

Distributed Deep Reinforcement Learning for Dynamic Task Scheduling in Multi-Robot Systems

Peng Song
Beijing Institute of Technology
Beijing, China
songpengcicc@bit.edu.cn

Yichen Xiao
Beijing Institute of Technology
Beijing, China
yichenxiao@bit.edu.cn

Kaixin Cui
Beijing Institute of Technology
Beijing, China
kaixincui@bit.edu.cn

Junzheng Wang
Beijing Institute of Technology
Beijing, China
wangjz@bit.edu.cn

Dawei Shi
Beijing Institute of Technology
Beijing, China
daweishi@bit.edu.cn

Abstract—With the increasing task scale and dynamic complexity of multi-robot systems in automated production lines, we design a dynamic distributed task scheduling framework to accelerate the convergence of the combinatorial optimization model, which is suitable for time-varying multi-task and multi-robot scenarios. By integrating an empirical learning model and a teammate collaboration model, a distributed deep reinforcement learning algorithm is formulated with a limited number of workstations. Each workstation is designed as an independent agent, interacting with the environment to learn the current allocation state of robots. Then, a Q-learning network is trained to extract high-dimensional features from the state and optimize the task scheduling policy. Besides, a greedy strategy is incorporated with the Q-learning network to favor actions that show an increasing trend in Q-values, which enables the algorithm to prioritize higher-priority tasks when facing resource limitations. Simulations with three different workload intensities demonstrate that our algorithm achieves a respective enhancement in overall performance of 3.50%, 8.16%, and 3.86% compared with the foundational deep reinforcement learning models.

Index Terms—Multi-robot system, dynamic task scheduling, distributed deep reinforcement learning.

I. INTRODUCTION

As the intelligent manufacturing industry progresses, multi-robot systems have been deployed to execute large-scale tasks in automotive manufacturing [1], [2], agriculture production [3], and healthcare [4]. In response to the increasing complexity of tasks and dynamic environments, the efficient coordination of a collective of robots to achieve optimal overall performance has received extensive attention.

The scope of this work belongs to the multi-task multi-robot scheduling field, which aims to address a multi-objective optimization problem under time-varying industrial manufacture constraints. A number of studies have been developed for task scheduling, mainly including exact or approximate methods [5]. Exact methods implicitly enumerate all possible

solutions in the solution space through recursively searching. For instance, Bererton [6] introduced a linear programming approach with integer constraints to establish a multi-agent cooperation model. Smriti *et al.* [7] developed a distributed Hungarian method to tackle multi-robot routing issues with spatiotemporal constraints. Approximate methods iteratively refine feasible solutions via construction, search or evolutionary algorithms [8]. Tuncer *et al.* [9] enhanced a genetic algorithm with a novel mutation operator to plan paths for multi-robot systems. Xia *et al.* [10] implemented an improved ant colony algorithm to create a conflict resolution model for multi-AGV systems. However, these algorithms are expensive to be computed in dynamic and large-scale robot scenarios.

In recent years, deep reinforcement learning (DRL) has achieved significant improvements and wide applications [11]. The DRL algorithms combine the advantages of reinforcement learning and deep neural networks, generating data through policy network parameters during environmental interactions, and updating these parameters by the reward feedback mechanism. Park *et al.* [8] adopted a DRL approach and a cross-attention mechanism to quantify the preference between robots and tasks. Zhu *et al.* [12] integrated the DRL technology into robotic mobile fulfillment systems, enabling the efficient processing of large-scale orders. With the exponential increase in task complexity and the number of robots, it is still not yet to know how to design a dynamic DRL optimization models with large-scale tasks, and how to establish an efficient collaboration scheme between multiple robots, which motivates the investigation in this work.

