

Multi-Agent Continuous Decision-Making for the Continuous Dynamic Flexible Job Shop Scheduling Problem

Dazzle Johnson[✉], Gang Chen[✉], *Senior Member, IEEE*, and Yuqian Lu[✉], *Member, IEEE*

Abstract—The Continuous Dynamic Flexible Job Shop Scheduling Problem (C-DFJSP) represents a critical challenge in smart manufacturing due to its continuous, dynamic nature, requiring real-time decision-making and high adaptability to diverse factory configurations. We provide a solution to the C-DFJSP problem, where the notion of a global optimum is inherently ill-defined. To solve the C-DFJSP, we propose a Multi-Agent Continuous Decision-Making (MACD) system. MACD leverages a novel state and observation design to extract novel factory-invariant features, enabling a learned scheduling policy to generalise across different factories, machine capabilities, and product libraries. MACD is an enhanced multi-agent continuous scheduling system that integrates Graph Neural Networks (GNNs) and Reinforcement Learning (RL) within a multi-agent framework. In addition, MACD employs an innovative negotiation strategy—a hybrid method combining Shortest Processing Time (SPT) and Least Loaded Machine (LLM) heuristics—supported by an empirical study to resolve rare conflicts among machine agents, enhancing adaptability and performance. Extensive experiments on public datasets validate MACD's superior ability to generalise across diverse factory configurations, allowing a trained scheduling policy to be directly used in different factories with varying machine flexibilities, utilisation levels, and product libraries.

Note to Practitioners—This work addresses the practical demand for a flexible and scalable scheduling policy capable of continuous decision-making in dynamic manufacturing environments while maintaining effectiveness and reliability across diverse factory configurations without the need to adjust/retrain the policy. Traditional scheduling methods struggle to adapt in today's fast-evolving manufacturing landscape, where job specifications and machine configurations constantly evolve. The Continuous Dynamic Flexible Job Shop Scheduling Problem (C-DFJSP) explicitly encapsulates these challenges, demanding real-time adaptability and efficient decision-making in modern factories. To address C-DFJSP, we introduce MACD, a Multi-Agent Continuous Decision-making system that utilises Graph Neural Networks (GNNs) to extract novel factory-invariant features, enabling learned scheduling policies to generalise across different factories, machine capabilities, and product libraries. By incorporating domain knowledge, such as prioritising faster

machines for higher efficiency and utilising a distributed multi-agent framework, MACD streamlines decision-making, enhances scalability, and enables a single trained policy to be applied across diverse factory configurations. It establishes a solid foundation for learning adaptive scheduling policies to tackle serious real-world disruptions like machine breakdowns. Compared to multiple state-of-the-art approaches, MACD significantly improves scheduling efficiency, adaptability, and productivity, leading to better resource utilisation and cost savings in smart manufacturing systems.

Index Terms—Intelligent and flexible manufacturing, manufacturing scheduling, production control, planning, scheduling and coordination, reinforcement learning.

I. INTRODUCTION

THE rise of mass personalisation is transforming manufacturing, demanding factories to produce diverse products in dynamic and flexible environments while maintaining the speed and efficiency of traditional mass production [1], [2]. Achieving this goal demands scheduling systems that can adapt to diverse products, fluctuating demand volumes and rates, and evolving factory configurations, including changes in machine count and flexibility [1], [2].

These scheduling challenges posed by mass personalisation far exceed the complexity of the traditional Flexible Job Shop Scheduling Problem (FJSP) [3], [4], [5], which is limited to handling fixed and static job orders. Meanwhile, existing dynamic problem variants (DFJSP), though accounting for the addition of a fixed number of new job orders, often ignore *continuous* job order arrivals and dynamic changes of the product library [6], [7], [8]. To bridge this gap, [9] formally introduced the *Continuous Dynamic Flexible Job Shop Scheduling Problem (C-DFJSP)* in 2024, a new problem formulation that captures three key characteristics: (i) continuous job arrivals rather than a static queue, (ii) asynchronous machine state transitions due to overlapping processing and idle times, and (iii) decision points triggered in real time rather than at synchronised epochs. These properties mirror the realities of many modern manufacturing systems—such as flexible robotic assembly [10] and machining workshops in automotive [11] and aerospace [12] manufacturing that operate continuously under mixed-product conditions, where multiple machines can perform similar operations and dynamic scheduling is essential to maintain throughput. Due to the unbounded time horizon, continuous arrivals, and lack of

Received 11 February 2025; revised 31 August 2025, 26 November 2025, and 13 March 2026; accepted 18 May 2026. Date of publication 2 June 2026; date of current version 9 June 2026. This article was recommended for publication by Associate Editor B. Yan and Editor Q.-S. Jia upon evaluation of the reviewers' comments. (*Corresponding author: Yuqian Lu.*)

Dazzle Johnson and Yuqian Lu are with the Department of Mechanical and Mechatronics Engineering, University of Auckland, 1010 Auckland, New Zealand (e-mail: yuqian.lu@auckland.ac.nz).

Gang Chen is with the School of Engineering and Computer Science, Victoria University of Wellington, 6140 Wellington, New Zealand.

Data is available on-line at https://github.com/uoa-iai/MACD_CDFJSP
Digital Object Identifier 10.1109/TASE.2026.3698853

future knowledge, C-DFJSP does not admit a well-defined global optimum, further distinguishing it from classical combinatorial scheduling problems. The C-DFJSP thus provides a generalised and realistic scheduling framework that challenges existing scheduling methods, which lack continuous adaptation and struggle to accommodate new products with updated production requirements [13].

Unlike DFJSP formulations, which typically considers a dynamic but finite set of job arrivals within a bounded planning period, and sometimes incorporates dynamic events such as breakdowns, the C-DFJSP models **continuous, infinite-horizon scheduling** with unbounded flow of jobs, diverse personalised products, and urgent orders. This distinction reflects the operational reality where modern manufacturing systems operate continuously with ongoing job arrivals, and therefore factories must continuously create and optimise schedules (e.g., daily or weekly), rather than compute one-off solutions for static batches of jobs. Accordingly, our novelty lies in addressing this modern formulation, which demands long-term optimisation and scalability across varying factory configurations.

Therefore, scheduling performance metrics need to change. For a continuous-time setup, metrics must reflect past, present, and future performance to provide holistic insights into scheduling effectiveness and stability. To this end, for C-DFJSP and in line with [9], we focus on two key metrics that are periodically evaluated: the **Mean Tardiness (MT)**, which captures how well deadlines were being met in the *past*, and the **number of Late Jobs remaining (LJ)** in the system, reflecting the *present* system backlog and stability that affect *future* performance.

An initial attempt to address C-DFJSP was made in [9], but its effectiveness declines when the factory configuration, including machine count and flexibility, differs significantly from the training environment. In view of this, we develop an innovative **Multi-Agent Continuous Decision-Making system**—MACD—that can effectively generalise to previously unseen factory configurations and handle varying product types and job order arrival patterns. We also clarify that MACD considers job and machine selection jointly rather than as independent stages.

For MACD to effectively cope with factories of different sizes and levels of machine flexibility, we must capture the intrinsic factory-invariant characteristics and carefully process machine-related information in the scheduling system. For this purpose, MACD leverages a novel state and observation design—utilising Graph Neural Networks (GNNs) [14] for job order feature extraction combined with critical machine information. MACD further exploits domain knowledge to filter irrelevant machines and employs attention mechanisms to process machine information for improved scalability and computational efficiency required for handling diverse factory configurations. This unique design ensures that machines are appropriately modelled, regardless of factory size and machine flexibility. Consequently, MACD can generalise effectively across a broad range of factory configurations.

In systems with distributed decision-makers, conflicts in machine selection are inevitable [15], [16]. Existing literature

often resolves disputes through bidding or heuristics rules; however, these methods have not been fully explored in continuous, dynamic environments with varying flexibility levels and job arrival rates. To address this gap, we incorporate a lightweight negotiation strategy within MACD that is triggered only when multiple agents select the same job order simultaneously. Importantly, this mechanism is independent of the reinforcement learning policy and is used only to resolve occasional conflicts. Our strategy combines the strengths of the *Shortest Processing Time (SPT)* and *Least Loaded Machine (LLM)* heuristics. Our design of this hybrid strategy is driven by thorough empirical evaluation, where we analysed the performance of different machine selection rules across diverse manufacturing environments (see Section V-G for details). By leveraging SPT for efficiency and LLM for machine load balancing, this hybrid approach effectively resolves conflicts in machine selection, offering an elegant solution to ensure high scheduling effectiveness across varying scenarios.

The primary contributions of this paper are as follows:

- **Generalisable scheduling framework (MACD):** We introduce a novel **Multi-Agent Continuous Decision-making (MACD)** framework that combines GNN-based feature extraction with shared decision policies to enable generalisation across diverse factory settings for solving the challenging C-DFJSP [9]. Unlike prior work that typically trains models tied to a specific factory or static job sets, our approach learns factory-invariant representations and supports real-time adaptability. Importantly, MACD considers job and machine selection jointly, rather than as separate decision stages, ensuring coordinated decision-making in decentralised environments.
- **Operation-centric graph representation:** We propose a carefully designed graph representation where **only operation nodes are embedded**, and machine-level constraints and capabilities are incorporated via attention-based aggregation. This design significantly improves scalability while preserving key decision context.
- **Rule-guided negotiation mechanism:** We design a lightweight **hybrid negotiation strategy** that leverages empirical preferences between SPT and LLM rules to resolve conflicts among machine agents. This mechanism is only triggered in rare contention scenarios, ensuring coordination without centralised control. Embedding this negotiation layer into the decentralised learning policy is novel for real-time dynamic job shops, as it provides stability and adaptability while preserving generalisation across factories.
- **Strong generalisation and scalability:** We demonstrate that a **single trained MACD model**, learned on a small-scale static environment, can be directly deployed in previously **unseen large-scale, dynamic factories without retraining**.

II. RELATED WORK

While much of the prior work has focused on static or finite-horizon scheduling, our study explicitly targets the C-DFJSP formulation, first proposed in [9]. This formulation

TABLE I
COMPARISON OF EXISTING SCHEDULING APPROACHES WITH THE PROPOSED MACD FRAMEWORK

Approach	Problem Setting	Methodology	Key Limitation / Gap
Heuristics & Metaheuristics [6], [7], [8], [17]	Static FJSP	Rule-based heuristics, GAs, TS, SA	Not adaptable to continuous or dynamic job arrivals.
Event-based Rescheduling [18], [19], [20]	Dynamic FJSP with disruptions	Reschedule when event occurs	High computational cost with frequent changes.
Reactive Dispatching [1], [2], [21]	Highly dynamic systems	Local, simple dispatching rules	Myopic; ignores long-term system stability.
RL-based Scheduling [21], [22], [23], [24]	Dynamic FJSP	Single-agent RL for dispatching or sequencing	Requires retraining for each new factory configuration.
GNN-RL Scheduling [13], [25], [26], [27]	Dynamic FJSP, small scale	Graph feature extraction + RL policy learning	Limited scalability and weak generalisation to unseen factories.
MACD (Proposed)	Continuous Dynamic FJSP	Multi-agent RL + GNN + hybrid negotiation	Captures factory-invariant features; generalises across unseen factories.

modernises the job shop problem by introducing continuous job arrivals and dynamic product libraries, which are essential for representing the realities of mass personalised production. Traditional FJSP or DFJSP methods are not directly applicable in this setting, as they lack mechanisms for continuous adaptation and scalability across unseen factory configurations. In contrast, MACD is designed to generalise across such dynamic environments, positioning it as a novel step beyond existing approaches.

