An Adaptive Meta-Reinforcement Learning Framework for Dynamic Flexible Job Shop Scheduling

Lincong Wu, Xiaoxia Li*, Xin Lu, Zaiwen Feng, Yanguo Jing

***Abstract*—The optimization of flexible job shop scheduling is essential for improving manufacturing efficiency and performance in dynamic production environments. However, existing scheduling methods face challenges in scalability and adaptability, which limits their effectiveness in such environments. To address these limitations, this paper proposes a generalized and modular DFJSP framework that systematically decomposes shop-floor elements into key modules, enabling flexible and resilient scheduling under dynamic conditions. Building on this framework, an Adaptive Markov Decision Process (AMDP) is formulated to capture real-time shop-floor states and guide optimal action selection. Leveraging Meta-Reinforcement Learning (MRL), the proposed approach integrates Model-Agnostic Meta-Learning (MAML) with Proximal Policy Optimization (PPO) to facilitate rapid adaptation to new scheduling tasks while enhancing policy generalization. Numerical experiments demonstrate that the framework effectively balances multiple dynamic objectives, including makespan and energy consumption, and adapts efficiently to real-time variations in job priorities, machine availability, and processing times. The results highlight the potential of combining modular problem formulation, AMDP modeling, and MRL for scalable, efficient, and robust DFJSP solutions in modern manufacturing environments.**

***Note to Practitioners*—This paper presents a MRL framework that offers an efficient solution for dynamic flexible job shop scheduling, enabling rapid adaptation to unexpected events such as machine breakdowns and urgent job insertions while simultaneously optimizing both makespan and energy consumption. Through a modular design, the framework can be flexibly adapted to various shop floor configurations. Leveraging the MAML-PPO algorithm, the model requires only a small amount of data to generate new scheduling solutions within seconds, demonstrating a significantly faster response compared to conventional methods such as genetic algorithms. For practical deployment, it is recommended to prioritize the integration of dynamic event definitions (e.g., failure probabilities) and to utilize GPU-accelerated computing. Additionally, module functionalities (e.g., logistics coordination) should be adjusted according to production requirements to achieve rapid improvements in operational efficiency and resource optimization.**

***Index Terms*—meta-learning, reinforcement learning, meta-reinforcement learning, dynamic flexible job shop scheduling**

I. Introduction

In recent years, manufacturing has become increasingly competitive due to stringent regulations, sustainability requirements, and rapidly evolving market demands. To remain competitive, manufacturers must continuously enhance operational efficiency, with shop-floor scheduling playing a central role in strategic planning and long-term sustainability. High-quality scheduling solutions are critical for improving key performance metrics, including productivity, energy efficiency, and overall operational flexibility [1], [2]. Within this context, flexible job shop scheduling problem (FJSP), in which each operation can be processed by any machine from a candidate set, has been widely studied. In real-world manufacturing, FJSPs are inherently dynamic: job arrivals, machine availability, and processing times vary over time, and unexpected events such as machine breakdowns or urgent job insertions frequently occur. This dynamic nature introduces significant challenges in accurately modelling the problem and constructing efficient scheduling solutions. Despite substantial research on FJSPs, there remains a lack of rapid, generalizable frameworks capable of efficiently formulating and solving dynamic FJSP (DFJSP) under diverse and uncertain shop-floor conditions.

Various approaches have been explored to tackle FJSP. Dispatching rules provide fast, experience-based decision-making but are sensitive to shop-specific configurations and often yield suboptimal results in complex or variable environments [3]. Meta-heuristic algorithms, such as genetic algorithms and particle swarm optimization, can explore near-optimal solutions by searching the scheduling space [4], however their efficiency declines with problem size, and their iterative nature limits responsiveness in dynamic settings. More recently, machine learning techniques, particularly reinforcement learning (RL), have shown promise in overcoming some of these limitations [5]–[7]. RL methods can learn effective scheduling policies from interaction with the environment but face challenges including sample inefficiency [8] and slow adaptation to diverse shop-floor conditions. Meta-learning offers a potential solution to these challenges by enabling rapid adaptation to new tasks based on prior experience [9]. When combined with RL, meta-reinforcement learning allows agents to leverage both prior knowledge and real-time interactions, representing a promising avenue for DFJSP. MRL has demonstrated success in other domains, such as traffic signal management [10] and intelligent robot control [11]. However, its application to shop-floor scheduling remains in its early stages. DFJSPs are inherently complex, making it difficult to represent job sequences, machine availability, and processing times in a way that MRL models can effectively process. The diversity of shop-floor environments further complicates model generalization, and real-world uncertainties, such as machine breakdowns and supply chain disruptions, must be incorporated to enhance robustness and reliability. As DFJSPs grow in size and complexity, the development of scalable MRL algorithms will become increasingly essential.

To bridge these gaps, this paper presents an innovative DFJSP framework that incorporates meta learning and RL algorithms to address the modelling, adaptability, and scalability challenges of current scheduling methodologies. The research demonstrates that combining advanced deep learning

algorithms with a distributed computing architecture provides an effective means of meeting the versatile requirements of modern manufacturing processes. Research innovations include the following aspects:

- A generalized and modular DFJSP framework is developed to address diverse real-world shop-floor environments. The proposed framework innovatively decomposes elements into distinct modules that map environmental variations, thereby enhancing flexibility and enabling robust scheduling under dynamic and uncertain conditions.
- An Adaptive Markov Decision Process is systematically formulated for DFJSP by mapping real-time shop-floor states to the modular framework for optimal action selection. The proposed AMDP enables MRL algorithms to effectively manage dynamic machining processes while enhancing scheduling efficiency and adaptability.
- A novel MRL framework is developed to enhance both the efficiency and generalization of policy learning in dynamic shop-floor environments. As a case study, the framework integrates MAML with PPO to enable rapid adaptation to new scheduling tasks through knowledge transfer across various tasks.

The remainder of this paper is organized as follows. Section II provides a review of related work. Section III introduces the modular architecture for DFJSP and the AMDP formulation. Section IV proposes the MRL framework for DFJSP and the MAML-PPO algorithm. Section V presents the experimental results and analysis. Finally, Section VI concludes the paper.

II. Related Work

The recent surge in scholarly attention towards solving FJSPs within the context of RL emphasizes the increasing significance of this field within both ML and operations research communities [12]. This section explores recent advancements in FJSP formulations and the development of MRL techniques tailored to these challenges.

A. FJSP framework and Markov Decision Process

In modern industrial production, the complexity arising from customization, product variety, and dynamic uncertainties has led scholars to extend FJSPs to better reflect real-world manufacturing environments. For instance, Wang et al. [13] proposed an improved FJSP framework that incorporates multiple dynamic events and objectives—such as job insertions, machine breakdowns, and varying job priorities—to more closely mirror real-world production environments. Kong et al. [14] enhanced the FJSP by integrating the transportation problem into the model, enabling a more accurate representation of manufacturing environments where machines are distributed across multiple shop floors. Meanwhile, [15] introduced an FJSP model with sequence-dependent setup times, enhancing the realism of the scheduling problem by reflecting the variability of machine setup durations and the complexities of actual manufacturing environments. These enhancements to FJSPs substantially increase their complexity, enabling more realistic representations of manufacturing environments but also expanding the search space and raising computational demands for identifying optimal solutions [16]. In addition, the absence of standardized benchmark problems that incorporate critical factors such as dynamic events and realistic shop-floor configurations restricts progress in the field. This lack of benchmarking not only hinders the development of universally applicable solutions but also underscores the urgent need for collaborative efforts to establish consistent metrics and test cases for FJSP research.

Additionally, Markov Decision Process (MDP) is widely used to model FJSP because it's the basis of RL. Numerous MDP-based RL approaches for FJSP have been proposed. For instance, Kun et al. [17] introduced a disjunctive graph representation, achieving superior results compared to rule-based methods by leveraging graph neural networks and reinforcement learning techniques. In addition, Zhu et al. [18] proposed an adaptive real-time scheduling method within a multi-agent system (MAS) framework for handling combined processing constraints in FJSPs. More recently, Zhang and Li [19] incorporated machine degradation into the dynamic modeling of FJSPs by formulating the problem as a joint scheduling and preventive maintenance decision-making process. The machine states were modelled as a discrete multi-state degradation process using a continuous-time Markov chain, which allowed the scheduling policy to explicitly account for stochastic degradation and maintenance costs. More recently, Li et al. [20] advanced the field by integrating a graph attention network within an MDP-RL framework to tackle multiple objectives simultaneously, such as makespan minimization and energy consumption, significantly improving Pareto-optimal results in flexible job shop environments. Furthermore, Xu et al. [21] transformed FJSP into an MDP and employed a Transformer-based deep reinforcement learning model to effectively learn scheduling policies across complex state–action spaces. Obviously, the real-world shop-floor's dynamics and complexity are neglected by the existing research works.

In conclusion, modeling FJSPs faces several challenges. FJSPs are inherently complex and vary across companies due to different constraints, priorities, and operational practices, resulting in a lack of standardized modeling approaches. Existing models are often time-consuming to develop and may fail to capture real-world shop-floor dynamics, such as multiple objectives and unexpected events. The absence of a comprehensive reference framework further limits the development of effective and adaptive solutions across industrial contexts.

B. Meta-reinforcement Learning for FJSP

In recent years, a variety of meta-learning algorithms combined with RL has emerged as a promising solution for achieving rapid adaptation to novel tasks with limited training data. Nagabandi et al. [8] introduced a sample-efficient meta-learning approach to enable swift adaptation of robotic systems within dynamic environments and Wang et al. [22] presented meta-learning strategies to boost learning efficacy across various robotic settings. Within the industrial domain, MRL methods have primarily been employed to optimize machining

processes. Changqing Liu et al. [23] developed an MRL model that can be updated with a minimal amount of real monitoring data, achieving optimization for the finishing process and deformation control. Qinge Xiao et al. [24] used a meta-actor-critic approach to optimize the machining parameters. Furthermore, McClement et al. [25] applied a context-based MRL method in industrial process control. The proposed system enables adaptation to new process dynamics and different control objectives within the same process. These diverse applications across industrial processes highlight the versatility and practical utility of MRL methods in tackling the challenges associated with rapid adaptation to new tasks within dynamic and complex real-world scenarios. MRL has been proven to be a valuable tool in addressing scheduling challenges. In heterogeneous edge computing systems, Liwen Niu et al. [26] introduced a multi-agent meta-PPO algorithm, which aims at optimizing task scheduling by adapting control policy learning to nonstationarity. Similarly, in cloud computing environments, Xi Xiu et al. [27] proposed an MRL-based method to enhance adaptability to diverse environments. The results showed that the proposed system can improve sample efficiency in various task scheduling problems. Furthermore, in home energy management systems, Luolin Xiong et al. [28] employed an MRL-based transferable scheduling strategy to address uncertainties in renewable energy sources and fluctuation in customer load. Regarding FJSP problems, Finn et al. [29] and Ahmedet et al. [30] presented MAML, which leverages past experiences to generalize to new scenarios. This makes it particularly suitable for DFJSP, where schedules must adapt quickly to changing conditions. Hierarchical MRL is another potential approach for solving FJSPs. It involves decomposing tasks into multiple sub-tasks to be solved across various hierarchical levels [31]. By leveraging the inherent structure within tasks, this approach enhances learning efficiency and improves adaptation to dynamic scheduling environments. In conclusion, integrating meta-learning with reinforcement learning holds significant promise for enabling rapid adaptation to novel tasks across various real-world domains. However, the application of MRL to shop-floor scheduling remains in its early stages, further research is required to overcome key challenges. DFJSPs are inherently complex, making it difficult to represent job sequences, machine availability, and processing times in a way that MRL models can effectively process, while the diversity of shop-floor environments presents challenges for generalization. Additionally, real-world uncertainties such as machine breakdowns and supply chain disruptions must be incorporated to enhance the robustness and reliability of these scheduling models. As DFJSPs grow in size and complexity, scalable MRL algorithms will become increasingly essential.

