Received 1 April 2024, accepted 16 April 2024, date of publication 23 April 2024, date of current version 1 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3392752

RESEARCH ARTICLE

Deep Reinforcement Learning for Task Assignment and Shelf Reallocation in Smart Warehouses

SHAO-CI WU[1], WEI-YU CHIU[2], (Senior Member, IEEE), AND CHIEN-FENG WU[3]
[1]Department of Electrical Engineering, National Tsing Hua University, Hsinchu 300044, Taiwan
[2]School of Information and Technology, Faculty of Science, Engineering and Built Environment, Deakin University, Waurn Ponds, VIC 3216, Australia
[3]Department of Electrical Engineering, National Taipei University, New Taipei City 237303, Taiwan

Corresponding author: Wei-Yu Chiu (chiuweiyu@gmail.com)

This work was supported by the National Science and Technology Council of Taiwan under Grant NSTC 112-2221-E-305-012.

ABSTRACT With the rapid development of online shopping and prosperity of the e-commerce industry in recent years, traditional warehouses are struggling to cope with increasing order volumes. Accordingly, smart warehouses have gained considerable attention for their relatively high efficiency and productivity. In such warehouses, robots transport shelves to picking stations on the basis of tasks assigned to them and then return to the inventory area. An accurate task assignment method must be developed to achieve high efficiency in smart warehouses; however, existing task assignment methods use limited information, resulting in a lack of insight regarding future tasks in warehouses. This paper proposes a method based on the deep Q-network (DQN) that considers inventory for task assignments. The developed DQN-based model determines shelf return locations on the basis of current states to improve warehouse performance. The proposed method was compared with a traditional task assignment method, namely regret and marginal-cost based task assignment algorithm; the results indicated the proposed approach is more efficient and faster than the traditional method and can accommodate more robots.

INDEX TERMS Automated warehouse, deep reinforcement learning, smart warehouse, task assignment.

NOMENCLATURE

$\mathcal{G}$	Graph representing the warehouse map.
$\mathcal{V}$	Set including all vertices in graph $\mathcal{G}$.
$\mathcal{E}$	Set including all edges in graph $\mathcal{G}$.
t	tth time slot.
B	Number of robots in the warehouse.
$p_b(t)$	Position of the bth robot at time t.
$c_b(t)$	Indication that the bth robot is carrying a shelf at time t.
$z(v, t)$	Indication of a static shelf at node v at time t.
U	Number of types of items in the warehouse.
F	Number of shelves in the warehouse.
$q_u(t)$	Quantity of the uth-type item in the warehouse.
$q_{u,f}(t)$	Quantity of the uth-type item on the f-th shelf.
$q_{u,f}^{\text{limit}}$	Upper limit of the quantity of the uth-type item for the fth shelf.
O	Number of orders.
d_u	Total demand of the uth-type item in all orders.
$d_{u,o}$	Demand of the uth-type item in the oth order.
W	Full task list.
W_o	Task list extracted from the oth order.
l_w	Position of the target shelf in task w.
g_w	Position of the target picking station in task w.
τ_w^{start}	Time when task w is assigned.
τ_w^{end}	Completion time of task w.
$P(j, k)$	Cost of the shortest path between node j and node k.

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad AlShabi.

N_a	Number of actions in the task assignment model.
N_v	Number of actions in the shelf reallocation model.
n	Length of the moving average.
$\mathcal{W}_n(t)$	Set of n completed tasks after time t.
$\mathcal{S}$	State space.
$\mathcal{A}$	Action space.
s_t	Observed state at time t.
a_t	Action selected at time t.
r_t	Reward received after action a_t.
R_t	Discounted return at time t.

I. INTRODUCTION

The rapid growth of the e-commerce [1] industry in recent years has presented new challenges and opportunities for logistics and supply chain management [2]. With increasing engagement in online shopping and the increasing demand for fast delivery, warehouses play a crucial role in ensuring efficient order fulfillment and customer satisfaction [3]. Traditional warehouses are struggling to cope with continual increases in the volumes of online orders [4], and this situation has led to growing interest in warehouse automation, which exhibits potential for achieving improved efficiency, accuracy, and cost-effectiveness in warehouses [5].

Smart warehouses–which are equipped with advanced technologies such as robotics, artificial intelligence, and the Internet of Things–have emerged as a solution to address the limitations of traditional warehouses [6]. These state-of-the-art facilities are designed to streamline processes, optimize inventory management, and expedite order fulfillment [7]. A critical aspect of warehouse automation is the efficient assignment of tasks within the warehouse [8].

In a smart warehouse, task assignment involves allocating specific tasks, such as order picking or replenishment [9], to a group of automated systems, including automated guided vehicles, robotic arms, and conveyor systems [10]. The task assignment process plays a vital role in ensuring smooth warehouse operation and maximizing overall warehouse productivity [11]. Efficient task assignment helps to minimize travel distances, reduce congestion, and optimize resource utilization.

The most typical task assignment problem is a classic generalized assignment problem [12], which involves assigning a finite number of tasks to an equal number of agents to minimize the total time or total number of steps required. Such an assignment is known as one-shot task assignment, in which each task is assigned exactly once and in which each agent must perform only one task. Although one-shot task assignment has advantages such as rapid computation and the ability to find the optimal solution, it cannot be used in continuously operating warehouses.

In traditional warehouse environments, the applied order-picking method is often the person-to-goods method, in which a picker travels through the aisles from a station, retrieves the required items from the shelves, and then returns to the station [13]. Many studies have proposed improvements in picking speed by using strategies such as order segmentation and zoning based on the aforementioned method. However, applying such strategies in high-throughput conditions is difficult because such warehouses often have limited pickers. Thus, the utilization of robots has considerable potential for item retrieval [14].

In recent times, goods-to-person warehousing facilities have been increasingly automated, with task assignment often being discussed in conjunction with path-finding methods. Such automation is often achieved through multi-agent picking and delivery [15]. In automated warehouses, robots must handle a continuous flow of tasks and a warehouse's operating system must assign tasks to robots while planning collision-free paths for their navigation. The efficiency of such warehouses is evaluated on the basis of metrics such as makespan and throughput. Chen et al. enhanced the efficiency of task assignment by using a heuristic approach [16]. Specifically, those researchers utilized prioritized planning to evaluate the cost of each assignment and then sequentially select assignments on this basis. This method achieved high efficiency; however, its high computational requirements resulted in a long initial computation time.

The aforementioned studies have assumed that the warehouse's operating system cannot obtain information regarding future orders from its current environment. However, in real-world scenarios, additional information regarding future task situations might be available. In this paper, we propose a warehouse scenario that includes settings for orders and item inventory. In previous studies, a robot's task has been assumed to involve only locations of a starting point and an end point, and the randomly generated locations have led to unpredictability. However, in the method proposed in this paper, tasks are generated on the basis of orders and the current inventory. Although future orders cannot be predicted, information regarding the current inventory is accessible. Intuitively, the higher the inventory amount on a shelf, the higher is the likelihood of this inventory being used in the future.

