2	$\overline{TEC}$ (49.6,76.8,106.4)	(78.4,100.8,132.8)	(98.0,126.0,166.0)	(117.6,151.2,199.2)	(137.2,176.4,232.4)	$\overline{TEC}$ (53.4,72.0,91.2)	(16458.2,24181.2,32436.8)	(8835.6,12896.2,17264.6)	(6924.6,10097.8,13508.6)	(5608.4,8286.6,11048.8)
<i>FMK09</i> – 2	$\tilde{C}_{\max}$ (138.6,200.4,259.2)	(184.8,268.8,342.4)	(231.0,336.0,428.0)	(277.2,403.2,513.6)	(323.4,470.4,599.2)	$\tilde{C}_{\max}$ (169.0,236.2,298.4)	(41468.0,59851.8,77271.6)	(24202.6,34810.2,45125.4)	(20098.0,29005.8,37503.6)	(20595.6,28964.8,37449.0)
<i>FMK10</i> – 2	$\overline{TEC}$ (121.8,178.8,232.2)	(162.4,238.4,309.6)	(203.0,298.0,387.0)	(243.6,357.6,464.4)	(284.2,417.2,541.8)	$\overline{TEC}$ (127.8,191.6,250.4)	(46037.4,63793.8,82486.4)	(24635.0,34439.0,44665.0)	(20448.6,28588.2,37070.0)	(19243.6,27253.8,35298.6)
	$\tilde{C}_{\max}$ (103.8,136.2,179.4)	(132.0,184.8,241.6)	(148.0,235.0,309.0)	(207.6,272.4,358.8)	(239.4,326.2,422.8)	$\tilde{C}_{\max}$ (116.9,165.4,217.8)	(47122.4,65170.6,83675.4)	(24708.0,34811.0,44989.0)	(20365.8,28079.4,36197.8)	(22609.0,31638.8,41511.8)
	$\overline{TEC}$ (47122.4,65170.6,83675.4)	(34931.6,48849.4,63004.4)	(24708.0,34811.0,44989.0)	(20365.8,28079.4,36197.8)	(16870.0,23529.8,30438.8)	$\overline{TEC}$ (16870.0,23529.8,30438.8)				

Table 10
Statistical results between MACMNG and MWKR+EET on both $\tilde{C}_{\max}$ and $\overline{TEC}$ for FMK instances.

Instances	MWKR+EET					MACMNG				
	V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	Best $\tilde{C}_{\max}$
<i>FMK01</i> – 2	$\tilde{C}_{\max}$ (13.2,22.8,31.2)	(17.6,30.4,41.6)	(22.0,38.0,52.0)	(26.4,45.6,62.4)	(30.8,53.2,72.8)	$\tilde{C}_{\max}$ (10.6,21.8,29.4)	(3277.6,5219.8,7185.4)	(1763.4,2803.8,3851.6)	(1458.0,2318.8,3190.4)	(1605.3,2590.8,3525.2)
<i>FMK02</i> – 2	$\overline{TEC}$ (12.0,19.2,27.0)	(15.2,23.2,34.4)	(20.0,32.0,43.0)	(24.0,38.4,54.0)	(28.0,42.0,61.6)	$\overline{TEC}$ (10.8,19.6,29.4)	(3393.8,5468.4,7604.4)	(1926.4,3048.8,4215.2)	(1490.4,2420.4,3396.4)	(1319.6,2247.8,3217.8)
<i>FMK03</i> – 2	$\tilde{C}_{\max}$ (3393.8,5468.4,7604.4)	(92.0,140.8,176.0)	(124.0,176.0,228.0)	(148.8,211.2,273.6)	(173.6,246.4,319.2)	$\tilde{C}_{\max}$ (62.6,96.4,126.8)	(26281.6,37944.8,49640.8)	(13870.6,20144.0,26346.2)	(11017.6,15899.0,20792.8)	(7782.6,11705.0,15187.0)
<i>FMK04</i> – 2	$\overline{TEC}$ (24.2,45.8,59.2)	(31.8,60.6,78.6)	(42.0,76.0,98.0)	(50.0,90.8,118.0)	(63.0,105.0,137.2)	$\overline{TEC}$ (20.8,39.8,54.0)	(7367.4,11879.6,15886.0)	(4670.2,7546.8,10077.4)	(3360.2,5447.2,7265.8)	(2528.8,4266.0,5837.6)
<i>FMK05</i> – 2	$\tilde{C}_{\max}$ (45.0,69.0,88.8)	(58.2,88.4,114.6)	(78.0,118.0,153.0)	(90.0,138.0,177.6)	(109.2,165.2,214.2)	$\tilde{C}_{\max}$ (49.0,74.0,98.0)	(12355.6,18927.2,25261.2)	(6721.0,10325.4,13693.2)	(5451.0,8370.4,11222.0)	(6303.6,9366.8,12522.0)
<i>FMK06</i> – 2	$\overline{TEC}$ (12355.6,18927.2,25261.2)	(44.2,74.8,104.6)	(59.0,95.0,124.0)	(74.4,111.6,150.0)	(86.8,130.2,175.0)	$\overline{TEC}$ (20.8,39.8,54.0)	(34.6,56.0,77.2)	(44.2,74.8,104.6)	(4127.2,6302.8,8377.6)	(2528.8,4266.0,5837.6)
<i>FMK07</i> – 2	$\tilde{C}_{\max}$ (12217.4,19051.0,25892.2)	(92.4,105.8,138.4)	(103.0,140.0,182.0)	(123.6,168.0,218.4)	(144.2,196.0,254.8)	$\tilde{C}_{\max}$ (37.2,60.2,80.0)	(54.8,85.4,113.4)	(7008.2,10820.6,14701.2)	(4594.8,7212.8,9797.2)	(49.0,74.0,98.0)
<i>FMK08</i> – 2	$\overline{TEC}$ (54.8,85.4,113.4)	(82.4,112.0,145.6)	(103.0,140.0,182.0)	(123.6,168.0,218.4)	(144.2,196.0,254.8)	$\overline{TEC}$ (53.4,72.0,91.2)	(156.6,218.4,286.8)	(208.8,291.2,382.4)	(5836.6,8394.4,10924.2)	(6303.6,9366.8,12522.0)
<i>FMK09</i> – 2	$\tilde{C}_{\max}$ (41984.0,60331.8,78498.2)	(135.2,189.4,251.4)	(176.8,255.0,334.0)	(233.8,318.0,406.2)	(280.4,381.4,487.2)	$\tilde{C}_{\max}$ (127.8,191.6,250.4)	(16792.2,24210.4,31659.2)	(9871.4,14216.4,18530.0)	(5836.6,8394.4,10924.2)	(37.2,60.2,80.0)
<i>FMK10</i> – 2	$\overline{TEC}$ (51472.0,72196.0,93674.2)	(131.2,187.2,240.8)	(179.0,245.0,323.0)	(230.8,291.0,377.4)	(250.6,343.0,452.8)	$\overline{TEC}$ (116.9,165.4,217.8)	(109.0,147.2,191.8)	(27054.0,37764.8,48899.8)	(18338.6,25687.2,33283.6)	(6303.6,9366.8,12522.0)
	$\tilde{C}_{\max}$ (49100.8,68949.4,89751.6)	(38219.8,53777.0,69741.4)	(28280.4,39670.0,51667.4)	(22569.6,31531.4,41126.4)	(17011.4,23896.6,31178.0)	$\tilde{C}_{\max}$ (22609.0,31638.8,41511.8)				

Table 11

Instances		MOPR + EET			MACMNG			
		V_1	V_2	V_3	V_4	V_5	Best $\tilde{C}_{\max}$	
FMK01	− 2	$\tilde{C}_{\max}$	(16.8, 25.8, 34.2)	(22.4, 34.4, 45.6)	(28.0, 43.0, 57.0)	(33.6, 51.6, 68.4)	(39.2, 60.2, 79.8)	(10.6, 21.8, 29.4)
	$\overline{TEC}$	(3119.4, 4922.2, 6692.6)	(2397.8, 3837.6, 5233.4)	(1740.0, 2788.8, 3827.2)	(1420.2, 2289.2, 3133.8)	(1222.2, 1961.4, 2695.0)	(1605.3, 2590.8, 3525.2)	
FMK02	− 2	$\tilde{C}_{\max}$	(13.8, 18.6, 25.2)	(15.6, 25.2, 35.6)	(21.0, 31.0, 43.0)	(23.8, 37.8, 53.0)	(28.0, 44.8, 65.8)	(10.8, 19.6, 29.4)
	$\overline{TEC}$	(3219.2, 5277.4, 7494.8)	(2770.0, 4469.4, 6338.8)	(1913.4, 3047.0, 4320.6)	(1369.0, 2234.4, 3180.6)	(1236.2, 1993.6, 2839.2)	(1319.6, 2247.8, 3217.8)	
FMK03	− 2	$\tilde{C}_{\max}$	(68.0, 105.0, 134.2)	(104.0, 135.2, 175.2)	(130.0, 169.0, 219.0)	(156.0, 202.8, 262.8)	(182.0, 236.6, 306.6)	(62.6, 96.4, 126.8)
	$\overline{TEC}$	(26751.2, 37565.8, 48833.0)	(20372.0, 28534.4, 37021.0)	(14274.4, 20267.2, 26350.0)	(11552.4, 16472.8, 21364.6)	(9886.8, 14061.6, 18183.2)	(7782.6, 11705.0, 15187.0)	
FMK04	− 2	$\tilde{C}_{\max}$	(26.4, 39.6, 54.0)	(31.6, 51.0, 67.2)	(44.0, 66.0, 90.0)	(52.8, 79.2, 108.0)	(61.6, 92.4, 126.0)	(20.8, 39.8, 54.0)
	$\overline{TEC}$	(7662.4, 11639.6, 15626.2)	(6008.6, 9537.0, 12703.6)	(4545.2, 6933.4, 9340.0)	(3639.2, 5595.2, 7508.4)	(2850.4, 4372.2, 5837.0)	(2528.8, 4266.0, 5837.6)	
FMK05	− 2	$\tilde{C}_{\max}$	(39.6, 66.0, 84.6)	(52.8, 88.0, 112.8)	(66.0, 110.0, 141.0)	(79.2, 132.0, 169.2)	(92.4, 154.0, 197.4)	(49.0, 74.4, 98.0)
	$\overline{TEC}$	(13123.8, 19884.4, 26294.6)	(10641.0, 16073.4, 21229.4)	(7885.4, 11971.6, 15800.4)	(6238.6, 9508.4, 12526.4)	(4069.8, 6235.6, 8246.0)	(6303.6, 9366.8, 12522.0)	
FMK06	− 2	$\tilde{C}_{\max}$	(33.8, 54.6, 77.8)	(52.0, 72.8, 96.8)	(57.0, 94.0, 123.0)	(78.0, 109.2, 145.2)	(91.0, 127.4, 169.4)	(37.2, 60.2, 80.0)
	$\overline{TEC}$	(14317.4, 21420.4, 29057.8)	(11715.0, 17527.2, 23805.6)	(8651.0, 12901.4, 17517.6)	(6743.4, 10022.6, 13579.0)	(5131.0, 7704.2, 10470.6)	(4662.2, 7420.8, 9933.6)	
FMK07	− 2	$\tilde{C}_{\max}$	(49.6, 76.8, 106.4)	(78.4, 100.8, 132.8)	(98.0, 126.0, 166.0)	(117.6, 151.2, 199.2)	(137.2, 176.4, 232.4)	(53.4, 72.0, 91.2)
	$\overline{TEC}$	(16458.2, 24181.2, 32436.8)	(13001.2, 18988.0, 25476.2)	(8835.6, 12896.2, 17264.6)	(6924.6, 10097.8, 13508.6)	(5608.4, 8286.6, 11048.8)	(6593.8, 9805.4, 12856.0)	
FMK08	− 2	$\tilde{C}_{\max}$	(143.0, 210.6, 277.0)	(198.8, 282.2, 354.2)	(257.0, 350.0, 444.0)	(289.2, 421.6, 549.8)	(350.0, 488.6, 635.6)	(169.0, 236.2, 298.4)
	$\overline{TEC}$	(50206.0, 71348.8, 92426.6)	(39829.6, 56614.2, 73272.8)	(31362.2, 44594.4, 57646.4)	(21675.4, 30910.6, 39824.0)	(18032.0, 25650.8, 33208.0)	(20595.6, 28964.8, 37449.0)	
FMK09	− 2	$\tilde{C}_{\max}$	(137.4, 192.0, 246.6)	(183.2, 256.0, 328.8)	(229.0, 320.0, 411.0)	(274.8, 384.0, 493.2)	(320.6, 448.0, 575.4)	(127.8, 191.6, 250.4)
	$\overline{TEC}$	(50516.8, 70304.0, 91227.6)	(39399.2, 54708.0, 70999.0)	(28459.0, 39442.2, 51060.6)	(22794.6, 31515.2, 40789.6)	(18701.2, 25593.0, 33628.0)	(19243.6, 27253.8, 35298.6)	
FMK10	− 2	$\tilde{C}_{\max}$	(87.0, 119.4, 153.0)	(116.0, 159.2, 204.0)	(154.2, 199.6, 244.6)	(174.0, 238.8, 306.0)	(203.0, 278.6, 357.0)	(116.9, 165.4, 217.8)
	$\overline{TEC}$	(46564.8, 66712.8, 86127.6)	(36617.6, 52487.0, 67630.0)	(26779.0, 38377.8, 49481.2)	(21007.2, 30025.2, 38701.0)	(16385.6, 23434.6, 30179.8)	(22609.0, 31638.8, 41511.8)	

produced by MACMNG and PDRs can be calculated by Eqs. (13)–(15) and optimized via the energy-efficient strategy (EES) detailed in Section 4.4. To ensure fairness, all speed levels were fixed uniformly across all composite PDRs due to their inability to adapt to variable speeds. The performance of MACMNG and PDRs was evaluated at five different speed levels (V_1 to V_5), with comparisons centered on their best achievable $\widetilde{C}_{\max}$ and $\widetilde{TEC}$.