In this work, we consider that robots operate in dynamic environments with substantial task demands on multi-workstation assembly lines. A multitude of robots navigate between a finite set of workstations to fulfill tasks that may arise at any moment. For each workstation, task generation is relatively independent, and upon the arrival of the next task, it can autonomously select an idle and compatible robot to undertake the task. However, each workstation lacks direct insight into the choices made by others and can only

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFE0204100 and in part by the National Natural Science Foundation of China under Grants 62261160575, 62333001 (Corresponding authors: Kaixin Cui.)

make decisions based on communication with the environment regarding the current availability of robots, while inferring the previous selections of other workstations. For the limited and independent workstations, we design each workstation as an intelligent agent and consider a distributed deep reinforcement learning (DDRL) strategy for dynamic task scheduling. Note that DDRL has been applied in some planning problems. Muller *et al.* [13] leveraged distributional RL to capture the full spectrum of reward outcomes and provided an interpretation of dopamine responses. Han *et al.* [14] proposed a potential function based on DDRL to achieve navigational path tracking and formation control. The key challenges of this work are to learn the time-varying task character and the interaction information between robot and workstations in an optimization framework, and to implement an efficient distributed multi-robot collaboration strategy, which can accelerate the network convergence. The main contributions of this work are summarized as follows:

- A distributed task generation and scheduling model, constrained by finite resources and time, is formulated to represent a dynamic set of tasks evolving within a finite Markov Decision Process. Then, a DDRL approach, where each workstation independently interacts with the environment, is designed to achieve the dynamic combinatorial optimization. A Q-learning network is developed to extract state features and optimize the robot allocation, which is updated with a gradient descent strategy.
- A distributed MaxNextQ scheduling algorithm is designed to facilitate collaboration among workstations under conditions of time-varying task information. The action selection policy is refined by integrating the Q-learning strategy with the ε -greedy policy, which favors exploration in the solution space for more promising action combinations. Simulation results show that this algorithm yields the improvement of 3.50%, 8.16% and 3.86% in different workload scenarios, surpassing the performance of conventional DRL algorithms.

II. PROBLEM STATEMENT

In this section, we develop a mathematical model for multi-purpose robots operating within automated factory assembly lines, and subsequently formulate a distributed scheduling problem to address a constrained optimization task.

A. Distributed Task Generation Model

In industrial manufacturing, robot-assisted operations are mainly performed on a grand scale under strenuous workloads, and the number of robots usually far exceeds that of workstations [15]. Moreover, addressing the task scheduling challenge requires the strategic to allocate a series of robots, each equipped with unique capabilities, to traverse among workstations and effectively accomplish a set of tasks generated from a limited number of workstations, as in Fig. 1. At the outset, let the set of workstations and the set of robots in the scenario be denoted by $\mathcal{W}$ and $\mathcal{R}$, respectively. As for workstations, they share the production line environment and can be informed

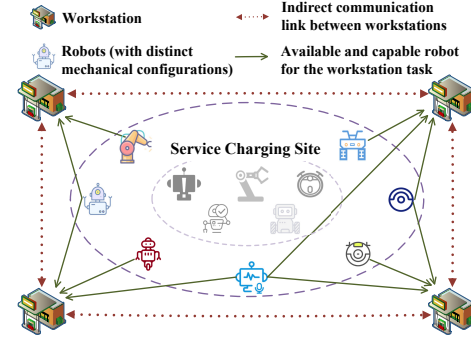


Fig. 1: Illustrative dynamic multi-task multi-robot scenarios.

of the allocation status of robots through communication devices. However, the tasks they are each responsible for are independently generated and executed. Thus, we introduce a distributed task generation model and conduct independent analyses for each workstation to improve the efficiency of task scheduling process. Define the set of tasks generated by the n -th workstation w_n (where $w_n \in \mathcal{W}$) as $\mathcal{H}_n$, where the i -th task is denoted by $h_{n,i}$. Upon the generation of task $h_{n,i}$, its associated task attributes are simultaneously acquired. These details, encompassing the initiation time, execution deadline, duration, and priority, are represented by $ts_{n,i}$, $te_{n,i}$, $d_{n,i}$, and $p_{n,i}$, respectively. A higher value of $p_{n,i} \in \{1, 2, \dots, P_{\max}\}$ means a greater urgency of the task. Note that tasks are produced at any moment, thus upcoming tasks are treated as unpredictable variables.

B. Distributed Task Scheduling Model

Subsequently, the workstations interact with the environment and consider the selection of a suitable robot, or to forgo the task. To address this dynamic optimization challenge, it is essential to initially define the constraints, decision variables, and the objective function.