Although we increased the quantity of available information in our scenario, we still required an algorithm that could handle complex data. Therefore, we focused on learning methods. Reinforcement learning (RL) is a machine learning approach that involves an agent engaging in learning through interactions with its environment and then using learned policies to make decisions regarding subsequent actions [17]. This method is suitable for handling uncertainty and complex decision-making [18] and has achieved success in multiple domains, including video gaming [19], energy management [20], job scheduling [21], and network intrusion detection [22]. Studies have utilized RL to address problems in warehouse environments. Sartoretti et al. proposed the use of RL for multi-agent path finding (MAPF) [23]. Those researchers considered the state to comprise information regarding the obstacles, robot positions, robot paths, and targets in the environment; treated robot movement as the action; and employed deep neural networks to achieve distributed path finding. Li et al. incorporated task assignments with MAPF in a warehouse environment [24]; however, they could not explicitly differentiate between individual tasks because of limitations in the action and state spaces. The present paper proposes a task assignment method based on the deep Q-network (DQN) [25]. The shelf positions and inventory information are considered to be the state, and the generated action determines the shelf that the robot must transport. The agent can gradually learn more effective strategies and implement task assignments on the basis of feedback and rewards.

In the aforementioned research, tasks were often simplified and the return positions of the shelves were not considered. However, in real-world scenarios, after a robot transports a shelf to the picking station, it often needs to return the shelf to the storage area. The location where the shelf must be returned can influence operation efficiency. For example, if the inventory on a shelf is depleted, the shelf is not selected again; in such a case, transporting the shelf to a location far away from the picking station might be appropriate. In this paper, we propose a shelf reallocation method based on the aforementioned concept and the DQN. In this method, shelves' return positions are generated on the basis of the state information for achieving effective shelf management.

The main contributions of this study are listed as follows.

- We developed a warehouse environment that provided additional inventory and order information for task assignment algorithms. In most related studies, tasks have been represented using only a starting point and an end point. However, in practical environments, tasks are extracted from orders in a warehouse. A single order might include multiple types of items, and the inventory on the shelves can change as a picking operation progresses. Accordingly, multiple robots may need to be deployed to fulfill a single order.
- We proposed a task assignment method based on the DQN; in this method, task assignment is performed on the basis of inventory information. After task assignment, a conflict-free path is generated by a path-finding algorithm.
- We proposed a shelf reallocation method based on the DQN which takes the inventory of the shelves into consideration. In this method, return positions of shelves are generated on the basis of the current state. The proposed approach involves the combination of the aforementioned two methods. Simulations were performed in this study to compare the efficiency and runtime of our approach with those of a modified version of regret and marginal-cost based task assignment algorithm (RMCA) [16].

The remainder of this paper is structured as follows: Section II provides an overview of the relevant literature, including that regarding multiagent path finding and RL. Section III presents the research problem and objectives. Section IV describes the RL algorithm proposed in this paper. Section V details the experimental data and relevant comparisons. Finally, Section VI presents the conclusions of this study.

II. RELATED WORK

This section provides an overview of related algorithms used in task assignment problems.

A. MULTI-AGENT PATH FINDING

The goal of multi-agent path finding (MAPF) is to find joint paths in a multiagent system [26], [27], guiding all agents to their destination without collisions. The existing MAPF methods can be broadly categorized into fast solvers, optimal solvers, and approximately solvers [28]. Fast solvers include methods such as Kornhauser's algorithm [29], hierarchical cooperative A* [30], push-and-swap [31], and push-and-rotate [32]. These solvers are characterized by short computation time and low algorithm complexity. However, such algorithms do not guarantee the quality of their solutions.

Optimal solvers include operator decomposition [33], independence detection, M* [34], increasing Cost tree search [35], conflict-based search [36], and constraint programming [37]. These solvers can find the solutions optimally. However, most of these algorithms have long computation times and poor scalability, making them unsuitable for real time applications in a large environment.

Approximate solvers, such as enhanced conflict-based search [38] and extended increasing cost tree search [39], offer a balance between computation time and solution quality. They can significantly reduce computation time and scale to larger map sizes than optimal solvers. They guarantee the quality of solutions within a specific ratio of the optimal solution. However, the runtime of the approximate solvers is still longer than the fast solvers. Therefore, we adopt a fast solver, hierarchical cooperative A*, to accelerate the learning process.

B. REINFORCEMENT LEARNING

RL is a type of machine learning paradigm that enables an agent to learn to make actions by interacting with an environment [18]. The fundamental principle of RL involves the agent learning from trial and error to maximize a notion of cumulative reward. Tabular methods such as Q-learning [40] and SARSA [18] are particularly suitable for small to medium-sized discrete state and action spaces, where the state and action spaces can be enumerated and stored in memory efficiently. Tabular methods are straightforward, easy to implement, and provide valuable insights into RL algorithms' underlying principles. However, they become computationally infeasible for large or continuous state and action spaces, which led to the development of function approximation techniques in RL, such as deep neural networks (DNNs), to address such challenges.

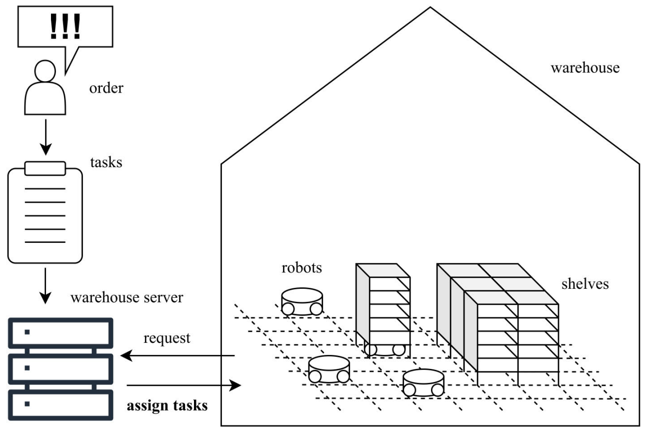

FIGURE 1. Task assignment problem in a multirobot warehouse.

Deep Reinforcement Learning (DRL) methods combine the principles of RL with DNNs, allowing agents to handle high-dimensional state and action spaces [41]. DRL can be categorized into value-based methods and policy-based methods. Value-based RL estimates the value function of different states or state-action pairs, and the policy is derived from the value function [25]. Deep Q-network (DQN) is a famous value-based DRL method known for its simplicity, sample efficiency, and ability to handle large state and action spaces [42]. Extensions such as Double DQN (DDQN) [43], dueling network architectures [44], and prioritized replay [45] are made to enhance the performance of DQN further. They are particularly well-suited for discrete environments. However, value-based methods may face challenges in continuous state and action spaces, where policy-based methods become more relevant.

Policy-based RL directly learns a policy function to make decisions in an environment [18]. Policy-based DRL methods like deep deterministic policy gradient [46], proximal policy optimization [47], and asynchronous advantage actor-critic [48] have demonstrated great success in various applications. These methods have several advantages, such as the ability to handle continuous action spaces and better convergence properties in high-dimensional and stochastic environments. However, they may require more data to achieve comparable performance to value-based methods, especially in high-dimensional state spaces.

In this study, we have adopted DQN with a few extensions to solve the task assignment problem since we are handling discrete action spaces.