As summarized in Tables 9–11, a comparative analysis on the effectiveness of MACMNG against the three PDRs at different speed levels is provided, with the best results achieved on each instance highlighted in **bold**. When speed levels are set to V_1 , composite PDRs can achieve comparable $\tilde{C}_{\max}$ optimization to MACMNG but demonstrate significantly inferior for optimizing $\widetilde{TEC}$. For example, in instance *FMK02-2*, FIFO+EET yields $\widetilde{TEC}$ value of (3507.0, 5918.0, 8189.4), which are greatly higher than MACMNG’s results (1319.6, 2247.8, 3217.8). When speed levels are decreased (e.g., to V_5), these three composite PDRs achieve marginal improvements in $\widetilde{TEC}$ for specific instances, but at the expense of drastically increased $\tilde{C}_{\max}$. Upon reducing speed level to V_5 , FIFO+EET reduces $\widetilde{TEC}$ in *FMK01-2* but results in a marked increase for $\tilde{C}_{\max}$, which is considerably higher than $\tilde{C}_{\max}$ of MACMNG by a significant margin. In contrast, MACMNG maintains a balanced performance of both objectives across different speed levels, effectively coordinating $\tilde{C}_{\max}$ and $\widetilde{TEC}$ to deliver stable and robust performance in tackling EE-DFFJSP. The experimental findings show a critical limitation of composite PDRs: their tendency to prioritize one objective while neglecting the other objective when adjusting speed levels, resulting in significant deterioration of neglected objectives for practical multi-objective scheduling scenarios. MACMNG achieves cooperative optimization in solving EE-DFFJSP. The MACMNG strikes a better balance between $\tilde{C}_{\max}$ and $\widetilde{TEC}$, making it more suitable for complex scenarios, where dynamic speed adaptation and multi-objective trade-offs are essential.

5.4. Comparison of MACMNG with existing algorithms

To further validate the efficacy of MACMNG, this section conducts a comprehensive comparative study against seven state-of-the-art multi-objective optimization algorithms: MOEA/D [72], NSGA-II [73], NSGA-III [65], KCA [74], MMMA [42], KBEA [45], and TSKEA [47]. These algorithms, selected for their specific strengths in solving MOPs, serve as baselines for evaluating MACMNG’s performance. As a well-known solver for MOPs, MOEA/D has been recognized for its robust search capabilities, driven by a decomposition-based method and a neighborhood-based optimization strategy. NSGA-II achieves excellent search behavior through an efficient nondominated sorting method and an elite selection strategy, while NSGA-III further improves the handling of MOPs via a reference point-based mechanism to preserve population diversity. KCA is developed to tackle the energy-efficient DFFSP, which incorporates problem-specific knowledge extraction, multi-operator cooperative exploration, and knowledge-based local intensification to balance both two objectives $C_{\max}$ and TEC . MMMA is devised for the energy-efficient DFFSP with variable machine speeds. MMMA employs a weighted NEH-based initialization inspired by MOEA/D and genetic global search operators to minimize both total weighted tardiness and TEC . KBEA is designed to solve the energy-efficient DFJSP with the objectives of minimizing $C_{\max}$ and TEC , which utilizes knowledge-guided local search and energy-efficient strategies (EES) to strengthen exploitation capability. TSKEA is devised for DGT2FJSP and adopts a two-stage framework combining problem-specific heuristics for population initialization, a Pareto-based evolution method, and multi-neighborhood search to balance convergence and diversity.

To fairly evaluate algorithms' efficacy, the Wilcoxon Signed-Rank Test is used to perform pairwise comparisons across all instances. The statistical results of MACMNG versus the seven baselines multi-objective

Table 12Statistical results of MACMNG versus other comparison algorithms on *HV* metric for two-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei01</i> – 2	0.087531(+)	0.494092(+)	0.227097(+)	0.032214(+)	0.074568(+)	0.385463(+)	0.173635(+)	0.690154
<i>Lei02</i> – 2	0.324517(+)	0.741414(+)	0.521762(+)	0.308014(+)	0.220847(+)	0.669506(+)	0.318517(+)	0.793409
<i>Lei03</i> – 2	0.320653(+)	0.576085(+)	0.502819(+)	0.133029(+)	0.206711(+)	0.565053(+)	0.300827(+)	0.824267
<i>Lei04</i> – 2	0.281270(+)	0.513076(+)	0.371261(+)	0.300834(+)	0.328726(+)	0.339419(+)	0.116377(+)	0.758543
<i>Lei05</i> – 2	0.030242(+)	0.279992(+)	0.052692(+)	0.005497(+)	0.000000(+)	0.372213(+)	0.091587(+)	0.554593
<i>FMK01</i> – 2	0.156262(+)	0.402439(+)	0.186110(+)	0.149997(+)	0.155291(+)	0.424252(+)	0.388192(+)	0.726711
<i>FMK02</i> – 2	0.536725(+)	0.461744(+)	0.355097(+)	0.461101(+)	0.401693(+)	0.463217(+)	0.367246(+)	0.842717
<i>FMK03</i> – 2	0.141642(+)	0.145938(+)	0.100767(+)	0.283566(+)	0.187103(+)	0.229547(+)	0.018089(+)	0.645952
<i>FMK04</i> – 2	0.000000(+)	0.258894(+)	0.012447(+)	0.012157(+)	0.000000(+)	0.047685(+)	0.012664(+)	0.638268
<i>FMK05</i> – 2	0.000000(+)	0.281305(+)	0.176642(+)	0.040635(+)	0.000000(+)	0.408618(+)	0.203358(+)	0.662474
<i>FMK06</i> – 2	0.000000(+)	0.149626(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.393899(+)	0.259619(+)	0.609442
<i>FMK07</i> – 2	0.022587(+)	0.288010(+)	0.186896(+)	0.049545(+)	0.005688(+)	0.308130(+)	0.199766(+)	0.661085
<i>FMK08</i> – 2	0.347167(+)	0.481231(+)	0.408340(+)	0.248606(+)	0.145555(+)	0.527104(+)	0.519127(+)	0.812578
<i>FMK09</i> – 2	0.051364(+)	0.127373(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.296051(+)	0.096846(+)	0.525400
<i>FMK10</i> – 2	0.198360(+)	0.363466(+)	0.228333(+)	0.189355(+)	0.228818(+)	0.504746(+)	0.215453(+)	0.757637
<i>Reman01</i> – 2	0.000000(+)	0.342234(+)	0.214625(+)	0.019404(+)	0.000000(+)	0.349722(+)	0.168641(+)	0.534985
<i>Reman02</i> – 2	0.111784(+)	0.397879(+)	0.167797(+)	0.004857(+)	0.002053(+)	0.491083(+)	0.152674(+)	0.569030
<i>Reman03</i> – 2	0.043863(+)	0.287209(+)	0.095877(+)	0.004460(+)	0.013796(+)	0.292175(+)	0.289605(+)	0.670915
<i>Reman04</i> – 2	0.398594(+)	0.591861(+)	0.443119(+)	0.252853(+)	0.302045(+)	0.666416(+)	0.548974(+)	0.861894
<i>Reman05</i> – 2	0.269665(+)	0.347437(+)	0.266390(+)	0.033443(+)	0.000000(+)	0.597041(+)	0.399246(+)	0.718405
<i>Reman06</i> – 2	0.333678(+)	0.560409(+)	0.444461(+)	0.219856(+)	0.290382(+)	0.693921(+)	0.452916(+)	0.758776
<i>Reman07</i> – 2	0.367455(+)	0.512598(+)	0.366114(+)	0.041216(+)	0.266424(+)	0.702518(+)	0.412017(+)	0.735358
<i>Reman08</i> – 2	0.297619(+)	0.549923(+)	0.430505(+)	0.000000(+)	0.000000(+)	0.748972(-)	0.197493(+)	0.641472
+ / $\approx$ / -	23/0/0	23/0/0	23/0/0	23/0/0	23/0/0	22/0/1	23/0/0	

Table 13Statistical results of MACMNG versus other comparison algorithms on *HV* metric for three-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei01</i> – 3	0.140213(+)	0.445011(+)	0.214870(+)	0.208752(+)	0.155451(+)	0.342550(+)	0.147323(+)	0.703619
<i>Lei02</i> – 3	0.128179(+)	0.390151(+)	0.208021(+)	0.310107(+)	0.159999(+)	0.482138(+)	0.225019(+)	0.724377
<i>Lei03</i> – 3	0.019124(+)	0.417853(+)	0.298453(+)	0.064168(+)	0.133425(+)	0.368243(+)	0.396846(+)	0.627037
<i>Lei04</i> – 3	0.203271(+)	0.395737(+)	0.338524(+)	0.047453(+)	0.128111(+)	0.285065(+)	0.379935(+)	0.611446
<i>Lei05</i> – 3	0.056826(+)	0.323051(+)	0.087202(+)	0.072144(+)	0.000000(+)	0.291098(+)	0.123226(+)	0.694067
<i>FMK01</i> – 3	0.066447(+)	0.324986(+)	0.320336(+)	0.200127(+)	0.284140(+)	0.385879(+)	0.309410(+)	0.656338
<i>FMK02</i> – 3	0.038507(+)	0.452840(+)	0.137527(+)	0.152822(+)	0.009851(+)	0.371761(+)	0.257815(+)	0.695251
<i>FMK03</i> – 3	0.000000(+)	0.219129(+)	0.040590(+)	0.000000(+)	0.000000(+)	0.317194(+)	0.002611(+)	0.729323
<i>FMK04</i> – 3	0.000000(+)	0.235967(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.128668(+)	0.159073(+)	0.668069
<i>FMK05</i> – 3	0.000000(+)	0.263831(+)	0.086369(+)	0.073722(+)	0.000000(+)	0.311943(+)	0.049430(+)	0.523183
<i>FMK06</i> – 3	0.322734(+)	0.420386(+)	0.344467(+)	0.000000(+)	0.046956(+)	0.320985(+)	0.410652(+)	0.703107
<i>FMK07</i> – 3	0.000000(+)	0.271115(+)	0.353950(+)	0.033173(+)	0.000000(+)	0.374574(+)	0.037556(+)	0.712502
<i>FMK08</i> – 3	0.034436(+)	0.193935(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.401938(+)	0.210793(+)	0.703601
<i>FMK09</i> – 3	0.151457(+)	0.511366(+)	0.385489(+)	0.164802(+)	0.152155(+)	0.555250(+)	0.434625(+)	0.700726
<i>FMK10</i> – 3	0.204032(+)	0.317740(+)	0.258919(+)	0.112299(+)	0.061974(+)	0.531354(+)	0.383294(+)	0.765737
<i>Reman01</i> – 3	0.044415(+)	0.375968(+)	0.272282(+)	0.007610(+)	0.000000(+)	0.166244(+)	0.233415(+)	0.544736
<i>Reman02</i> – 3	0.182185(+)	0.526979(+)	0.336493(+)	0.064177(+)	0.050750(+)	0.488679(+)	0.142682(+)	0.750562
<i>Reman03</i> – 3	0.173956(+)	0.379461(+)	0.179022(+)	0.003990(+)	0.056357(+)	0.293678(+)	0.195842(+)	0.672932
<i>Reman04</i> – 3	0.071811(+)	0.372966(+)	0.163252(+)	0.000098(+)	0.011457(+)	0.212705(+)	0.094239(+)	0.729142
<i>Reman05</i> – 3	0.271574(+)	0.477071(+)	0.390259(+)	0.049277(+)	0.035320(+)	0.665350(+)	0.276801(+)	0.871616
<i>Reman06</i> – 3	0.132063(+)	0.389655(+)	0.256215(+)	0.000000(+)	0.000000(+)	0.513640(+)	0.118687(+)	0.852399
<i>Reman07</i> – 3	0.248437(+)	0.433718(+)	0.308485(+)	0.000000(+)	0.000000(+)	0.639640(+)	0.219063(+)	0.731447
<i>Reman08</i> – 3	0.474395(+)	0.530443(+)	0.426874(+)	0.046042(+)	0.117430(+)	0.474957(+)	0.643091(+)	0.647709
+ / $\approx$ / -	23/0/0	23/0/0	23/0/0	23/0/0	23/0/0	23/0/0	23/0/0	

algorithms on *HV*, *GD*, and *IGD* metrics are summarized in [Tables 12–20](#), grouped by the number of factories. The symbols ‘+’, ‘ $\approx$ ’, and ‘-’ indicate whether the baseline algorithms (i.e., MOEA/D, NSGA-II, NSGA-III, KCA, MMMA, TSKEA, and KBEA) are inferior to, similar to, or superior to the MACMNG, respectively. Furthermore, the Friedman Test is utilized to provide an overall performance ranking of MACMNG against other algorithms across all instances to determine the statistical significance of MACMNG’s superiority.