III. DFJSP Framework and Model

As highlighted in the literature, existing scheduling systems and FJSP models suffer from limited adaptability to diverse production environments, insufficient responsiveness to dynamic changes, and a lack of modular design, restricting their broader applicability and effectiveness. To address these issues in FJSP modeling and frameworks, this section proposes a modular framework for DFJSP to enhance adaptability, responsiveness, and scalability across various production environments, along with an AMDP formulation to support structured decision-making using MRL.

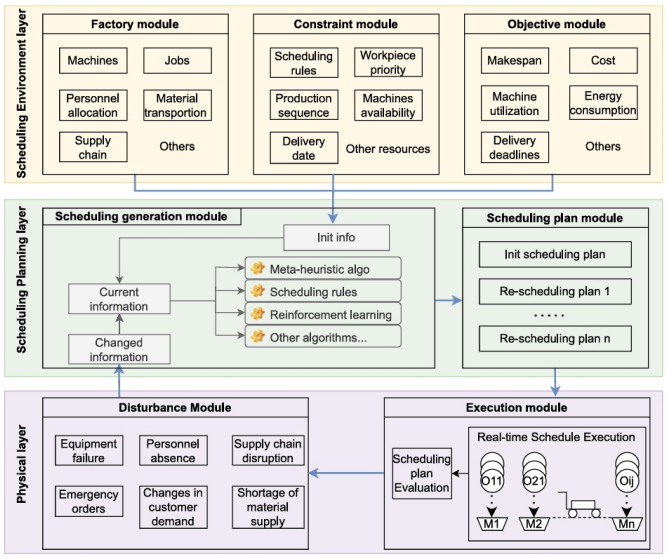

Fig. 1: A Hierarchical and Modular General Framework for DFJSP

A. Modular Framework for DFJSP

The evolution of modern industrial enterprises has turned the FJSPs into complex spatio-temporal dynamic challenges that require flexible expansion and robust contraction based on varying operational scenarios, which includes incorporating transpiration logistics, handing out multiple objectives, managing additional resources, and responding to dynamic events. To address these dynamic complexities, a generalized and modular DFJSP framework with hierarchical layers has been developed, as illustrated in Fig. 1. The framework is organized into three layers, each comprising multiple modules tailored to dynamic factors such as multiple objectives, machine breakdowns, urgent job insertions, and variable processing times. By systematically decomposing these challenges into modular components, the framework enhances transparency in decision-making and enables efficient reconfiguration of scheduling strategies in real time, thereby reinforcing flexibility and adaptability across diverse industrial contexts. The top layer, named Scheduling Environment layer, is responsible for the initial setup and configuration of the DFJSP. It establishes the foundational components essential for initiating the scheduling process and comprises three modules: a Factory Module, an Objective Module and a Constraint Module, respectively.

1) Factory Module: is designed to manage the initial configuration information of the factory, including machines, operations, personnel allocation, raw materials, material transportation capacity, and more. This information may be sourced from higher-level enterprise management systems such as MES (Manufacturing Execution System) or ERP (Enterprise Resource Planning).

2) Objective Module: is designed to manage and balance multiple, often conflicting production objectives, including completion time, energy consumption, machine utilization, inventory levels, and delivery deadlines. The relative importance of these objectives may vary depending on company-specific priorities and operational strategies.

3) Constraint Module: The constraint module is responsible for setting various constraints in the production process to ensure smooth operations. Typically, workshop constraints include scheduling rules, workpiece priorities, production sequences, machine availability, order delivery dates, and other resource limitations. When dealing with issues of collaborative optimization with other systems, such as logistics, warehousing, and scheduling systems, this module also reflects spatio-temporal coupling constraints or resource competition constraints.

The scheduling planning layer is the intermediate component of the proposed framework, consisting of two important modules: the scheduling generation module and the scheduling planning module. These modules support the use of different scheduling algorithms to develop scheduling plans.

1) Schedule generation module: The scheduling generation module is responsible for generating the optimal production schedule by integrating inputs from the factory, goal, constraint, and dynamic event modules. This module uses multiple scheduling agents and supports various scheduling techniques, including metaheuristic algorithms, scheduling rules, or reinforcement learning strategies.

2) Scheduling plan module: The scheduling planning module is responsible for deploying the current scheduling plan and managing the data of historical scheduling plans.

The physical layer includes the execution module and the disruption module. This layer is responsible for the actual execution of the scheduling plan and real-time monitoring of the production process. It uses an "execution-feedback" control loop, which ensures that production activities follow the optimized schedule while also managing any disruptions or deviations that arise during operations.

1) Execution Module: The execution module ensures the scheduling plan is implemented smoothly on the shop floor and continuously monitored. It consists of two parts: real-time scheduling execution, which coordinates machines, workers, and materials according to the plan, and scheduling plan evaluation, which gathers feedback and performance data to support continuous improvement of future schedules.

2) Disturbance Module: is designed to handle a wide range of dynamic events that can disrupt production, covering both internal and external factors. Internal disturbances include unexpected machine breakdowns, variations in processing times, or staff unavailability, while external disturbances may arise from urgent job insertions, order changes, supply chain delays, or logistical constraints. By using feedback from the plan evaluation, this module makes timely adjustments to reduce downtime, mitigate risks, and strengthen the robustness of the production system.

In conclusion, the proposed hierarchical DFJSP framework is designed to seamlessly offer a flexible and adaptable solution for managing complex scenarios in dynamic industrial environments. Its modular structure allows components to be added or removed based on specific production needs, making it easy to integrate functions like transportation logistics or expanded resource management.

B. AMDP-Based Formulation of DFJSP

Within the proposed modular framework, the DFJSP is integrated to establish a comprehensive, generalised definition. This modular approach allows the problem to be systematically decomposed into key elements, including factory resources, operational constraints, performance objectives, and potential disturbances. Building on this foundation, we develop an AMDP modle, which is tailored to capture the dynamic and stochastic nature of DFJSP. The model is designed not only to optimise primary objectives, such as minimizing makespan and energy consumption, but also to incorporate relevant operational constraints and to adapt to diverse dynamic events occurring on the shop floor. To develop the AMDP model, the DFJSP is systematically decomposed into four modules within the proposed framework. These modules, which capture the essential elements, constraints, objectives, and disturbances, are summarised in Table I and details as follows:

1) Factory Module Definition: Let n be the number of jobs to be machined. Let m be the number of machines in the shop floor. The jobs and the machines can be denoted as $J = \{J_1, J_2, \ldots, J_n\}$ and $M = \{M_1, M_2, \ldots, M_m\}$ respectively. Each job J_i consists of n_i operations $O_{i,j}$ $(j = 1, 2, \ldots, n_i)$. Each operation $O_{i,j}$ can be processed by any machine from a set of available machines $M_{i,j} \subseteq M$. The processing time of each operation on different machines is expressed as $t_{i,j,k}$, where k indexes the machine.

2) Objective module Definition: The criteria that are considered by the manufacturers to improve their manufacturing performances, are integrated into the objective module to be used as the optimization objectives. Since productivity and sustainability are the manufacturing performances focus, both makespan and machining energy consumption are contained in the objective module. The former is to minimize the maximum completion time of all jobs and operations:

$$C_{max} = \max_{i,j}\{C_{i,j}\} \tag{1}$$

where $C_{i,j}$ is the completion time of the operation $O_{i,j}$, and C_{max} (i.e. makespan) is the maximum completion time. Minimizing makespan reduces the maximum completion time, enhances resource utilization, and shortens lead time. The model accounts for job priorities, machine availability, and dependencies to achieve this objective.

The latter is to minimize the machining energy consumption, focusing on the idle and processing phases. The energy consumption during these phases for machine M_k is:

$$E^{proc}_{i,j,k} = P^{proc}_{k} \times t_{i,j,k} \tag{2}$$

where $E^{proc}_{i,j,k}$ is the energy consumed by machine M_k during the processing of operation $O_{i,j}$, and P^{proc}_{k} is the processing power of machine M_k. The idle energy consumption is:

$$E^{idle}_{k} = P^{idle}_{k} \times t^{idle}_{k} \tag{3}$$

TABLE I: A general DFJSP definition

Factory elements module	Constraint module	Objective module	Disturbance
$J = \{J_1, J_2, \ldots, J_n\}$	Operation order	makespan	Emergency job insertion
$M = \{M_1, M_2, \ldots, M_m\}$	Operation allocation	Energy consumption	Machine breakdown
$J_{info} \quad 0 < i \leq n$	Machine capacity		Machine Maintenance
$M_{info} \quad 0 < j \leq m$			

where P_k^{proc} is the idle power of machine M_k. Thus, the total energy consumption for machine M_k is:

$$EC_k = \sum_{O_{i,j} \in J} E_{i,j,k}^{proc} + E_k^{idle} \tag{4}$$

Minimizing EC improves the sustainability and efficiency of the manufacturing system.

3) Constraint Module Definition: Three common constraints are considered in this paper, listed in Table I. The constraint details are as follows:

Operation sequence constraint: The start time of operation $O_{i,j}$ should be later than the completion time of its preceding operation, which is formulated as follows:

$$S_{i,j} \geq C_{i,j-1} \tag{5}$$

where $S_{i,j}$ represents the start time of operation $O_{i,j}$, and $C_{i,j-1}$ is the completion time of its preceding operation. For the first operation ($j = 1$), it is assumed that $C_{i,0} = 0$ to ensure production can start immediately.

Operation assignment constraint: Each operation must be assigned to a specific machine, which is implemented through the binary variable $X_{i,j,k}$:

$$\sum_{k \in M} X_{i,j,k} = 1 \quad \forall i, j \tag{6}$$

When operation $O_{i,j}$ is assigned to machine M_k, $X_{i,j,k} = 1$; otherwise, $X_{i,j,k} = 0$. This constraint prevents operations from being unassigned or redundantly assigned to multiple machines.

Machine capacity constraint: A machine can only perform one operation at any given time:

$$\sum_{i,j} X_{i,j,k,t} \leq 1 \quad \forall k, \forall t \tag{7}$$

This constraint limits the number of operations processed by machine M_k at time t, ensuring that it does not become overloaded, which could cause the scheduling plan to fail.

4) Disturbance Module Definition: It is known that there are many different disturbances that may occur on the shop floor. Also, all these disturbances can be integrated into the disturbance module to be responsed to. Table I lists three most commonly occurred disturbances (i.e. emergency job insertion, machine breakdown and machine maintenance), which are related to jobs and machines.

As illustrated in Fig. 2, a FJSP instance can be effectively represented as a disjunctive graph (DG), described by the tuple $DG = (O, \mathcal{P}, U)$ [7]. In this formulation, the set of vertices, O, represents the operations to be scheduled, where each vertex corresponds to a specific operation in the job shop environment. The directed arc set, $\mathcal{P}$, denotes precedence constraints between operations within the same job. These arcs dictate the sequence in which operations must be executed, ensuring the correct order of processing. The set of undirected arcs, U, known as the disjunction set. Each subset represents operations that can be executed on the same machine, allowing flexibility in assigning operations to machines. The undirected arcs within each subset connect operations competing for the same machine resources. In the disjunctive graph, undirected arcs (disjunctions) link operations that could be scheduled on the same machine, while directed arcs ensure the proper order of operations within a job. This Fig. 2 illustrates the iterative process of solving the DFJSP using a DG . The process begins with a complex graph where directed arcs represent the sequential order of operations within a job, while undirected arcs signify conflicts where different jobs compete for the same machine. Each step in the solving process is a decision point: a set of undirected arcs is chosen and converted into directed arcs to determine the processing order of operations. With each decision, the undirected arcs in the graph are gradually "oriented" until all operations are assigned and sequenced. Ultimately, all undirected arcs are transformed into directed ones, forming a complete and feasible scheduling solution that specifies which machine performs each operation and in what order. Essentially, the process resolves potential machine conflicts through a series of decisions to create a finalized, coherent schedule.