III. SYSTEM MODEL AND PROBLEM FORMULATION

This section describes the formulation of the task assignment problem in multirobot warehouses (Fig. 1). After orders are received by a warehouse, they are decomposed into tasks and stored as a list. The warehouse server then allocates these tasks and suitable paths to available robots. The robots pick up items in the warehouse by transporting the required shelves to the picking station. After picking is performed, the robots take the shelves to the assigned locations in the storage area. It is desirable to minimize the cost and maximize the throughput through task assignment and shelf reallocation.

A. WAREHOUSE MODEL

The warehouse considered in this study was formulated as a two-dimensional, four-connection grid map, in which each vertex is connected to its neighbors in four directions. The term $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes the warehouse map, where the vertex set $\mathcal{V}$ includes all positions and where the edge set $\mathcal{E}$ represents the connections between vertices [16]. In the research problem, we considered a discrete-time model $t \in \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. The term $B \in \mathbb{N}$ denotes the number of robots in the warehouse, and $p_b(t) \in \mathcal{V}$ denotes the position of the bth robot in time slot t, where $b \in \mathbb{N}$ and $b \leq B$. In every time slot, robot b can stay at the same position (i.e., $p_b(t) = p_b(t+1)$) or move along the edge such that $(p_b(t), p_b(t+1)) \in \mathcal{E}$. Each robot can carry one shelf in any time slot. Let $c_b(t) \in \{0, 1\}$ indicate whether the bth robot carries a shelf in time slot t, and let $z(v, t) \in \{0, 1\}$ indicate a static shelf on vertex $v \in \mathcal{V}$ in time slot t. A robot can move under static shelves when it is not carrying another shelf; that is, $c_b(t) = 0$. Once a robot picks up a shelf, it can no longer travel under static shelves. To avoid collisions, the following constrains are made:

$$p_b(t) \neq p_{b'}(t) \quad \forall b, \forall t, b \neq b', b' \leq B \tag{1}$$

$$(p_b(t), p_b(t+1)) \neq (p_{b'}(t+1), p_{b'}(t)) \quad \forall b, \forall t, b \neq b', b' \leq B \tag{2}$$

$$c_b(t) + z(p_b(t), t) \leq 1 \quad \forall b \leq B, \forall t. \tag{3}$$

Constrain (1) indicates that robots cannot pass through the same position in the same time slot;constrain (2) prevents robots from moving along the same edge in opposite directions in the same time slot; Constrain (3) implies that a robot cannot travel through shelves when carrying one.

B. ORDER AND INVENTORY MODEL

The warehouse is assumed to contain specified quantities of $U \in \mathbb{N}$ types of items. The term $q_u(t)$ represents the total quantity of the uth item type in time slot t, in which $u, q_u(t) \in \mathbb{N}$ and $u \leq U$. All the items are randomly placed on shelves, and each shelf has a limitation on the quantity of each type of item that it can hold. The total number of shelves is denoted as F, the quantity of the uth item type on the fth shelf in time slot t is denoted as $q_{u,f}(t)$, and the upper limit of the quantity of each item type on a shelf is denoted as $q_{u,f}^{\text{limit}}$. The shelves are assumed to be fully stocked initially; that is, $q_{u,f}(0) = q_{u,f}^{\text{limit}}$. Moreover, all the items are assumed to be located on the shelves initially; thus, we can calculate the total quantities of item by $q_u(t) = \sum_{f=1}^{F} q_{u,f}(t), \forall u; \forall t$.

An order is composed of required quantities of one or multiple items. The term d_u represents the total demand for the uth item type. Assuming that $O \in \mathbb{N}$ orders exist, the demand for the uth item type in the oth order is denoted as $d_{u,o}$, where $o, d_{u,o} \in \mathbb{N}$ and $o \leq O$. Thus, we can calculate the total demand as $d_u = \sum_{o=1}^{O} d_{u,o}, \forall u$.

The total demand for all the orders should not exceed the inventory in the warehouse; that is, $d_u \leq q_u(0), \forall u$. To fulfill the demand, every order must be divided into more minor tasks, each of which must include the positions of a shelf and picking station. The term W_o denotes the list of tasks obtained from the oth order. Each task $w \in W_o$ is associated with a tuple $(l_w, g_w, \tau_w^{\text{start}}, \tau_w^{\text{end}})$, where l_w represents the coordinates of the target shelf of w, g_w represents the coordinates of the picking station of w, τ_w^{start} denotes the time when w is assigned, and τ_w^{end} represents the completion time of w. The complete task list W constitutes the union of task lists obtained from all the orders and can be expressed as $W = \bigcup_{o=1}^{O} W_o$.

C. OBJECTIVES

Objectives such as the makespan and sum of costs can be optimized in task assignment problems. Makespan represents the time required to complete all given tasks, whereas the sum of costs represents the total cost incurred by all the robots. These objectives are often used in one-shot problems. However, in a lifelong scenario, the optimization of a metric that indicates throughput is preferred to enhance efficiency. In the present scenario, we made cost per task (CPT) as one of our optimization objectives, which can be calculated as

$$CPT = \sum_{w \in W} \frac{\tau_w^{\text{end}} - \tau_w^{\text{start}}}{|W|}. \tag{4}$$

where CPT stands for the average time consumed by a task. Another objective optimized in the research problem is total relative cost (TRC), which can be expressed as:

$$TRC = \sum_{w \in W} (\tau_w^{\text{end}} - \tau_w^{\text{start}} - P(l_w, g_w)) \tag{5}$$

where $P(j, k)$ denote the lowest path cost between nodes j, $k \in V$.

Equation (5) calculates the difference between the actual time spent by robots in performing a set of tasks and the ideal time required to perform the same tasks. The relative cost provides the agent with an understanding of how much additional time has been spent by a robot on a task and prevents the agent from selecting actions solely according to the task duration.

IV. METHODOLOGY

This section introduces a DQN-based dispatch system for task assignment and shelf reallocation. Fig. 2 displays the flowchart of the entire process performed by robots in a warehouse. This process comprises a path-finding module and two DQNs that solve the task assignment problem and determine the positions of returning shelves. If a robot is available, it sends a request to the task assignment system. This system then selects a task from the task list according to the current state in the warehouse and generates a collision-free path for the robot by using the path-finding module. After receiving a task from the task assignment system, the robot follows the provided path and completes its work at a picking station. It then sends a request to the shelf reallocation system, which selects a returning position and generates a path to this position by using the path-finding module. The robot becomes available again upon its return to the storage area.

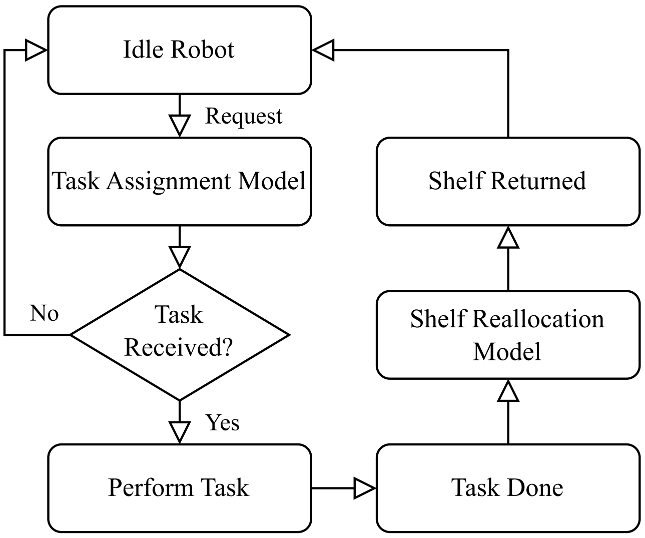

FIGURE 2. Flowchart of the entire process performed by robots in a warehouse.