As illustrated in [Tables 12–20](#) and [Fig. 14](#), MACMNG has demonstrated statistically significant superiority over state-of-the-art multi-objective optimization algorithms across various datasets and scenarios with different numbers of factories, consistently delivering superior results even in medium- and large-scale cases. In terms of *HV* and *IGD* metrics, MACMNG achieves the best values in all instances, indicating superior solution quality, diversity, and coverage of the Pareto front.

Although MACMNG is slightly less competitive in a few cases in terms of *GD* metrics, its overall performance is still deemed to be the best among all metrics. Specifically, MACMNG yields the highest *HV* values, indicating superior convergence and diversity, simultaneously achieving the lowest *GD* and *IGD* values, reflecting the fact that the solutions found by MACMNG are closer to the true Pareto front (PF^*). Furthermore, MACMNG also exhibits remarkable stability, which is a highly desirable property for its practical multi-objective scheduling applications. As shown in [Table 21](#) the Wilcoxon Signed-Rank Test results further validate MACMNG’s superiority over its baseline counterparts, with p -values < 0.05 indicating statistically significant differences. Here, the notation R^+ denotes the sum of positive ranks, which corresponds to instances that MACMNG outperforms its competitors; while R^- represents the sum of negative ranks, reflecting instances that all baseline algorithms outperform MACMNG. The consistently higher R^+ and lower

Table 14

Statistical results of MACMNG versus other comparison algorithms on *HV* metric for four-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei01</i> – 4	0.024379(+)	0.494640(+)	0.241569(+)	0.111834(+)	0.024881(+)	0.416402(+)	0.453951(+)	0.684065
<i>Lei02</i> – 4	0.113313(+)	0.611337(-)	0.212003(+)	0.061314(+)	0.000000(+)	0.464292(+)	0.144726(+)	0.476064
<i>Lei03</i> – 4	0.227950(+)	0.578064(-)	0.431874(+)	0.000000(+)	0.090249(+)	0.209650(+)	0.273726(+)	0.453617
<i>Lei04</i> – 4	0.276406(+)	0.669603(-)	0.470494(+)	0.000000(+)	0.000000(+)	0.500128(-)	0.313963(+)	0.482131
<i>Lei05</i> – 4	0.000000(+)	0.302039(+)	0.233041(+)	0.000000(+)	0.000000(+)	0.211939(+)	0.142523(+)	0.484159
<i>FMK01</i> – 4	0.000000(+)	0.343751(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.370402(+)	0.297324(+)	0.765854
<i>FMK02</i> – 4	0.190992(+)	0.463874(+)	0.451150(+)	0.192845(+)	0.318614(+)	0.305756(+)	0.481784(+)	0.683297
<i>FMK03</i> – 4	0.000000(+)	0.199589(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.377679(+)	0.212088(+)	0.705123
<i>FMK04</i> – 4	0.000000(+)	0.330738(+)	0.135648(+)	0.000000(+)	0.000000(+)	0.414983(+)	0.203957(+)	0.725346
<i>FMK05</i> – 4	0.000000(+)	0.190716(+)	0.048415(+)	0.047531(+)	0.000000(+)	0.331191(+)	0.263917(+)	0.601520
<i>FMK06</i> – 4	0.021675(+)	0.215988(+)	0.066125(+)	0.000000(+)	0.000000(+)	0.363338(+)	0.031554(+)	0.600555
<i>FMK07</i> – 4	0.037610(+)	0.261092(+)	0.035879(+)	0.000000(+)	0.000000(+)	0.296379(+)	0.168628(+)	0.658522
<i>FMK08</i> – 4	0.000000(+)	0.064155(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.142196(+)	0.302545(+)	0.684858
<i>FMK09</i> – 4	0.000000(+)	0.267329(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.346433(+)	0.091289(+)	0.575558
<i>FMK10</i> – 4	0.000000(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.000000(+)	0.245899(+)	0.136216(+)	0.508342
<i>Reman01</i> – 4	0.041059(+)	0.482212(+)	0.336083(+)	0.087747(+)	0.144142(+)	0.464270(+)	0.250964(+)	0.663708
<i>Reman02</i> – 4	0.000000(+)	0.448362(+)	0.179424(+)	0.021424(+)	0.000000(+)	0.436868(+)	0.287350(+)	0.634532
<i>Reman03</i> – 4	0.000000(+)	0.238522(+)	0.040507(+)	0.000000(+)	0.000000(+)	0.328602(+)	0.293428(+)	0.692838
<i>Reman04</i> – 4	0.131898(+)	0.449824(+)	0.151383(+)	0.000000(+)	0.000000(+)	0.160359(+)	0.203153(+)	0.638720
<i>Reman05</i> – 4	0.249610(+)	0.397067(+)	0.238978(+)	0.185413(+)	0.232936(+)	0.463852(+)	0.435041(+)	0.811389
<i>Reman06</i> – 4	0.074977(+)	0.402865(+)	0.229811(+)	0.000000(+)	0.114053(+)	0.448748(+)	0.344117(+)	0.724775
<i>Reman07</i> – 4	0.000000(+)	0.101888(+)	0.000000(+)	0.000000(+)	0.002480(+)	0.240265(+)	0.157030(+)	0.449302
<i>Reman08</i> – 4	0.193243(+)	0.301290(+)	0.218624(+)	0.011574(+)	0.000000(+)	0.392357(+)	0.291082(+)	0.449102
+ / $\approx$ / –	23/0/0	20/0/3	23/0/0	23/0/0	23/0/0	22/0/1	23/0/0	

Table 15

Statistical results of MACMNG versus other comparison algorithms on *GD* metric for two-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei01</i> – 2	0.613850(+)	0.332633(+)	0.363106(+)	0.337757(+)	0.402369(+)	0.463351(+)	0.202689(+)	0.000000
<i>Lei02</i> – 2	0.450926(+)	0.150930(+)	0.244405(+)	0.150265(+)	0.333575(+)	0.262744(+)	0.146900(+)	0.000000
<i>Lei03</i> – 2	0.381711(+)	0.109262(+)	0.269688(+)	0.235516(+)	0.387688(+)	0.226392(+)	0.137948(+)	0.000000
<i>Lei04</i> – 2	0.393444(+)	0.072885(+)	0.104940(+)	0.160861(+)	0.173878(+)	0.274827(+)	0.185897(+)	0.006771
<i>Lei05</i> – 2	0.382673(+)	0.133679(+)	0.293938(+)	0.616022(+)	0.521176(+)	0.125044(+)	0.272483(+)	0.000000
<i>FMK01</i> – 2	0.355438(+)	0.227441(+)	0.304265(+)	0.385562(+)	0.435524(+)	0.177410(+)	0.174347(+)	0.000000
<i>FMK02</i> – 2	0.157583(+)	0.120509(+)	0.161600(+)	0.181520(+)	0.247094(+)	0.425619(+)	0.092873(+)	0.000000
<i>FMK03</i> – 2	0.324209(+)	0.193448(+)	0.196468(+)	0.349734(+)	0.298153(+)	0.709920(+)	0.425078(+)	0.000000
<i>FMK04</i> – 2	0.928590(+)	0.266978(+)	0.505554(+)	0.524774(+)	0.840276(+)	0.887235(+)	0.335724(+)	0.000000
<i>FMK05</i> – 2	0.575921(+)	0.152370(+)	0.169961(+)	0.386193(+)	0.501280(+)	0.300983(+)	0.164486(+)	0.000000
<i>FMK06</i> – 2	0.431380(+)	0.115795(+)	0.243604(+)	0.810566(+)	0.661986(+)	0.436197(+)	0.046406(+)	0.023633
<i>FMK07</i> – 2	0.490244(+)	0.102246(+)	0.161341(+)	0.530518(+)	0.474064(+)	0.502575(+)	0.165729(+)	0.000000
<i>FMK08</i> – 2	0.273012(+)	0.121073(+)	0.208557(+)	0.431033(+)	0.359156(+)	0.148996(+)	0.133382(+)	0.000000
<i>FMK09</i> – 2	0.569490(+)	0.204889(+)	0.346404(+)	0.527295(+)	0.748245(+)	0.378660(+)	0.197409(+)	0.000000
<i>FMK10</i> – 2	0.183060(+)	0.041106(+)	0.051792(+)	0.363090(+)	0.267812(+)	0.133114(+)	0.057005(+)	0.000000
<i>Reman01</i> – 2	0.499696(+)	0.290682(+)	0.394322(+)	0.541965(+)	0.757197(+)	0.154751(+)	0.235884(+)	0.000000
<i>Reman02</i> – 2	0.344084(+)	0.120488(-)	0.170174(-)	0.595985(+)	0.739059(+)	0.017760(-)	0.193262(-)	0.275595
<i>Reman03</i> – 2	0.440491(+)	0.181211(+)	0.290267(+)	0.722202(+)	0.719972(+)	0.225720(+)	0.136614(+)	0.000000
<i>Reman04</i> – 2	0.158831(+)	0.070026(+)	0.101051(+)	0.436251(+)	0.371501(+)	0.076036(+)	0.051864(+)	0.000000
<i>Reman05</i> – 2	0.160100(+)	0.030277(+)	0.035960(+)	0.660341(+)	0.379874(+)	0.071948(+)	0.075945(+)	0.009202
<i>Reman06</i> – 2	0.222162(+)	0.004103(-)	0.081290(+)	0.471133(+)	0.261364(+)	0.019109(+)	0.046403(+)	0.017816
<i>Reman07</i> – 2	0.091777(+)	0.000000(-)	0.058831(+)	0.652870(+)	0.236733(+)	0.069938(+)	0.078799(+)	0.017525
<i>Reman08</i> – 2	0.207935(+)	0.014514(-)	0.086049(-)	0.901182(+)	0.503897(+)	0.000000(-)	0.197606(+)	0.107712
+ / $\approx$ / –	23/0/0	19/0/4	21/0/2	23/0/0	23/0/0	21/0/2	22/0/1	