The literature on job shop scheduling is extensive, covering various approaches addressing the challenges of static and dynamic manufacturing environments [5]. Table I provides an overview of representative approaches and highlights the key limitations that motivate our proposed MACD framework.

Traditional scheduling methods, such as heuristic methods [17] and genetic algorithms [6], [7], [8], have been widely applied to static scheduling problems. However, these methods often lack the flexibility to adapt to dynamic and continuous problems [9], where system states and job configurations evolve.

To address dynamic scheduling challenges, variants of DFJSP have been explored [20]. These approaches introduced event-based rescheduling [18], [19] and reactive dispatching [21] techniques. Event-based rescheduling works well for infrequent disruptions but becomes computationally impractical for frequent changes. Conversely, reactive dispatching performs well in highly dynamic systems but does not account for the long-term impacts of its decisions. These approaches struggle to support continuous scheduling with varying urgencies and dynamically changing product diversity. These limitations stem from the absence of a problem model that accurately reflects real-world manufacturing challenges [1], [2]. C-DFJSP was hence introduced in [9] to bridge these gaps by explicitly modelling continuous job arrivals and dynamic product libraries. However, the initial work on C-DFJSP lacked scalability and generalisation to diverse factory configurations and machine flexibilities, leaving key challenges unresolved.

Recent advancements in reinforcement learning (RL) [22], [23] have introduced novel strategies and frameworks for dynamic scheduling. RL-based methods [21], [24] have shown promise in improving adaptability and decision-making in real-time manufacturing environments by learning directly

from the environment. However, their scalability and generalisation remain constrained by the need for extensive retraining when applied to new factory configurations.

GNNs have been recently integrated into the RL framework for enhanced feature extraction and state representation [13], [25], [26], [27]. GNNs enable the encoding of relational information between jobs, machines, and operations, improving scalability and adaptability by generalising scheduling methods in dynamic settings. Nevertheless, these prior works often focus on static and small DFJSPs and fail to generalise effectively to unseen factory configurations and continuous scheduling scenarios.

Our proposed method, MACD, builds upon these early insights and address their limitations. Through the novel design of the state and observation space, MACD effectively captures intrinsic factory-invariant characteristics of C-DFJSP. This is achieved using a multi-agent reinforcement learning framework combined with GNNs for enhanced feature extraction. Additionally, by incorporating domain knowledge for machine filtering and employing attention mechanisms, MACD improves computational efficiency and scalability, enabling effective generalisation to diverse and dynamic factory configurations.

III. PROBLEM FORMULATION

This paper addresses C-DFJSP, first proposed in [9] to modernise traditional scheduling problems. In this formulation, a *factory* (see Fig. 1) is modelled as a set of m machines, $\mathcal{M}$, that process job orders arriving dynamically according to a Poisson process. These job orders are drawn from a predefined product library, $\mathcal{L}$, which contains l distinct products. Each job order $J_{j,i} \in \mathcal{J}$ corresponds to the j^{th} order of product $L_i \in \mathcal{L}$.

Each product L_i and the corresponding job order requires processing a sequence of operation steps. The k^{th} operation of the job order $J_{j,i}$, denoted as $O_{k,j} \in \mathcal{O}$, can be performed by any machine M_d belonging to a set of capable machines $\mathcal{M}_{k,i} \subseteq \mathcal{M}$. The processing time for operation $O_{k,j}$ on machine M_d is $p_{k,i}^d$. $\bar{p}_{k,i}$ is the mean processing time for operation $O_{k,i}$ across all capable machines.

In line with prior studies and the queuing theory [28], job order arrivals follow a Poisson distribution, with the arrival rate λ , mean service rate μ , and machine load level $U = \frac{\lambda}{m\mu}$ jointly

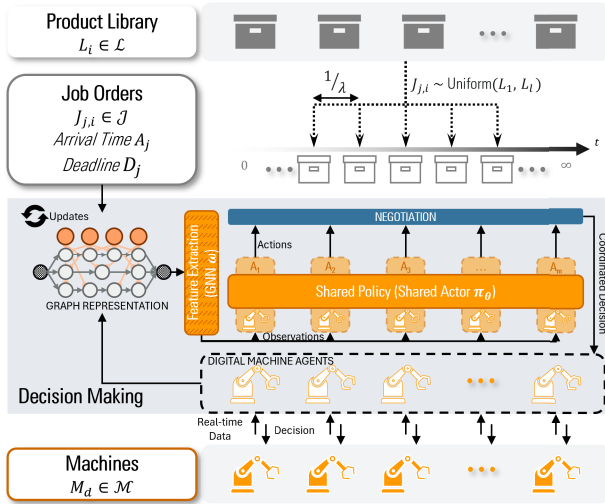


Fig. 1. C-DFJSP problem where Job Orders are continuously generated from a Product Library. In the Decision-Making layer, each machine is represented by a machine agent. For feature extraction, job order and machine information are captured in a graph structure and processed by graph neural networks. Extracted features comprise the input to a shared policy utilised by each machine agent to select a job order. Negotiations then forge a coordinated decision among machine agents, which is subsequently executed by the physical machines.

defining the system utilisation. Unlike many existing studies [13], [26], job order arrivals are modelled as a continuous process that reflects the dynamic nature of real manufacturing systems. Each incoming job order is assigned a fulfilment deadline D_j :

$$D_j = A_j + DDT \times \sum_k \bar{p}_{k,i}, \quad (1)$$

where A_j represents the arrival time of the job order, $DDT \sim U(low, high)$ is the *Due Date Tightness*.

Given the continuous nature of C-DFJSP, scheduling performance is evaluated based on two metrics assessed over consecutive periods of length T . The *primary metric* MT is defined as:

$$MT = \frac{\sum_j \max(0, C_j - D_j)}{N_T}, \quad (2)$$

where N_T denotes the total number of job orders completed during T . C_j and D_j are the completion time and fulfilment deadline of the j^{th} completed job order during T , respectively. The *secondary metric* LJ is defined as:

$$LJ = \sum_{\hat{j}} X_{\hat{j}}, \quad (3)$$

where $X_{\hat{j}} = 1$ if the $\hat{j}^{th}$ uncompleted job order has exceeded its deadline at the end of the current period, else $X_{\hat{j}} = 0$.

MT measures the mean tardiness of *completed* job orders, reflecting periodic system performance. In contrast, LJ tracks the number of *uncompleted* late jobs, offering insights into current delays and trends over consecutive periods. LJ complements MT by providing a measure of system stability and assists in predicting potential future delays resulting from incomplete job orders. Both metrics enable a holistic evaluation of the scheduling performance in dynamic, continuous

manufacturing environments, supported by our experiments reported in Section V.

IV. METHOD

To tackle C-DFJSP, we propose the MACD system to address the challenges introduced by diverse product libraries, varying factory configurations, and dynamically arriving jobs with different urgencies. MACD, as illustrated in Fig. 2, employs a Multi-Agent (MA) framework powered by GNNs and Proximal Policy Optimisation (PPO) [29] for end-to-end training of an adaptive policy. Training is conducted on small-scale static instances, while deployment is extended to factories with arbitrary configurations and dynamic, continuous job orders. Inter-agent conflicts are resolved using a rule-based negotiation strategy during training and execution.

Below, we discuss the technical details of our Multi-Agent Reinforcement Learning Framework, Negotiation Strategy, and GNNs for Feature Extraction.

A. Multi-Agent Reinforcement Learning Framework

MACD models C-DFJSP as a Multi-Agent partially observable Markov decision process (MA-POMDP) denoted by the tuple $(\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega)$, where:

- $\mathcal{M}$ is the set of agents (machines),
- $\mathcal{S}$ is the state space, supported by a heterogeneous graph representation G_t ;
- $\mathcal{A}$ is the action space, where each agent selects a job order operation as its action to process;
- $\mathcal{P}$ defines the state transition probabilities;
- $\mathcal{R}$ is the reward function based on system tardiness; and
- Ω is the observation space for each machine agent.

At any decision step t , agent $M_d \in \mathcal{M}$ receives a partial observation $s_t^d \in \Omega$, selects and performs an action $a_t^d \in \mathcal{A}$, and receives a shared reward $R_t \in \mathcal{R}$. The joint action $\hat{a}_t$ forces the scheduling system to transit from state $s_t \in \mathcal{S}$ to the next state $s_{t+1} \in \mathcal{S}$ with probability $\mathcal{P}$. All agents collaborate to maximise the system's expected average return, $\mathbb{E} \left[\sum_{t \geq 0} (R_t - R(\pi_\theta)) \right]$, where $R(\pi_\theta)$ represents the average total reward obtainable by following policy π_θ [30].

1) *States and Observations*: In MACD, machines, job operations, and their relationships are modelled as a heterogeneous graph G_t , defined as $\mathcal{G} = (\mathcal{O}, \mathcal{M}, \mathcal{C}, \mathcal{E})$, where $\mathcal{O}$ represents the set of operation nodes and $\mathcal{M}$ is the set of machine nodes. Among all operation nodes, some stand for *ready operations*, i.e., operations whose preceding operations in the job orders have been processed. $\mathcal{C}$ is the set of *operation-operation edges* used to model the sequential order over job operations. $\mathcal{E}$ denotes the set of *operation-machine edges*, indicating all machines that can execute any operation in $\mathcal{O}$.