A. REINFORCEMENT LEARNING

To effectively address uncertainty and utilize the large quantity of data available within a warehouse, we employed RL to solve the task assignment problem. This subsection provides a brief description of RL. In time slot t, the RL agent observes the current state $s_t \in \mathcal{S}$ and selects an action from the policy $a_t = \pi(s_t) \in \mathcal{A}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ denotes the action space, and $\pi(s_t)$ is the policy that generates an action based on state s_t. After interacting with the environment, the agent receives a reward r_t and observes the next state s_{t+1}. The goal is to find a policy that maximizes the discounted return $R_t = \Sigma_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$, where $\gamma \in [0, 1]$, where $\gamma \in [0, 1]$ denotes the discount rate. Three value functions are used to evaluate and improve the policy, namely the Q-value, state value function, and advantage function. The Q-value of a state-action pair (s, a) is defined as the expectation of the discounted return:

$$Q(s, a) = \mathbb{E}\big[R_t | s_t = s, a_t = a\big]. \tag{6}$$

The return obtained on the basis of the Q-value in the next state by using the policy π can be calculated as:

$$Q(s, a) = \mathbb{E}\big[r_t + \gamma \mathbb{E}[Q(s_{t+1}, \pi(s_{t+1})) \,|\pi] \,|s_t = s, a_t = a\big]. \tag{7}$$

The state value function can be expressed as:

$$V(s) = \mathbb{E}\big[Q(s, \pi(s)) \,|\pi\big]. \tag{8}$$

The advantage function is expressed as:

$$A(s, a) = Q(s, a) - V(s). \tag{9}$$

The advantage function represents the importance of each action [49]. Tabular solution methods often use the value functions in (6)-(9) to address small state spaces. In high-dimensional state and action spaces, deep neural networks are used to approximate the aforementioned three value functions. The functions $Q(s, a; \theta)$, $V(s; \theta)$, and $A(s, a; \theta)$ denote the value functions approximated using the network weight θ.

B. TASK ASSIGNMENT METHOD

This subsection describes the task assignment problem in the RL framework. In particular, details regarding the state representation, action space, and reward design are provided.

1) STATE SPACE FOR TASK ASSIGNMENT:

Because the task assignment problem is handled by a centralized system, all information except that regarding future orders is assumed to be available. To obtain sufficient data, a map of the entire warehouse was used to determine the state. The available information was divided into the following channels:

1) Channel 1: The position of a selected robot
2) Channel 2: The positions of all other robots
3) Channel 3: The positions of all obstacles
4) Channel 4: The positions and inventory of all static shelves
5) Channel 5: The traffic information related to the other robots
6) Channel 6: channels for task information

Channels 1-3 comprise binary maps that indicate the positions of objects. Channel 4 indicates the locations of all static shelves, with the corresponding values representing the inventory ratio of goods on each shelf. This ratio can be calculated as:

$$inventoryRatio(f) = \sum_{u=1}^{U} \frac{q_{u,f}(t)}{q_{u,f}^{\text{limit}} \times U} \tag{10}$$

where f is the index of the shelf. Previous studies have assumed that all tasks are unknown to the system before their release time. However, we believe that the inventory information can provide the learning network with insights regarding future tasks. The intuition is that the more items a shelf holds, the more likely is that shelf to be included in a task.

Channel 5 represents the path information of other robots. Our aim was to integrate all the path information into the state. However, the complete representation of the paths of numerous robots can lead to an increase in the number of state dimension, which results in a decrease in learning speed. To avoid this problem, all the path information can be merged into an abstract map, which represents the total time that each position will be occupied by a robot in the future.

In channel 6, N_a channels are added to the input, where N_a is the number of actions customized by the user. A total of N_a tasks are selected from the task list in advance, and each task is converted into a map. Each map contains information regarding the shortest path required to complete that task. The N_a tasks are selected according to the shortest distance between the shelves and the robots, and the shortest path is calculated by a single-agent A*, with collisions among all agents being ignored.

2) ACTION SPACE FOR TASK ASSIGNMENT:

In the state space, we selected N_a tasks as candidates. Consequently, in the present case, the actions involved choosing one of the N_a tasks. Training may involve situations where invalid actions are generated. For example, when the number of remaining tasks is smaller than N_a, resulting in vacancies in the action space, actions with no actual tasks cannot be selected. In such cases, the final action is forced to be chosen from one of the valid actions.

3) REWARD FUNCTION FOR TASK ASSIGNMENT:

As mentioned in section III, our goal was to minimize TRC through (5), which involved maximizing its negative value. In RL, this value can be treated as the return, which is the accumulation of discounted rewards. Consequently, our reward was simply designed as the relative cost:

$$reward = P(l_w, g_w) - \tau_w^{\text{end}} + \tau_w^{\text{start}}. \tag{11}$$

This reward design prevents the agent from relying solely on the path length to judge the quality of tasks. If the task duration is directly used as the reward, the agent might be misled into prioritizing the selection of the shortest tasks; however, the longer tasks must still be performed eventually. It is desired to enable agents to select suitable tasks on the basis of factors such as traffic conditions and shelf positions to prevent these tasks from getting caught in traffic, which results in decreased efficiency. Therefore, we used a reward that subtracted the actual time spent on performing a task from the theoretical time that should be spent on performing said task. This approach ensured that the longer the required waiting time for a task, the lower was the reward.

C. SHELF REALLOCATION METHOD

This subsection describes an RL method for shelf reallocation, including details regarding the state representation, action space, and reward design.

1) STATE SPACE FOR SHELF REALLOCATION:

The RL model used for shelf reallocation shares similarities with the task assignment model, with both models using similar information. The states used in the shelf reallocation approach are listed as follows:

1) Channel 1: The position and inventory of the returning shelf
2) Channel 2: The positions of all robots apart from the selected robot

3) Channel 3: The positions of all obstacles
4) Channel 4: The positions of the starting points of the current tasks
5) Channel 5: The traffic information related to the other robots
6) Channel 6: Additional channels for shelf vacancies

Channel 1 indicates the location of the returning shelf, with the value representing the inventory ratio of goods on the shelf (10). Intuitively, shelves with lower inventory amounts are suitable for placement farther from the picking station, and this information might help the agent determine a more suitable return location. Channels 2-4 comprise binary maps that indicate the positions of objects. As in the task assignment model, channel 5 in the shelf reallocation model contains an abstract map representing future traffic conditions. Finally, channel 6 comprises N_v channels added to the input. A total of N_v candidates are preselected for return positions, and each of these candidates is converted into a map that contains information regarding the shortest path to the shelf. These candidates are selected at various distances from the picking station to maximize the degree of variety in the actions.

2) ACTION SPACE FOR SHELF REALLOCATION:

Similarly, because N_v candidates are selected in the state space, the corresponding actions involve selecting one of N_v positions. During training, if invalid actions are generated, the actual action is forced to be selected from the set of valid actions.