1) *Time-resource constraints*: The robot's functional architecture must be compatible with the task requirements, and the operational lifespan has to exclude downtime for charging, malfunctions, and maintenance. By integrating these two criteria, the time constraints for assigning task $h_{n,i}$ to robot r_m (where $r_m \in \mathcal{R}$) can be expressed as:

$$\begin{cases} TS_{n,i,m} \geq \max\{ts_{n,i}, as_{n,i,m}\}, \\ TE_{n,i,m} \leq \min\{te_{n,i}, ae_{n,i,m}\}, \end{cases} \quad (1)$$

where $as_{n,i,m}$ and $ae_{n,i,m}$ represent the earliest feasible start time and the latest permissible end time respectively, while $TS_{n,i,m}$ and $TE_{n,i,m}$ represent the real start and end time of the task execution respectively. Besides, the cumulative duration required to execute all tasks may surpass the total unoccupied time for robot set $\mathcal{R}$, indicating the resources limitation. Define the time required for robot to transition as $d_{m,n}$, which is proportional to the distance between robot r_m and workstation w_n . Thus, the total time required to complete

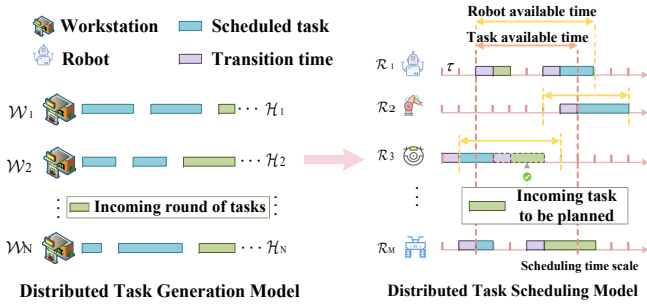


Fig. 2: Schematic of the distributed task generation and scheduling problem.

task $h_{n,i}$ can be expressed as the sum of task duration and transition duration:

$$TE_{n,i,m} - TS_{n,i,m} = d_{m,n} + d_{n,i}. \quad (2)$$

2) *Decision variables*: In the continuous scheduling timeline, we segment the total interval T into $D \in \mathbb{R}^+$ discrete time units, each of duration τ , as shown in Fig. 2. A smaller value of τ corresponds to a more detailed planning granularity at the expense of increased computational complexity. Then, the decision variable is defined as the robot selected to execute the i -th task within $\mathcal{H}_n$, represented by $o_{n,i,m,k}$:

$$o_{m,i,n,k} = \begin{cases} 1, & \text{if } h_{n,i} \text{ is assigned to} \\ & r_m \text{ at the } k\text{-th round,} \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

3) *Objective function*: During each round, every workstation independently make an action selection based on the current state, and the latter of two conflicting actions is considered an invalid action. By optimizing task allocation strategies, the model is designed to accomplish a greater number of tasks, prioritize those with higher urgency, and minimize the dispatch time of robots across various workstations. The distributed task scheduling model is characterized as a multi-objective optimization framework:

$$\begin{aligned} & \text{maximize} \quad \omega_{o,n} O_n + \omega_{p,n} P_n - \omega_{d,n} D_n, \\ & \text{subject to} \quad (1), (2), \end{aligned} \quad (4)$$

where ω_o , ω_p , and ω_d are weight parameters, and O_n , P_n and D_n are the cumulative sum of completed task numbers, priorities, and the movement time, respectively. On a production assembly line, the number of workstations and robots are M and N , and set $\mathcal{H}_n$ contains I_n tasks. Then the objective function is calculated as follows:

$$O_n = \sum_{m=1}^M \sum_{i=1}^{I_n} \sum_{k=1}^{k_{\max}} o_{m,i,n,k}, \quad (5)$$

$$P_n = \sum_{m=1}^M \sum_{i=1}^{I_n} \sum_{k=1}^{k_{\max}} o_{m,i,n,k} \times p_{n,i}, \quad (6)$$

$$D_n = \sum_{m=1}^M \sum_{i=1}^{I_n} \sum_{k=1}^{k_{\max}} o_{m,i,n,k} \times d_{m,n}. \quad (7)$$

Based on the scheduling problem, we aim to address the following questions in this work:

- How to design an efficient planning algorithm to prioritize the completion of tasks with higher importance?
- How to achieve distributed robot collaboration scheduling with only knowledge of the current environment?