Table 16
Statistical results of MACMNG versus other comparison algorithms on *GD* metric for three-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei</i> 01 – 3	0.505169(+)	0.204777(+)	0.446991(+)	0.157390(+)	0.268286(+)	0.431469(+)	0.168857(+)	0.000000
<i>Lei</i> 02 – 3	0.516363(+)	0.268382(+)	0.434028(+)	0.206147(+)	0.331547(+)	0.341359(+)	0.136541(+)	0.000000
<i>Lei</i> 03 – 3	0.549768(+)	0.157783(+)	0.171643(+)	0.509101(+)	0.293602(+)	0.239180(+)	0.116873(+)	0.000000
<i>Lei</i> 04 – 3	0.551210(+)	0.169588(+)	0.319355(+)	0.420931(+)	0.278568(+)	0.411597(+)	0.139270(+)	0.000000
<i>Lei</i> 05 – 3	0.567777(+)	0.170537(+)	0.462369(+)	0.554236(+)	0.581392(+)	0.329988(+)	0.228981(+)	0.000000
<i>FMK</i> 01 – 3	0.416677(+)	0.250175(+)	0.213964(+)	0.374016(+)	0.303217(+)	0.206339(+)	0.278865(+)	0.000000
<i>FMK</i> 02 – 3	0.390581(+)	0.114849(+)	0.252612(+)	0.488652(+)	0.507989(+)	0.294353(+)	0.036592(+)	0.000000
<i>FMK</i> 03 – 3	0.527116(+)	0.233080(+)	0.282011(+)	0.854081(+)	0.595528(+)	0.356481(+)	0.285226(+)	0.000000
<i>FMK</i> 04 – 3	0.899630(+)	0.320477(+)	0.475870(+)	0.751464(+)	0.872460(+)	0.704126(+)	0.353527(+)	0.000000
<i>FMK</i> 05 – 3	0.738602(+)	0.237908(+)	0.363560(+)	0.552408(+)	0.707072(+)	0.167607(+)	0.302768(+)	0.000000
<i>FMK</i> 06 – 3	0.241076(+)	0.195767(+)	0.232399(+)	0.944696(+)	0.660048(+)	0.234782(+)	0.094025(+)	0.000000
<i>FMK</i> 07 – 3	0.503021(+)	0.108410(+)	0.106970(+)	0.577444(+)	0.431849(+)	0.125554(+)	0.120828(+)	0.000000
<i>FMK</i> 08 – 3	0.463937(+)	0.197165(+)	0.146304(+)	0.194471(+)	0.454154(+)	0.110734(+)	0.158405(+)	0.000000
<i>FMK</i> 09 – 3	0.475040(+)	0.097347(+)	0.098737(+)	0.408176(+)	0.376741(+)	0.156626(+)	0.078818(+)	0.000000
<i>FMK</i> 10 – 3	0.295119(+)	0.134224(+)	0.137228(+)	0.560613(+)	0.284774(+)	0.080787(+)	0.064418(+)	0.001614
<i>Reman</i> 01 – 3	0.516788(+)	0.300650(+)	0.359669(+)	0.613982(+)	0.784746(+)	0.345143(+)	0.251699(+)	0.000000
<i>Reman</i> 02 – 3	0.333290(+)	0.125892(+)	0.210016(+)	0.567703(+)	0.640726(+)	0.159280(+)	0.235646(+)	0.000000
<i>Reman</i> 03 – 3	0.291913(+)	0.131665(+)	0.224284(+)	0.838287(+)	0.449486(+)	0.290936(+)	0.149891(+)	0.000000
<i>Reman</i> 04 – 3	0.349751(+)	0.081840(+)	0.148989(+)	0.758472(+)	0.425286(+)	0.129105(+)	0.176173(+)	0.000000
<i>Reman</i> 05 – 3	0.171383(+)	0.078694(+)	0.057648(+)	0.649102(+)	0.411243(+)	0.080300(+)	0.059355(+)	0.000000
<i>Reman</i> 06 – 3	0.210470(+)	0.131200(+)	0.106830(+)	0.648348(+)	0.347407(+)	0.185126(+)	0.129478(+)	0.000000
<i>Reman</i> 07 – 3	0.205761(+)	0.000000 (-)	0.085846(-)	0.888761(+)	0.503802(+)	0.034924(-)	0.141483(+)	0.111704
<i>Reman</i> 08 – 3	0.140445(+)	0.065598(-)	0.010440(-)	0.610210(+)	0.448566(+)	0.128733(+)	0.000000 (-)	0.123417
+ / $\approx$ / -	23/0/0	21/0/2	21/0/2	23/0/0	23/0/0	22/0/1	22/0/1	

Table 17
Statistical results of MACMNG versus other comparison algorithms on *GD* metric for four-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei</i> 01 – 4	0.655075(+)	0.184101(+)	0.373730(+)	0.509195(+)	0.550586(+)	0.083900(+)	0.088716(+)	0.006289
<i>Lei</i> 02 – 4	0.598395(+)	0.221528(+)	0.458295(+)	0.371279(+)	0.619276(+)	0.404383(+)	0.113365(+)	0.000000
<i>Lei</i> 03 – 4	0.404850(+)	0.032891(-)	0.070758(-)	0.793242(+)	0.562286(+)	0.221367(+)	0.128718(+)	0.077529
<i>Lei</i> 04 – 4	0.394553(+)	0.099426(-)	0.213767(+)	0.634859(+)	0.777317(+)	0.212361(+)	0.129090 (-)	0.162822
<i>Lei</i> 05 – 4	0.848949(+)	0.301769(+)	0.271241(+)	0.945336(+)	0.756343(+)	0.287206(+)	0.293823(+)	0.000000
<i>FMK</i> 01 – 4	0.790487(+)	0.385491(+)	0.912348(+)	0.997257(+)	0.986987(+)	0.266235(+)	0.281375(+)	0.000000
<i>FMK</i> 02 – 4	0.313196(+)	0.153092(+)	0.132222(+)	0.393257(+)	0.258585(+)	0.256775(+)	0.141908(+)	0.030968
<i>FMK</i> 03 – 4	0.528948(+)	0.257443(+)	0.374395(+)	0.836415(+)	0.564624(+)	0.239727(+)	0.122896(+)	0.000000
<i>FMK</i> 04 – 4	0.809777(+)	0.233657(+)	0.398984(+)	0.790873(+)	0.768587(+)	0.154994(+)	0.152733(+)	0.000000
<i>FMK</i> 05 – 4	0.803499(+)	0.292247(+)	0.446961(+)	0.568710(+)	0.738957(+)	0.145882(+)	0.130679(+)	0.000000
<i>FMK</i> 06 – 4	0.294315(+)	0.132022(+)	0.267815(+)	0.782237(+)	0.423058(+)	0.119127(+)	0.148809(+)	0.000000
<i>FMK</i> 07 – 4	0.508197(+)	0.200048(+)	0.282623(+)	0.744133(+)	0.597229(+)	0.225657(+)	0.315864(+)	0.000000
<i>FMK</i> 08 – 4	0.729496(+)	0.307651(+)	0.510976(+)	0.775487(+)	0.797300(+)	0.409290(+)	0.168907(+)	0.000000
<i>FMK</i> 09 – 4	0.559376(+)	0.106750(+)	0.301557(+)	0.543694(+)	0.747309(+)	0.141415(+)	0.233883(+)	0.000000
<i>FMK</i> 10 – 4	0.626772(+)	0.420758(+)	0.731689(+)	0.722771(+)	0.666480(+)	0.363364(+)	0.150566(+)	0.000000
<i>Reman</i> 01 – 4	0.439263(+)	0.295247(+)	0.345779(+)	0.471939(+)	0.421137(+)	0.153569(+)	0.330256(+)	0.000000
<i>Reman</i> 02 – 4	0.724874(+)	0.261485(+)	0.410612(+)	0.738434(+)	0.857047(+)	0.538682(+)	0.155753(+)	0.048854
<i>Reman</i> 03 – 4	0.570860(+)	0.307647(+)	0.350396(+)	0.954803(+)	0.848105(+)	0.562666(+)	0.253797(+)	0.000000
<i>Reman</i> 04 – 4	0.216101(+)	0.096768(+)	0.310036(+)	0.833621(+)	0.733811(+)	0.645044(+)	0.103983(+)	0.000000
<i>Reman</i> 05 – 4	0.367231(+)	0.206421(+)	0.124809(+)	0.505543(+)	0.430234(+)	0.243322(+)	0.162043(+)	0.000000
<i>Reman</i> 06 – 4	0.484433(+)	0.188109(+)	0.217116(+)	0.800385(+)	0.833843(+)	0.152051(+)	0.211874(+)	0.000000
<i>Reman</i> 07 – 4	0.159755(+)	0.019877(+)	0.129293(+)	0.754758(+)	0.624850(+)	0.186807(+)	0.176804(+)	0.000000
<i>Reman</i> 08 – 4	0.165927(-)	0.010756 (-)	0.071263(-)	0.774911(+)	0.691413(+)	0.309836(+)	0.024669(-)	0.227839
+ / $\approx$ / -	22/0/1	20/0/3	21/0/2	23/0/0	23/0/0	23/0/0	21/0/2	

Table 18Statistical results of MACMNG versus other comparison algorithms on *IGD* metric for two-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei</i> 01 – 2	0.533975(+)	0.179299(+)	0.364857(+)	0.540691(+)	0.531414(+)	0.289412(+)	0.402491(+)	0.000000
<i>Lei</i> 02 – 2	0.435304(+)	0.141379(+)	0.294426(+)	0.424837(+)	0.496850(+)	0.215639(+)	0.404909(+)	0.000000
<i>Lei</i> 03 – 2	0.379515(+)	0.244705(+)	0.236562(+)	0.626168(+)	0.508976(+)	0.249146(+)	0.474622(+)	0.000000
<i>Lei</i> 04 – 2	0.363158(+)	0.128124(+)	0.240189(+)	0.291322(+)	0.275058(+)	0.273129(+)	0.478203(+)	0.015208
<i>Lei</i> 05 – 2	0.576870(+)	0.302654(+)	0.561725(+)	0.680226(+)	0.725859(+)	0.151705(+)	0.504644(+)	0.032531
<i>FMK</i> 01 – 2	0.510982(+)	0.261604(+)	0.473117(+)	0.496995(+)	0.501304(+)	0.260573(+)	0.292865(+)	0.000000
<i>FMK</i> 02 – 2	0.206758(+)	0.239164(+)	0.321351(+)	0.265965(+)	0.312058(+)	0.272139(+)	0.307776(+)	0.000000
<i>FMK</i> 03 – 2	0.491246(+)	0.445961(+)	0.520989(+)	0.379888(+)	0.422150(+)	0.410566(+)	0.577888(+)	0.000000
<i>FMK</i> 04 – 2	0.901040(+)	0.396381(+)	0.639999(+)	0.652604(+)	0.928317(+)	0.560189(+)	0.571426(+)	0.000000
<i>FMK</i> 05 – 2	0.751025(+)	0.337995(+)	0.428852(+)	0.592429(+)	0.724326(+)	0.254604(+)	0.388409(+)	0.000000
<i>FMK</i> 06 – 2	0.706791(+)	0.430822(+)	0.604228(+)	0.920384(+)	0.785911(+)	0.193906(+)	0.310183(+)	0.011999
<i>FMK</i> 07 – 2	0.562905(+)	0.319739(+)	0.389645(+)	0.606350(+)	0.633040(+)	0.324557(+)	0.355930(+)	0.000000
<i>FMK</i> 08 – 2	0.390942(+)	0.327663(+)	0.359849(+)	0.470653(+)	0.574407(+)	0.197354(+)	0.295900(+)	0.065686
<i>FMK</i> 09 – 2	0.591202(+)	0.433755(+)	0.712877(+)	0.705885(+)	0.824455(+)	0.201127(+)	0.465398(+)	0.020297
<i>FMK</i> 10 – 2	0.426853(+)	0.346155(+)	0.429571(+)	0.455395(+)	0.414321(+)	0.213012(+)	0.440766(+)	0.025849
<i>Reman</i> 01 – 2	0.652764(+)	0.217887(+)	0.359705(+)	0.646939(+)	0.840811(+)	0.163922(+)	0.396124(+)	0.000000
<i>Reman</i> 02 – 2	0.470588(+)	0.243703(+)	0.398024(+)	0.670402(+)	0.762384(+)	0.077830(+)	0.432907(+)	0.043810
<i>Reman</i> 03 – 2	0.592085(+)	0.369341(+)	0.540669(+)	0.741295(+)	0.742740(+)	0.296338(+)	0.381216(+)	0.000000
<i>Reman</i> 04 – 2	0.369969(+)	0.259520(+)	0.377223(+)	0.486267(+)	0.432348(+)	0.159514(+)	0.300826(+)	0.020644
<i>Reman</i> 05 – 2	0.284434(+)	0.206400(+)	0.267774(+)	0.695450(+)	0.674944(+)	0.150227(+)	0.196465(+)	0.089671
<i>Reman</i> 06 – 2	0.304915(+)	0.192011(+)	0.253776(+)	0.441545(+)	0.376855(+)	0.106216(+)	0.251220(+)	0.100102
<i>Reman</i> 07 – 2	0.346893(+)	0.280390(+)	0.398229(+)	0.727886(+)	0.416112(+)	0.159376(+)	0.321221(+)	0.062476
<i>Reman</i> 08 – 2	0.334845(+)	0.224975(-)	0.257152(+)	0.984664(+)	0.763491(+)	0.231018(-)	0.373285(+)	0.233621
+ / $\approx$ / -	23/0/0	22/0/1	23/0/0	23/0/0	23/0/0	22/0/1	23/0/0	