3) REWARD FUNCTION FOR SHELF REALLOCATION:

The target of optimization is CPT, which represents the average cost incurred by the tasks. The CPT value provides the agent with an understanding of the level of efficiency within the facility. Notably, CPT serves as a valuable metric for computing throughput in a lifelong environment. However, in (4), information regarding all tasks, including future ones, is required for value calculation. A practical solution to this problem is to calculate the moving average of CPT (mCPT) as:

$$mCPT(t) = \sum_{w' \in \mathcal{W}_n(t)} \frac{P_{w'}}{n} \tag{12}$$

where n is the length of the moving average defined by the user and where $\mathcal{W}_n(t)$ denotes the set of n completed tasks that start after time t. In RL, mCPT is minimized through the maximization of its negative value. Therefore, the reward for completion of task w is expressed as:

$$reward = -mCPT(\tau_w^{\text{end}}) \tag{13}$$

where τ_w^{end} is the time required to complete task w.

D. DEEP Q-NETWORK

The algorithm proposed in this paper is based on a dueling double DQN with prioritized replay that incorporates several extensions [44]. To avoid overoptimistic value estimates, a double DQN uses target weights to evaluate the Q-value of an action while calculating the loss [43]. The target in a double DQN can be expressed as:

$$T_i \equiv r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_i); \theta_{i-1}) \tag{14}$$

where i represents the number of updates. The parameter θ_{i-1} represents the weights of the target network, which are a periodic copy of the online weights θ_i. The actions are determined according to the online weights as:

$$a_t = \arg\max_{a'} Q(s_t, a'; \theta_i). \tag{15}$$

In a dueling DQN, an advantage function is used to compute Q-values [44]; the computed Q-values can then be divided into two components: the state value and advantage values for a specific action. This separation enables the dueling DQN to learn more efficiently and conduct generalization across multiple states. The shared convolutional layers and fully connected layers of the DQN calculate advantage values, whereas separated fully connected layers calculate the state value. The advantage values A and state value V can be combined as follows to acquire the Q-value:

$$\begin{aligned} Q(s_t, a_t; \xi, \alpha, \beta) = V(s_t; \xi, \beta) + A(s_t, a_t; \xi, \alpha) \\ - \frac{\sum_{a'} A(s_t, a'; \xi, \alpha)}{|A|} \end{aligned} \tag{16}$$

where ξ, α and β represent the weights of the convolutional layers, the fully connected layers used for calculating A, and the fully connected layers used for calculating V, respectively.

Typically, all transitions stored in the replay buffer are randomly selected with equal probability. However, some transitions provide little information and thus might not be helpful for learning. In practice, attempts should be made to frequently sample transitions that can accelerate the learning process. Accordingly, we adopted prioritized replay, which uses the probability based on the transition priority for sampling [45]. The priority of a transition is set as the transition's maximum value when its first enters the replay buffer. During learning, the priority of the selected transitions is updated simultaneously, and the priority of transition j is calculated as:

$$\rho_j = \left| r_j + \gamma Q(s'_j, \arg\max_{a'} Q(s'_j, a'; \theta_i); \theta_{i-1}) - Q(s_j, a_j; \theta_i) \right| \tag{17}$$

where ρ_j denotes the priority of the jth transition. The term $\psi(j)$ denotes the probability of sampling transition j:

$$\psi(j) = \frac{\rho_j}{\sum_k \rho_j}. \tag{18}$$

Finally, the loss $L_i(\theta_i)$ is expressed as:

$$L_i(\theta_i) = \mathbb{E}_{j \sim J}\left[\left(T_i - Q\left(s_j, a_j; \theta_i\right) \right)^2 \right] \tag{19}$$

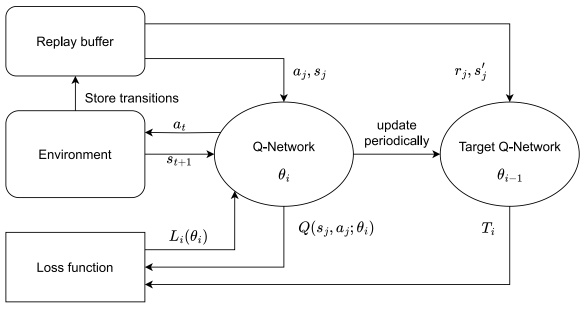

FIGURE 3. **Diagram of the DDQN algorithm.**

where J is the minibatch sampled from the replay buffer with the probability calculated by (18).

Fig. 3 shows the diagram of the DDQN algorithm. After the Q-network interacts with the environment, the reward is given by (11)-(13) and the transitions are stored in the replay buffer. During the training process, samples are selected based on their priority calculated by (17), and the loss is calculated by (14)-(16),(19) using the Q-network and the target Q-network.

V. NUMERICAL RESULTS

This section details our numerical results for task assignment and shelf reallocation in warehouses with sizes of 25 × 22, 37 × 34, and 48 × 46. Fig. 4 displays the warehouse with a size of 25 × 22; in the figure, the blue squares represent shelves, the orange squares represent picking stations, and the red squares represent robots. In the 25 × 22 warehouse, the number of robots was $B \in \{20, 25, \ldots, 70\}$. In the 37 × 34 and 48 × 46 warehouses, the number of robots was $B \in \{40, 50, \ldots, 90\}$. The numbers of shelves in the 25 × 22, 37 × 34, and 48 × 46 warehouses were 224, 528, and 960, respectively. Each warehouse contained $U = 20$ item types, and the maximum quantity for each item type on a shelf was randomly generated from the range $\{5, 6, \ldots, 20\}$. In the 25 × 22 warehouse, we tested $O = 50$ orders, and in the other warehouses, we tested $O = 100$ orders. The parameters of the network of the task assignment agent are displayed in Fig. 5. Because we considered the entire map as our state, the output sizes of the convolutional layers changed with the size of the warehouse. The shelf reallocation model had the same network architecture as did the task allocation model; however, in the shelf reallocation model, the number of input channels was $N_v + 5$, and the size of the action space was N_v. During training, we used a discount rate of $\gamma = 0.95$ and adopted the Adam optimizer with a learning rate of 2×10^{-3}. The adopted networks were trained for 100 episodes for each warehouse. The numbers of actions (N_a, N_v) was set as 5. We computed the moving average of CPT by setting $n = 10$. For convenience in learning, we equalized the lengths of all sides of each warehouse by padding the shorter sides with 0s.