The global state s_t and each machine agent's local observation s_t^d are constructed from three components: (i) machine node features (F_M), (ii) features of operation-machine edges (F_{OM}) linked to ready operations, and (iii) embeddings of ready operation nodes (E_O). Since each component may involve multiple nodes or edges, we apply a pooling operator to obtain a fixed-length representation. Specifically, $\text{pool}(\cdot)$

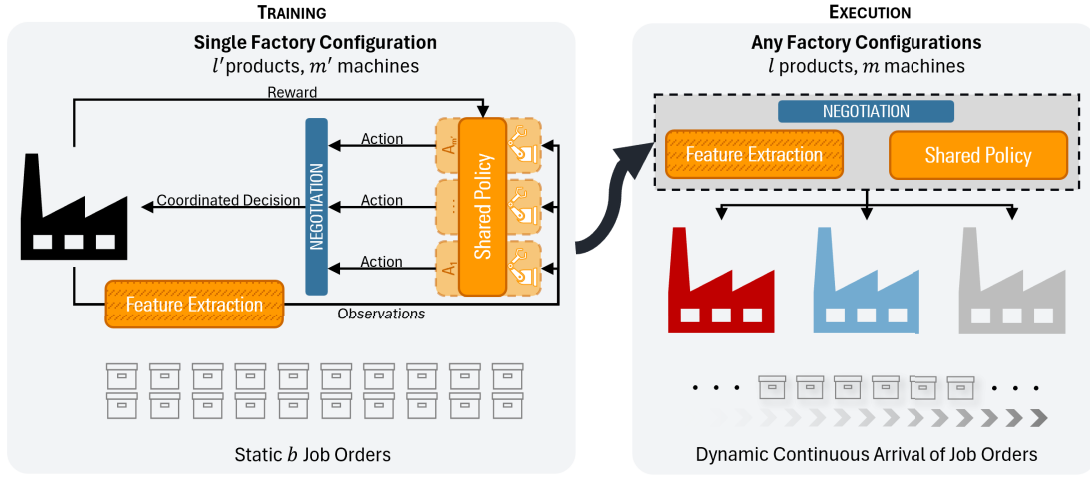


Fig. 2. Overall solution to the C-DFJSP. Training is conducted on *small, static instances* with l' products in the Product Library and m' machines. n job orders are generated from the product library at $t = 0$. Feature Extraction and Shared Policy are trained using RL under a Multi-Agent (MA) framework with a negotiation strategy. The learned Feature Extraction and Shared Policy can be executed in factories with *any* configuration of l products and m machines with the dynamic and continuous arrival of job orders.

TABLE II
GRAPH NODE AND EDGE FEATURES

Node/Edge Type	Feature	Description	Formula	Normalisation Method
Operation Node	Time to Operation Start Time	Defined as the sum of the preceding uncompleted operation processing times	$u_1^{k'} = \sum_{k=0}^{k'-1} \bar{p}_{k,i} \times \mathbf{1}(O_{k,j} \text{ is incomplete})$	$x_{\text{norm}} = \frac{x}{\max(\text{time to completion})}$
	Time to Job Order Completion Time	Defined as the sum of the remaining processing time of the job order	$u_2^{k'} = \sum_{k=0}^{k'-1} \bar{p}_{k,i} \times \mathbf{1}(O_{k,j} \text{ is incomplete})$	
	Time till Job Order Deadline	Defined as the difference between the deadline and current time	$u_3^{k'} = D_j - t$	
	Number of Neighbouring Machines	Defined as the count of the capable machines for the operation	$u_4^{k'} = \deg(O_{k',i})$	
Machine Node	Idle Time	Defined as the difference between the current time and the time that the machine becomes available	$v_1^d = \max(0, t - t_{\text{available}}^d)$	$x_{\text{norm}} = \frac{x - \mu}{\max(x) - \min(x)}$
	Time till Machine Available	Defined as the difference between the time that the machine becomes available and the current time	$v_2^d = \max(0, t_{\text{available}}^d - t)$	
	Number of Neighbouring Operations	Defined as the count of the operations the machine is capable of processing	$v_3^d = \deg(M_d)$	
Operation-Machine Edge	Processing time	Defined as the processing time of machine M_d for operation $O_{k,j}$	$w_1^{d,k}, w_2^{d,k} = p_{k,i}^d$	$x_{\text{norm}} = \frac{x}{\max(\text{time to completion})}$ $x_{\text{norm}} = \frac{x - \mu}{\max(x) - \min(x)}$

denotes an aggregation function (mean pooling [31] in our implementation, though other functions such as max or sum pooling are possible) that compresses the set of features into a single vector. The state and observation are then given by:

$$s_t = \text{pool}(F_M) \oplus \text{pool}(F_{OM}) \oplus \text{pool}(E_O), \quad (4)$$

$$s_t^d = F_M^d \oplus F_{OM}^d \oplus E_O^d \quad (5)$$

where $\oplus$ denotes vector concatenation. In the local observation s_t^d , F_M^d represents the features specific to machine M_d , while E_O^d corresponds to the ready operations that M_d can process.

We normalise the node and edge features (F) (see Table II) of graph G_t to properly describe the system-wide status at each decision step. The design of these features avoids redundancy while capturing essential information about the factory environment. The two normalisation methods, max normalisation and mean normalisation, ensure comparability across different time-related features and machine-specific features.

The embeddings of the ready-operation nodes are extracted using GNNs (see Section IV-C) from G_t to obtain E_O , which is directly combined with the machine-related features F_M

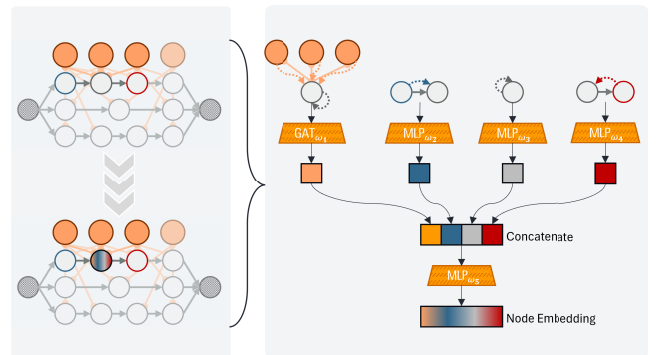


Fig. 3. The embedding process for a single operation node (grey) in the graph on the left is illustrated on the right. Three MLPs process information from preceding (blue), self (grey), and succeeding (red) operation nodes. A GAT processes information from machine nodes (orange). Finally, an MLP integrates the processed information to generate the node embedding.

and F_{OM} to encode global states and local observations (see Fig. 4).

The global state s_t is constructed using mean pooling to aggregate information from all nodes and edges in the

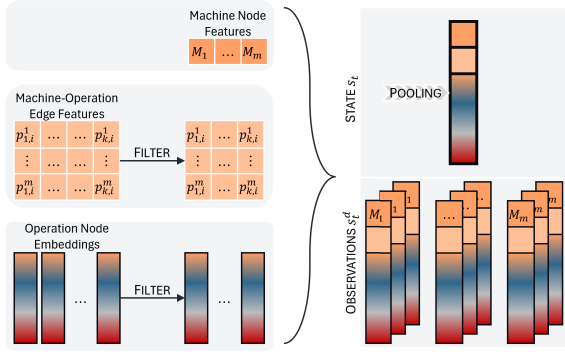


Fig. 4. Operation node embeddings are filtered to correspond to ready-operations. Filtered embeddings are concatenated with machine features to generate observations for each machine agent. Features are pooled to form the system state.

heterogeneous state graph, providing a high-level representation of the entire system. In contrast, the local observation s_t^d focuses on information specific to the agent's machine, ensuring a comprehensive but focused local context for decision-making.

2) *Actions*: At each decision step t , an agent d selects an action a_t^d from feasible job orders $\mathcal{A}_t^d \subseteq \mathcal{A}$. If $\mathcal{A}_t^d = \emptyset$, the agent will choose not to process any new job orders.

3) *Transition*: At each decision step, the execution of joint actions triggers the system to transition to the next state s_{t+1} . Non-selected operation-machine edges are removed when operations are scheduled. Meanwhile, operation nodes corresponding to completed job orders are removed, preventing unbounded graph growth and ensuring efficiency. Node features in G_t are recalculated using the equations shown in Table II,

4) *Reward*: The reward function is designed to minimise the average tardiness across all uncompleted job orders:

$$R_t = -\frac{1}{|\hat{\mathcal{J}}|} \sum_{\hat{j}} \max(0, C_{j,t} - D_j), \quad (6)$$

where $C_{j,t}$ is the estimated completion time of the $\hat{j}^{th}$ uncompleted job order at decision step t . $C_{j,t}$ is estimated by summing the remaining average processing times of all incomplete operations in the job order: $(\sum \bar{p}_{k,i} \times \mathbf{1}(O_{k,i} \text{ is incomplete}))$. Note that Eq. (6) aligns with the MT objective and is used as the training reward. The LJ objective is separately reported as an evaluation metric to capture backlog and stability, as it is less suitable for per-step reward shaping.

The reward design penalises tardiness, which discourages policies from letting backlog accumulate. This mechanism helps avoid degenerate behaviours (e.g., ignoring urgent jobs), and has shown empirical robustness across factory sizes and job arrival rates.

5) *Policy*: The policy $\pi_\theta(a|s)$ parametrised by θ determines the probability distribution over the action space $\mathcal{A}$ based on the current state s . Guided by this distribution, an agent selects a job order to process probabilistically during training and greedily during testing at each decision step.

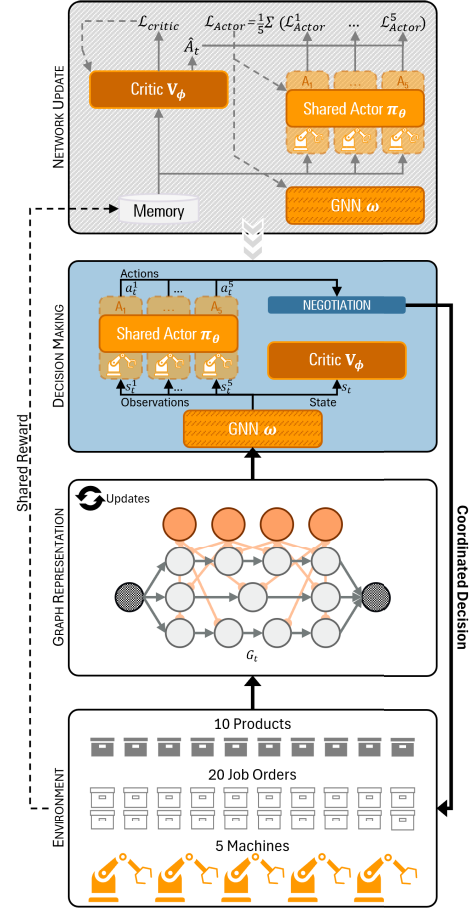


Fig. 5. Training and decision-making framework for learning the Feature Extraction Neural Network (GNN) and the Shared Policy Neural Network (Shared Actor) across all machine agents. The simulated factory is first represented as a graph, which the GNN encodes into agent observations and a global system state. These are passed to the Shared Actor network to select job orders. A negotiation strategy then resolves conflicts to produce the coordinated decision (highlighted box). Bold arrows trace the forward flow from factory to graph to decision-making, while dashed arrows indicate training updates, where the mean actor loss is used to update both the Actor and GNN networks. Training is conducted under static conditions with 10 products, 20 job orders, and 5 machines.

B. Negotiation Strategy

Conflicts arise when multiple agents select the same job order operation to process. Such conflicts are unavoidable in a multi-agent, partially decentralised setting, and when they occur, they must be resolved quickly and fairly to maintain system efficiency. In MACD, these conflicts are handled through a lightweight rule-based negotiation layer rather than being embedded within the MARL policy itself.