Following these definitions, the DFJSP scheduling problem can be formally formulated as an AMDP, which is defined by the set $\{S, A, R, T, \gamma, D\}$, where S represents the set of states, A represents the set of actions, and $R(r_{t+1}|s_t, a_t, s_{t+1})$ is the reward function. $T(s_{t+1}|s_t, a_t)$ denotes the transition equation, discount factor $\gamma = 1$ indicates that there is no decay in the accumulation of rewards (in this study, it is assumed that rewards are not discounted). Specifically, D refers to the Disturbance Event Set, originating from the disturbance module in the DFJSP architecture. This set encompasses all types of disturbances, such as equipment failures and emergency order insertions, that may affect the workshop environment. The specific AMDP definition for the proposed DFJSP is as follows:

1) State: The state space S_t represents the complete configuration of the job shop at time t, including operation-level, machine-level, and operation–machine interaction level. The corresponding features are defined as follows:

- *Operation features* $\mathbf{f}_t^{op}$: minimum energy consumption $E_t^{op,\min}$, scheduling status flag $Flag_t^{op}$, completion time lower bound LB_t^{op}, remaining energy consumption $E_t^{op,rem}$, waiting time W_t^{op}, and remaining processing time P_t^{op}.
- *Machine features* $\mathbf{f}_t^{ma}$: unscheduled operations count C_t^{ma}, machine free time F_t^{ma}, waiting time W_t^{ma}, machine status S_t^{ma}, remaining processing time P_t^{ma}, and

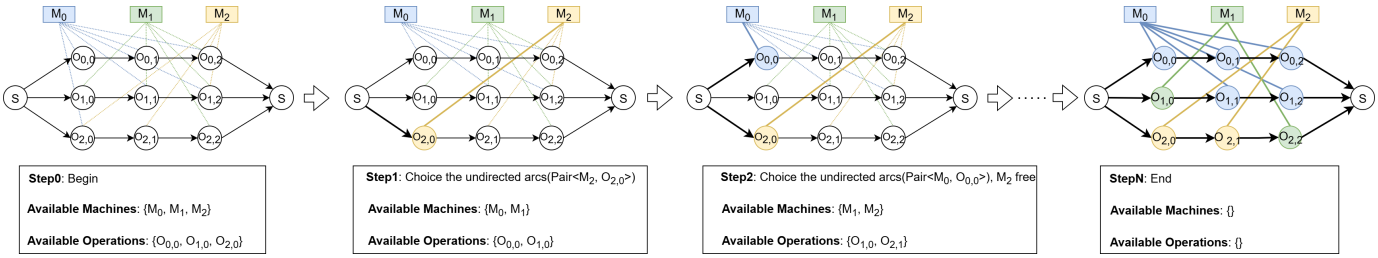

Fig. 2: A Process Instance for Addressing a FJSP

current energy consumption rate R_t^{ma}.

- *Operation–machine Pair features* $\mathbf{f}_t^{pa}$: processing time and workload ratios R_t^{pa}, and operation energy consumption E_t^{pa}.

Thus, the state vector can be formulated as:

$$S_t = \{\mathbf{f}_t^{op},\ \mathbf{f}_t^{ma},\ \mathbf{f}_t^{pa}\}. \tag{8}$$

The features in the state vector are adaptive and can be adjusted with the dynamic events occurring in the shop floor. For example, when machine breakdown occurs, the corresponding machine's features will be masked. When new jobs are inserted, the jobs' operation features will be added. All detailed feature definitions and calculation methods are provided in the Supplemental Items.

2) Action: In a DFJSP, an action involves selecting a machine for a specific operation or determining the sequence of operations on a machine. As shown in Fig. 2, an action at time step t is defined as selecting an operation–machine pair for scheduling. Formally, let $\mathcal{M}_t^{av}$ denote the set of available machines and $\mathcal{O}_t^{av}$ denote the set of available operations. Then an action can be expressed as:

$$a_t = \langle M_k, O_{i,j} \rangle, \quad M_k \in \mathcal{M}_t^{av},\ O_{i,j} \in \mathcal{O}_t^{av}, \tag{9}$$

where M_k is the selected machine , and $O_{i,j}$ is the j-th operation of job i.

Taking such an action means assigning operation $O_{i,j}$ to machine M_k, thereby fixing the corresponding undirected arc in the disjunctive graph representation. This gradually reduces the available sets $\mathcal{O}_t^{av}$ and $\mathcal{M}_t^{av}$, until all operations are scheduled and the process ends. When dynamic events occur on the shop floor, the action set will be adjusted following the current state to support the regeneration of new solutions.

3) Reward: In this work, the reward, r_t, is designed to simultaneously minimize the makespan and reduce energy consumption, while adhering to the constraints of the system. It is defined at time step t as follows:

$$r_t = \lambda_1 \times r_{ec} + \lambda_2 \times r_{mk} \tag{10}$$

where r_{ec} and r_{mk} represent the rewards obtained from reducing the total energy consumption and the maximum completion time of all operations (makespan), respectively. λ_1 and λ_2 are the weights associated with energy consumption and makespan rewards, respectively.

The reward for energy consumption r_{ec} is based on the reduction in energy used by the machines between two consecutive time steps t and $t+1$:

$$r_{ec} = \sum_{k=1}^{N_M} EC_k(s_t) - \sum_{k=1}^{N_M} EC_k(s_{t+1}) \tag{11}$$

in which the energy consumption EC_k of machine k is calculated based on the equation referenced in Equation (4). Assuming a complete scheduling cycle consists of $|\mathcal{O}|$ time steps, the cumulative energy consumption reward can be expressed as:

$$\sum_{t=0}^{|\mathcal{O}|} r_{ec} = \left(\sum_{k=1}^{N_M} EC_k(s_0)\right) - \left(\sum_{k=1}^{N_M} EC_k(s_{|\mathcal{O}|})\right) \tag{12}$$

The total energy consumption at the initial state $\sum_{k=1}^{N_M} EC_k(s_0)$ is a constant, unaffected by the scheduling policy. Therefore, maximizing the cumulative reward $\sum_{t=0}^{|\mathcal{O}|} r_{ec}$ is equivalent to minimizing the total energy consumption in the terminal state $\sum_{k=1}^{N_M} EC_k(s_{|\mathcal{O}|})$.

The reward for makespan r_{mk} is based on the reduction in the makespan between two consecutive steps:

$$r_{mk} = C_{\max}(s_t) - C_{\max}(s_{t+1}) \tag{13}$$

where $C_{\max}(s_t)$ represents the makespan at step t. Similarly, it can be deduced that maximizing the cumulative reward $\sum_{t=0}^{|\mathcal{O}|} r_{mk}$ is equivalent to minimizing the total makespan at the terminal state $C_{max}(s_{|\mathcal{O}|})$.

4) Disturbance Event Set: The disturbance event set $D = \{d_1, d_2, \ldots, d_p\}$ captures all unexpected or planned events that can disrupt the production environment, requiring real-time adaptations to maintain optimal performance. In this study, we consider three representative types of disturbances:

(1) Emergency Job Insertion. To simulate this event, a new job J^{new} is inserted into the operation queue during execution. Each new job consists of a sequence of operations $O^{new} = \{O_1^{new}, O_2^{new}, \ldots, O_m^{new}\}$, which follow the same precedence constraints as regular jobs.

(2) Machine Breakdown. To model this disturbance, a masking mechanism that disables the broken machine M_k, is applied to prevent any operation assignment to it within the breakdown window. Once the machine recovers, the mask is removed and the machine returns to the available pool. This approach ensures that scheduling decisions automatically adapt to a reduced set of feasible machines.

(3) Machine Maintenance. Planned maintenance events share a similar implementation to breakdowns, but with known

start times and durations. During the maintenance window, the corresponding machine M_k is masked and cannot be scheduled for new operations. Unlike breakdowns, maintenance events are typically pre-specified, enabling the scheduler to proactively adjust the allocation strategy, reduce potential delays, and improve system resilience.

The proposed disturbance mechanisms enable the scheduling framework to effectively manage both unplanned disruptions and planned interruptions. By dynamically updating the operation queue and machine availability through masking, the system can maintain adaptive, real-time scheduling in complex and uncertain production environments.

In summary, the proposed AMDP, formulated based on the DFJSP framework, is capable of addressing multiple dynamic objectives, exemplified in this study by makespan and energy consumption. This formulation underscores the flexibility of the approach, demonstrating its applicability across a wide range of real-world shop-floor environments. The model's inherent adaptability enables real-time responsiveness to fluctuations in job priorities, machine availability, and processing times, thereby supporting efficient, robust, and resilient scheduling under dynamic production conditions. Furthermore, the developed AMDP provides a versatile foundation for the implementation of various machine learning algorithms.

IV. MRL BASED METHOD FOR DFJSP

As mentioned above, MRL can be employed to quickly adapt to new tasks with minimal data and make decision, which is required by DFJSP. A general conceptual framework, tailored for applying MRL to DFJSP scenarios, is proposed in this section. This framework serves as a blueprint for integrating the necessary components that facilitate efficient learning and adaptation in response to real-time changes on the shop floor. Following this, the integration of MAML and PPO is explored and implemented within the proposed framework to evaluate its performance in addressing DFJSP challenges, particularly in terms of rapid adaptation and optimization under fluctuating operational conditions.

A. The Overall MRL Framework for DFJSP

As shown in Fig. 3, the conceptual framework of MRL for DFJSP consists of two phases: training and adapting. The training phase is responsible for training the MRL model using a diverse group of tasks where each task interaction is modeled as an MDP. The adapting phase fine-tunes the trained MRL model, enabling it to respond quickly to changes in the manufacturing environment. The following sections provide a detailed explanation of each phase. In the training phase, meta-learning and RL are combined with each other to train the MRL model for DFJSP. Several iterations are executed in the training process. In each iteration, the task-specific parameters θ_i are initialized by the Meta Learner. The current MRL model with θ_i is used to interact with the scheduling environment. During the interaction, a tuple including the state-action-reward-state transitions, denoted as $\{(s, a, r, s')\}$, is obtained. This tuple is commonly referred to as "experience

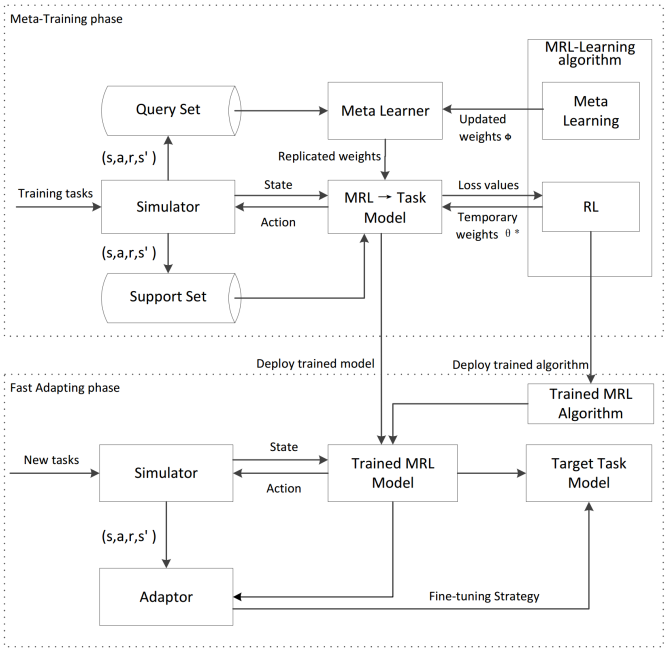

Fig. 3: Overall framework of the MRL based method for DFJSP.

data." To facilitate effective learning, the experience dataset is partitioned into two subsets: the support set and the query set. Specifically, we follow a ratio of 9:1, where 90% of the data is allocated to the support set and 10% to the query set. The support set is employed by the RL component to adjust and refine its parameters based on task-specific loss values, thereby capturing expertise within the given task. In contrast, the query set is used by the Meta Learner to update meta-parameters, ensuring that the learned initialization acquires generalization capabilities for unseen tasks. As a result, the final MRL model is trained by jointly considering both task-specific and cross-task expertise, enabling rapid and adaptive decision-making in DFJSP environments.