Table 19Statistical results of MACMNG versus other comparison algorithms on *IGD* metric for three-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei</i> 01 – 3	0.482582(+)	0.206979(+)	0.406968(+)	0.447616(+)	0.473414(+)	0.274107(+)	0.492600(+)	0.000000
<i>Lei</i> 02 – 3	0.513029(+)	0.229169(+)	0.385133(+)	0.321642(+)	0.431864(+)	0.202736(+)	0.393755(+)	0.004366
<i>Lei</i> 03 – 3	0.688933(+)	0.195187(+)	0.339990(+)	0.599691(+)	0.516419(+)	0.209304(+)	0.276232(+)	0.005004
<i>Lei</i> 04 – 3	0.411945(+)	0.195822(+)	0.216808(+)	0.549981(+)	0.471892(+)	0.228350(+)	0.502402(+)	0.004546
<i>Lei</i> 05 – 3	0.622780(+)	0.344129(+)	0.549828(+)	0.621766(+)	0.844234(+)	0.350917(+)	0.520628(+)	0.000000
<i>FMK</i> 01 – 3	0.554497(+)	0.310048(+)	0.312898(+)	0.479266(+)	0.335116(+)	0.188264(+)	0.311101(+)	0.000000
<i>FMK</i> 02 – 3	0.586804(+)	0.265422(+)	0.503230(+)	0.469877(+)	0.632000(+)	0.310081(+)	0.442554(+)	0.000000
<i>FMK</i> 03 – 3	0.682478(+)	0.403153(+)	0.572144(+)	0.967067(+)	0.811696(+)	0.322390(+)	0.614746(+)	0.000000
<i>FMK</i> 04 – 3	1.022100(+)	0.336460(+)	0.627698(+)	0.863471(+)	0.980529(+)	0.442009(+)	0.388660(+)	0.000000
<i>FMK</i> 05 – 3	0.847375(+)	0.365872(+)	0.508530(+)	0.587490(+)	0.862707(+)	0.270073(+)	0.522142(+)	0.000000
<i>FMK</i> 06 – 3	0.336887(+)	0.238446(+)	0.294906(+)	0.992782(+)	0.682136(+)	0.260606(+)	0.303688(+)	0.033250
<i>FMK</i> 07 – 3	0.800904(+)	0.400154(+)	0.365899(+)	0.697767(+)	0.802345(+)	0.303405(+)	0.644463(+)	0.000000
<i>FMK</i> 08 – 3	0.647138(+)	0.498342(+)	0.752006(+)	0.926422(+)	0.872731(+)	0.247709(+)	0.489978(+)	0.000000
<i>FMK</i> 09 – 3	0.514976(+)	0.261172(+)	0.396632(+)	0.517613(+)	0.525074(+)	0.102532(-)	0.343795(+)	0.104668
<i>FMK</i> 10 – 3	0.460221(+)	0.391727(+)	0.448761(+)	0.579372(+)	0.625954(+)	0.233146(+)	0.409007(+)	0.028480
<i>Reman</i> 01 – 3	0.530204(+)	0.176082(+)	0.266694(+)	0.666391(+)	0.870955(+)	0.324303(+)	0.301980(+)	0.041957
<i>Reman</i> 02 – 3	0.473143(+)	0.197221(+)	0.318311(+)	0.660072(+)	0.630151(+)	0.178017(+)	0.533006(+)	0.000000
<i>Reman</i> 03 – 3	0.411963(+)	0.251766(+)	0.411299(+)	0.810197(+)	0.577929(+)	0.284152(+)	0.412575(+)	0.000000
<i>Reman</i> 04 – 3	0.568339(+)	0.346179(+)	0.518119(+)	0.783977(+)	0.695640(+)	0.451845(+)	0.583473(+)	0.000000
<i>Reman</i> 05 – 3	0.500650(+)	0.400052(+)	0.453205(+)	0.774981(+)	0.715117(+)	0.203727(+)	0.528717(+)	0.087621
<i>Reman</i> 06 – 3	0.572555(+)	0.362888(+)	0.475255(+)	0.901472(+)	0.923616(+)	0.216313(+)	0.591049(+)	0.049651
<i>Reman</i> 07 – 3	0.390590(+)	0.290268(+)	0.385484(+)	0.945961(+)	0.705238(+)	0.133432(+)	0.439272(+)	0.106718
<i>Reman</i> 08 – 3	0.212311(+)	0.212790(+)	0.308662(+)	0.691101(+)	0.531461(+)	0.204417(+)	0.242330(+)	0.194168
+ / $\approx$ / -	23/0/0	23/0/0	23/0/0	23/0/0	23/0/0	22/0/1	23/0/0	

Table 20Statistical results of MACMNG versus other comparison algorithms on *IGD* metric for four-factory instances.

Instances	MOEA/D	NSGA-II	NSGA-III	KCA	MMMA	KBEA	TSKEA	MACMNG
<i>Lei01</i> – 4	0.693209(+)	0.111830(+)	0.361179(+)	0.548396(+)	0.677917(+)	0.219424(+)	0.194846(+)	0.030937
<i>Lei02</i> – 4	0.530888(+)	0.048324(-)	0.386745(+)	0.562156(+)	0.779715(+)	0.189772(+)	0.470465(+)	0.176534
<i>Lei03</i> – 4	0.340010(+)	0.036469(-)	0.201283(+)	0.878328(+)	0.518021(+)	0.286533(+)	0.335934(+)	0.158675
<i>Lei04</i> – 4	0.412838(+)	0.049650(-)	0.249827(-)	0.924517(+)	0.929971(+)	0.158633(-)	0.449324(+)	0.276423
<i>Lei05</i> – 4	0.926234(+)	0.201426(+)	0.271103(+)	0.960208(+)	0.848535(+)	0.288106(+)	0.358962(+)	0.000000
<i>FMK01</i> – 4	0.955864(+)	0.324135(+)	0.781751(+)	0.959721(+)	0.987733(+)	0.241574(+)	0.378007(+)	0.000000
<i>FMK02</i> – 4	0.397661(+)	0.170081(+)	0.177066(+)	0.546515(+)	0.322165(+)	0.221117(+)	0.216176(+)	0.012619
<i>FMK03</i> – 4	0.869015(+)	0.439206(+)	0.733141(+)	0.822819(+)	0.834450(+)	0.336659(+)	0.478480(+)	0.000000
<i>FMK04</i> – 4	0.814340(+)	0.299367(+)	0.478360(+)	0.689612(+)	0.933998(+)	0.184791(+)	0.406110(+)	0.000000
<i>FMK05</i> – 4	0.845196(+)	0.375121(+)	0.532446(+)	0.637898(+)	0.784390(+)	0.267007(+)	0.381562(+)	0.000000
<i>FMK06</i> – 4	0.606287(+)	0.414539(+)	0.602154(+)	0.793028(+)	0.798470(+)	0.190303(+)	0.683878(+)	0.000000
<i>FMK07</i> – 4	0.666763(+)	0.423304(+)	0.695297(+)	0.744536(+)	0.901574(+)	0.360597(+)	0.518743(+)	0.000000
<i>FMK08</i> – 4	0.715294(+)	0.508414(+)	0.728405(+)	0.790510(+)	0.766208(+)	0.458133(+)	0.315634(+)	0.000000
<i>FMK09</i> – 4	0.697904(+)	0.267392(+)	0.649259(+)	0.685933(+)	0.911289(+)	0.210140(+)	0.424645(+)	0.000000
<i>FMK10</i> – 4	0.823280(+)	0.479976(+)	0.756676(+)	0.781348(+)	0.687927(+)	0.197433(+)	0.361833(+)	0.000000
<i>Reman01</i> – 4	0.618731(+)	0.199573(+)	0.295730(+)	0.563550(+)	0.504519(+)	0.240536(+)	0.370486(+)	0.007495
<i>Reman02</i> – 4	0.896668(+)	0.164437(+)	0.450131(+)	0.734762(+)	0.884498(+)	0.236255(+)	0.394529(+)	0.048041
<i>Reman03</i> – 4	0.702020(+)	0.387431(+)	0.560119(+)	0.927009(+)	0.860319(+)	0.315389(+)	0.327372(+)	0.000000
<i>Reman04</i> – 4	0.355610(+)	0.166989(+)	0.360778(+)	0.833003(+)	0.764824(+)	0.447268(+)	0.352894(+)	0.020542
<i>Reman05</i> – 4	0.444605(+)	0.330360(+)	0.486198(+)	0.523713(+)	0.468015(+)	0.258598(+)	0.311834(+)	0.022168
<i>Reman06</i> – 4	0.582006(+)	0.280475(+)	0.405102(+)	0.907548(+)	0.561877(+)	0.270488(+)	0.328099(+)	0.125730
<i>Reman07</i> – 4	0.390426(+)	0.233903(+)	0.354941(+)	0.793420(+)	0.630871(+)	0.187328(+)	0.437334(+)	0.125094
<i>Reman08</i> – 4	0.256097(+)	0.140602(-)	0.190276(-)	0.751622(+)	0.730771(+)	0.229663(-)	0.174053(-)	0.232759
+ / $\approx$ / -	23/0/0	19/0/4	21/0/2	23/0/0	23/0/0	21/0/2	22/0/1	

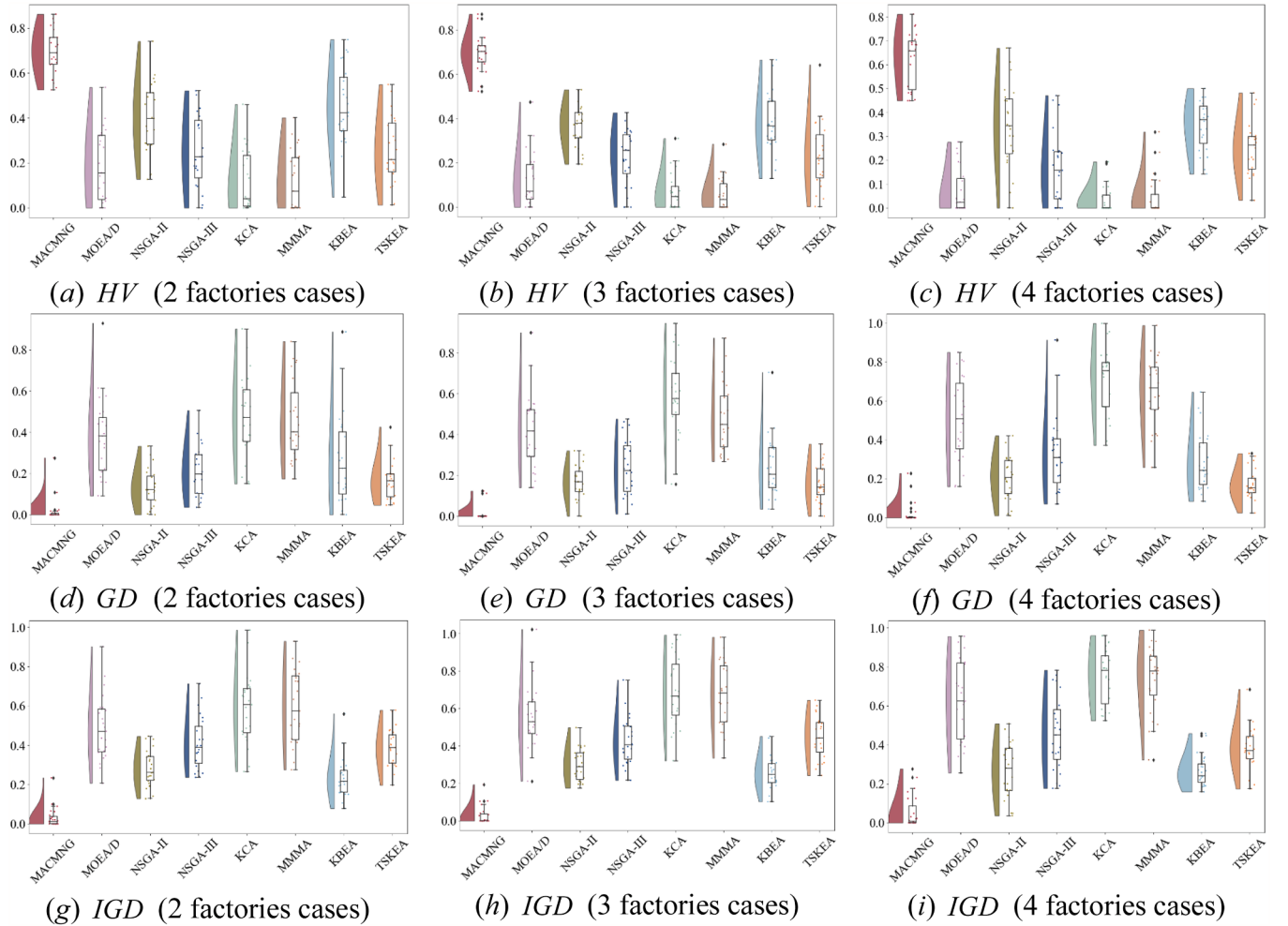
**Fig. 14.** Performance comparison of MACMNG with seven algorithms on metrics *HV*, *GD*, and *IGD*.

Table 21Results of Wilcoxon Test for MACMNG with all compared algorithms on *HV*, *GD*, and *IGD* metrics at $\alpha = 0.05$.

Algorithms	<i>HV</i>				<i>GD</i>				<i>IGD</i>			
	R^+	R^-	Z	p - value	R^+	R^-	Z	p - value	R^+	R^-	Z	p - value
MOEA/D	2415	0	7.219578	5.214926e-13	2413	2	7.207620	5.693823e-13	2415	0	7.219578	5.214926e-13
NSGA-II	2394	21	7.094020	1.302714e-12	2276	139	6.388504	1.675162e-10	2357	58	6.872799	6.295432e-12
NSGA-III	2415	0	7.219578	5.214926e-13	2355	60	6.860841	6.845636e-12	2410	5	7.189683	6.494193e-13
KCA	2415	0	7.219578	5.214926e-13	2415	0	7.219578	5.214926e-13	2415	0	7.219578	5.214926e-13
MMMA	2415	0	7.219578	5.214926e-13	2415	0	7.219578	5.214926e-13	2415	0	7.219578	5.214926e-13
KBEA	2405	10	7.159788	8.080178e-13	2355	60	6.860841	6.845636e-12	2396	19	7.105978	1.194736e-12
TSKEA	2415	0	7.219578	5.214926e-13	2319	96	6.645599	3.019867e-11	2413	2	7.207620	5.693833e-13

Table 22

Average Friedman rankings of all algorithms.