In many one-shot task assignment and MAPF problems, makespan, which represents the time taken to complete all tasks, is commonly optimized. However, in lifelong

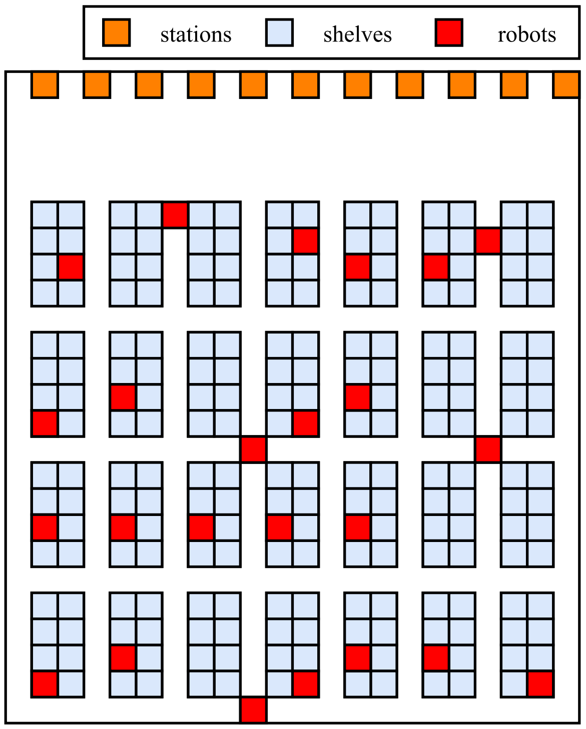

FIGURE 4. **Example of a warehouse with a size of 25 × 22.**

problems, in which tasks arrive continuously, this practice might not be appropriate. Accordingly, we used CPT and throughput to measure the efficiency of our model and compared this efficiency with that of a mature task assignment method, namely RMCA. CPT quantifies the average time consumed in each task. Because our algorithm is designed for lifelong systems, CPT is a more suitable optimization objective than makespan, and a lower CPT indicates higher efficiency. Furthermore, for relatively intuitive understanding of the model performance, we considered throughput, which is calculated as:

$$Throughput = \frac{B}{CPT} \times 60. \tag{20}$$

In (20), throughput represents the number of tasks completed per minute. We selected RMCA as the comparative benchmark and modified it to suit our environment. To enable the RMCA model to handle shelf returns, we modified its path planning method to generate returning paths, and the departing and returning paths cannot be separated during the process. In addition, the end points were divided into picking stations and shelves, and robots were prohibited from moving under shelves while transporting a shelf. Each robot's capacity was set as 1, which enabled it to carry only one shelf at a time. The value of k-robust was set as 0, which indicated that no additional time had to be reserved for path planning. In addition, we examined the efficiency of the original DQN model to evaluate the benefits associated with the proposed and RMCA models.

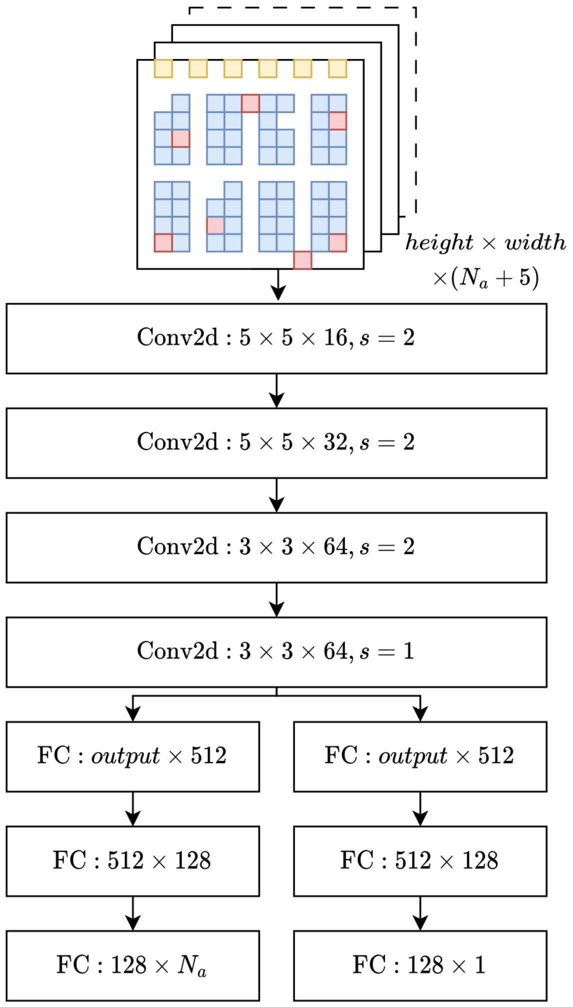

FIGURE 5. Neural network architecture and parameters used in our numerical analysis.

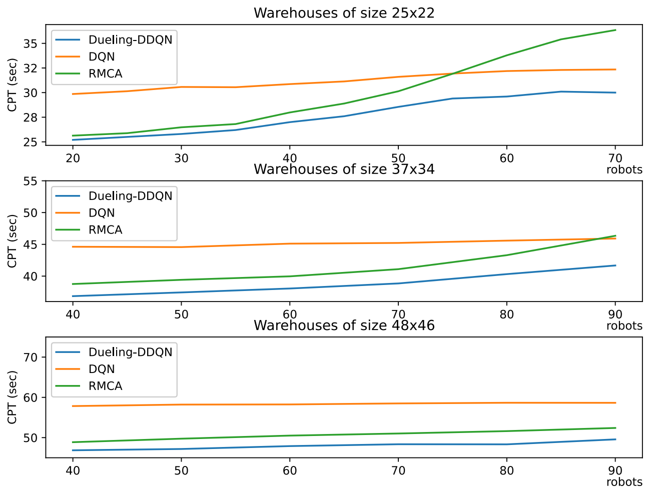

FIGURE 6. CPT values of the proposed model, RMCA model, and original DQN model.

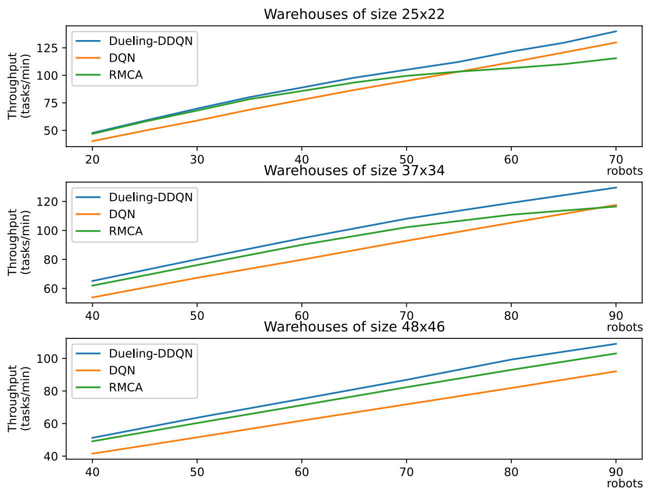

FIGURE 7. Throughput values of the three compared models.

In the model comparison, we generated random initial robot positions and order contents. To ensure fairness, we ran all the models by using the same configuration. In the simulation, we conducted 30 tests for each combination of warehouse size and number of robots, and the obtained values were then averaged. Fig. 6 displays the CPT values of the compared models. The CPT increased with the number of robots. A large number of robots led to traffic congestion within a warehouse, which resulted in long path lengths. Moreover, an increase in the warehouse size caused an increase in the CPT because the distances that the robots had to travel became longer. The proposed model had lower average CPT values than did the other two models in multiple scenarios, indicating the proposed model's suitability for the developed environment. The throughput values of the three models are displayed in Fig. 7; the proposed model achieved the highest throughput in all scenarios. And due to the poor performance of the DQN method, we will exclude it from further discussion.