During the adapting phase, the trained MRL model is fine-tuned to accommodate new scheduling tasks introduced by the dynamic manufacturing environment. Since the new tasks create new states within the manufacturing environment, the trained MRL model is used to decide which action to execute. Based on the execution of the action on the Simulator, a number of trajectories $\{(s, a, r, s')\}$ are generated and passed to the adapter for further fine-tuning of the model. After several iterations of fine-tuning on the new tasks, a target model suitable for the new manufacturing environment can be obtained. This model can then be used to make optimal decisions for the DFJSP.

B. MAML-PPO Algorithm Applied in the MRL framework

MAML is one of the most representative meta-learning methods [29]. It has been shown to be highly effective because of its flexibility and efficiency in adapting to new tasks. PPO is a highly stable policy gradient RL algorithm that maintain a delicate balance between exploration (trying new actions) and exploitation (leveraging known solutions). Following the

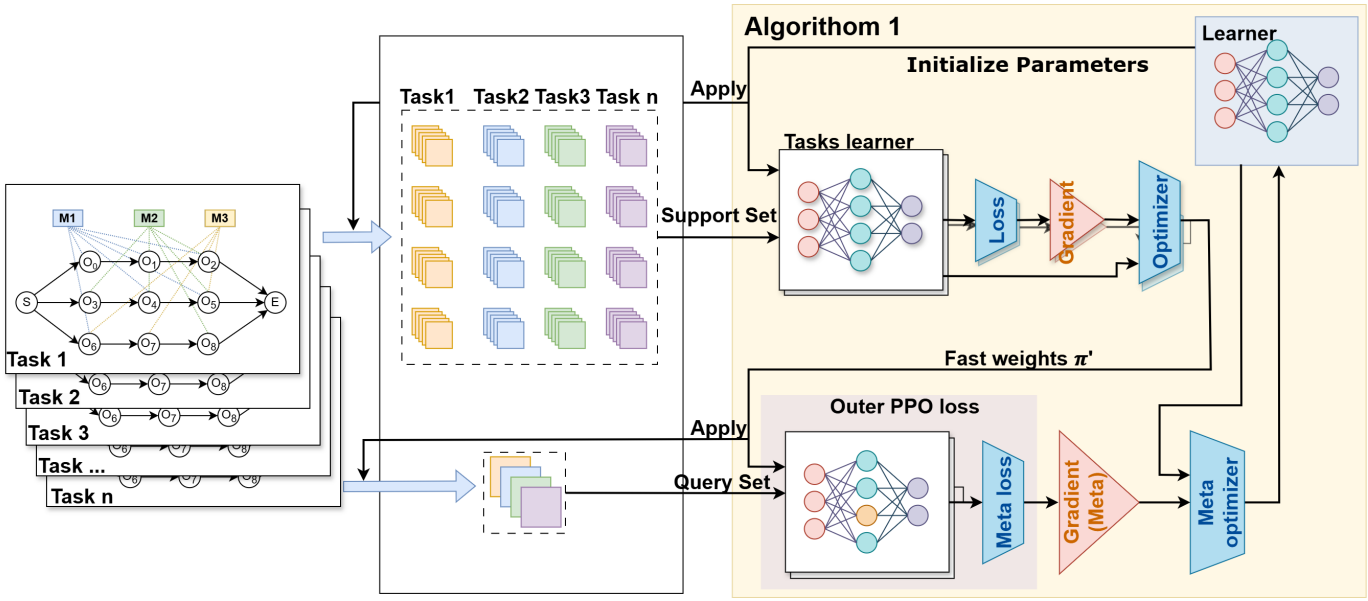

Fig. 4: Illustration of MAML-PPO for Training Process

framework presented in the former section, the MAML-PPO will be described in detail for both the training and the adaptation phases in this section.

1) MAML-PPO for the training phase: As illustrated in Fig. 4, multiple different scheduling tasks are employed to generate the training set. As mentioned previously, the training set is divided into two parts: the support set and the query set. The support set is used by the task learner (i.e. the inner loop of the MAML-PPO) to train the task-specific parameters θ_i for each task's RL model. Meanwhile, the query set is used by the MAML-PPO outer loop to train the MRL parameters Φ. As listed in Algorithm 1, the MAML-PPO for the training phase is composed of two loops: the inner loop (see Line 9-14) and the outer loop (see Line 8-16). All the tasks in the training task set $\mathcal{E}_{train}$ is addressed one by one in the inner loop. The parameters of RL models (i.e. PPO's actor and critic networks), denoted as θ_i are first initialized by the current meta-parameters Φ. These parameters are called task-specific because they are obtained using the data set of a task. Then, the RL models are used to interact with the scheduling environment, which has been modeled as a MDP, to generate policy π_{θ_i} and decide the action to be executed. This interaction yields a support set $\mathcal{M}_{\text{sup}}$, which is composed of some collections of state transitions formatted as $\{(s, a, r, s')\}$. This set captures a sequence of states, actions taken, rewards, and the subsequent states resulting from those actions, thereby forming the basis for the agent to learn. Based on the obtained support set, the task-specific parameters θ_i are updated and optimized using gradient descent method for the PPO loss $\mathcal{L}(\pi_{\theta_i})$, which consists of three main components: the clipped policy loss, the value function loss, and the entropy bonus. Specifically, the clipped policy loss is given by:

$$\begin{aligned}\mathcal{L}^{\text{PPO-clip}}(\pi_{\theta_i}) =& \mathbb{E}_t\left[\min\left(r_t(\theta_i)A_t,\right.\right.\\ &\left.\left.\text{clip}\left(r_t(\theta_i), 1-\epsilon, 1+\epsilon\right)A_t\right)\right]\end{aligned} \tag{14}$$

Algorithm 1 Training Process for Meta-Learning based on MAML-PPO

1: **Input:**
2: Training environment tasks $\mathcal{E}_{train}$
3: Learning rates α, β
4: Meta-parameters Φ
5: **Output:**
6: Trained meta-parameters Φ
7: **Function** TRAIN
8: **for** each iteration $t = 1, 2, \ldots, T$ **do**: *// Outer-loop*
9: **for** $\mathcal{E}_i \in \mathcal{E}_{train}$ **do**: *// Inner-loop*
10: Initialize task-specific parameters $\theta_i \leftarrow \Phi$
11: Generate support set $\mathcal{M}_{sup}$ using current π_{θ_i}
12: Update task-specific parameters

$$\theta_i \leftarrow \theta_i - \alpha\nabla_{\theta_i}\mathcal{L}(\pi_{\theta_i}, \mathcal{M}_{sup})$$

13: Generate task i query replays $\mathcal{M}_{qry,i}$ using π'_{θ_i}
14: **end for** *// End Inner-loop*
15: Update meta-parameters

$$\Phi \leftarrow \Phi - \beta\nabla_{\Phi}\sum_{i=1}^{|\mathcal{E}_{train}|}\mathcal{L}(\pi'_{\theta_i}, \mathcal{M}_{qry,i})$$

16: **end for** *// End Outer-loop*
17: **end function**

where $r_t(\theta_i) = \frac{\pi_{\theta_i}(a_t|s_t)}{\pi_{\theta_i^{\text{old}}}(a_t|s_t)}$ is the importance sampling ratio, A_t is the advantage function, and ϵ is a hyper-parameter that controls the range of the ratio to avoid excessively large updates.

The complete PPO loss function is defined as:

$$\mathcal{L}(\pi_{\theta_i}) = \mathcal{L}^{\text{PPO-clip}}(\pi_{\theta_i}) + c_1\mathcal{L}^{\text{value}} + c_2\mathcal{L}^{\text{entropy}} \tag{15}$$

where:

- $\mathcal{L}^{\text{value}} = \frac{1}{2}(V_{\theta_i}(s_t) - G_t)^2$ is the value function loss, measuring the mean squared error between the estimated value $V_{\theta_i}(s_t)$ and the target return G_t.
- $\mathcal{L}^{\text{entropy}} = -\sum_a \pi_{\theta_i}(a|s_t) \log \pi_{\theta_i}(a|s_t)$ is the entropy bonus, which encourages exploration by penalizing deterministic policies.
- c_1 and c_2 are hyper-parameters controlling the weights of the value function loss and the entropy bonus, respectively.

RL models with updated task-specific parameters are further used to generate improved policies π'_{θ_i} to collect the query set $\mathcal{M}_{\text{qry},i}$ from the environment. Clearly, because of the task-specific training, the generalization and robustness of the current RL models are needed to be improved to fit the dynamic scheduling environment. Thus, the query sets associated with all tasks in $\mathcal{E}_{\text{train}}$ are further used in the outer loop to train the RL models. The meta-loss which is a sum of the losses of all tasks is employed to refine the meta-parameters of the RL models. After several iterations of the inner and outer loops, both task-specific and generalized have been taken into account, and thus the obtained MRL models can be used to support the further quick fine-tuning of the models to respond to dynamic manufacturing environments.

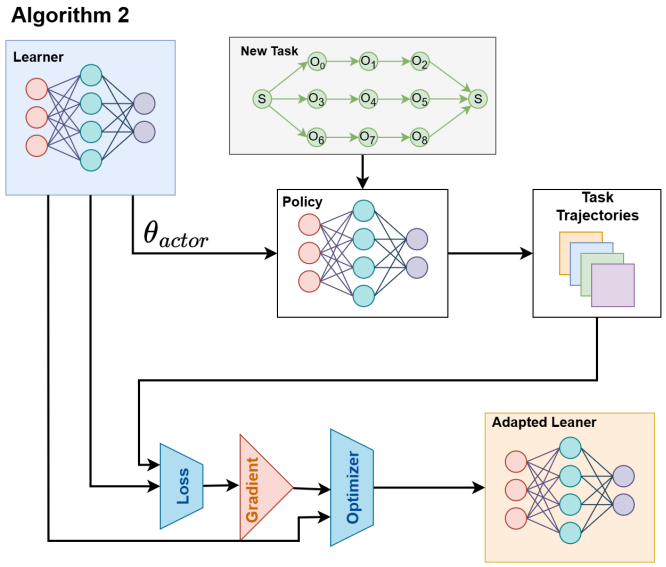

Fig. 5: Illustration of the MAML-PPO for Adapting Process

2) MAML-PPO for the Adapting phase: As illustrated in Fig. 5 and detailed in Algorithm 2, the adaptation phase of MAML-PPO enables the meta-trained model to quickly adjust to new scheduling tasks that are caused by the disturbances occurring in the dynamic manufacturing environment. The process begins by initializing task-specific parameters θ_1 with the meta-trained parameters Φ. The learner (i.e. the agent deployed with the trained MRL models) then interacts with the environment using the current policy π_{θ_j} to generate a small number of task trajectories(typically no more than 5), forming the dataset $\mathcal{D}_{\text{train}}$ (line 5). These trajectories encapsulate essential information about the new task while keeping the sample size minimal to ensure fast adaptation. Based on $\mathcal{D}_{\text{train}}$, the loss $\mathcal{L}(\theta_j, \mathcal{D}_{\text{train}})$ is computed (line 6), and gradient updates are applied iteratively to refine the actor and critic networks (line 7). After N_{iter} iterations (typically $N_{iter} < 10$), the final adapted parameters $\theta_{adapted} = \theta_N$ are obtained. This procedure equips the RL policy with robust adaptability, allowing it to effectively respond to the unique requirements of new scheduling tasks and maintain stable performance under varying operational conditions.

Algorithm 2 MAML-PPO Adapting Phase

1: **Input:** Meta-trained parameters Φ
2: **Output:** Adapted parameters θ_{adapted}
3: Initialize task-specific parameters $\theta_1 = \Phi$
4: **for** each gradient step j from 1 to N **do**
5: Generating $\mathcal{D}_{\text{train}}$ for task using π_{θ_j}
6: Compute loss $\mathcal{L}(\theta_j, \mathcal{D}_{\text{train}})$
7: Update parameters: $\theta_j \leftarrow \theta_j - \alpha \nabla_{\theta_j} \mathcal{L}$
8: **end for**
9: Update adapted parameters $\theta_{adapted} \leftarrow \theta_N$

V. Experiments

A series of experiments were conducted to evaluate the performance of the proposed MRL framework for DFJSPs. To assess the framework's effectiveness of scheduling, generalization capability, and adaptability to novel environments, the experimental instances incorporated various disturbance events, simulating dynamic shop floor conditions. The experiments were carried out on a high performance computational platform, which provided the necessary processing power and memory to handle the computational demands of MRL training and adaptation. The hardware setup consists of two Hygon 7185 processors, each featuring 32 cores with a clock speed of 2.0 GHz, 256GB of DDR4 RAM, and an NVIDIA RTX 3080 GPU equipped with 10GB of GDDR6X memory.