Algorithm	<i>HV</i> - Ranking	<i>GD</i> - Ranking	<i>IGD</i> - Ranking
MACMNG	1.072464	1.333333	1.173913
MOEA/D	6.333333	6.376812	6.144928
NSGA-II	2.927536	2.840580	2.724638
NSGA-III	4.963768	4.159420	4.666667
KCA	6.739130	7.028986	6.942029
MMMA	6.934783	6.826087	7.057971
KBEA	2.710145	4.275362	2.637681
TSKEA	4.318841	3.159420	4.637681
p -value	1.172184e-76	9.526443e-71	5.490583e-78

R^- values across all metrics (*HV*, *GD*, *IGD*) indicate that MACMNG's pervasive dominance. It is observed that the standardized test statistic Z further corroborates these findings, with higher positive Z values attesting to MACMNG's superior performance. Table 22 summarizes the Friedman Test rankings of all algorithms, where MACMNG ranks in the top-performing position across all metrics. The Friedman test also shows p - values < 0.05 , providing further validation of the superiority of MACMNG, delivering both stability and scalability.

As depicted in Fig. 15, the Pareto front distributions of MACMNG and comparative algorithms are visualized across 12 benchmark instances, demonstrating MACMNG's consistent dominance in solution quality and diversity. It is observed that MACMNG always achieves superior Pareto fronts across all test cases, outperforming baseline algorithms in both convergence and coverage. Notably, MACMNG's dynamic adaptation to bi-objective optimization ($\tilde{C}_{\max}$ and $\tilde{TEC}$) during the iterative training process is illustrated in Fig. 16, which shows how MACMNG adapts to varying weights to solve subproblems effectively. To illustrate the details, here we take the training curve of the FMK01-2 case as an example. The initial objective weight vector $\lambda_1 = (\lambda_1^{q_1}, \lambda_1^{q_2}) = (1, 0)$ is iteratively adjusted over 400 training epochs, with training of each subnet Subnet_h undergoing 20-epoch cycles. The dashed section in the figure marks the completion of the initial training phase for subnets $\text{Subnet}_1 = (\text{Subnet}_1^I, \text{Subnet}_1^M)$. As the weight vector $\lambda_h^{q_1}$ gradually decreases, its corresponding increases, while a higher weight vector $\lambda_h^{q_2}$ results in a lower $\tilde{TEC}$.

The Fig. 17 presents a visual representation of the fuzzy Gantt charts for instance FMK03 - 2. Fig. 17(a) and (b) display the fuzzy Gantt charts with the best $\tilde{C}_{\max}$ for each factory, yielding $\tilde{C}_{\max}$ values of (62.6, 96.4, 126) and $\tilde{TEC}$ values of (7782.6, 11705.0, 15187.0), respectively. Fig. 17(c) and (d) illustrate the fuzzy Gantt charts fine-tuned for $\tilde{TEC}$, resulting in the $\tilde{C}_{\max}$ values of (147.0, 209.0, 264.8) and $\tilde{TEC}$ values of (4777.8, 7070.0, 9196.6). Furthermore, Fig. 18 provides an illustration of the fuzzy Gantt charts for the instance Reman05 - 2, corresponding to the best values for $\tilde{C}_{\max} = (47.6, 78.8, 111.4)$ and $\tilde{TEC} = (2547.2, 4227.4, 5796.6)$.

6. Conclusions, limitations, and future research directions

The MACMNG framework introduced in this study has significant advantages against traditional advanced approaches, demonstrating robust adaptability and optimization potential for coping with complex constraints and coupled correlations. The MACMNG is capable of decomposing complex challenges into correlated subproblems and creating subnetworks for cooperative parallel processing. Experimental findings and statistical insights demonstrate that MACMNG outperforms state-of-the-art multi-objective algorithms in terms of both economic and energy efficiency criteria. However, as the number of subproblems increases, the model's internal subnets expand accordingly, resulting in a sharp increase in storage requirements, which poses critical challenges in resource-limited scenarios.

Future research directions should focus on several key aspects to enhance MACMNG's efficiency and applicability. First, incorporating GCNs and multi-scale attention mechanisms (MSAMs) into the scheduling environment can enhance MACMNG's capacity to capture complex interdependencies among critical features, thereby enriching multi-state feature representation and multi-modal feature extraction. Furthermore, investigating structural optimization of the DNNs and lightweight surrogate models, particularly focusing on the design of compact network group architectures, contributes to controlling the complexity of the models, facilitates cooperative optimization across network groups, significantly improving the effectiveness and computational efficiency of MACMNG.

In conclusion, cooperative optimization of multi-network groups for distributed energy-efficient scheduling through the CTDE-based MARL framework represents an emerging research direction, with current research reports remaining nascent. Focusing on EE-DFFJSP within IIoT-enabled smart manufacturing systems, this study provides a solid foundation and fresh perspective to promote future research and the development of energy-efficient scheduling research in DFM environments.

CRedit authorship contribution statement

Qian Bin: Writing – review & editing, Methodology, Investigation, Funding acquisition. **Xiao-Wei Li:** Writing – original draft, Software, Methodology, Investigation. **Rong Hu:** Writing – review & editing, Supervision, Methodology, Funding acquisition. **Huai-Ping Jin:** Software, Investigation. **Jian-Bo Yang:** Project administration. **Zi-Qi Zhang:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence

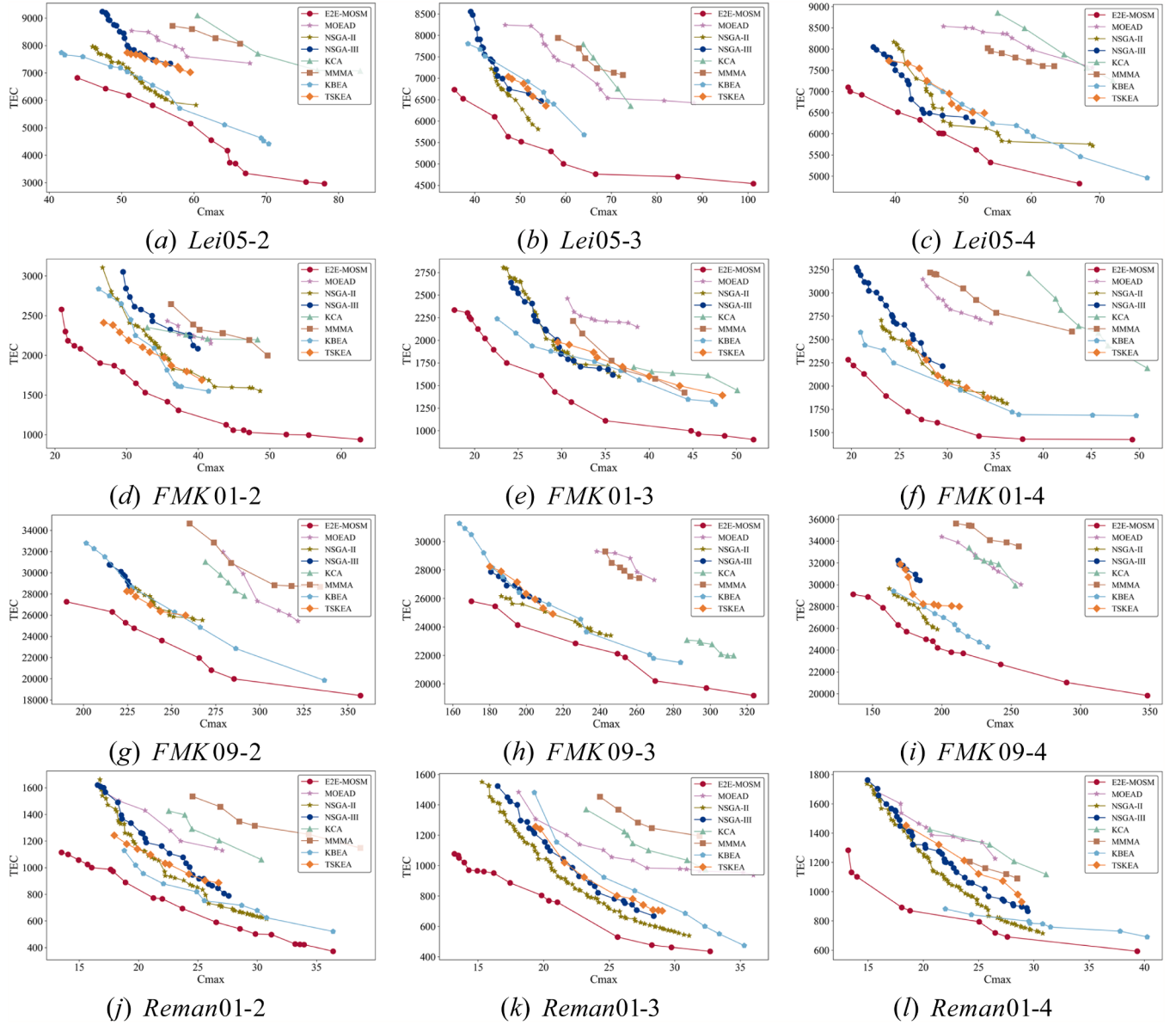


Fig. 15. Pareto fronts obtained by MACMNG and seven algorithms across 12 instances.

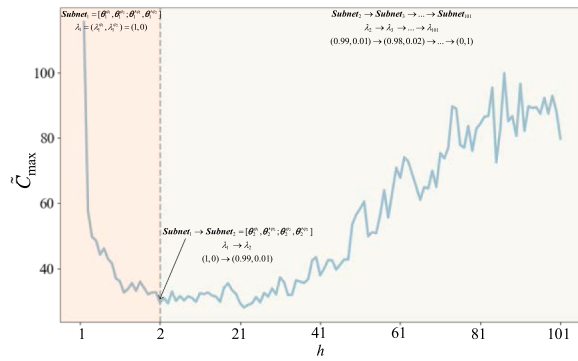
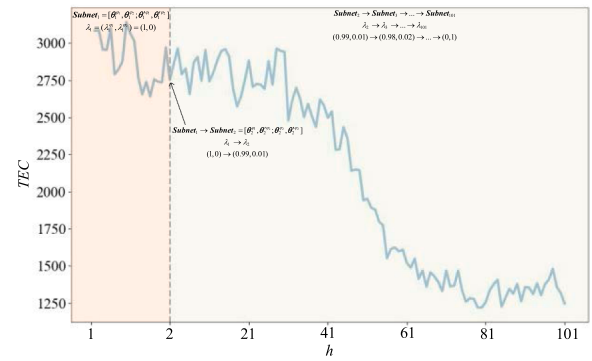
(a) Training curve on $\tilde{C}_{\max}$ (b) Training curve on TEC

Fig. 16. Training curve for MACMNG.