This design choice is grounded in practicality. Under typical operating conditions, conflicts occur infrequently (see Fig. 7), and incorporating conflict handling directly into the learning loop would significantly increase coordination overhead while providing limited benefit. Moreover, learning an exclusivity constraint in decentralised MARL typically requires centralised critics or synchronous communication, which contradicts MACD's scalability objectives. By resolving conflicts externally, MACD preserves stable convergence, avoids unnecessary non-stationarity, and maintains decentralised execution.

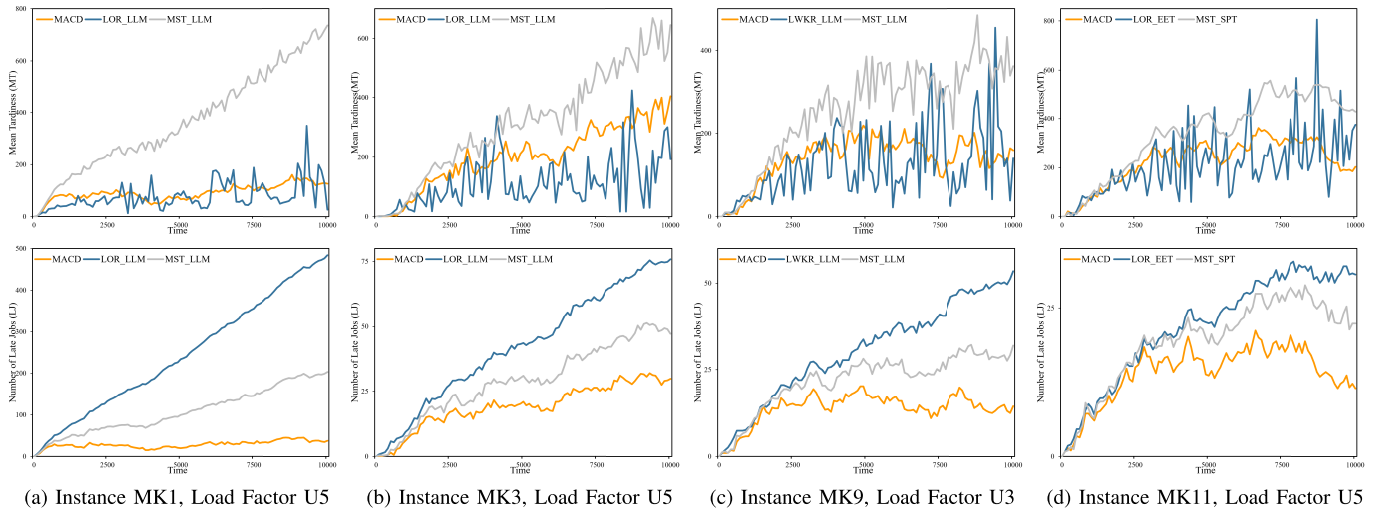


Fig. 6. Selection of representative instances showing the two objectives MT and LJ over 100 periods. MT results are shown in the top row, and LJ results in the bottom row.

TABLE III

AVERAGE MEAN TARDINESS (OBJ. 1 MT) AND AVERAGE LATE NUMBER OF JOBS (OBJ. 2 LJ) FOR BRANDIMARTE MK1-MK7 UNDER DIFFERENT LOAD LEVELS (U1 - U5) WHERE MACD IS THE DEVELOPED METHOD, MACS IS THE METHOD DEVELOPED IN [9], BEST MT IS THE BEST PERFORMING HEURISTIC UNDER THE MT METRIC, AND BEST LJ IS THE BEST PERFORMING HEURISTIC UNDER THE LJ METRIC

10x6 MK01 MK02					15x8 MK03 MK04					15x4 MK05					10x15 MK06					20x5 MK07				
	MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ
U1	Obj 1. MT	2.07	3.59	4.11	4.11	4.26	4.66	5.01	5.01	15.96	17.99	19.99	20.61	0.21	0.38	0.19	0.19	2.38	3.39	3.48	3.59			
	Rel. Gain	0.0%	72.9%	98.4%	98.4%	0.0%	9.5%	17.6%	17.6%	0.0%	12.7%	25.3%	29.1%	8.1%	98.9%	0.0%	0.0%	0.0%	42.1%	46.1%	50.6%			
	Obj 2. LJ	1.17	1.62	1.75	1.75	1.19	1.26	1.32	1.32	2.30	2.43	2.63	2.63	0.31	0.39	0.29	0.29	0.66	0.79	0.82	0.80			
U2	Rel. Gain	0.0%	38.9%	49.8%	49.8%	0.0%	5.7%	11.1%	11.1%	0.0%	5.6%	14.3%	14.1%	7.7%	37.8%	0.0%	0.0%	0.0%	20.1%	24.4%	22.3%			
	Obj 1. MT	4.01	7.89	8.82	8.82	8.41	10.09	11.90	11.90	24.28	30.52	33.69	33.69	0.27	0.50	0.36	0.36	3.05	5.95	5.41	5.41			
	Rel. Gain	0.0%	96.8%	119.9%	119.9%	0.0%	20.0%	41.5%	41.5%	0.0%	25.7%	38.8%	38.8%	0.0%	87.6%	34.4%	34.4%	0.0%	95.1%	77.2%	77.2%			
U3	Obj 2. LJ	2.10	3.02	3.26	3.26	2.02	2.30	2.53	2.53	3.40	3.86	4.16	4.16	0.35	0.49	0.41	0.41	0.79	1.23	1.14	1.14			
	Rel. Gain	0.0%	43.7%	55.3%	55.3%	0.0%	13.9%	25.5%	25.5%	0.0%	13.5%	22.4%	22.4%	0.0%	40.2%	19.0%	19.0%	0.0%	56.6%	45.2%	45.2%			
	Obj 1. MT	8.12	16.05	23.13	23.13	19.75	25.36	43.68	43.68	40.75	59.45	64.44	64.44	0.36	1.15	0.72	0.72	4.00	9.44	9.86	10.46			
U4	Rel. Gain	0.0%	97.6%	184.8%	184.8%	0.0%	28.4%	121.1%	121.1%	0.0%	45.9%	58.1%	58.1%	0.0%	215.6%	98.6%	98.6%	0.0%	136.0%	146.6%	161.6%			
	Obj 2. LJ	3.43	5.79	7.49	7.49	3.74	4.69	6.55	6.55	5.28	6.88	7.25	7.25	0.42	0.78	0.53	0.53	1.10	1.78	1.95	1.92			
	Rel. Gain	0.0%	68.8%	118.4%	118.4%	0.0%	25.3%	75.0%	75.0%	0.0%	30.4%	37.3%	37.3%	0.0%	86.2%	26.7%	26.7%	0.0%	62.8%	78.1%	74.8%			
U5	Obj 1. MT	17.56	35.06	41.91	93.18	52.84	77.71	77.69	107.84	68.90	126.68	125.18	126.54	0.37	4.21	3.50	3.50	5.99	20.65	20.45	20.45			
	Rel. Gain	0.0%	99.6%	138.6%	430.6%	0.0%	47.1%	47.0%	104.1%	0.0%	83.0%	81.7%	83.7%	0.0%	1047.0%	853.7%	853.7%	0.0%	244.7%	241.4%	241.4%			
	Obj 2. LJ	6.51	11.93	123.97	26.92	8.44	11.85	21.81	14.71	8.17	12.92	29.58	13.00	0.48	1.94	1.68	1.68	1.53	3.56	3.46	3.46			
U6	Rel. Gain	0.0%	83.1%	1803.4%	313.3%	0.0%	40.4%	158.5%	74.3%	0.0%	58.2%	262.3%	59.2%	0.0%	307.6%	252.1%	252.1%	0.0%	133.0%	126.7%	126.7%			
	Obj 1. MT	47.89	118.18	58.10	204.00	131.61	170.35	87.30	234.10	142.19	268.52	151.46	284.58	0.56	59.20	47.67	47.67	7.89	75.56	57.11	57.11			
	Rel. Gain	0.0%	146.8%	21.3%	326.0%	50.8%	95.1%	0.0%	168.2%	0.0%	88.8%	6.5%	100.1%	0.0%	10497.9%	8433.0%	8433.0%	0.0%	857.2%	623.4%	623.4%			
U7	Obj 2. LJ	15.81	38.37	130.46	61.79	18.91	26.68	113.71	34.78	15.27	26.88	45.43	27.04	0.61	13.10	11.15	11.15	1.91	10.23	8.27	8.27			
	Rel. Gain	0.0%	142.6%	725.0%	290.7%	0.0%	41.1%	501.3%	83.9%	0.0%	76.0%	197.5%	77.1%	0.0%	2047.9%	1728.5%	1728.5%	0.0%	434.5%	332.1%	332.1%			

TABLE IV

AVERAGE MEAN TARDINESS (OBJ. 1 MT) AND AVERAGE LATE NUMBER OF JOBS (OBJ. 2 LJ) FOR BRANDIMARTE MK8-MK15 UNDER DIFFERENT LOAD LEVELS (U1 - U5) WHERE MACD IS THE DEVELOPED METHOD, MACS IS THE METHOD DEVELOPED IN [9], BEST MT IS THE BEST PERFORMING HEURISTIC UNDER THE MT METRIC, AND BEST LJ IS THE BEST PERFORMING HEURISTIC UNDER THE LJ METRIC