III. DISTRIBUTED DEEP Q-LEARNING FOR THE TASK SCHEDULING ALGORITHM

The robot scheduling problem is regarded as a finite Markov decision process [16]. In recent years, DRL algorithms have become prominent in the decision-making optimization field [17]. When confronted with large-scale tasks, the complexity of the model increases significantly, resulting in a substantial slowdown in convergence rates. In this work, we employ DDRL to execute DRL algorithms in parallel, based on the relative independence of task generation and execution at each workstation. Consequently, this methodology enhances the speed of learning and the ability to manage large-scale issues, making it well-suited for complex decision-making spaces.

A. Finite Markov Decision Process

The process is represented by a quintuple comprising states, actions, rewards, discount factors, and learning rates, expressed as $(s, a, r, \gamma, \alpha)$ [16]. In the DDRL algorithm, each workstation acts as an autonomous agent. Therefore, it is essential to evaluate the global properties of these five key elements, which are organized as follows:

1) *State Space*: In each round of robot scheduling, workstations make decisions sequentially based on (3), which entails choosing an unoccupied robot to perform the k -th task. Subsequently, the state space is updated to reflect the allocation status of all robots across the entire time scale. This is represented by an $M \times D$ matrix:

$$S_k^{M \times D} = \begin{bmatrix} s_{k,1,1} & s_{k,1,2} & \cdots & s_{k,1,D} \\ s_{k,2,1} & s_{k,2,2} & \cdots & s_{k,2,D} \\ \vdots & \vdots & \ddots & \vdots \\ s_{k,M,1} & s_{k,M,2} & \cdots & s_{k,M,D} \end{bmatrix}, \quad (8)$$

where k represents the scheduling round. Since that the state space matrix is influenced by the decision at each workstation, it is considered a global variable. Moreover, the elements of this matrix correspond to the mappings of decision variables from all executed decisions:

$$s_{k,m,d} = \begin{cases} 1, & \text{if robot } r_m \text{ is occupied} \\ & \text{at the } d\text{-th time unit,} \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

2) *Action Space*: When workstations interact with the environment, they observe the identical state space $S_k^{M \times D}$. Nonetheless, the task generation and action selection are executed simultaneously among them, without prior knowledge or mutual communication. Therefore, the action space is a distributed variable, which can be expressed as:

$$a_{n,i,m} = \begin{cases} m, & \text{if } r_m \text{ is selected for } h_{n,i}, \\ 0, & \text{if } h_{n,i} \text{ is forgone.} \end{cases} \quad (10)$$

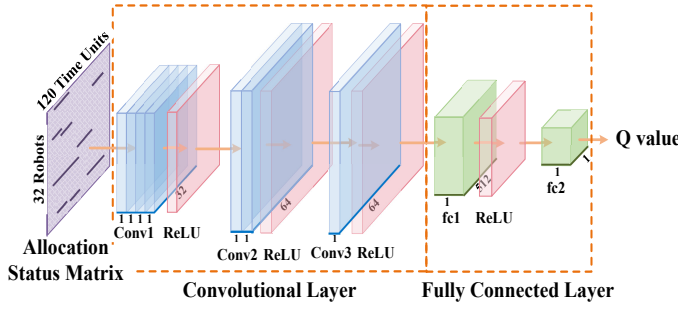


Fig. 3: Structural diagram of Deep Q-Network.

Considering the lack of differentiation among the available execution time slots for the chosen robot r_m , this algorithm prioritizes selecting the earliest continuous units for task execution. Subsequently, the state matrix, as outlined in (8), is updated accordingly.