A. Preparatory Works

In this experiment, two preparatory tasks were carried out: dataset preparation and network design. These steps were essential for training and evaluating the proposed MRL framework for DFJSP, ensuring robust performance and adaptability to dynamic manufacturing environments.

1) Datasets: Multiple scheduling tasks are generated to support the training and testing of the MRL models used in the proposed framework for DFJSP. In this context, N_j and N_m denote the number of jobs and machines in the scheduling tasks, respectively. These scheduling tasks are different from each other in N_j or N_m because multiple tasks that share underlying similarities and exhibit distinct differences are required to achieve meta-learning. To simulate the dynamic and complex nature of real shop floor environments, two key factors were taken into consideration. First, dynamic and unpredictable events were explicitly incorporated, including emergency job insertions, machine breakdowns, and scheduled maintenance. Second, the optimization objectives were carefully selected. In manufacturing, makespan and energy

consumption are key performance metrics, but they often conflict—optimizing one may worsen the other. The performance of these two criteria is shown in Fig. 6. The red line represents energy consumption performance, while the blue line shows makespan performance, both normalized across iterations. Between iterations 100 and 250, a clear conflict emerges: optimizing for makespan leads to a significant rise in energy consumption. This highlights the inherent conflict between optimizing makespan and minimizing energy consumption in real-world production settings. Thus, both makespan and energy consumption are incorporated into our dataset, making it a more accurate representation of the challenges encountered in dynamic manufacturing environments.

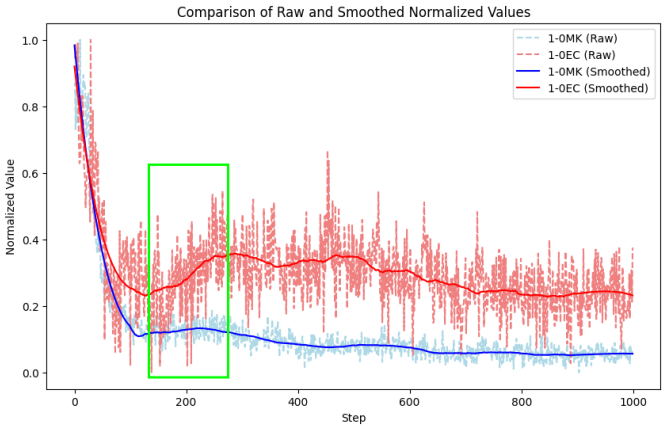

Fig. 6: The trade-off between makespan and Energy Consumption in Scheduling Environment

2) Network Design: Since the DFJSP involves both machines and the operations executed on them, key information is essential for efficient scheduling. This information includes operation attributes, machine characteristics and the interactions between them. To handle this information, our framework deploys several networks. As shown in Fig. 7, the operation and machine data are processed through separate attention networks to retrieve features specific to operations and machines. The output of these attention networks, along with the interactions between operations and machines, is then combined to form a comprehensive feature set. This combined feature set is passed to the PPO networks for decision-making. The PPO network consists of two components: the policy network, which generates actions, and the critic network, which predicts value estimates. Both networks are implemented as multilayer perceptrons (MLPs) with multiple fully connected layers, allowing for efficient processing and decision-making in dynamic scheduling environments.

To ensure robust training, different configurations were systematically tested, and the final hyper-parameters adopted in our MAML-PPO framework are summarized in Table II. Consistent with industrial practice, where meeting due times is often more critical than reducing energy usage, makespan was given higher priority. Specifically, we applied a weighting scheme of 0.8 for makespan and 0.2 for energy consumption. The two objectives can align under certain conditions, extensive testing confirmed that this weighting provides the best

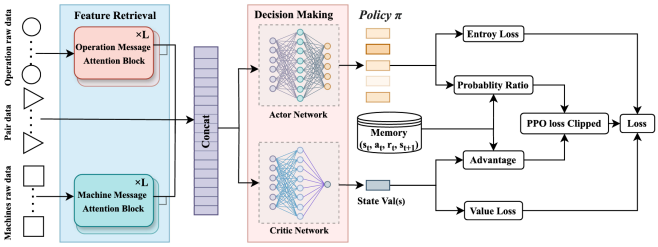

Fig. 7: Network Structure Used in Our Framework

trade-off. This design balances productivity with sustainability, ensuring that the learned scheduling policy remains both efficient and practical for real-world applications.

B. Training Model

TABLE II: Hyper-parameters for MAML-PPO

Hyperparameter	Value
The learning rate α for the inner loop	6e-4
The learning rate β for the outer loop	6e-4
The number of training steps H	1500
The number of tasks in a batch K	4
Update epoch J	4
The rate of samples for each task's support set N	0.8
The rate of samples for each task's query set N'	0.2
The learning rate δ for the fine-tuning	6e-4
The number of fine-tuning steps U	5
Factor makespan	0.8
Factor EC	0.2
Discounting factor	1

The training progress, depicted in Fig. 8, demonstrates the performance of the MAML-PPO approach in optimizing different objectives within the DFJSP. The figure is divided into three rows, each representing the training process under different optimization objectives: makespan (MK), energy consumption (EC), and the combined objective of 0.8MK+0.2EC. For each optimization objective, the corresponding training loss, the mean rewards, and the mean objective values are plotted over time. The results shown in the figure indicate a continuous improvement in model performance as training progresses. The decrease in training loss, increase in the mean reward, and reduction in the mean makespan or energy consumption demonstrate the effectiveness of the combined MAML-PPO approach for solving DFJSPs. Moreover, the figure shows a steady decline in training loss, while the mean rewards increase. This indicates that the model is effectively learning and improving its ability to make better scheduling decisions over time. Additionally, as shown in the figure, the mean objective values (MK and EC) gradually decrease over time. This trend further demonstrates the model's ability to effectively optimize the job shop environment by improving both productivity (through makespan reduction) and energy efficiency (through lower energy consumption). The decreasing values of these objectives indicate that the model is progressively improving its performance in achieving the dual optimization goals. This highlights the model's efficiency in managing competing objectives within the dynamic production environment. However, as depicted in the training data,

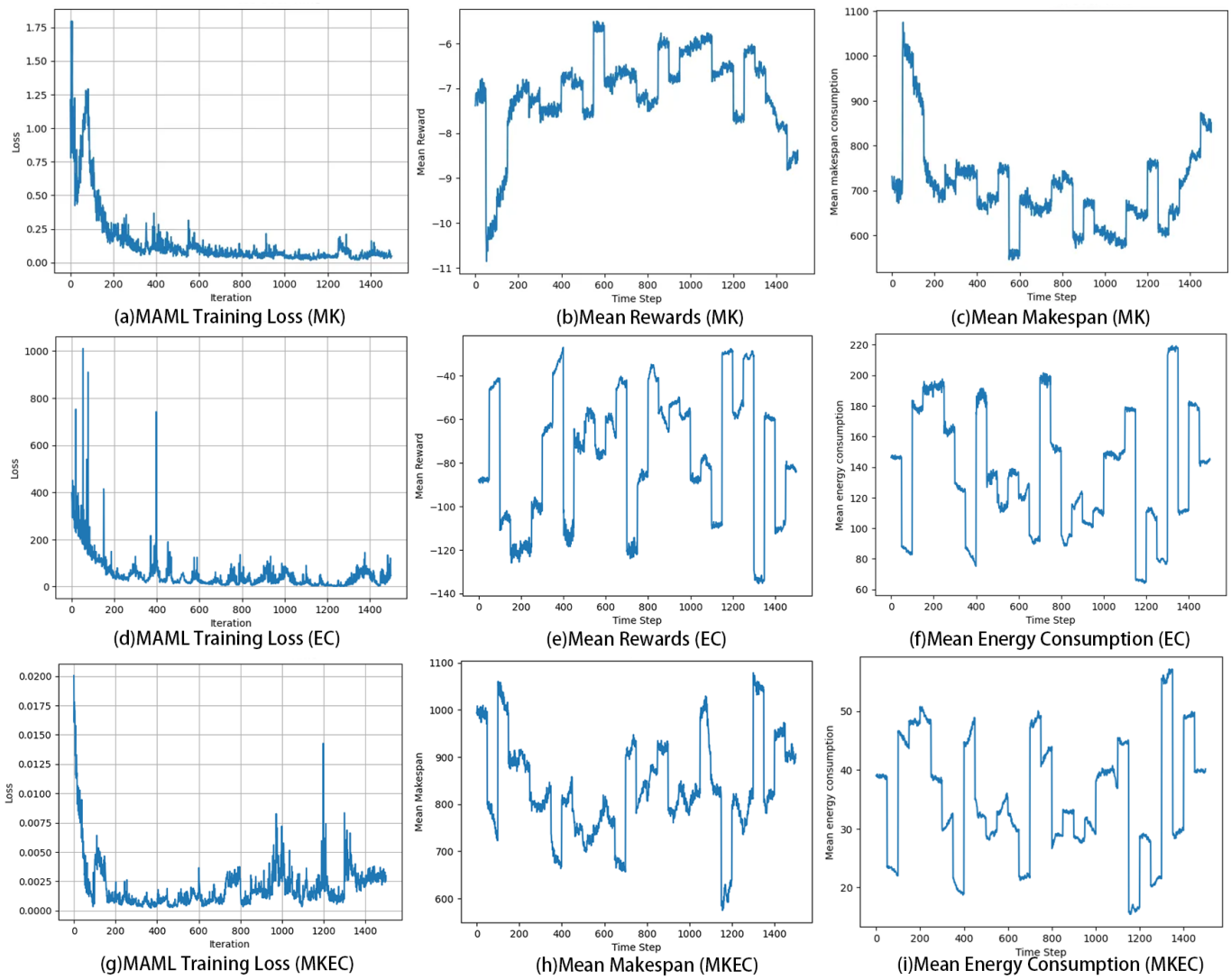

Fig. 8: Training Curves of MAML-PPO on makespan or EC Tasks in the FJSP. **makespan Objective**: (a)MAML Training Loss(MK). (b)Mean Rewards(MK). (c)Mean makespan(MK); **Energy Consumption Objective**: (d)MAML Training Loss(EC). (e)Mean Rewards(EC). (f)Mean Energy Consumption(EC); **0.2EC + 0.8MK Objective**: (g)MAML Training Loss(MKEC). (h)Mean makespan(MKEC). (i)Mean Energy Consumption(MKEC).

fluctuations in rewards and objective values appear every 50 iterations. These fluctuations are linked to the MAML meta-learning environment, where new scheduling tasks are introduced every 50 iterations. As a result, the model experiences a brief period of adaptation to changing tasks and this adaptation leads to a temporary instability in performance. This behavior is not a sign of training failure. It is an expected outcome of the model adjusting to evolving conditions, which is a common feature of meta-learning processes. Another significant observation is that instability in the loss curve emerges during the later stages of MAML multi-objective training. This instability indicates a conflict between optimizing MK and EC, which is a common challenge in multi-objective optimization problems. As the model focuses on improving one objective at a time, it can accidentally degrade the other. This creates a challenge in optimizing both objectives simultaneously. This trade-off between objectives becomes more evident in the later stages of training.

C. Evaluation of Model Generalization and Adaptation

To evaluate the generalization and adaptability of the trained MAML-PPO models in adapting to new tasks, we compared their performance with three baseline models:

1) Independent Task-Specific Training: In this baseline, the PPO models are trained separately on each new task. That is, a specific model can be built for each new task. Thus, it can serve as a reference for evaluating how well each model performs under various optimization criteria.