20x10 MK08 MK09					20x15 MK10					30x5 MK11					30x10 MK12 MK13					30x15 MK14 MK15				
	MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ		MACD	MACS	Best MT	Best LJ
U1	Obj 1. MT	26.70	29.58	34.89	34.89	18.78	75.69	103.30	103.30	32.93	34.97	41.22	41.22	16.52	16.42	17.98	17.98	11.56	12.00	14.01	14.03			
	Rel. Gain	0.0%	10.8%	30.7%	30.7%	0.0%	303.0%	450.0%	450.0%	0.0%	6.2%	25.2%	25.2%	0.6%	0.0%	9.5%	9.5%	0.0%	3.8%	21.2%	21.3%			
	Obj 2. LJ	3.02	3.14	3.68	3.68	4.47	11.13	14.10	14.10	2.68	2.80	3.08	3.08	1.85	1.90	1.96	1.96	1.76	1.85	2.03	1.97			
U2	Rel. Gain	0.0%	3.7%	21.7%	21.7%	0.0%	149.0%	215.4%	215.4%	0.0%	4.6%	15.1%	15.1%	0.0%	2.6%	5.7%	5.7%	0.0%	5.3%	15.4%	12.1%			
	Obj 1. MT	47.58	59.43	74.48	78.96	47.65	281.12	197.65	330.62	48.07	57.26	65.29	65.53	25.42	25.05	31.33	31.33	16.83	17.86	23.43	23.63			
	Rel. Gain	0.0%	24.9%	56.6%	66.0%	0.0%	490.0%	314.8%	593.8%	0.0%	19.1%	35.8%	36.3%	1.5%	0.0%	25.0%	25.0%	0.0%	6.1%	39.3%	40.4%			
U3	Obj 2. LJ	4.88	5.62	10.76	7.08	9.56	32.61	47.93	36.84	3.80	4.10	4.72	4.68	2.69	2.68	3.04	3.04	2.51	2.65	3.23	3.21			
	Rel. Gain	0.0%	15.3%	120.6%	45.1%	0.0%	241.0%	401.2%	285.2%	0.0%	8.0%	24.4%	23.3%	0.3%	0.0%	13.1%	13.1%	0.0%	5.5%	28.7%	28.1%			
	Obj 1. MT	117.33	136.27	106.69	173.87	130.83	493.55	229.80	602.02	74.21	97.49	111.99	111.99	49.03	51.30	66.90	68.57	31.27	36.28	53.10	53.10			
U4	Rel. Gain	10.0%	27.7%	0.0%	63.0%	0.0%	277.2%	75.6%	360.2%	0.0%	31.4%	50.9%	50.9%	0.0%	4.6%	36.5%	39.9%	0.0%	16.0%	69.8%	69.8%			
	Obj 2. LJ	9.78	11.06	18.04	13.67	20.70	60.75	74.77	64.70	5.53	6.55	7.34	7.34	4.67	4.81	5.81	5.74	3.95	4.48	5.86	5.86			
	Rel. Gain	0.0%	13.0%	84.3%	39.7%	0.0%	193.5%	261.3%	212.6%	0.0%	18.6%	32.7%	32.7%	0.0%	3.1%	24.5%	23.0%	0.0%	13.6%	48.4%	48.4%			
U5	Obj 1. MT	271.74	296.42	185.10	323.43	264.86	623.42	263.86	860.39	129.72	182.68	176.46	210.25	105.59	116.46	131.68	137.04	75.02	91.23	147.84	150.29			
	Rel. Gain	46.8%	60.1%	0.0%	74.7%	0.4%	136.3%	0.0%	226.1%	0.0%	40.8%	36.0%	62.1%	0.0%	10.3%	24.7%	29.8%	0.0%	21.6%	97.1%	100.3%			
	Obj 2. LJ	19.18	21.00	27.31	24.06	36.27	100.09	93.82	93.37	8.73	11.19	14.74	12.44	8.76	9.55	12.76	11.48	7.47	9.16	15.57	13.38			
U6	Rel. Gain	0.0%	9.4%	42.3%	25.4%	0.0%	175.9%	158.7%	157.4%	0.0%	28.2%	68.9%	42.5%	0.0%	9.0%	45.7%	31.0%	0.0%	22.6%	108.3%	79.1%			
	Obj 1. MT	501.21	489.53	224.04	434.92	418.48	613.11	250.26	250.26	228.23	301.95	213.33	342.99	216.62	228.25	157.10	157.62	169.11	230.55	185.94	317.73			
	Rel. Gain	123.7%	118.5%	0.0%	94.1%	67.2%	145.0%	0.0%	0.0%	7.0%	41.5%	60.8%	60.8%	0.0%	37.9%	45.3%	0.0%	0.3%	36.3%	9.9%	87.9%			
U7	Obj 2. LJ	33.22	35.32	38.59	36.38	54.20	144.46	119.04	119.04	14.15	17.61	22.03	19.53	16.90	18.96	26.68	20.29	14.51	19.72	40.11	25.51			
	Rel. Gain	0.0%	6.3%	16.2%	9.5%	0.0%	166.5%	119.6%	119.6%	0.0%	24.4%	55.7%	38.0%	0.0%	12.2%	57.8%	20.1%	0.0%	35.9%	176.4%	75.8%			

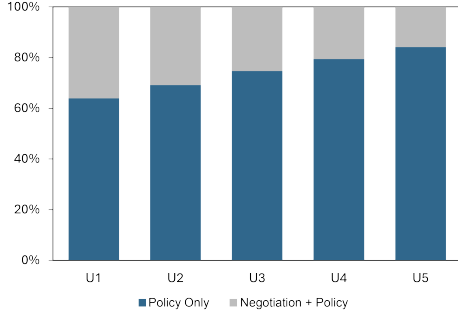


Fig. 7. Proportion of Policy Only decisions compared with decisions supported by the Negotiation Strategy across different machine utilisation levels. The results show that as utilisation increases, a larger share of decisions are made solely by the learned policy.

TABLE V
KEY HYPERPARAMETERS USED IN TRAINING
AND EVALUATION

Category	Value
Number of machines	5
Number of jobs	10
Hidden size (operation encoder)	128
Latent dimension (actor/critic)	64
Hidden layers (actor/critic)	3
Attention heads	1
Learning rate	0.0005
PPO epochs (K)	5
Clip range (ϵ)	0.1
Entropy coefficient	0.01
Minibatch size	1024

TABLE VI
AVERAGE MEAN TARDINESS (MT) AND AVERAGE LATE NUMBER OF JOBS
(LJ) FOR HURINK 5 MACHINES

		E			R			V		
		MACD	Best MT	Best LJ	MACD	Best MT	Best LJ	MACD	Best MT	Best LJ
U1	Obj 1. MT	41.79	42.16	42.54	44.24	47.58	47.77	30.23	32.03	32.35
	Rel. Gain	0.0%	0.9%	1.8%	0.0%	7.5%	8.0%	0.0%	6.0%	7.0%
	Obj 2. LJ	0.98	1.02	1.00	1.42	1.49	1.48	1.09	1.11	1.10
U2	Rel. Gain	0.0%	3.5%	1.2%	0.0%	4.9%	4.3%	0.0%	2.2%	1.1%
	Obj 1. MT	49.98	52.28	52.46	56.14	60.82	60.87	37.77	39.61	40.08
	Rel. Gain	0.0%	4.6%	5.0%	0.0%	8.3%	8.4%	0.0%	4.9%	6.1%
U3	Obj 2. LJ	1.27	1.28	1.27	1.88	1.96	1.93	1.45	1.46	1.44
	Rel. Gain	0.0%	1.4%	0.1%	0.0%	4.7%	2.6%	0.6%	1.5%	0.0%
	Obj 1. MT	64.92	65.82	66.14	74.60	81.73	81.81	49.87	52.35	52.77
U4	Rel. Gain	0.0%	1.4%	1.9%	0.0%	9.6%	9.7%	0.0%	5.0%	5.8%
	Obj 2. LJ	1.66	1.69	1.68	2.46	2.60	2.58	1.84	1.90	1.89
	Rel. Gain	0.0%	1.7%	1.2%	0.0%	5.6%	4.8%	0.0%	3.4%	3.1%
U5	Obj 1. MT	84.72	89.35	89.83	106.60	117.72	117.72	69.95	74.04	74.28
	Rel. Gain	0.0%	5.5%	6.0%	0.0%	10.4%	10.4%	0.0%	5.9%	6.2%
	Obj 2. LJ	2.28	2.34	2.34	3.51	3.70	3.70	2.60	2.64	2.62
U5	Rel. Gain	0.0%	2.9%	2.5%	0.0%	5.6%	5.6%	0.0%	1.4%	0.7%
	Obj 1. MT	117.90	127.41	127.95	172.93	197.09	199.14	118.29	122.26	122.26
	Rel. Gain	0.0%	8.1%	8.5%	0.0%	14.0%	15.2%	0.0%	3.4%	3.4%
U5	Obj 2. LJ	3.04	3.18	3.16	5.30	5.76	5.76	4.01	4.00	4.00
	Rel. Gain	0.0%	4.7%	4.1%	0.0%	8.8%	8.7%	0.2%	0.0%	0.0%

Empirical analysis (Table IX) shows that the Shortest Processing Time (SPT) rule consistently offers the best performance across most flexibility levels, while the Least Loaded Machine (LLM) rule performs better in low-flexibility settings. This aligns with domain knowledge: faster machines minimise tardiness under typical conditions, whereas load balancing becomes more important when few machines are eligible. The hybrid SPT-LLM negotiation strategy adopted in MACD directly reflects these observations and its implementation is shown in Algorithm 1.

TABLE VII
AVERAGE MEAN TARDINESS (MT) AND AVERAGE LATE NUMBER OF JOBS
(LJ) FOR HURINK 10 MACHINES

		E			R			V		
		MACD	Best MT	Best LJ	MACD	Best MT	Best LJ	MACD	Best MT	Best LJ
U1	Obj 1. MT	73.78	73.60	73.82	70.13	71.89	72.29	31.90	31.96	32.27
	Rel. Gain	0.3%	0.0%	0.3%	0.0%	2.5%	3.1%	0.0%	0.2%	1.2%
	Obj 2. LJ	1.82	1.81	1.80	2.53	2.58	2.56	1.48	1.49	1.47
U2	Rel. Gain	1.0%	0.3%	0.0%	0.0%	1.9%	1.3%	0.4%	1.1%	0.0%
	Obj 1. MT	87.19	88.18	88.59	84.24	88.04	88.23	33.73	33.75	34.01
	Rel. Gain	0.0%	1.1%	1.6%	0.0%	4.5%	4.7%	0.0%	0.1%	0.8%
U3	Obj 2. LJ	2.27	2.31	2.27	3.06	3.14	3.12	1.54	1.55	1.54
	Rel. Gain	0.0%	2.0%	0.3%	0.0%	2.3%	2.0%	0.0%	0.8%	0.0%
	Obj 1. MT	111.11	111.17	111.25	107.62	117.59	117.94	35.78	35.87	36.30
U4	Rel. Gain	0.0%	0.1%	0.1%	0.0%	9.3%	9.6%	0.0%	0.3%	1.5%
	Obj 2. LJ	3.07	3.04	3.01	4.09	4.30	4.27	1.80	1.79	1.79
	Rel. Gain	1.8%	0.8%	0.0%	0.0%	5.2%	4.4%	0.8%	0.4%	0.0%
U5	Obj 1. MT	151.01	155.04	155.93	154.08	175.90	176.05	39.74	39.85	40.40
	Rel. Gain	0.0%	2.7%	3.3%	0.0%	14.2%	14.3%	0.0%	0.3%	1.6%
	Obj 2. LJ	4.24	4.28	4.27	5.67	6.10	6.07	2.02	2.01	2.00
U5	Rel. Gain	0.0%	1.0%	0.6%	0.0%	7.6%	7.0%	0.9%	0.5%	0.0%
	Obj 1. MT	222.35	233.13	234.93	252.94	302.48	302.91	50.92	50.78	51.14
	Rel. Gain	0.0%	4.8%	5.7%	0.0%	19.6%	19.8%	0.3%	0.0%	0.7%
U5	Obj 2. LJ	6.14	6.38	6.28	8.81	9.77	9.75	2.74	2.70	2.70
	Rel. Gain	0.0%	3.9%	2.4%	0.0%	10.9%	10.6%	1.8%	0.2%	0.0%