3) *Reward function*: At workstation w_n , the reward function, representing the feedback signal received by the state matrix after the action execution, is also a distributed variable. According to (4), we are intended to save energy resources and allocate tasks as rationally as possible. Consequently, the reward function can be designed accordingly as:

$$r(S_k, a_{n,i,m}) = \omega_k S_k + \omega_p P_k - \omega_d D_k. \quad (11)$$

4) *Deep Q-Network*: Discount factors γ and learning rates α are both distributed parameters across the Q-learning algorithm. $Q_n(S, a)$ assesses the value assigned to the next action based on the current state and the Q-tables are computed through the Bellman equation [18], thereby determining the optimal action selection strategy. At round k , the workstation w_n selects a robot r_m based on the current task and evaluates the value of this choice through:

$$Q_{n,k+1}(S_k, a_k) = (1 - \alpha)Q_{n,k}(S_k, a_k) + \alpha[r(S_k, a_k) + \gamma \max_a Q_{n,k}(S_{k+1}, a_{k+1})]. \quad (12)$$

In this work, we introduce Deep Q-Networks in a distributed manner for each workstation to calculate Q-values, simulating the process of each workstation interacting independently with the environment to learn optimal strategies. The structural diagram of these networks is depicted in Fig. 3. Given that each workstation observes the environment synchronously and selects actions in parallel, this work has constructed neural networks with identical structure but distinct parameters for each workstation. Concretely, the networks accept the state matrix S_k as input and produces Q-value as output. Firstly, the state matrix is padded with zeros to adjust the dimensions of the input data. Secondly, a series of three 1D convolutional layers progressively diminish the input length while augmenting the feature channels, facilitating feature extraction and transformation. These features derived from the convolutional layers are then flattened and integrated through dense, fully connected layers. Ultimately, the fully connected layers generate the final Q-values utilized for action evaluation within the scheduling problem.

Algorithm 1 The MaxNextQ Scheduling Algorithm

Input: learning rate α , discount rate γ , exploration rate ε , network weights θ , current state S

```

1: Initialize  $L_1, L_2, \dots, L_N$  and  $L$  to be empty lists;
2: for workstation  $n = 1, \dots, N$  do
3:   Initialize the possible action  $A_n = \{0, 1, \dots, M\}$ ;
4:   for action  $a_n \in A_n$  do
5:     if  $a_n = \arg \max_{b \in A_n} Q_n(S, b)$  then
6:        $L_n \leftarrow a_n$ ;
7:     end if
8:   end for
9: end for
10: for  $(a_1, \dots, a_N) \in L_1 \times \dots \times L_N$  do
11:   if  $a_n$  is not conflict with  $a_{n-1}$  then
12:      $S' \leftarrow T(S, a_n)$ ;
13:   end if
14:    $\maxNextQ = \max Q(S', a'_1, \dots, a'_N)$  where  $a'_n$  is
     determined following steps [4:8];
15:   if  $Q(S, a_1, \dots, a_N) < \maxNextQ$  then
16:      $L \leftarrow (a_1, \dots, a_N)$ ;
17:   end if
18: end for
19: if  $L$  is not empty then
20:   Choose a random action pair for  $(a_1, \dots, a_N) \in L$ ;
21: else
22:   Choose a random action pair for  $(a_1, \dots, a_N) \in L_1 \times \dots \times L_N$ ;
23: end if
24: for workstation  $n = 1, \dots, N$  do
25:   if the  $\varepsilon$ -greedy policy chooses to explore at workstation
      $n$  then
26:     Replace  $a_n$  with a random action in  $\{0, 1, \dots, M\}$ ;
27:   end if
28: end for
Output: action pairs  $(a_1, \dots, a_N)$  under the state  $S$ 

```

B. The DDRL Algorithm

After identifying the five key elements mentioned above, we construct and resolve the task scheduling optimization problem in accordance with the flowchart outlined in Algorithm 1.

In the DDRL framework, devising an appropriate action selection algorithm is a pivotal task. The computation of Q-values evaluates the quality of action choices in the current state. However, rather than merely opting for the action with the maximal Q-value, a more efficient decision is to identify actions that exhibit potential for future Q-value enhancement, suggesting an upward trend in Q-values for future decisions. Based on [19], we design a distributed MaxNextQ scheduling algorithm integrated with an ε -greedy policy, to achieve the optimal scheduling performance.