2) Pre-trained PPO models: The PPO models are pre-trained in the same tasks used to train the MAML-PPO models. The parameters obtained from this pretraining serve as the initial values for the PPO models, which are then fine-tuned to adapt and make decisions for the new tasks.

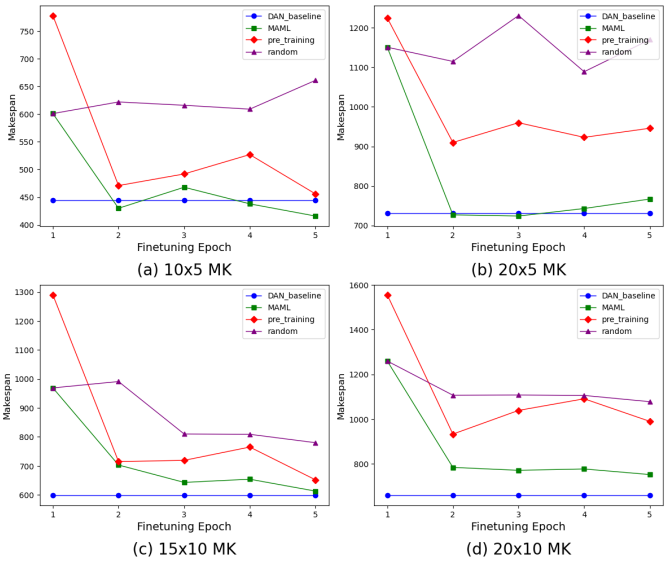

Fig. 9: Adaptation Performance Comparison Among the Models in Scene: makespan. (a) 10×5. (b) 20×5. (c) 15×10, (d)20×10.

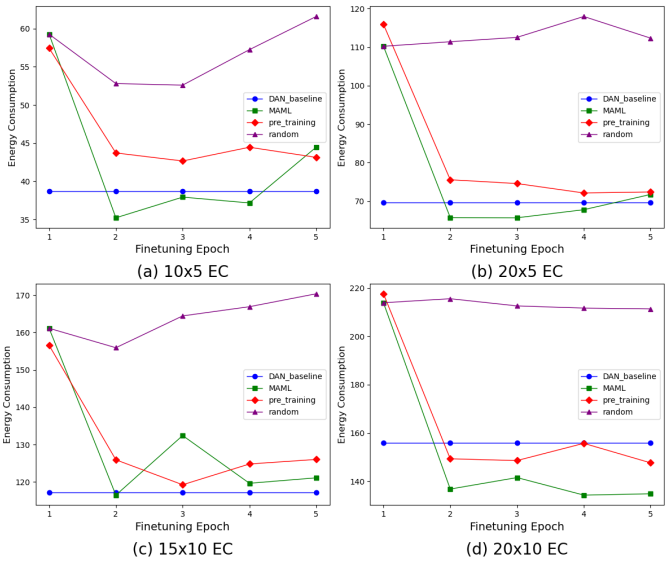

Fig. 10: Adaptation Performance Comparison Among the Models in Scene: Energy Consumption. (a) 10×5. (b) 20×5. (c) 15×10, (d)20×10.

3) Random Initialized models: The PPO models are initialized with random parameters and then fine-tuned via gradient descent to adapt to the new tasks.

For the first baseline, training the models independently on each task allowed for a direct assessment of performance without the influence of prior knowledge or shared initialization. For baselines (2) and (3), the models were fine-tuned to adapt to the new tasks using multiple steps of gradient descent. In the experiments, each fine-tuning process involved up to five gradient updates with 10 sampled trajectories.

Fig.9-11 illustrate the performance comparison between the MAML-PPO model and the baseline models to adapt to the

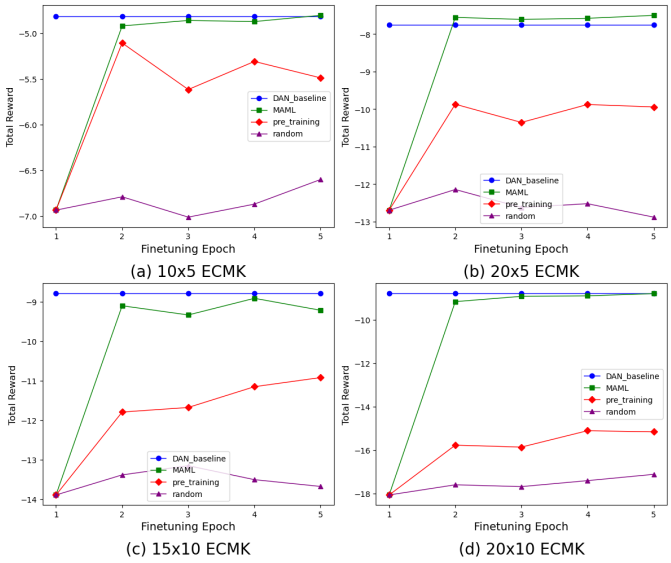

Fig. 11: Adaptation Performance Comparison Among the Models in Scene: 0.8mk + 0.2ec Rewards. (a) 10×5. (b) 20×5. (c) 15×10, (d)20×10.

new task with different sizes: 10×5,20×5, 15×10, and 20×10 (denoting the number of jobs and machines, respectively). The results shown in Fig. 9 were obtained according to the makespan criteria. It is clear that the task-specific training model achieved the best results in scenarios (c and d), as these models are tailored specifically for the new tasks. The results achieved by fine-tuning the proposed MAML-PPO model outperformed those of the task-specific training model in scenarios (a and b). Compared to the pretrained and randomly initialized models, the proposed MAML-PPO model consistently showed the quickest adaptability to new tasks with only five fine-tuning steps. Although the pretrained PPO model showed some improvements during fine-tuning, its performance in the 20×5 and 20×10 tasks was less significant. This was due to the differences between the tasks it was trained on and the new tasks to which it needed to adapt. The randomly initialized model performed the worst in all scenarios. This was because there was a significant gap between its random parameters and the optimal ones required for the new tasks. Similar results can be observed in Fig. 10 where energy consumption was used as the evaluation criterion. It is important to note that some fluctuation appeared in the curves of the proposed MAML-PPO model due to the incorporation of random exploration in the decision-making process. Additionally, Fig. 11 presents the fine-tuning curves in multi-objective environments, where both makespan and energy consumption are included in the reward criteria (see Equations 17-19). The results in Fig. 11 further demonstrate the superior adaptability of the proposed MAML-PPO model compared to the baseline models. These findings collectively show that MAML-based meta-learning allows decision-making models to swiftly adapt to new environments with minimal fine-tuning steps.

As shown in Table III and IV, the proposed MAML-PPO was compared with three representative methods for the DFJSP. The optimal results and the computation time for

TABLE III: Comparison of Adaption Time (AT) (s) and makespan (MK) across Different Methods

Size		MAML-PPO	DANRL	PDR(FIFO)	GA
10 × 5	MK	444.4	416.0	569.4	359.0
	AT	1.50	0.16	0.44	1.01
20 × 5	MK	730.7	767.0	1045.8	718.0
	AT	3.05	0.33	0.90	2.62
15 × 10	MK	598.4	613.0	871.1	756.0
	AT	4.60	0.51	1.36	4.83
20 × 10	MK	659.0	752.0	1088.1	953.0
	AT	6.22	0.70	1.79	5.06

Note: AT denotes the adaptation or model solving time. For MAML-PPO, AT includes both adaptation and problem-solving time.

TABLE IV: Comparison of Adaption Time (AT) (s) and EC (KJ) across Different Methods

Size		MAML-PPO	DANRL	PDR(FIFO)	GA
10 × 5	EC	7379	6432	9789	7344
	AT	1.75	0.62	0.17	4.31
20 × 5	EC	11622	11277	19633	14552
	AT	3.45	1.25	0.33	8.08
15 × 10	EC	17582	17002	32806	21898
	AT	5.02	1.94	1.36	11.19
20 × 10	EC	19412	21617	44061	27453
	AT	7.60	2.65	0.84	14.91

Note: AT denotes the adaptation or model solving time. For MAML-PPO, AT includes both adaptation and problem-solving time.

various instances with different scales were listed in the tables. It is clear that the fine-tuning of the MAML-PPO models was completed in a few seconds that meets the real-world factory response requirements for new scheduling tasks. Just focusing on the makespan objective (see Table III), the fine-tuning time for the MAML-PPO model slightly increased as the task size grew, ranging from 1.50s for smaller instances to 6.22s for larger ones. Although this performance is slower than the DANRL method, the difference is mainly due to the additional adaptation processes inherent in the meta-learning approach. In particular, for specific problem configurations such as 20×10 and 15×10, the MAML-PPO method yielded better results. This highlights its effectiveness in adapting to different shop floor scenarios. While the rule-based method (FIFO) is recognized for its efficiency, its overall performance in optimizing makespan fell short in comparison to the adaptability and effectiveness of our MAML-PPO approach. The genetic algorithm (GA) showed advantages in smaller-scale problems (e.g. 10×5 and 20×5), but its performance declined significantly in larger-scale problems due to the high number of iterations required in the algorithm. For the criteria of energy consumption (see Table IV), the fine-tuning time for the MAML-PPO model increased from 1.75 seconds to 7.60 seconds as the task size grew. This increase is attributed to the more complex nature of the EC objective, which involves additional features (see Equations 3-5) that require computation. The FIFO method continues to maintain high efficiency but became increasingly inefficient as task size grew. Similar to the makespan scenario, the GA experienced a rapid rise in fine-tuning time with larger task sizes. However, it should be noted that the performance of MAML-PPO may not always surpass that of models specifically trained for each individual task, especially when dealing with highly complex problems. In such cases, the performance of MAML-based models may slightly decline, as MAML is inherently designed to generalize across a broad range of tasks rather than deeply optimize for any single one. Despite this, the speed advantage offered by MAML remains significant. For small-scale problems, MAML might only require a few seconds to complete fine-tuning, whereas for larger-scale problems, it may take only a few minutes. In contrast, traditional models trained from scratch for a single task could take several hours or more. The substantial reduction in time and resource costs for training makes MAML an ideal choice for dynamic environments where rapid adaptation to new tasks is crucial. This rapid adaptability makes MAML particularly valuable in time-sensitive situations or environments that require frequent model updates and fine-tuning. MAML's efficiency in adjusting to new conditions makes it a highly promising learning strategy for real-world applications, such as industrial scheduling, autonomous systems, and dynamic resource allocation. By reducing adaptation time while maintaining strong performance, MAML offers great potential for environments that require both quick responses and continuous optimization.

TABLE V: Comparison of Makespan (MK) and Adaption Time (AT)(s) across Different Methods on Large-Scale Instances

Size		MAML-PPO	DANRL	PDR	GA
40 × 5	MK	1302	1299	1660	1589
	AT	4.85	0.75	1.27	3.00
40 × 10	MK	1109	1017	1735	1412
	AT	10.87	1.52	2.75	5.80
40 × 15	MK	895	881	1830	1578
	AT	19.76	2.80	4.60	8.63
40 × 20	MK	913	921	2230	1313
	AT	31.62	3.12	5.83	11.68

Note: AT denotes the adaptation or model solving time. For MAML-PPO, AT includes both adaptation and problem-solving time.

As shown in Table V, the proposed MAML-PPO model was compared with DANRL, PDR, and GA on large-scale DFJSP instances. Two aspects are reported: makespan (MK) and adaptation time (AT). In terms of makespan, MAML-PPO consistently produced competitive or superior results, especially in larger instances (e.g., 40×20, MK = 913), where it significantly outperformed GA (1313) and PDR (2230). This shows that the meta-learning approach remains stable and effective as the problem size grows. More importantly, AT in MAML-PPO does not mean training from scratch. Instead, it represents the deployment time needed to adapt the pre-trained model to a new scheduling task. Even for the largest instances, this time was about 30 seconds, which is acceptable in industrial practice. Compared with traditional methods that require long retraining or many iterations, MAML-PPO can be put into use almost immediately once deployed. Therefore, beyond algorithmic performance, MAML-PPO offers clear industrial value: it eliminates costly retraining, enables rapid deployment, and delivers high-quality schedules under dynamic and large-scale conditions. This makes it particularly

suitable for modern manufacturing systems where both quick responses and reliable optimization are required.