TABLE VIII
AVERAGE MEAN TARDINESS (MT) AND AVERAGE LATE NUMBER OF JOBS
(LJ) FOR HURINK 15 MACHINES

		E			R			V		
		MACD	Best MT	Best LJ	MACD	Best MT	Best LJ	MACD	Best MT	Best LJ
U1	Obj 1. MT	108.99	108.25	108.25	100.79	103.39	104.36	42.64	42.57	42.60
	Rel. Gain	0.7%	0.0%	0.0%	0.0%	2.6%	3.5%	0.2%	0.0%	0.1%
	Obj 2. LJ	2.73	2.69	2.69	3.63	3.77	3.66	2.11	2.12	2.09
U2	Rel. Gain	1.6%	0.0%	0.0%	0.0%	3.6%	0.7%	1.1%	1.5%	0.0%
	Obj 1. MT	126.39	123.89	123.89	121.28	124.41	124.41	44.29	44.27	45.14
	Rel. Gain	2.0%	0.0%	0.0%	0.0%	2.6%	2.6%	0.1%	0.0%	2.0%
U3	Obj 2. LJ	3.35	3.28	3.28	4.53	4.52	4.52	2.16	2.18	2.16
	Rel. Gain	2.3%	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%	1.0%	0.0%
	Obj 1. MT	157.35	160.84	160.84	160.25	164.89	164.89	45.53	45.45	45.45
U4	Rel. Gain	0.0%	2.2%	2.2%	0.0%	2.9%	2.9%	0.2%	0.0%	0.0%
	Obj 2. LJ	4.37	4.45	4.45	6.11	6.08	6.08	2.41	2.38	2.38
	Rel. Gain	0.0%	1.8%	1.8%	0.0%	0.5%	0.0%	1.1%	0.0%	0.0%
U5	Obj 1. MT	234.34	233.05	233.05	235.64	270.90	270.90	47.82	47.51	47.59
	Rel. Gain	0.6%	0.0%	0.0%	0.0%	15.0%	15.0%	0.6%	0.0%	0.2%
	Obj 2. LJ	6.65	6.56	6.56	8.89	9.67	9.67	2.61	2.61	2.60
U5	Rel. Gain	1.2%	0.0%	0.0%	0.0%	8.8%	8.8%	0.5%	0.4%	0.0%
	Obj 1. MT	351.29	361.07	361.07	418.40	530.18	530.18	51.87	50.85	50.85
	Rel. Gain	0.0%	2.8%	2.8%	0.0%	26.7%	26.7%	2.0%	0.0%	0.0%
U5	Obj 2. LJ	9.83	9.79	9.79	14.47	16.41	16.41	2.96	2.88	2.88
	Rel. Gain	0.4%	0.0%	0.0%	0.0%	13.4%	13.4%	2.8%	0.0%	0.0%

TABLE IX
AVERAGE MEAN TARDINESS (MT) AND AVERAGE LATE NUMBER OF JOBS
(LJ) FOR NEGOTIATION STRATEGY LPT, SPT, LLM, AND EET

		F2				F3				F4				F5				F1-5			
		LPT	SPT	LLM	EET	LPT	SPT	LLM	EET	LPT	SPT	LLM	EET	LPT	SPT	LLM	EET	LPT	SPT	LLM	EET
U1	Obj 1. MT	14.73	14.93	13.58	14.06	2.72	2.21	2.51	2.50	2.61	1.89	2.28	2.28	1.78	1.17	1.50	1.51	5.15	4.38	4.69	4.74
	Rel. Gain	8.5%	9.9%	0.0%	3.6%	23.2%	0.0%	13.4%	12.9%	38.2%	0.0%	20.6%	20.9%	52.0%	0.0%	28.7%	29.1%	17.6%	0.0%	7.2%	8.3%
	Obj 2. LJ	2.78	2.81	2.60	2.65	0.80	0.75	0.77	0.68	0.61	0.66	0.64	0.62	0.49	0.56	0.57	1.06	0.92	0.99	0.99	
U2	Rel. Gain	7.0%	8.0%	0.0%	2.1%	13.1%	0.0%	6.5%	8.8%	10.1%	0.0%	7.8%	4.9%	26.0%	0.0%	14.2%	16.3%	4.7%	0.0%	7.4%	6.9%
	Obj 1. MT	29.06	28.68	27.39	28.85	3.27	2.83	3.04	3.07	3.29	2.62	2.92	2.91	2.11	1.44	1.74	1.76	7.05	5.91	6.27	6.26
	Rel. Gain	6.1%	4.7%	0.0%	5.3%	15.8%	0.0%	7.7%	8.7%	25.5%	0.0%	11.5%	11.2%	46.6%	0.0%	20.7%	22.7%	19.3%	0.0%	6.1%	5.9%
U3	Obj 2. LJ	5.06	4.97	4.80	4.98	0.95	0.91	0.97	0.95	0.85	0.72	0.82	0.79	0.69	0.56	0.64	0.63	1.60	1.36	1.41	1.47
	Rel. Gain	5.3%	3.4%	0.0%	3.6%	4.2%	0.0%	6.4%	4.6%	17.7%	0.0%	12.7%	8.6%	24.5%	0.0%	14.4%	13.3%	18.3%	0.0%	4.1%	8.7%
	Obj 1. MT	71.60	76.23	67.02	73.34	4.64	4.04	4.34	4.40	4.62	3.80	4.21	4.28	2.99	2.18	2.59	2.58	9.06	9.00	9.35	9.58
U4	Rel. Gain	6.8%	13.7%	0.0%	9.4%	14.8%	0.0%	7.4%	8.9%	21.6%	0.0%	10.8%	12.7%	37.4%	0.0%	18.7%	18.4%	9.6%	0.0%	5.1%	5.5%
	Obj 2. LJ	10.74	11.28	10.17	10.94	1.41	1.27	1.32	1.35	1.22	1.09	1.13	1.15	1.08	0.91	1.00	0.99	2.11	1.99	2.05	2.04
	Rel. Gain	5.5%	10.8%	0.0%	7.5%	10.9%	0.0%	3.9%	6.2%	11.9%	0.0%	3.3%	5.9%	19.2%	0.0%	10.6%	8.6%	6.1%	0.0%	3.1%	2.2%
U5	Obj 1. MT	221.04	227.92	192.73	200.24	7.73	7.88	7.26	7.45	7.33	6.11	6.81	6.82	4.77	3.64	4.26	4.17	16.96	15.39	16.07	15.97
	Rel. Gain	14.7%	18.3%	0.0%	3.9%	9.2%	0.0%	2.5%	3.3%	19.9%	0.0%	11.6%	11.6%	30.9%	0.0%	17.0%	14.6%	10.2%	0.0%	4.4%	3.8%
	Obj 2. LJ	30.23	31.11	26.78	27.68	2.13	2.00	2.03	2.07	1.82	1.58	1.72	1.71	1.56	1.30	1.45	1.41	3.47	3.21	3.30	3.32
U6	Rel. Gain	12.9%	16.2%	0.0%	3.4%	6.7%	0.0%	1.7%	3.5%	15.1%	0.0%	8.6%	8.1%	20.5%	0.0%	11.7%	8.9%	8.1%	0.0%	2.9%	3.5%
	Obj 1. MT	598.37	613.34	567.81	584.86	13.78	12.59	13.04	13.55	14.87	13.03	13.83	13.81	8.90	7.31	8.12	8.25	34.97	34.71	33.26	33.59
	Rel. Gain	8.4%	10.0%	0.0%	4.8%	10.0%	0.0%	2.3%	3.4%	13.7%	0.0%	4.2%	4.3%	14.4%	0.0%	10.0%	9.4%	7.4%	0.0%	3.4%	3.4%
U7	Obj 1. MT	80.68	82.82	76.82	78.90	3.52	3.27	3.37	3.42	3.42	3.05	3.18	3.23	2.64	2.25	2.41	2.47	6.38	5.92	6.12	6.20
	Rel. Gain	5.2%	8.0%	0.0%	2.8%	7.8%	0.0%	1.8%	1.6%	13.0%	0.0%	4.2%	5.9%	17.3%	0.0%	7.2%	9.6%	7.7%	0.0%	3.4%	4.7%

Algorithm 1 Negotiation Strategy

```

while conflicting job orders exist do
  for each conflicting job order  $j^*$  do
     $C \leftarrow$  conflicting agents
    if  $|C| \leq 2$  then
      Prioritise using LLM; break ties with SPT
    else
      Prioritise using SPT; break ties with LLM
    Assign  $j^*$  to the highest priority agent
    Store losing agents in list  $B$ 
    Mask  $j^*$  to avoid further selection
  for each  $d \in B$  do
    Select new job order using policy  $\pi_\theta(a|s)$ 
  Check conflicting job orders exist

```

(MLPs) handle heterogeneous information from the preceding, succeeding, and current operation nodes.

Neighbouring machines, which are capable of processing a given operation, hold essential information in their features. A Graph Attention Network (GAT) processes these features, assigning attention coefficients to prioritize the most relevant machines. However, performing GAT operations on all neighbouring machines can be computationally expensive. To manage this overhead, we use the *neighbour sampling technique* [31] to select up to x machine nodes before applying the GAT operation. Rather than random sampling, we found empirically (see Section V-G) that prioritising machines with faster processing times can often reduce tardiness. Based on this, we identify the x fastest machines for each operation, ensuring accurate prioritisation. In cases where multiple machines can process an operation at the same speed, the machine with the longest idle time is favoured to improve load balancing among all machines. This method ensures effective prioritisation, improving MACD's generalisability and scalability across factory configurations.

D. Training

The overall training process is illustrated in Fig. 5 and Algorithm 2. Training is conducted on batches of small, fixed-size, static FJSP instances. To address the computational challenges of training directly in a dynamic setting, we transform the dynamic system into a sequence of static problems. Each static problem presents a snapshot of the dynamic system at a specific moment, reducing computational demands and enabling efficient training on representative static scenarios. The batch of instances is updated periodically in two ways: *product library update* and *job orders update*. PPO periodically trains the critic and actor neural networks in a centralised manner. For each machine agent, individual actor loss $\mathcal{L}_{Actor}^d$ is calculated, and the average is taken to determine the overall actor loss $\mathcal{L}_{Actor}$. The critic network leverages global state information, while the actor network utilises local observations. $\mathcal{L}_{Actor}$ is used to jointly train the shared actor network and the GNN for learning state embeddings. This ensures that the GNN produces embeddings optimised for the actor's policy, enabling end-to-end learning.