The lists $L_1, L_2, \dots, L_N$ store the set of actions with the maximum Q-values in the current state, while the list L stores the combinations of actions from the aforementioned lists that exhibit a Q-values increase in the next round. After all

workstations simultaneously observe the environmental state, the decision to explore or exploit is determined based on the ε -greedy strategy:

$$a_m = \begin{cases} a_{m,TM}, & \text{with probability } 1 - \varepsilon, \\ & \text{where } a_{m,TM} \text{ in steps}[19:23], \\ a_{m,R}, & \text{otherwise, where } a_{m,R} \sim R(A), \end{cases} \quad (13)$$

where $R(A)$ represents a function for random action selection. This approach ensures sufficient exploration during the initial phases and favors actions that lead to increases in Q-values, thus effectively pursuing optimization objectives.

In the updating phase, an experience replay mechanism is employed as well. For workstation w_n , experience pools D_n are established distributedly to archive experiences consisting of past states, actions, and rewards. The corresponding deep neural network then extracts a batch of experiences from D_m and compute the Q-value estimate y . Meanwhile, the Q-value target $\hat{y}$ is calculated by (12). Thus, the network parameters are iteratively refined using the gradient descent optimization method:

$$\theta_n^{(l+1)} \leftarrow \theta_n^{(l)} - \alpha \nabla_{\theta_n^{(l)}} \mathcal{L}(\theta_n^{(l)}), \quad (14)$$

where $l \in \{1, \dots, L\}$ represents the iteration time, and $\mathcal{L}$ is the loss function used to evaluate the difference between the actual output y and the predicted output $\hat{y}$.

IV. EXPERIMENT RESULTS

To verify the validity of the proposed DDRL algorithm, we simulate a multi-robot task scheduling environment via Python in this section and perform a comparative analysis of the outcomes generated by the DDRL and DRL algorithms under same input conditions and constraints.

A. Parameter Setting

Firstly, we construct a deep neural network based on the PyTorch framework, as depicted in Fig. 3, for valuing action-state pairs and gradually optimizing the decision-making policy through experiential learning. In the forward propagation pathway of this network, the robot's allocation state is inputted, reshaped, and sequentially passed through three convolutional layers: the first convolutional layer (conv1) has 32 filters and a stride of 4, the second (conv2) has 64 filters and a stride of 2, and the third (conv3) has 64 filters with a stride of 1. Collectively, these convolutional operations extract multi-level features from the input state, progressively reducing the dimensionality through downsampling. Subsequently, the features are passed through two fully connected layers: the first (fc1) projects the features into a higher-dimensional space with 512 neurons to augment the model's representational capacity, and the second (fc2) outputs a single value representing the estimated value of an action. A rectified linear unit (ReLU) activation function is applied after each convolutional and fully connected layer to introduce nonlinear characteristics and prevent the vanishing gradient problem. Furthermore, the detailed parameters of the model, after fine-tuning, are presented in Table I.

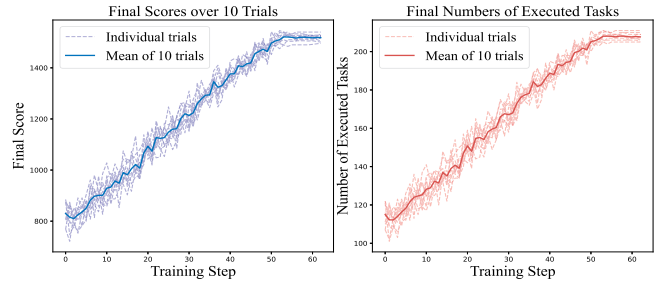


Fig. 4: DDRL Algorithm Performance. The dashed lines illustrate the results of individual simulations, while the solid lines illustrate the average outcomes across the ten simulations.