To demonstrate the adaptability of the proposed method to different environments, we tested the efficiency of each method in a longer size of 67×34 and a wider size of 37×64. From Fig. 8 and Fig. 9, we observed that due to the longer layout, increasing the distances traveled, the efficiency of both methods decreased. However, because of the increased number of picking stations, the efficiency associated with the wider layout did not decrease significantly. The proposed method exhibited higher efficiency in all these cases.

In Fig. 7, as the number of robots increases, the slope of the throughput curve gradually flattens, indicating that the contribution of each newly added robot to efficiency diminishes. We can utilize this characteristic to determine the capacity of robots in each environment. In this study, we define the throughput of the first ten robots in each scenario as the baseline efficiency. If the throughput provided by the newly added ten robots is less than half of the baseline efficiency, it is considered to have reached the capacity. Fig. 10 illustrates the capacity of each model in different warehouse sizes, where the proposed method achieves a higher capacity than RMCA.

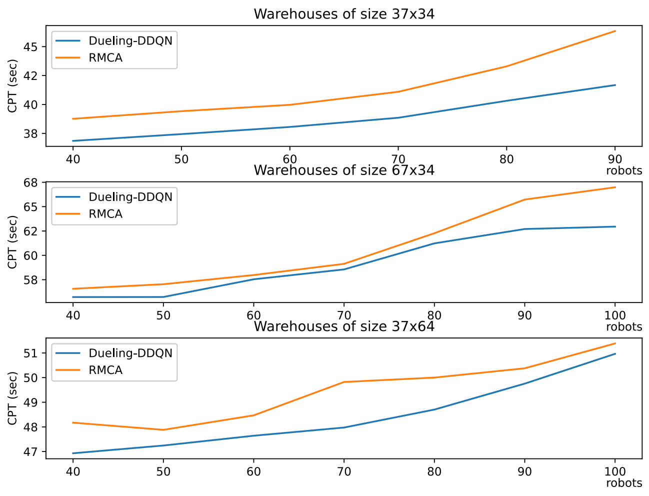

FIGURE 8. CPT values of different sizes of warehouses.

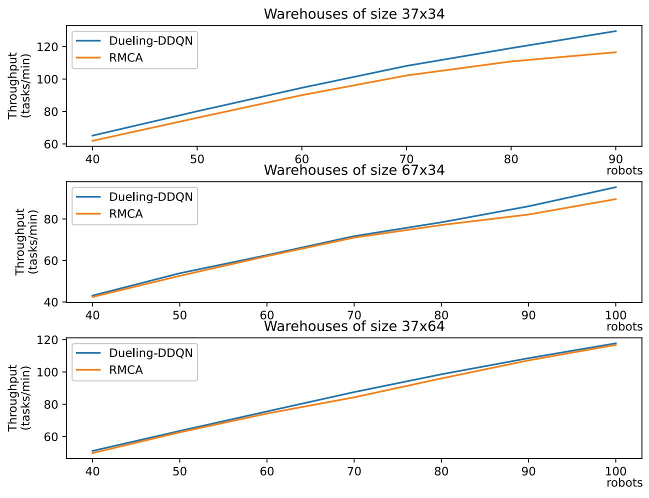

FIGURE 9. Throughput values of different sizes of warehouses.

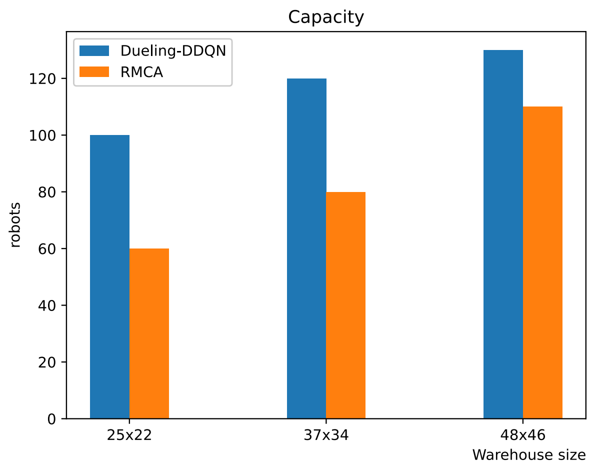

FIGURE 10. Capacities of the compared models in different warehouse sizes.

Another metric utilized for model comparison was runtime, which represents the time required to run the simulation.

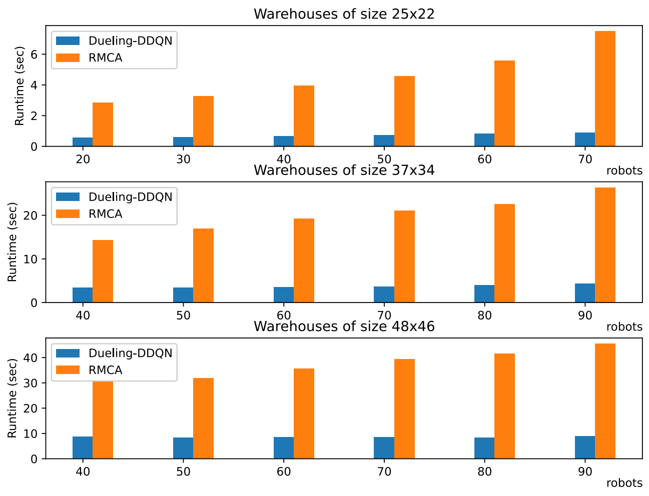

FIGURE 11. Runtime of the proposed and RMCA models.

Because the proposed and RMCA models must operate in real time, they must be able to compute subsequent actions within a specified time limit; failure to do so might lead to prolonged robot downtime, which would render the planned paths ineffective. Thus, runtime can indicate the feasibility of the aforementioned models. Fig. 11 displays the runtime of the aforementioned models. The proposed model had a shorter runtime than did the RMCA model. Moreover, the runtime of the proposed model did not increase with the number of robots. By contrast, the RMCA model had to perform extensive computations in each time slot, and its runtime increased with the map size and number of robots. In summary, the proposed model is capable of operating in real time in scenarios involving a relatively large number of robots.

VI. CONCLUSION

In this study, we introduced the task assignment problem in warehouses and formulated warehouse environments. It is desired to continuously assign incoming tasks to a group of robots in a warehouse to maximize efficiency. We proposed a DQN-based task assignment model and a DQN-based shelf reallocation model that included shelf inventory in the state. These models were combined to develop the final model of this study. In addition, we performed simulations to compare the throughput and runtime of the final proposed model with those of the RMCA model. The proposed model had a higher throughput and shorter runtime than did the RMCA model. The simulation results indicated that the proposed model is suitable for application in warehouses with a large number of robots. In the future, several improvements could be made to the warehouse system developed in this study. First, a replenishment strategy could be incorporated into this system. Because of inventory constraints, the developed system cannot operate indefinitely unless the inventory on the shelves is replenished. Our preliminary idea was to set up replenishment stations across from the picking stations to facilitate replenishment. Second, the quality of the order decomposition in the developed system could be enhanced by considering the warehouse state in this process. Last but not least, the practical application of the algorithms is crucial, and we will conduct real-world testing.

REFERENCES

[1] J. F. Rayport and B. J. Jaworski, *Introduction to E-Commerce*. New York, NY, USA: McGraw-Hill, 2003.