V. EXPERIMENTS

We design experiments to evaluate the performance and generalisability of the proposed MACD approach for solving

Algorithm 2 Training Algorithm Using PPO

```

Initialise GNN, Actor and Critic networks with parameters  $\omega, \theta, \phi$ 
for iteration  $iter = 1, 2, \dots, \mathcal{I}$  do
  for Instance  $b = 1, 2, \dots, \mathcal{B}$  do
    while not done do
      Extract embeddings using GNN
      for each  $M_d \in \mathcal{M}$  do Sample  $a_t^d \sim \pi_\theta(\cdot|s)$ 
      Resolve conflicts using Algorithm 1
      Execute actions,  $s_t \leftarrow s_{t+1}$ , Get reward  $R_t$ 

    if  $iter \% \text{update\_library} = 0$  then Update library  $\mathcal{L}$ 
    if  $iter \% \text{update\_job\_orders} = 0$  then Update selection of job orders
    if  $iter \% \text{update\_networks} = 0$  then
      for each training epoch in  $K$  do
        for each minibatch do
          Compute generalised advantage estimates
           $\hat{A}_t = \sum_{k=0}^{\infty} (\gamma \lambda)^k \delta_t^{(k)}$ ,  $\delta_t^{(k)} = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ 
          Compute Critic loss
           $\mathcal{L}_{Critic} = \frac{1}{|\mathcal{B}|} \sum_t (V_\phi(s_t) - G_t)^2$ 
          for each  $M_d \in \mathcal{M}$  do
            Compute Actor loss  $\mathcal{L}_{Actor}^d = \mathcal{L}_{Actor}^d = \mathbb{E}_t [\min(r_t \hat{A}_t, \text{clip}(r_t, 1 - \epsilon, 1 + \epsilon) \hat{A}_t)]$ 
           $\mathcal{L}_{Actor} = \frac{1}{m} \sum_d \mathcal{L}_{Actor}^d$ 
          Using  $\mathcal{L}_{Critic}$  update  $\phi$ 
          Using  $\mathcal{L}_{Actor}$  update  $\omega$  and  $\theta$ 

```

C-DFJSP. A shared decision-making policy network, introduced in Section IV, is first trained on static FJSP problem instances before being applied and evaluated on unseen dynamic problem instances.

In the following subsections, we describe the training and validation settings (see Table V for hyperparameters), followed by the testing approach, and outline the competing methods. We conclude this section by reporting our experiment results and associated analysis.

A. Training and Validation Settings

Following existing works [9], synthetic training instances were generated using a product library of size $l = 10$ and $m = 5$ machines. The training was conducted on batches of 10 instances where each instance contains $n = 20$ job orders randomly selected from the product library, with due dates determined by the tightness factor $DDT \sim U(1.0, 3.0)$. To align with the training setting, up to $x = 5$ machines are filtered prior to the GAT operation. Robust training was ensured through periodic evaluations conducted on 100 validation instances (larger than the training batch size to ensure the model is not overfitting to the training instances) generated under identical settings. We keep track of the best-performing model on the validation set for testing.

B. Testing Framework

The trained policy is evaluated on publicly available FJSP instances, including Brandimarte [32] and Hurink [33], under the C-DFJSP setting. The metrics MT and LJ were recorded periodically over extended simulations to quantify the system's stability and adaptability.

Two datasets are used to test generalisability:

- **Brandimarte Dataset (MK01-MK15):** Cover mixed flexibility and product, machine, and operation characteristics variations.

- **Hurink Dataset (E, R, V, 19-43):** Provide multiple levels of flexibility and introduce challenges such as uniform processing times across capable machines, making it difficult to differentiate between machines and select the most suitable one.

These datasets were selected because of their widespread use in FJSP research and their ability to test adaptability across diverse factory configurations. These FJSP instances formed the product libraries for the C-DFJSP problem, allowing job orders to be continuously generated from this library.

C. Competing Approaches

A combination of widely used job order selection and machine selection heuristics serve as competing approaches. The job order rules [13] include First In First Out (FIFO), Earliest Due Date (EDD), Critical Ratio (CR), Mean Slack Time (MST), Least Operations Remaining (LOR), and Least Work Remaining (LWKR). The machine selection rules [13] include Longest Processing Time (LPT), Least Loaded Machine (LLM), Earliest End Time (EET), and Shortest Processing Time (SPT). Particular attention is given to heuristics targeting tardiness, such as EDD, CR, and MST.

Additionally, we compare against MAC-Sched [9], a recently proposed deep reinforcement learning framework and the first attempt to solve the C-DFJSP. This provides a direct DRL baseline for assessing the effectiveness and generalisation of MACD.

It is important to note that while metaheuristic methods such as Genetic Algorithms, Simulated Annealing, or Tabu Search have demonstrated success in static or finite-horizon FJSPs, they are not practical for the continuous setting considered here. In C-DFJSP, job orders arrive continuously and factory states change at each step; metaheuristics would therefore require full re-optimisation after every arrival, resulting in prohibitive computational overhead. Prior studies on dynamic scheduling (e.g., [13], [27]) similarly rely on heuristic or learning-based baselines rather than metaheuristics for this reason.

Furthermore, unlike classical static FJSPs, the C-DFJSP problem does not have a well-defined theoretical lower bound or global optimum. This is due to its continuous, open-ended nature: job arrivals are not known in advance, machine states evolve asynchronously, and scheduling decisions must be made in real time under partial observability.

Accordingly, our comparisons include:

- A broad set of heuristic dispatching rules, which remain the most widely used in practice due to their speed and suitability for online scheduling.
- The DRL-based MAC-Sched method [9], which represents the state-of-the-art in C-DFJSP research.

These choices ensure that the evaluation reflects both realistic factory constraints (low-latency decision-making) and the closest available DRL baseline.

The results, summarised in Tables III, IV and VI, VII, VIII, highlight the *MT* and *LJ* outcomes from the best-performing heuristic methods under both objectives. The tables also highlight the relative performance gain (Rel. Gain), defined as the

percentage deviation between MACD and the best baseline method—either MAC-Sched or the top-performing heuristic under each objective. When MACD achieves the best result, the percentages are computed relative to MACD. The best-performing method in each case is highlighted in bold.

D. Brandimarte Dataset Results

The Brandimarte dataset introduces substantial variations in product and machine configurations, with the number of products ranging from 10 to 30 and the number of machines ranging from 4 to 15. Job orders were generated using $DDT \sim U(0.5, 3.0)$, and arrival rates were set based on five load levels from U1-5, ordered by the increasing arrival rate. This creates both urgent and non-urgent job orders under normal to busy conditions.

A continuous scheduling environment is simulated for 10,000 time steps, divided into 100 periods of $T = 100$ time steps each. This allows for periodic evaluation over a time duration that is much longer than the average processing time of a job order. Performance metrics, including the warm-up period, were recorded from the start to evaluate the performance over the entire simulation period. MACD completes 980 job orders on average per simulation, showing its competence for continuous scheduling. In contrast, existing methods are typically evaluated on problems with less than 100 jobs [13].

The results in Tables III and IV demonstrate MACD's ability to generalise from static to dynamic conditions effectively and to large unseen problem instances using the trained policy. MACD outperforms the heuristic methods and MAC-Sched (MACS) on most problem instances regarding *MT* and *LJ*. It also performs competitively on the remaining problem instances. The relative performance gain between MACD and the other methods increases as the arrival rate increases, demonstrating MACD's superior ability to manage busy manufacturing systems. This is particularly evident in instances with high machine counts (e.g., 15 machines), where MAC-Sched struggled to perform well. Furthermore, while MACD may not achieve the lowest *MT* in certain instances, such as 15×8 under U5, its strong performance in *LJ* confirms its effectiveness at maintaining system stability by preventing the accumulation of delayed jobs.

Fig. 6 highlights instances where MACD initially seems to underperform in *MT*. However, when considering both *MT* and *LJ* together, MACD outperforms the best heuristic methods. Notably, Fig. 6 shows that MACD can maintain a stable *LJ* count over time, unlike heuristics that exhibit increasing instability.

E. Hurink Dataset Results

The Hurink dataset expands the evaluation scope by incorporating varying flexibility levels:

- **E:** Low flexibility, with few operations assignable to multiple machines.
- **R:** Medium flexibility, with most operations assignable to some machines.

- **V:** High flexibility, with nearly all operations assignable to most machines.

Testing was performed on 40 instances for each subset (E, R, V), covering 10-30 product library sizes and 5-15 machines. Results are averaged and grouped by the number of machines. Simulations were run for 100 periods of:

$$T = \frac{\max(\text{job processing time})}{m} \times 10,$$

time steps each. This accounts for longer and more variable processing times than the Brandimarte dataset. On average, 1300 job orders were completed across all test instances, demonstrating MACD's robust scheduling performance under high workloads.

Furthermore, Tables VI, VII, and VIII confirm MACD's adaptability across varying flexibility levels. While its performance in high-flexibility settings is comparable to the best heuristics, MACD outperforms them at medium flexibility levels, showcasing its strength in diverse scenarios. MACD consistently delivers robust average performance across all cases, unlike heuristics, whose effectiveness varies by scenario. Notably, MACD achieves up to 26.7% better *MT* performance than the best heuristic in some cases, with only a 2.0% drop in relative performance in the worst case. This enhanced performance boosts efficiency and offers significant cost savings for smart manufacturing.

Although MACD achieves comparable results in high-flexibility scenarios with large machine counts, such configurations may require further refinement to optimise performance fully. Nevertheless, MACD is a robust base scheduling system, capable of generalising effectively across a wide range of factory configurations and operational conditions, with the flexibility to be fine-tuned for specific requirements.

F. Scalability and Robustness

Our evaluation extends beyond traditional benchmarks (e.g., Brandimarte, Hurink), scaling up to 30 products, 15 machines, and over 1,000 job orders per simulation—approximately three times larger than the training configuration. These large-scale scenarios enable us to assess MACD's ability to generalise under sustained, high-load conditions.

To profile computational efficiency, we measured the per-decision inference time on a Linux workstation with a 16-core CPU (no GPU; Python 3.10.12, kernel 5.15). Even on the largest test cases, each scheduling decision averaged 54 ms, which is negligible compared to real processing times (minutes per job). Peak memory consumption remained well within 16GB RAM, dominated by graph construction and GNN embeddings. These results confirm that MACD can operate in real time on standard industrial hardware without specialised accelerators.

The computational complexity of MACD is dominated by GNN inference, which scales linearly with the number of nodes and edges in the heterogeneous graph, i.e., $(|O| + |M| + |C| + |E|)$. In practice, this yields lightweight and predictable scaling characteristics.

We further evaluated MACD across varying arrival rates and machine flexibility levels, where performance remained stable. Nevertheless, extreme scenarios—such as thousands of machines, abrupt failures, or highly stochastic demand spikes—remain challenging. Future work should investigate batching, sparse attention, hierarchical decomposition, and robust reinforcement learning techniques to extend scalability and provide stronger safety guarantees under such conditions.

G. Negotiation Strategy Empirical Analysis

To design a principled and explainable negotiation mechanism, we conducted empirical evaluations using synthetic test instances (10 products $\times$ 5 machines) under five machine flexibility levels (2, 3, 4, 5, and mixed). Three representative machine selection rules were tested individually—Longest Processing Time (LPT), Shortest Processing Time (SPT), Least Loaded Machine (LLM), and Earliest End Time (EET)—as summarised in Table IX.