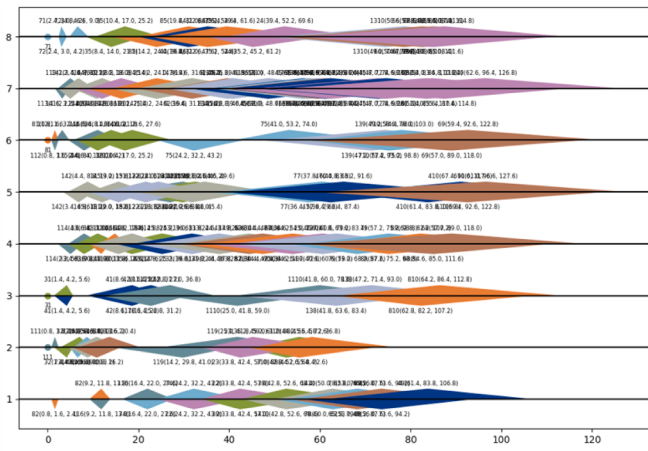
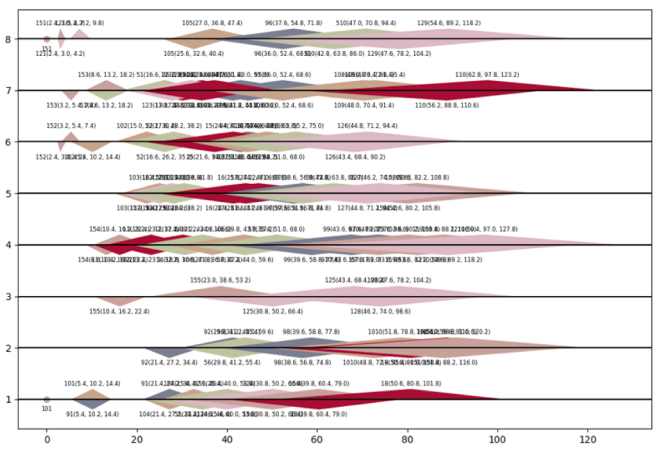
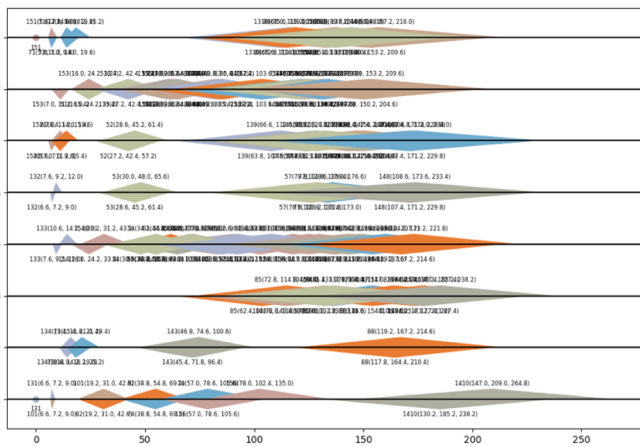
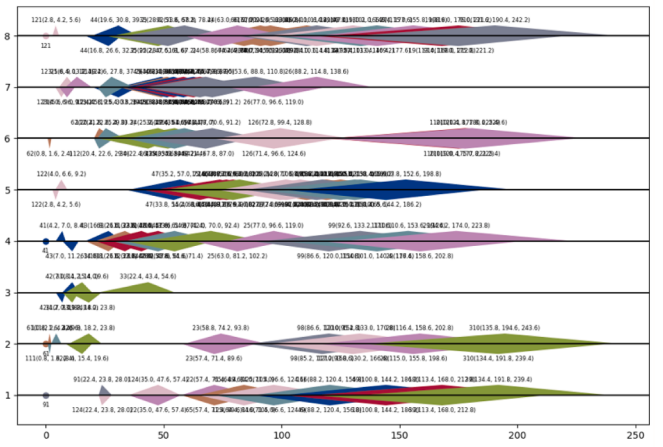
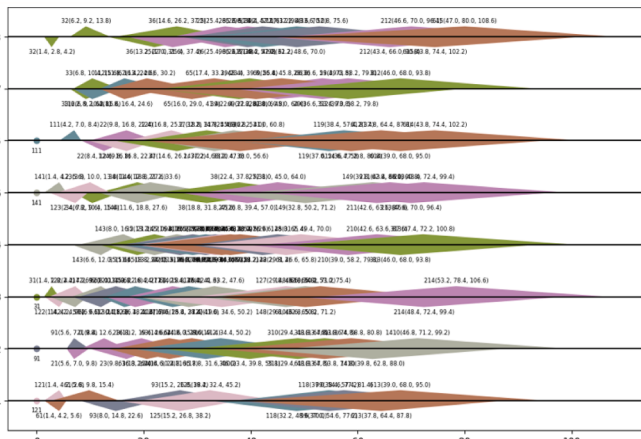
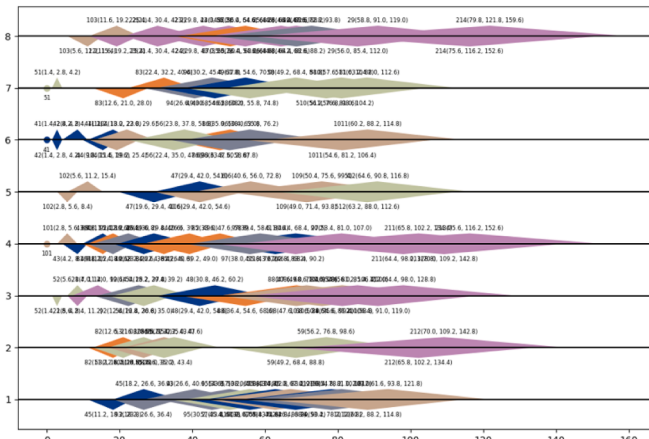
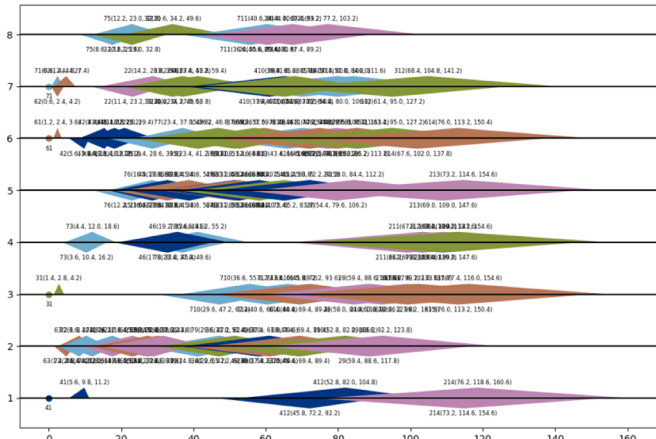
(a) Fuzzy Gantt chart with best $\tilde{C}_{\max}$ in factory 1.(b) Fuzzy Gantt chart with best $\tilde{C}_{\max}$ in factory 2.(c) Fuzzy Gantt chart with best TEC in factory 1.(d) Fuzzy Gantt chart with best TEC in factory 2.

Fig. 17. Fuzzy Gantt charts for the instance FMK03 – 2.

(a) Fuzzy Gantt chart with best $\tilde{C}_{max}$ in factory 1.(b) Fuzzy Gantt chart with best $\tilde{C}_{max}$ in factory 2.(c) Fuzzy Gantt chart with best TEC in factory 1.(d) Fuzzy Gantt chart with best TEC in factory 2.Fig. 18. Fuzzy Gantt charts for the instance *Reman05* – 2.

the work reported in this paper.

Acknowledgments

The authors are sincerely grateful to the anonymous reviewers for their insightful comments and suggestions, which greatly improve this paper. This work was financially supported by the National Natural Science Foundation of China (Grant Nos. 72201115, 62173169, and U24A20273), Yunnan Fundamental Research Projects (Grant Nos. 202501AW070010, 202201BE070001-050 and 202301AU070069), the Basic Research Key Project of Yunnan Province (Grant No. 202201AS070030), and the Construction Project of Higher Educational Key Laboratory for Industrial Intelligence and Systems of Yunnan Province (Grant Number: KKP202403003).

Data availability

Data will be made available on request.

References

- [1] P. Aruchamy, L. Balraj, K.K.D. Sowndarya, An energy-aware link fault detection and recovery scheme for QoS enhancement in internet of things-enabled wireless sensor network, *Comput. Electr. Eng.* 123 (2025) 110092, <https://doi.org/10.1016/j.compeleceng.2025.110092>.
- [2] S. Madadi-Barough, P. Ruiz-Blanco, J.D. Lin, R. Vidal, C. Gomez, Matter: IoT interoperability for smart homes, *IEEE Commun. Mag.* 63 (2025) 106–112, <https://doi.org/10.1109/Mcom.001.2400274>.
- [3] A.D. Prasanth, L. Dhanaraj, R.K. Balusamy, B. P. C. S. Technological Advancement in Internet of Medical Things and Blockchain for Personalized Healthcare: Applications and Use Cases, CRC Press, 2024, <https://doi.org/10.1201/97810033405450>.
- [4] F. Puig, M. Garcia-Vila, M.A. Soriano, J.A. Rodríguez-Díaz, AquaCrop-IoT: a smart irrigation platform integrating real-time images and weather forecasting, *Comput. Electron. Agric.* 235 (2025) 110372, <https://doi.org/10.1016/j.compag.2025.110372>.
- [5] L. Balraj, A. Prasanth, K.K.D. Sowndarya, T. Kuntavai, A lightweight blockchain scheme for secure data communication in internet of things-enabled wireless sensor network, *Proceedings of the International Conference on Smart Systems for applications in Electrical Sciences (ICSSSES)*, 2024, pp. 1–6, <https://doi.org/10.1109/ICSSSES62373.2024.10561348>.
- [6] J.C. Serrano-Ruiz, J. Mula, R. Poler, Smart manufacturing scheduling: a literature review, *J. Manuf. Syst.* 61 (2021) 265–287, <https://doi.org/10.1016/j.jmsy.2021.09.011>.
- [7] L. Meng, C. Zhang, Y. Ren, B. Zhang, C. Lv, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, *Comput. Ind. Eng.* 142 (2020) 106347, <https://doi.org/10.1016/j.cie.2020.106347>.
- [8] J.P. Huang, L. Gao, X.Y. Li, C.J. Zhang, A novel priority dispatch rule generation method based on graph neural network and reinforcement learning for distributed job-shop scheduling, *J. Manuf. Syst.* 69 (2023) 119–134, <https://doi.org/10.1016/j.jmsy.2023.06.007>.
- [9] C.E. Okwudire, H.V. Madhyastha, Distributed manufacturing for and by the masses, *Science* 372 (2021) 341–342, <https://doi.org/10.1126/science.abg4924>.
- [10] J.-q. Li, Q.-k. Pan, Chemical-reaction optimization for solving fuzzy job-shop scheduling problem with flexible maintenance activities, *Int. J. Prod. Econ.* 145 (2013) 4–17, <https://doi.org/10.1016/j.jpe.2012.11.005>.