TABLE I: Hyperparameters in the DDRL Algorithm

Parameter	Value	Meaning
L	60	Iteration size
B	200	Batch size
RM	300	Replay memory size
α	10^{-4}	Learning rate
γ	0.99	Discount factor
ε_0	1	Initial exploration rate
ε_1	10^{-3}	Ultimate exploration rate

B. Performance Comparison

During the simulation phase, the scheduling time scale is confined to 10 minutes, segmented into 120 discrete time units, each lasting 5 seconds. Within this framework, we simulate a scheduling system composed of 3 workstations and 32 robots. Each workstation independently generates tasks and their associated attributes across the 120 time units. We then employ the DDRL algorithm to optimize task allocation for ten repeated times in the same experimental scenario to mitigate the errors introduced by the algorithm's stochastic nature. The outcomes, including the cumulative scores and the total number of tasks across all iterations, are graphically represented in Fig. 4. Across the initial 55 iterations, the learning curves for the final scores and numbers display a sustained upward trajectory with small fluctuations, indicating the model's exploration phase. Following this, from iterations 55 to 60, the output results exhibit a trend towards stabilization, indicating the model's exploitation phase. These results successfully verify the capability of DDRL algorithm to converge to a stable state subsequent to the learning of the optimal policy.

Then, we simulate different scenarios by varying the robot-to-workstation ratio and record the performances of the algorithms across ten repeated experiments. In each scenario, we conduct comparative tests using both DDRL and DRL algorithms under the same constraints and task inputs. The optimization results via two algorithms under varying workloads are visually presented in Fig. 5. Across the three depicted scenarios, the trend lines of both algorithms are initially comparable; however, after the 30th iteration, the DDRL algorithm exhibited enhanced learning capabilities, leading to a certain degree of improvement in both task numbers and task scores. Aggregated results from 60 simulations indicate that

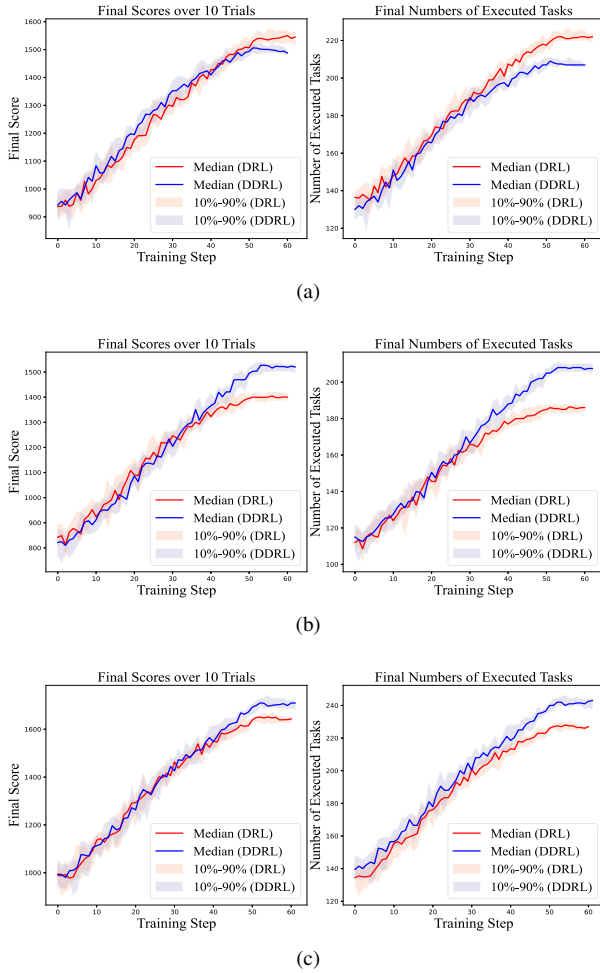


Fig. 5: DDRL and DRL Performance Comparison. Panels (a), (b), and (c) depict three distinct scenarios with robot-to-workstation ratios of 36:3, 32:3, and 28:3, and the total task numbers are 262, 268, and 291, respectively.

the mean total numbers show increases of 7.31%, 11.61% and 6.67%, respectively, and the mean total scores show increases of 3.50%, 8.16% and 3.86%, respectively. These results validate the enhanced exploratory capacity of DDRL algorithm in tackling scheduling optimization challenges. Moreover, it shows that the distributed strategy enables each workstation to account for preferences among different robot types, resulting in more effective environmental observation and more efficient large-scale task scheduling.