These rules respectively emphasise operational efficiency, load balancing, and completion time. Other machine rules, such as Minimum Completion Time (MCT), Least Work Remaining (LWKR), and Most Work Remaining (MWKR), were excluded as they were unsuitable for real-time decision making under the tardiness objective.

The results show that SPT consistently achieves the lowest mean tardiness in most scenarios, while LLM outperforms SPT only when machine flexibility is minimal. This observation is consistent with domain intuition: when only a few machines can process an operation, balancing load prevents bottlenecks, whereas under higher flexibility, prioritising the fastest machine is more effective.

To quantify how much negotiation contributes to overall behaviour, we measured the proportion of decisions requiring negotiation at five utilisation levels (Fig. 7). Most decisions are made solely by the learned policy, especially in well-utilised systems where machines become available less frequently and are less likely to compete for the same job. Negotiation is more common under low utilisation when machines are frequently idle and therefore more likely to target the same operation.

Crucially, even when negotiation occurs, both the candidate job and the set of competing machines are determined by the learned policy. Thus, negotiation resolves only the final assignment among RL-proposed options. Because (i) negotiation events are relatively rare, and (ii) negotiated choices originate from RL proposals, the specific negotiation rule has limited overall impact on performance. This supports the use of a simple, interpretable mechanism rather than a more complex learned coordination strategy.

VI. CONCLUSION

This paper introduced **MACD**, a Multi-Agent Continuous Decision-Making system for the Continuous Dynamic Flexible Job Shop Scheduling Problem (C-DFJSP). By integrating Graph Neural Networks (GNNs) and Reinforcement Learning (RL) within a multi-agent framework, MACD achieved consistently high performance across diverse factory configurations and dynamic job arrivals. Its novel state and observation

design, domain knowledge-based machine filtering, and attention mechanisms ensure scalability and robust performance across unseen factory scenarios. MACD also introduced a mixed negotiation strategy that combines the Shortest Processing Time (SPT) and Least Loaded Machine (LLM) heuristics, achieving a desirable balance between efficiency and load distribution. Extensive experiments validated its superior generalisation ability, outperforming existing methods in mean tardiness and system stability.

Although our experiments evaluate factories with up to 15 machines, industrial environments may involve hundreds. In such large-scale settings, the main limitation lies in the growth of the shop-floor graph and the cost of GNN-based message passing. While MACD's decentralised decision-making scales conceptually, practical constraints emerge in the embedding stage. We therefore highlight several promising avenues for large-scale deployment: hierarchical decomposition of factory regions, sparse or locality-aware attention, batching of agent observations to leverage GPU parallelism, and distributed graph encoding.

Future work will extend MACD to more realistic domains—stochastic processing times, machine failures, and distributed multi-factory networks—and explore *learned* negotiation strategies enabled by its modular design. This may introduce challenges such as credit assignment and coordination instability, which merit focused investigation. Strengthening robustness, including through risk-sensitive optimisation and fallback mechanisms for disruptions, remains an essential step toward reliable deployment in industrial smart manufacturing settings.

REFERENCES

- [1] X. Gu and Y. Koren, "Smart factories for mass individualization," *Encyclopedia*, vol. 4, no. 1, pp. 415–429, Feb. 2024.
- [2] Y. Lu, X. Xu, and L. Wang, "Smart manufacturing process and system automation—A critical review of the standards and envisioned scenarios," *J. Manuf. Syst.*, vol. 56, pp. 312–325, Jul. 2020, doi: [10.1016/j.jmsy.2020.06.010](https://doi.org/10.1016/j.jmsy.2020.06.010).
- [3] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Comput.*, vol. 45, no. 4, pp. 369–375, 1990.
- [4] I. A. Chaudhry and A. A. Khan, "A research survey: Review of flexible job shop scheduling techniques," *Int. Trans. Oper. Res.*, vol. 23, no. 3, pp. 551–591, May 2016.
- [5] J. Xie, L. Gao, K. Peng, X. Li, and H. Li, "Review on flexible job shop scheduling," *IET Collaborative Intell. Manuf.*, vol. 1, no. 3, pp. 67–77, Sep. 2019, doi: [10.1049/iet-cim.2018.0009](https://doi.org/10.1049/iet-cim.2018.0009).
- [6] M. Gholami and M. Zandieh, "Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop," *J. Intell. Manuf.*, vol. 20, no. 4, pp. 481–498, Aug. 2009.
- [7] X. Li and L. Gao, *A Hybrid Genetic Algorithm and Tabu Search for Multi-objective Dynamic JSP*. Berlin, Germany: Springer, 2020, pp. 377–403, doi: [10.1007/978-3-662-55305-3_18](https://doi.org/10.1007/978-3-662-55305-3_18).
- [8] S. K. Fuladi and C.-S. Kim, "Dynamic events in the flexible job-shop scheduling problem: Rescheduling with a hybrid metaheuristic algorithm," *Algorithms*, vol. 17, no. 4, p. 142, Mar. 2024. [Online]. Available: <https://www.mdpi.com/1999-4893/17/4/142>
- [9] D. Johnson, G. Chen, and Y. Lu, "Multi-agent scheduler for the continuous dynamic flexible job shop scheduling problem," in *Proc. IEEE 20th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2024, pp. 2924–2930.
- [10] *Automated Assembly With Kuka Assembly Robots — Kuka*. Accessed: May 15, 2026. [Online]. Available: <https://www.kuka.com/en-de/applications/joining-process-automation/assembly-robot>
- [11] *Robotic Flex Feeding Systems for Automotive Manufacturing — PMI2*. Accessed: May 15, 2026. [Online]. Available: <https://pmi2sc.com/blog/automated-robotic-flex-feeding-systems-for-automotive-manufacturing>
- [12] T. Borreguero-Sanchidrian, R. Pulido, A. Garcia-Sanchez, and M. Ortega-Mier, "Flexible job shop scheduling with operators in aeronautical manufacturing: A case study," *IEEE Access*, vol. 6, pp. 224–233, 2018.
- [13] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 20, no. 1, pp. 1007–1018, Jan. 2024.
- [14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2008.
- [15] C. Nam and D. A. Shell, "Assignment algorithms for modeling resource contention in multirobot task allocation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 889–900, Jul. 2015.
- [16] D.-H. Lee, S. A. Zaheer, and J.-H. Kim, "A resource-oriented, decentralized auction algorithm for multirobot task allocation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 4, pp. 1469–1481, Oct. 2015.
- [17] V. Subramaniam, G. K. Lee, T. Ramesh, G. S. Hong, and Y. S. Wong, "Machine selection rules in a dynamic job shop," *Int. J. Adv. Manuf. Technol.*, vol. 16, no. 12, pp. 902–908, Oct. 2000.
- [18] N. Kundakci and O. Kulak, "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem," *Comput. Ind. Eng.*, vol. 96, pp. 31–51, Jun. 2016.
- [19] S. Zhang, F. Tang, X. Li, J. Liu, and B. Zhang, "A hybrid multi-objective approach for real-time flexible production scheduling and rescheduling under dynamic environment in Industry 4.0 context," *Comput. Oper. Res.*, vol. 132, Aug. 2021, Art. no. 105267, doi: [10.1016/j.cor.2021.105267](https://doi.org/10.1016/j.cor.2021.105267).
- [20] L. Renke, R. Piplani, and C. Toro, "A review of dynamic scheduling: Context, techniques and prospects," in *Implementing Industry 4.0* (Intelligent Systems Reference Library), vol. 202. Cham, Switzerland: Springer, 2021, pp. 229–258, doi: [10.1007/978-3-030-67270-6_9](https://doi.org/10.1007/978-3-030-67270-6_9).
- [21] L. Zhao, J. Fan, C. Zhang, W. Shen, and J. Zhuang, "A DRL-based reactive scheduling policy for flexible job shops with random job arrivals," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 2912–2923, Jul. 2024.
- [22] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [23] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: [10.1038/nature16961](https://doi.org/10.1038/nature16961).
- [24] S. Luo, L. Zhang, and Y. Fan, "Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3020–3038, Oct. 2022.
- [25] J.-H. Lee and H.-J. Kim, "Graph-based imitation learning for real-time job shop dispatcher," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 8593–8606, 2024.
- [26] J.-P. Huang, L. Gao, and 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.*, vol. 22, pp. 2501–2513, 2025.
- [27] W. Song, X. Chen, Q. Li, and Z. Cao, "Flexible job-shop scheduling via graph neural network and deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1600–1610, Feb. 2023.
- [28] A. O. Allen, *Probability, Statistics, and Queuing Theory: With Computerscience Applications*, 2nd ed., New York, NY, USA: Academic, 1990.
- [29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [30] S. Mahadevan, *Average Reward Reinforcement Learning: Found., Algorithms, Empirical Results*. Boston, MA, USA: Springer, 1996, pp. 159–195, doi: [10.1007/978-0-585-33656-5_8](https://doi.org/10.1007/978-0-585-33656-5_8).
- [31] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017, pp. 1024–1034.
- [32] P. Brandimarte, "Routing and scheduling in a flexible job shop by Tabu search," *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, Sep. 1993.
- [33] J. Hurink, B. Jurisch, and M. Thole, "Tabu search for the job-shop scheduling problem with multi-purpose machines," *OR Spektrum*, vol. 15, no. 4, pp. 205–215, Dec. 1994.

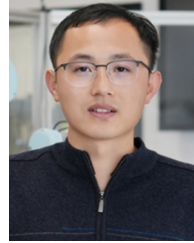


Dazzle Johnson received the joint B.E. degree (Hons.) in mechatronics engineering and the B.Sc. degree in computer science and the Ph.D. degree in mechatronics engineering from the University of Auckland, New Zealand, in 2020 and 2026, respectively. She is currently a Graduate Engineer with Powerco, New Zealand. Her research interests include artificial intelligence applications for power systems, data analytics, and intelligent decision-making systems.

He has served as the Senior Area Chair or the Area Chair for major AI conferences, including NeurIPS, ICML, and ICLR, and as a Senior Program Committee Member for AAAI and IJCAI.



Gang Chen (Senior Member, IEEE) received the Ph.D. degree in computer science from Nanyang Technological University, Singapore. He is currently an Associate Professor with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. He has published extensively in leading journals and conferences in artificial intelligence, machine learning, evolutionary computation, and distributed computing. His research interests include reinforcement learning, deep learning, evolutionary computation, multi-agent systems, cloud and service computing, and combinatorial optimization.



Yuqian Lu (Member, IEEE) received the B.E. degree (Hons.) in mechatronics engineering from Dalian University of Technology, Dalian, China, in 2012, and the Ph.D. degree in mechatronics engineering from the University of Auckland, Auckland, New Zealand, in 2017. He is currently a Senior Lecturer (Above the Bar) with the University of Auckland, where he is also the Director of the Industrial AI Research Group. His research interests include industrial artificial intelligence, smart manufacturing, and human–robot collaboration. He serves as an Associate Editor for IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS and IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.