[2] S. L. Golicic, D. F. Davis, T. M. McCarthy, and J. T. Mentzer, "The impact of e-commerce on supply chain relationships," *Int. J. Phys. Distrib. Logistics Managemen*, vol. 32, no. 10, pp. 851–871, Dec. 2002.

[3] Y. Yu, X. Wang, R. Y. Zhong, and G. Q. Huang, "E-commerce logistics in supply chain management: Practice perspective," *Proc. CIRP*, vol. 52, pp. 179–185, Sep. 2016.

[4] N. Boysen, R. de Koster, and F. Weidinger, "Warehousing in the e-commerce era: A survey," *Eur. J. Oper. Res.*, vol. 277, no. 2, pp. 396–411, Sep. 2019.

[5] J. Gu, M. Goetschalckx, and L. F. McGinnis, "Research on warehouse design and performance evaluation: A comprehensive review," *Eur. J. Oper. Res.*, vol. 203, no. 3, pp. 539–549, Jun. 2010.

[6] E. Žunić, S. Delalić, K. Hodžić, A. Beširević, and H. Hindija, "Smart warehouse management system concept with implementation," in *Proc. 14th Symp. Neural Netw. Appl.*, Belgrade, Serbia, Nov. 2018, pp. 1–5.

[7] L. Zhen and H. Li, "A literature review of smart warehouse operations management," *Frontiers Eng. Manage.*, vol. 9, no. 1, pp. 31–55, Jan. 2022.

[8] X. Liu, J. Cao, Y. Yang, and S. Jiang, "CPS-based smart warehouse for Industry 4.0: A survey of the underlying technologies," *Computers*, vol. 7, no. 1, p. 13, Feb. 2018.

[9] N. Seenu, R. M. K. Chetty, M. M. Ramya, and M. N. Janardhanan, "Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems," *Ind. Robot. Int. J. Robot. Res. Appl.*, vol. 47, no. 6, pp. 929–942, Sep. 2020.

[10] Z. Tan, H. Li, and X. He, "Optimizing parcel sorting process of vertical sorting system in e-commerce warehouse," *Adv. Eng. Informat.*, vol. 48, Apr. 2021, Art. no. 101279.

[11] A. Bolu and Ö. Korçak, "Adaptive task planning for multi-robot smart warehouse," *IEEE Access*, vol. 9, pp. 27346–27358, 2021.

[12] D. G. Cattrysse and L. N. Van Wassenhove, "A survey of algorithms for the generalized assignment problem," *Eur. J. Oper. Res.*, vol. 60, no. 3, pp. 260–272, Aug. 1992.

[13] R. de Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *Eur. J. Oper. Res.*, vol. 182, no. 2, pp. 481–501, Oct. 2007.

[14] J. Zhang, X. Wang, and K. Huang, "On-line scheduling of order picking and delivery with multiple zones and limited vehicle capacity," *Omega*, vol. 79, pp. 104–115, Sep. 2018.

[15] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," 2017, *arXiv:1705.10868*.

[16] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5816–5823, Jul. 2021.

[17] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, U.K.: MIT Press, 2018.

[19] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.

[20] B.-C. Lai, W.-Y. Chiu, and Y.-P. Tsai, "Multiagent reinforcement learning for community energy management to mitigate peak rebounds under renewable energy uncertainty," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 3, pp. 568–579, Jun. 2022.

[21] Y. Du, J.-Q. Li, X.-L. Chen, P.-Y. Duan, and Q.-K. Pan, "Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1036–1050, Aug. 2022.

[22] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[23] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. S. Kumar, S. Koenig, and H. Choset, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[24] M. P. Li, P. Sankaran, M. E. Kuhl, R. Ptucha, A. Ganguly, and A. Kwasinski, "Task selection by autonomous mobile robots in a warehouse using deep reinforcement learning," in *Proc. Winter Simul. Conf.*, Dec. 2019, pp. 680–689.

[25] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.

[26] Q. Hou and J. Dong, "Distributed dynamic event-triggered consensus control for multiagent systems with guaranteed L_2 performance and positive inter-event times," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 1, pp. 746–757, Dec. 2004.

[27] Q. Hou and J. Dong, "Robust adaptive event-triggered fault-tolerant consensus control of multiagent systems with a positive minimum interevent time," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 7, pp. 4003–4014, Feb. 2023.

[28] R. Stern, "Multi-agent path finding—An overview," in *Proc. 5th RAAI Summer School Artif. Intell.*, 2019, pp. 96–115.

[29] D. Kornhauser, G. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," in *Proc. 25th Annu. Symp. Found. Comput. Sci.*, Oct. 1984, pp. 241–250.

[30] D. Silver, "Cooperative pathfinding," in *Proc. 1st AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[31] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 3268–3275.

[32] B. D. Wilde, A. W. Ter Mors, and C. Witteveen, "Push and rotate: A complete multi-agent pathfinding algorithm," *J. Artif. Intell. Res.*, vol. 51, pp. 443–492, Oct. 2014.

[33] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proc. 24th AAAI Conf. Artif. Intell.*, vol. 24, no. 1, 2010, pp. 173–178.

[34] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artif. Intell.*, vol. 219, pp. 1–24, Feb. 2015.

[35] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 195, pp. 470–495, Feb. 2013.

[36] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, Feb. 2015.

[37] A. Felner, R. Stern, S. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Proc. Int. Symp. Combinat. Search*, vol. 8, no. 1, 2017, pp. 29–37.

[38] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. Int. Symp. Combinat. Search*, vol. 5, no. 1, 2014, pp. 19–27.

[39] T. T. Walker, N. R. Sturtevant, and A. Felner, "Extended increasing cost tree search for non-unit cost domains," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 534–540.

[40] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, May 1992.

[41] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[43] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th Conf. Artif. Intell.*, Mar. 2016, pp. 2094–2100.

[44] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.

[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[48] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[49] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.

SHAO-CI WU was born in Taipei, Taiwan, in 1997. He received the B.S. and M.S. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2019 and 2023, respectively.

WEI-YU CHIU (Senior Member, IEEE) received the Ph.D. degree in communications engineering from National Tsing Hua University, Hsinchu, Taiwan. He is currently an Associate Professor with the School of Information and Technology, Deakin University, Melbourne, VIC, Australia. His research interests include multiobjective optimization and reinforcement learning, and their applications to control systems, robotics, and smart energy systems. He was a recipient of the Youth Automatic Control Engineering Award bestowed by Chinese Automatic Control Society, the Outstanding Young Scholar Academic Award bestowed by Taiwan Association of Systems Science and Engineering, the Erasmus+Programme Fellowship funded by European Union (staff mobility for teaching), and the Outstanding Youth Electrical Engineer Award bestowed by Chinese Institute of Electrical Engineering. He was an Organizer/Chair of the International Workshop on Integrating Communications, Control, and Computing Technologies for Smart Grid (ICT4SG). He is a Subject Editor of *IET Smart Grid*.

CHIEN-FENG WU received the Ph.D. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2017. From 2020 to 2023, he was an Assistant Professor with the Department of Electrical Engineering and Computer Science, Tamkang University. He is currently an Assistant Professor with the Department of Electrical Engineering, National Taipei University, New Taipei City, Taiwan. His current research interests include robust control, fuzzy control, multiobjective optimization, automated driving systems, and nonlinear stochastic systems.