V. CONCLUSION

In response to the challenges in large-scale task scheduling, we designed a distributed generation and scheduling model of tasks, tailored to dynamic multi-objective optimization. Then, we proposed a DDRL algorithm to acquire action selection strategies for multi-robot systems with a limited number of workstations, considering constraints on both resources and time. For the workstations, their Q-learning networks shared the same structure but processed distinct parameters, indicative

of the diverse preferences for robot types at each workstation. Additionally, we enhanced the greedy algorithm by incorporating the MaxNextQ scheduling algorithm, which empowers the model to identify and select more promising actions. Our future work will focus on further refining the action selection strategies to facilitate swifter decision-making and execution.

REFERENCES

- [1] S. Karthik, M. N. Rajan, M. N., and V. K., "Creating a 6-axis robot for enhanced assembly line safety in automotive manufacturing," in *2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)*, 2023, pp. 1729–1735.
- [2] G. Chen and W. Zhang, "Hierarchical coordinated control method for unmanned robot applied to automotive test," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1039–1051, 2016.
- [3] X. Han, L. Xu, Y. Peng, and Z. Wang, "Trend of intelligent robot application based on intelligent agriculture system," in *2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM)*, 2021, pp. 205–209.
- [4] Y. Hu, H. Su, J. Fu, H. R. Karimi, G. Ferrigno, E. D. Momi, and A. Knoll, "Nonlinear model predictive control for mobile medical robot using neural optimization," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 12, pp. 12 636–12 645, 2021.
- [5] S. Yoo, Y. Jo, and H. Bahn, "Integrated scheduling of real-time and interactive tasks for configurable industrial systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 631–641, 2022.
- [6] C. A. Bererton, *Multi-robot coordination and competition using mixed integer and linear programs*. Carnegie Mellon University, 2004.
- [7] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, 2017.
- [8] T. Inoue, T. Furuhashi, H. Maeda, and M. Takaba, "A proposal of combined method of evolutionary algorithm and heuristics for nurse scheduling support system," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 833–838, 2003.
- [9] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 2012.
- [10] T. Xia and N. Wang, "Application of improved ant colony algorithm in multiple agv scheduling," *Logistics Technology*, vol. 34, no. 23, pp. 87–89, 2015.
- [11] K. Cui, J. Song, L. Zhang, Y. Tao, W. Liu, and D. Shi, "Event-triggered deep reinforcement learning for dynamic task scheduling in multisatellite resource allocation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 4, pp. 3766–3777, 2023.
- [12] Z.-R. Wang, X.-X. Hu, P. Wei, and B. Yuan, "An improved particle swarm optimization algorithm for scheduling tasks in cloud environment," *Expert Systems*, vol. 41, no. 7, p. e13529, 2024.
- [13] T. H. Muller, J. L. Butler, S. Veselic, B. Miranda, J. D. Wallis, P. Dayan, T. E. Behrens, Z. Kurth-Nelson, and S. W. Kennerley, "Distributional reinforcement learning in prefrontal cortex," *Nature Neuroscience*, vol. 27, no. 3, pp. 403–408, 2024.
- [14] Z. Han, Y. Wang, and Q. Sun, "Straight-path following and formation control of usvs using distributed deep reinforcement learning and adaptive neural network," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 2, pp. 572–574, 2023.
- [15] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A distributed task allocation algorithm for a multi-robot system in healthcare facilities," *Journal of Intelligent & Robotic Systems*, vol. 80, pp. 33–58, 2015.
- [16] M. Vidyasagar, "A tutorial introduction to reinforcement learning," *SICE Journal of Control, Measurement, and System Integration*, vol. 16, no. 1, pp. 172–191, 2023.
- [17] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962–4971, 2023.
- [18] H. J. Kappen, "Optimal control theory and the linear bellman equation," 2011.
- [19] W. Zenzem and M. Tagina, "Cooperative multi-agent systems using distributed reinforcement learning techniques," *Procedia Computer Science*, vol. 126, pp. 517–526, 2018.