- [11] B. Liu, Y. Fan, Y. Liu, A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem, *Comput. Ind. Eng.* 87 (2015) 193–201, <https://doi.org/10.1016/j.cie.2015.04.029>.
- [12] J.G. Duan, J.H. Wang, Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection, *Comput. Ind. Eng.* 161 (2021) 107677, <https://doi.org/10.1016/j.cie.2021.107677>.
- [13] Z.Q. Zhang, F.C. Wu, B. Qian, R. Hu, L. Wang, H.P. Jin, A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation, *Expert Syst. Appl.* 234 (2023) 121050, <https://doi.org/10.1016/j.eswa.2023.121050>.
- [14] E. Levén, A. Segerstedt, A scheduling policy for adjusting economic lot quantities to a feasible solution, *Eur. J. Oper. Res.* 179 (2007) 414–423, <https://doi.org/10.1016/j.ejor.2006.03.038>.
- [15] M. Abdellatif, A.E. Hassanien, M. Mortagi, H. Hamouda, Geopolymer foam concrete: a review of pore characteristics, compressive strength and artificial intelligence in GFC strength simulations, *Discov. Concr. Cem.* 1 (2025) 4, <https://doi.org/10.1007/s44416-025-00003-x>.
- [16] M. Abdellatif, G. Murali, S. Dixit, Leveraging machine learning to evaluate the effect of raw materials on the compressive strength of ultra-high-performance concrete, *Results Eng.* 25 (2025) 104542, <https://doi.org/10.1016/j.rineng.2025.104542>.
- [17] M. Abdellatif, L.S. Wong, N.M. Din, A.N. Ahmed, A.M. Hassan, Z. Ibrahim, G. Murali, K.H. Mo, A. El-Shafie, Sustainable foam glass property prediction using machine learning: a comprehensive comparison of predictive methods and techniques, *Results Eng.* 25 (2025) 104089, <https://doi.org/10.1016/j.rineng.2025.104089>.
- [18] J.-D. Zhang, Z. He, W.-H. Chan, C.-Y. Chow, DeepMAG: deep reinforcement learning with multi-agent graphs for flexible job shop scheduling, *Knowl. Based Syst.* 259 (2023) 110083, <https://doi.org/10.1016/j.knsys.2022.110083>.
- [19] Y. Du, J.-q. Li, A deep reinforcement learning based algorithm for a distributed precast concrete production scheduling, *Int. J. Prod. Econ.* 268 (2024) 109102, <https://doi.org/10.1016/j.ijpe.2023.109102>.
- [20] R. Wang, G. Wang, J. Sun, F. Deng, J. Chen, Flexible Job shop scheduling via dual attention network-based reinforcement learning, *IEEE Trans. Neural Netw. Learn. Syst.* 35 (2024) 3091–3102, <https://doi.org/10.1109/TNNLS.2023.3306421>.
- [21] C. Li, Q. Chang, H.-T. Fan, Multi-agent reinforcement learning for integrated manufacturing system-process control, *J. Manuf. Syst.* 76 (2024) 585–598, <https://doi.org/10.1016/j.jmsy.2024.08.021>.
- [22] X. Jing, X. Yao, M. Liu, J. Zhou, Multi-agent reinforcement learning based on graph convolutional network for flexible job shop scheduling, *J. Intell. Manuf.* 35 (2022) 75–93, <https://doi.org/10.1007/s10845-022-02037-5>.
- [23] P. Rokhforoz, M. Montazeri, O. Fink, Multi-agent reinforcement learning with graph convolutional neural networks for optimal bidding strategies of generation units in electricity markets, *Expert Syst. Appl.* 225 (2023) 120010, <https://doi.org/10.1016/j.eswa.2023.120010>.
- [24] M. Schilling, A. Melnik, F.W. Ohl, H.J. Ritter, B. Hammer, Decentralized control and local information for robust and adaptive decentralized deep reinforcement learning, *Neural Netw.* 144 (2021) 699–725, <https://doi.org/10.1016/j.neunet.2021.09.017>.
- [25] T.T. Nguyen, N.D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications, *IEEE Trans. Cybern.* 50 (2020) 3826–3839, <https://doi.org/10.1109/Tcyb.2020.2977374>.
- [26] Z. Feng, M. Huang, Y. Wu, D. Wu, J. Cao, I. Korovin, S. Gorbachev, N. Gorbacheva, Approximating nash equilibrium for anti-UAV jamming markov game using a novel event-triggered multi-agent reinforcement learning, *Neural Netw.* 161 (2023) 330–342, <https://doi.org/10.1016/j.neunet.2022.12.022>.
- [27] F. Ren, W. Dong, X. Zhao, F. Zhang, Y. Kong, Q. Yang, Two-layer coordinated reinforcement learning for traffic signal control in traffic network, *Expert Syst. Appl.* 235 (2024) 121111, <https://doi.org/10.1016/j.eswa.2023.121111>.
- [28] X. Wu, Y. Sun, A green scheduling algorithm for flexible job shop with energy-saving measures, *J. Clean. Prod.* 172 (2018) 3249–3264, <https://doi.org/10.1016/j.jclepro.2017.10.342>.
- [29] L.P. Zhang, Q.H. Tang, Z.J. Wu, F. Wang, Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops, *Energy* 138 (2017) 210–227, <https://doi.org/10.1016/j.energy.2017.07.005>.
- [30] M. Dai, D.B. Tang, A. Giret, M.A. Salido, W.D. Li, Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm, *Robot. Comput. Integr. Manuf.* 29 (2013) 418–429, <https://doi.org/10.1016/j.rcim.2013.04.001>.
- [31] L. Meng, C. Zhang, B. Zhang, Y. Ren, Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility, *IEEE Access* 7 (2019) 68043–68059, <https://doi.org/10.1109/ACCESS.2019.2916468>.
- [32] G.L. Gong, R. Chiong, Q.W. Deng, W.W. Han, L.K. Zhang, D. Huang, Energy-efficient production scheduling through machine on/off control during preventive maintenance, *Eng. Appl. Artif. Intell.* 104 (2021) 104359, <https://doi.org/10.1016/j.engappai.2021.104359>.
- [33] M.J. Park, A. Ham, Energy-aware flexible job shop scheduling under time-of-use pricing, *Int. J. Prod. Econ.* 248 (2022) 108507, <https://doi.org/10.1016/j.ijpe.2022.108507>.
- [34] D. Catanzaro, R. Pesenti, R. Ronco, Job scheduling under time-of-use energy tariffs for sustainable manufacturing: a survey, *Eur. J. Oper. Res.* 308 (2023) 1091–1109, <https://doi.org/10.1016/j.ejor.2023.01.029>.
- [35] B. Zhang, Q.-k. Pan, L. Gao, X.-y. Li, L.-l. Meng, K.-k. Peng, A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem, *Comput. Ind. Eng.* 136 (2019) 325–344, <https://doi.org/10.1016/j.cie.2019.07.036>.
- [36] J. Duan, J. Wang, Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection, *Comput. Ind. Eng.* 161 (2021) 107677, <https://doi.org/10.1016/j.cie.2021.107677>.
- [37] C.S. Lin, P.Y. Li, J.M. Wei, M.C. Wu, Integration of process planning and scheduling for distributed flexible job shops, *Comput. Oper. Res.* 124 (2020) 105053, <https://doi.org/10.1016/j.cor.2020.105053>.
- [38] W.X. Xu, Y.W. Hu, W. Luo, L. Wang, R. Wu, A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission, *Comput. Ind. Eng.* 157 (2021) 107318, <https://doi.org/10.1016/j.cie.2021.107318>.
- [39] Y. Du, J.Q. Li, C. Luo, L.L. Meng, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, *Swarm Evolut. Comput.* 62 (2021) 100861, <https://doi.org/10.1016/j.swevo.2021.100861>.
- [40] Q. Luo, Q.W. Deng, G.L. Gong, L.K. Zhang, W.W. Han, K.X. Li, An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers, *Expert Syst. Appl.* 160 (2020) 113721, <https://doi.org/10.1016/j.eswa.2020.113721>.
- [41] J.J. Wang, L. Wang, A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop, *IEEE Trans. Syst. Man Cybern. Syst.* 50 (2020) 1805–1819, <https://doi.org/10.1109/Tsmc.2017.2788879>.
- [42] W.S. Shao, Z.S. Shao, D.C. Pi, A multi-neighborhood-based multi-objective memetic algorithm for the energy-efficient distributed flexible flow shop scheduling problem, *Neural Comput. Appl.* 34 (2022) 22303–22330, <https://doi.org/10.1007/s00521-022-07714-3>.
- [43] Z.Q. Zhang, X.P. Zhu, Y.X. Xu, B. Qian, B. Yang, R. Hu, MEDHEA: Multidimensional estimation of distribution based hyper-heuristic evolutionary algorithm for energy-efficient distributed assembly no-wait flow-shop scheduling problem, *Expert Syst. Appl.* 271 (2025) 126526, <https://doi.org/10.1016/j.eswa.2025.126526>.
- [44] L.L. Meng, Y.P. Ren, B. Zhang, J.Q. Li, H.Y. Sang, C.Y. Zhang, MILP Modeling and optimization of energy-efficient distributed flexible job shop scheduling problem, *IEEE Access* 8 (2020) 191191–191203, <https://doi.org/10.1109/Access.2020.3032548>.
- [45] F. Yu, C. Lu, J. Zhou, L. Yin, K. Wang, A knowledge-guided bi-population evolutionary algorithm for energy-efficient scheduling of distributed flexible job shop problem, *Eng. Appl. Artif. Intell.* 128 (2024) 107458, <https://doi.org/10.1016/j.engappai.2023.107458>.
- [46] Z.-Q. Zhang, Y. Li, B. Qian, R. Hu, J.-B. Yang, A multidimensional probabilistic model based evolutionary algorithm for the energy-efficient distributed flexible job-shop scheduling problem, *Eng. Appl. Artif. Intell.* 135 (2024) 108841, <https://doi.org/10.1016/j.engappai.2024.108841>.
- [47] R. Li, W.Y. Gong, L. Wang, C. Lu, S.N. Jiang, Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time, *Swarm Evolut. Comput.* 74 (2022) 101139, <https://doi.org/10.1016/j.swevo.2022.101139>.
- [48] K. Lei, P. Guo, W.C. Zhao, Y. Wang, L.M. Qian, X.Y. Meng, L.S. Tang, A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem, *Expert Syst. Appl.* 205 (2022) 117796, <https://doi.org/10.1016/j.eswa.2022.117796>.
- [49] W. Song, X.Y. Chen, Q.Q. Li, Z.G. Cao, Flexible Job-shop scheduling via graph neural network and deep reinforcement learning, *IEEE Trans. Ind. Inform.* 19 (2023) 1600–1610, <https://doi.org/10.1109/Tii.2022.3189725>.
- [50] E.D. Yuan, L.J. Wang, S.L. Cheng, S.J. Song, W. Fan, Y.M. Li, Solving flexible job shop scheduling problems via deep reinforcement learning, *Expert Syst. Appl.* 245 (2024) 123019, <https://doi.org/10.1016/j.eswa.2023.123019>.
- [51] W. Sun, Z. Li, J. Shi, Z. Bai, F. Wang, T.Q.S. Quek, MAHTD-DDPG-based multi-objective resource allocation for UAV-assisted wireless network, *IEEE J. Miniatur. Syst.* 2024 (2024), <https://doi.org/10.1109/jmass.2024.3420893>, 1–1.
- [52] Y. Birman, S. Hindi, G. Katz, A. Shabtai, Cost-effective ensemble models selection using deep reinforcement learning, *Inf. Fusion* 77 (2022) 133–148, <https://doi.org/10.1016/j.inffus.2021.07.011>.
- [53] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, *Nature* 529 (2016) 484, <https://doi.org/10.1038/nature16961>.
- [54] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, *Nature* 518 (2015) 529–533, <https://doi.org/10.1038/nature14236>.
- [55] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y.T. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, *Nature* 550 (2017) 354, <https://doi.org/10.1038/nature24270>.
- [56] R. Liu, R. Piplani, C. Toro, Deep reinforcement learning for dynamic scheduling of a flexible job shop, *Int. J. Prod. Res.* 60 (2022) 4049–4069, <https://doi.org/10.1080/00207543.2022.2058432>.
- [57] L. Wan, L. Fu, C. Li, K. Li, An effective multi-agent-based graph reinforcement learning method for solving flexible job shop scheduling problem, *Eng. Appl. Artif. Intell.* 139 (2025) 109557, <https://doi.org/10.1016/j.engappai.2024.109557>.
- [58] J.P. Huang, L. Gao, X.Y. Li, A hierarchical multi-action deep reinforcement learning method for dynamic distributed job-shop scheduling problem with job

- arrivals, *IEEE Trans. Autom. Sci. Eng.* 22 (2025) 2501–2513, <https://doi.org/10.1109/Tase.2024.3380644>.
- [59] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Ann. Oper. Res.* 41 (1993) 157–183, <https://doi.org/10.1007/bf02023073>.
- [60] G. Bortolan, R. Degani, A review of some methods for ranking fuzzy subsets, *Fuzzy Sets Syst.* 15 (1985) 1–19, [https://doi.org/10.1016/0165-0114\(85\)90012-0](https://doi.org/10.1016/0165-0114(85)90012-0).
- [61] J. Lin, A hybrid biogeography-based optimization for the fuzzy flexible job-shop scheduling problem, *Knowl. Based Syst.* 78 (2015) 59–74, <https://doi.org/10.1016/j.knosys.2015.01.017>.
- [62] D.M. Lei, A genetic algorithm for flexible job shop scheduling with fuzzy processing time, *Int. J. Prod. Res.* 48 (2010) 2995–3013, <https://doi.org/10.1080/00207540902814348>.
- [63] M. Sakawa, R. Kubota, Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy due date through genetic algorithms, *Eur. J. Oper. Res.* 120 (2000) 393–407, [https://doi.org/10.1016/S0377-2217\(99\)00094-6](https://doi.org/10.1016/S0377-2217(99)00094-6).
- [64] K. Li, K. Deb, Q.F. Zhang, S. Kwong, An evolutionary many-objective optimization algorithm based on dominance and decomposition, *IEEE Trans. Evolut. Comput.* 19 (2015) 694–716, <https://doi.org/10.1109/Tevc.2014.2373386>.
- [65] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints, *IEEE Trans. Evolut. Comput.* 18 (2014) 577–601, <https://doi.org/10.1109/TEVC.2013.2281535>.
- [66] K. Miettinen, *Nonlinear Multiobjective Optimization*, Springer Science & Business Media, 1999.
- [67] J.P. Huang, L. Gao, X.Y. Li, C.J. Zhang, A cooperative hierarchical deep reinforcement learning based multi-agent method for distributed job shop scheduling problem with random job arrivals, *Comput. Ind. Eng.* 185 (2023) 109650, <https://doi.org/10.1016/j.cie.2023.109650>.
- [68] D.M. Lei, Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling, *Appl. Soft Comput.* 12 (2012) 2237–2245, <https://doi.org/10.1016/j.asoc.2012.03.025>.
- [69] K.Z. Gao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time, *Int. J. Prod. Res.* 53 (2015) 5896–5911, <https://doi.org/10.1080/00207543.2015.1020174>.
- [70] R. Li, W.Y. Gong, C. Lu, Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time, *Comput. Ind. Eng.* 168 (2022) 108099, <https://doi.org/10.1016/j.cie.2022.108099>.
- [71] L. While, P. Hingston, L. Barone, S. Huband, A faster algorithm for calculating hypervolume, *IEEE Trans. Evolut. Comput.* 10 (2006) 29–38, <https://doi.org/10.1109/TEVC.2005.851275>.
- [72] M. Zangari, A. Mendiburu, R. Santana, A. Pozo, Multiobjective decomposition-based mallow models estimation of distribution algorithm. a case of study for permutation flowshop scheduling problem, *Inf. Sci.* 397 (2017) 137–154, <https://doi.org/10.1016/j.ins.2017.02.034>.
- [73] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evolut. Comput.* 6 (2002) 182–197, <https://doi.org/10.1109/4235.996017>.
- [74] J.J. Wang, L. Wang, A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop, *IEEE Trans. Syst. Man Cybern. Syst.* 50 (2020) 1805–1819, <https://doi.org/10.1109/TSMC.2017.2788879>.