D. Response to Disturbance Events in various Shop Floor Configurations

In real-world manufacturing environments, additional factors such as machine reliability, workforce availability, material supply delays, and unplanned urgent orders often introduce complexities beyond those captured in controlled computational experiments. To evaluate the proposed solution under varying conditions, a set of generalized environment settings was created, varying both the number of jobs and machines. The number of jobs ranged from 5 to 25 in increments of 3, while the number of machines followed the same intervals. The models, including the MAML-PPO model, the model pretrained on the same task sets, and the randomly initialized model, were fine-tuned using five gradient updates.

Fig. 12 illustrated the generalizability of the MAML-PPO model in addressing different FJSPs. Colors were used to represent the ratio of the completion time of the MAML-PPO model relative to the other models (pretrained, specific-task, or randomly initialized models). Importantly, a ratio of less than 1 indicates that the MAML-PPO model performed better than the other models under the same number of fine-tuning steps. Specifically, Figures (a), (b), and (c) highlighted the performance of the MAML-PPO model when it was fine-tuned under makespan. It is clear that the MAML-PPO model outperformed the other methods in most configurations, especially when the number of jobs exceeded the number of machines. This highlights the ability of the MAML-PPO model to adapt effectively to complex scenarios. In Figures (d), (e), and (f), the performance of the MAML-PPO model is shown under energy consumption. Similarly, the MAML-PPO model consistently outperformed the pretrained, specific task, and randomly initialized models in the majority of configurations. This further emphasizes the effectiveness of the model in minimizing energy consumption, even when dealing with varying complexity in job-to-machine ratios. Overall, the MAML-PPO model demonstrates excellent generalization and adaptability across different shop floor configurations.

TABLE VI: Performance Comparison of Different Scheduling Algorithms Under Dynamic Disturbance Events

Event		MAML	DANRL	GA	FIFO
Original	makespan	585	588	949	1012
Schedule	Time(s)	1.56	1.55	2.33	0.31
Periodic	makespan	672	714	984	1071
Maintenance	Time(s)	1.45	1.33	2.85	0.27
Machine	makespan	639	661	977	1087
Breakdown	Time(s)	1.02	1.02	2.14	0.20
Emergency Job	makespan	645	671	1013	1189
Insertion	Time(s)	1.15	1.34	2.48	0.27

To further assess the effectiveness of the proposed MRL framework for DFJSP, three typical disturbance scenarios commonly observed in real-world shop floors were simulated. As illustrated in Fig. 13, these disturbances include periodic maintenance, predictive maintenance with machine breakdown, and emergency job insertions, all occurring during the execution of the original schedule for an initial configuration comprising 17 jobs and 11 machines (17×11). The proposed MRL method was employed to regenerate updated schedules to effectively address and respond to these disturbances. The details of each scenario are elaborated below.

Response to Periodic Maintenance: In this scenario, certain machines temporarily become unavailable due to periodic maintenance. A Gantt chart, illustrated in Fig. 14, visually compares the original scheduling plan with the regenerated schedule. From the figure, machine 4, 5, and 6 (M4, M5 and M6) do not execute any tasks between time units 150 and 270. When the factory specifies the machines scheduled for maintenance and their respective maintenance periods, these machines are set to an unavailable state during the given period to prevent any processing. To maintain production continuity, operations originally assigned to machines undergoing maintenance can be dynamically reallocated to other available machines. However, idle times may still occur before the start or after the completion of the maintenance period, primarily due to the constraints imposed by the FJSP rules. During this period, the system transitions from the initial configuration (17 jobs and 11 machines, labeled as A) to a reduced configuration (17 jobs and 8 machines, labeled as B) and this transition is represented by an $A \rightarrow B$ arrow in Fig. 13, illustrating the shift in machine availability due to maintenance activities. The comparison, shown in Fig. 14, clearly highlights the adjustments made to accommodate machine maintenance. Furthermore, to address this disruption, the original scheduling solution required adjustments. In our proposed MRL framework, the adaptation phase was executed to fine-tune the MRL model and quickly generate an updated scheduling solution, as shown in Table.VI. According to the table, the rescheduling process took only 1.45 seconds, demonstrating the ability of the framework to meet real-time response requirements in industrial applications. Furthermore, the optimal result obtained by our MAML-PPO is much better than the ones obtained by both the dual attention network-based reinforcement learning (DANRL) [6] and GA approaches. These results highlight the effectiveness of the MAML-PPO model in adapting to changes in machine availability.

Response to Predictive Maintenance with Machine Breakdown: Unlike periodic maintenance, predictive maintenance occurs without prior planning, as it depends on predictive analysis of the machine state. Additionally, machine breakdowns often happen suddenly in this scenario. As a result, the scheduling system needs to immediately re-assess the availability of the remaining machines and reallocate the affected jobs to ensure the continuity of the machining process. For example, the process from A (17×11) to C (17×10) in Fig. 13 demonstrates the fine-tuning process when a machine faces predictive maintenance and simultaneously experiences a breakdown. This transition reflects a new task with fewer machines being input into the MRL framework for the DFJSP. As shown in Fig. 15, a machine failure occurred in machine M4 at time unit 120 that caused an interruption. As a result, rescheduling was required to reallocate the operations, originally assigned to M4, to other available machines. From

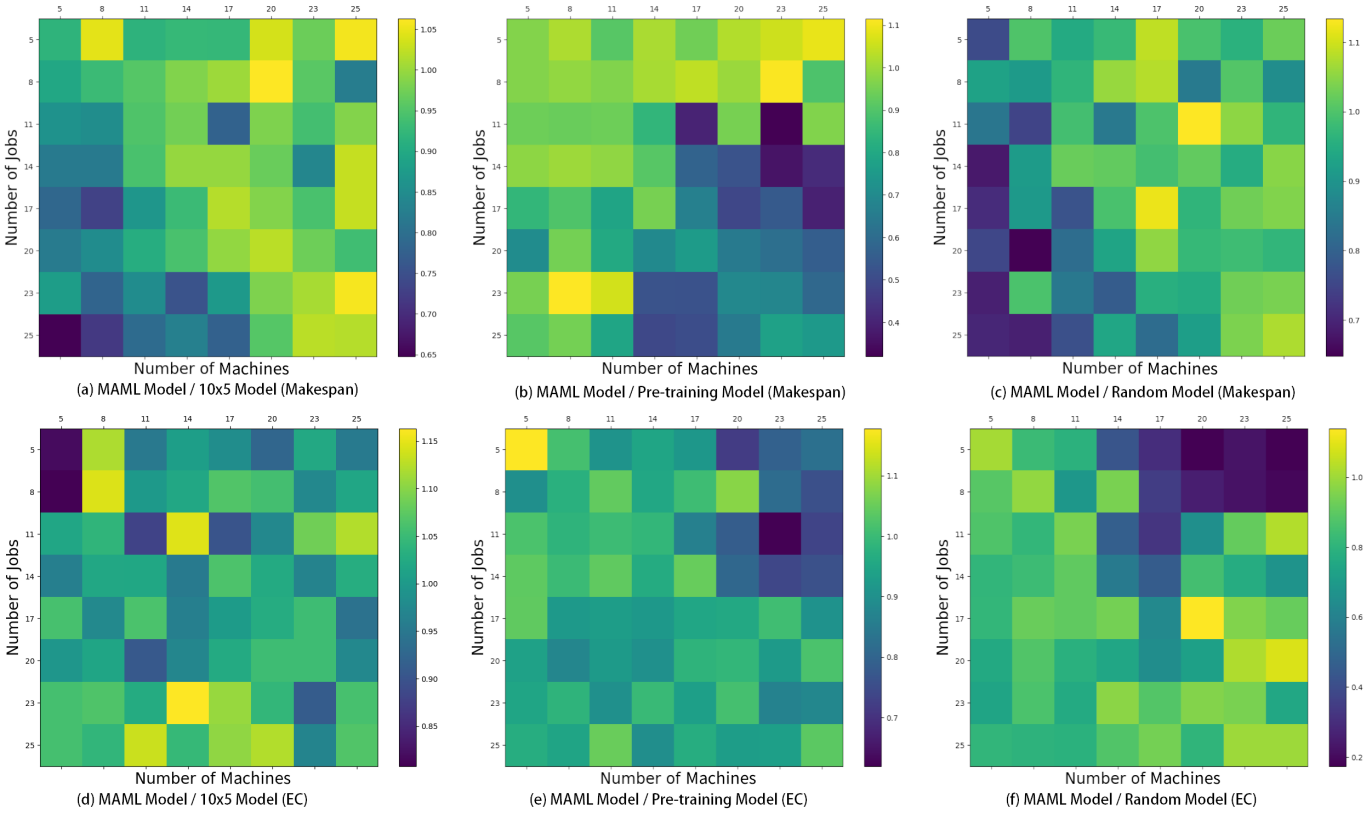

Fig. 12: Comparison of Generalization Performance of Different Models Under Various Disturbance Events. (a) MAML Model / 10×5 Model (makespan). (b) MAML Model / Pre-training Model (makespan). (c) MAML Model / Random Model (makespan). (d) MAML Model / 10×5 Model (EC). (e) MAML Model / Pre-training Model (EC). (f) MAML Model / Random Model (EC).

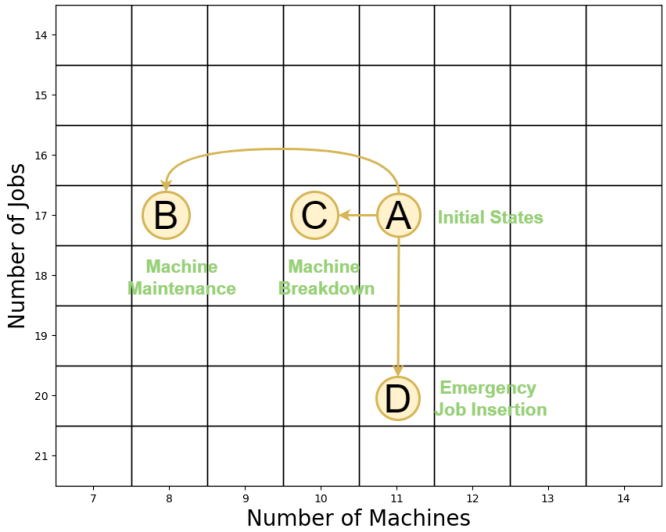

Fig. 13: Adaptive Performance of MAML Model in Response to Disturbance Events

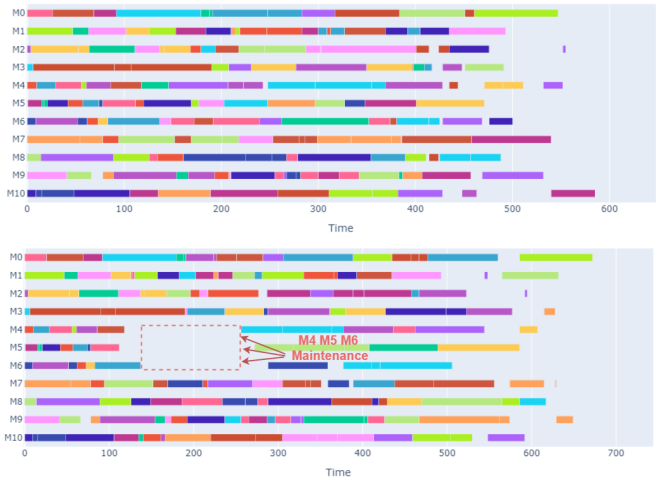

Fig. 14: Gantt Chart Representing Predictive Maintenance

the Gantt chart, a newly generated scheduling plan, which successfully reassigned the extra tasks from M4 to others, was produced through our proposed approach. Due to a new scheduling plan immediately in response to machine failure or predictive maintenance is critical, we compared the new scheduling generation time with other algorithms, listed in Table VI. The results showed that the MAML-PPO model generated a solution in just 1.02 seconds, outperforming the results of the DANRL and GA approaches. This demonstrates that the MRL model can quickly identify jobs that need to be reassigned and evaluate the capabilities of the remaining

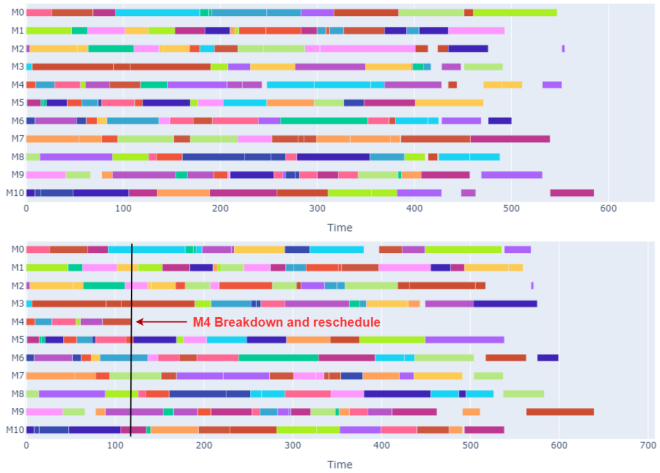

Fig. 15: Gantt Chart Representing Machine Breakdown

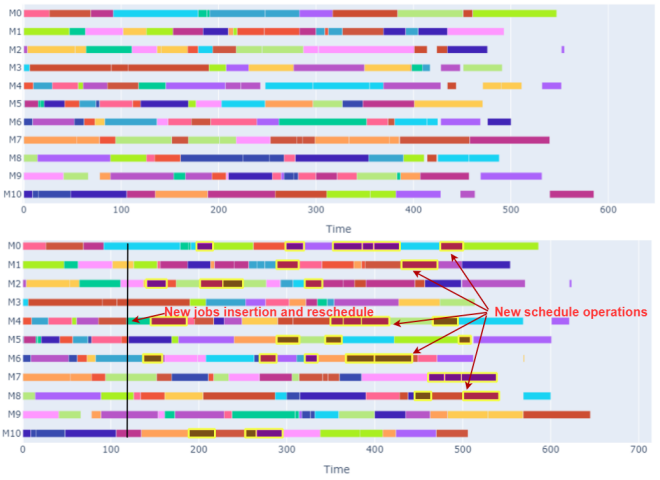

Fig. 16: Gantt Chart Representing Job Insertion

machines. Moreover, the flexibility of the MAML model enables it to adapt to various breakdown scenarios, whether caused by a single machine failure or multiple machines going offline simultaneously. By efficiently redistributing tasks, the model prevents bottlenecks, maintains throughput, and ensures that production targets are met in industrial environments.

Response to Emergency Job Insertion: In contrast to previous disturbances, the insertion of emergency work involves an increase in the number of jobs. For instance (see Fig. 13), the configuration changed from managing 17 jobs on 11 machines A (17×11) to accommodating 20 jobs with the same number of machines D (20×11). Upon receiving the new job request, the job assignments in the existing schedule and the capacity of the available machines are analyzed to develop a new scheduling task. As shown in Fig. 16, when an urgent job is inserted in time unit 120, the proposed MRL model generates a new scheduling plan by leveraging its rapid adaptation capabilities. In the new schedule, the newly added job operations are highlighted in fluorescent yellow in the Gantt chart. The results showed that the MAML model effectively balances the emergency job need with ongoing production demands by optimizing the overall scheduling strategy. This involves making real-time adjustments to the job queue and machine assignments, helping minimize idle time and maximize throughput. Furthermore, shown in Table VI, the proposed MAML model took only 1.15 seconds to generate a new plan to address this issue. This demonstrates the effectiveness of the MAML model in handling such disruptions and highlights its suitability for dynamic manufacturing environments.

These case studies illustrate the exceptional quick response capability and adaptability of the proposed MRL model in managing various disturbance events. It is known that various disturbance events may occur in the real-world manufacturing environment. Additionally, a fast and high-quality response is expected to deal with the sudden disturbances. The proposed MRL model can not only minimize downtime but also maximize throughput by swiftly reallocating resources and optimizing job assignments in response to disturbances—such as machine maintenance, breakdowns, or emergency job insertions. The inherent flexibility of the MRL model ensures that the scheduling system can promptly adjust to real-world production environments, even facing with sudden and unforeseen disruptions.

VI. Conclusion

This paper introduced a novel MRL framework for addressing the complex and dynamic nature of FJSPs. By combining the flexibility of the AMDP with the adaptability of meta-learning, the proposed approach effectively tackles key challenges of a dynamic shop floor environment, such as scalability, adaptability to dynamic events, and sample inefficiency. The modular architecture of the framework accommodates a wide range of dynamic configurations on the shop floor, including machine failures, insertions of emergency work, and machine maintenance, allowing real-time adjustments to scheduling decisions. The incorporation of MAML-PPO algorithms within the proposed MRL framework significantly enhances the system's capacity to generalize across a variety of tasks and environments. Extensive benchmarking of the proposed MAML-PPO algorithm across a range of FJSP scenarios, which include diverse shop-floor configurations, operational constraints, and unforeseen events, reveals that the MRL framework outperforms traditional baseline methods, such as pre-training, random initialization, and task-specific training, particularly in task adaptation efficiency and addressing the complex and dynamic disturbance challenges. Overall, the proposed MRL framework presents a robust and scalable solution for DFJSP, offering substantial improvements in both operational efficiency and flexibility. Future work will explore additional meta-learning and RL techniques, incorporate more diverse and realistic dynamic events, and further evaluate the framework's adaptability in real-world manufacturing contexts, aiming to enhance overall manufacturing performance.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant 61803169 and the Funda-

mental Research Funds for Central Universities under Grant 2662018JC029.

REFERENCES

[1] J. Xie, L. Gao, K. Peng, X. Li, and H. Li, "Review on flexible job shop scheduling," IET collaborative intelligent manufacturing, vol. 1, no. 3, pp. 67–77, 2019.

[2] J. Para, J. Del Ser, and A. J. Nebro, "Energy-aware multi-objective job shop scheduling optimization with metaheuristics in manufacturing industries: a critical survey, results, and perspectives," Applied Sciences, vol. 12, no. 3, p. 1491, 2022.

[3] C. Pickardt, J. Branke, T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness," in Proceedings of the 2010 winter simulation conference. IEEE, 2010, pp. 2504–2515.

[4] L. Zhang and T. Wong, "Solving integrated process planning and scheduling problem with constructive meta-heuristics," Information Sciences, vol. 340-341, pp. 1–16, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516000037

[5] W. Song, X. Chen, Q. Li, and Z. Cao, "Flexible job-shop scheduling via graph neural network and deep reinforcement learning," IEEE Transactions on Industrial Informatics, vol. 19, no. 2, pp. 1600–1610, 2022.

[6] R. Wang, G. Wang, J. Sun, F. Deng, and J. Chen, "Flexible job shop scheduling via dual attention network-based reinforcement learning," IEEE Transactions on Neural Networks and Learning Systems, 2023.

[7] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, "Learning to dispatch for job shop scheduling via deep reinforcement learning," Advances in neural information processing systems, vol. 33, pp. 1621–1632, 2020.

[8] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," arXiv preprint arXiv:1803.11347, 2018.

[9] S.-F. Huang, C.-J. Lin, D.-R. Liu, Y.-C. Chen, and H.-y. Lee, "Meta-tts: Meta-learning for few-shot speaker adaptive text-to-speech," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 30, pp. 1558–1571, 2022.

[10] X. Du, J. Wang, and S. Chen, "Multi-agent meta-reinforcement learning with coordination and reward shaping for traffic signal control," in Advances in Knowledge Discovery and Data Mining, H. Kashima, T. Ide, and W.-C. Peng, Eds. Cham: Springer Nature Switzerland, 2023, pp. 349–360.

[11] A. Dargazany, "Drl: Deep reinforcement learning for intelligent robot control – concept, literature, and future," 2021.

[12] X. Zhang and G.-Y. Zhu, "A literature review of reinforcement learning methods applied to job-shop scheduling problems," Computers Operations Research, vol. 175, p. 106929, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054824004015

[13] H. Wang, J. Cheng, C. Liu, Y. Zhang, S. Hu, and L. Chen, "Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events," Applied Soft Computing, vol. 131, p. 109717, 2022.

[14] J. Kong and Y. Yang, "Research on multi-objective flexible job shop scheduling problem with setup and handling based on an improved shuffled frog leaping algorithm," Applied Sciences, vol. 14, no. 10, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/10/4029

[15] S. Hatami, S. Ebrahimnejad, R. Tavakkoli-Moghaddam, and Y. Maboudian, "Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times," The International Journal of Advanced Manufacturing Technology, vol. 50, no. 9, pp. 1153–1164, 2010. [Online]. Available: https://doi.org/10.1007/s00170-010-2579-5

[16] S. Dauzère-Pérès, J. Ding, L. Shen, and K. Tamssaouet, "The flexible job shop scheduling problem: A review," European Journal of Operational Research, vol. 314, no. 2, pp. 409–432, 2024.

[17] K. Lei, P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, and L. Tang, "A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem," Expert Systems with Applications, vol. 205, p. 117796, 2022.

[18] H. Zhu, M. Chen, Z. Zhang, and D. Tang, "An adaptive real-time scheduling method for flexible job shop scheduling problem with combined processing constraint," IEEE Access, vol. 7, pp. 125 113–125 121, 2019.

[19] Y. Zhang and Y. Li, "Flexible job-shop scheduling problem based on markov state decision process," Journal of Physics: Conference Series, vol. 2825, no. 1, p. 012013, aug 2024. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2825/1/012013

[20] Y. Li, W. Zhong, and Y. Wu, "Multi-objective flexible job-shop scheduling via graph attention network and reinforcement learning," The Journal of Supercomputing, vol. 81, no. 1, p. 293, 2025.

[21] S. Xu, Y. Li, and Q. Li, "A deep reinforcement learning method based on a transformer model for the flexible job shop scheduling problem," Electronics, vol. 13, no. 18, p. 3696, 2024.

[22] H. Wang, J. Zhou, and X. He, "Learning context-aware task reasoning for efficient meta-reinforcement learning," arXiv preprint arXiv:2003.01373, 2020.

[23] C. Liu, Y. Li, C. Huang, Y. Zhao, and Z. Zhao, "A meta-reinforcement learning method by incorporating simulation and real data for machining deformation control of finishing process," International Journal of Production Research, vol. 61, no. 4, pp. 1114–1128, 2023.

[24] Q. Xiao, C. Li, Y. Tang, and L. Li, "Meta-reinforcement learning of machining parameters for energy-efficient process control of flexible turning operations," IEEE Transactions on Automation Science and Engineering, vol. 18, no. 1, pp. 5–18, 2019.

[25] D. G. McClement, N. P. Lawrence, P. D. Loewen, M. G. Forbes, J. U. Backström, and R. B. Gopaluni, "A meta-reinforcement learning approach to process control," IFAC-PapersOnLine, vol. 54, no. 3, pp. 685–692, 2021.

[26] L. Niu, X. Chen, N. Zhang, Y. Zhu, R. Yin, C. Wu, and Y. Cao, "Multi-agent meta-reinforcement learning for optimized task scheduling in heterogeneous edge computing systems," IEEE Internet of Things Journal, 2023.

[27] X. Xiu, J. Li, Y. Long, and W. Wu, "Mrlcc: an adaptive cloud task scheduling method based on meta reinforcement learning," Journal of Cloud Computing, vol. 12, no. 1, p. 75, 2023.

[28] L. Xiong, Y. Tang, C. Liu, S. Mao, K. Meng, Z. Dong, and F. Qian, "Meta-reinforcement learning-based transferable scheduling strategy for energy management," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 70, no. 4, pp. 1685–1695, 2023.

[29] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in International conference on machine learning. PMLR, 2017, pp. 1126–1135.

[30] I. Ahmed, M. Quinones-Grueiro, and G. Biswas, "Performance-weighed policy sampling for meta-reinforcement learning," arXiv preprint arXiv:2012.06016, 2020.

[31] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," arXiv preprint arXiv:1710.09767, 2017.