2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Reinforcement Learning of Multi-robot Task Allocation for Multi-object Transportation with Infeasible Tasks

Yuma Shida[1], Tomohiko Jimbo[2,1], Tadashi Odashima[1], and Takamitsu Matsubara[3]

Abstract— Multi-object transport using multi-robot systems has the potential for diverse practical applications such as delivery services owing to its efficient individual and scalable cooperative transport. However, allocating transportation tasks of objects with unknown weights remains challenging. Moreover, the presence of infeasible tasks (untransportable objects) can lead to robot stoppage (deadlock). This paper proposes a framework for dynamic task allocation that involves storing task experiences for each task in a scalable manner with respect to the number of robots. First, these experiences are broadcasted from the cloud server to the entire robot system. Subsequently, each robot learns the exclusion levels for each task based on those task experiences, enabling it to exclude infeasible tasks and reset its task priorities. Finally, individual transportation, cooperative transportation, and the temporary exclusion of tasks considered infeasible are achieved. The scalability and versatility of the proposed method were confirmed through numerical experiments with an increased number of robots and objects, including unlearned weight objects. The effectiveness of the temporary deadlock avoidance was also confirmed by introducing additional robots within an episode. The proposed method enables the implementation of task allocation strategies that are feasible for different numbers of robots and various transport tasks without prior consideration of feasibility.

I. INTRODUCTION

In recent years, multi-robot transportation tasks have attracted considerable attention in various fields such as delivery services, factory logistics, search and rescue, and precision agriculture. Systems in which multiple robots are controlled via a cloud server to execute various transportation tasks within facilities have been developed. These multi-robot systems achieve efficient transportation because each robot can cover a wide area independently [1] (Fig. 1 (a)). In addition, scalability is realized by enabling multiple robots to cooperate with each other when transporting objects that individual robots find infeasible [2]–[5] (Fig. 1 (c)). Furthermore, distributed control improves the resilience of robots by facilitating the seamless addition of robots to the system.

Multi-robot task allocation (MRTA) is important for achieving multiple transportation tasks with multi-robot systems [1], [6]. Garkely et al. (2004) [1] categorized MRTA approaches into centralized algorithms [7], [8], distributed algorithms [9], and hybrid algorithms that combine centralized and distributed approaches, such as auction-based methods [10]–[12].

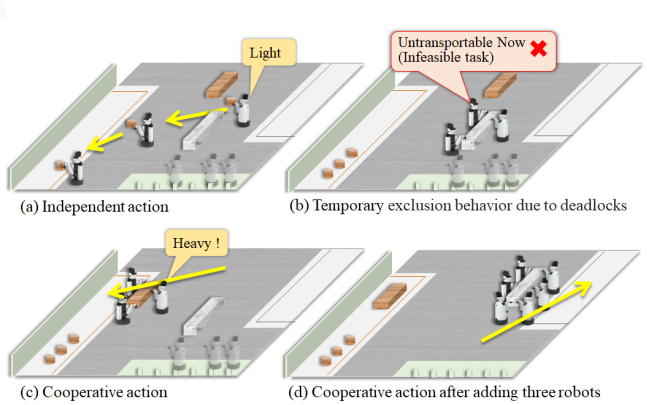

Fig. 1. Multi-object transport using a multi-robot system. (a) Robots independently perform actions when they can carry the selected objects. (b) Deadlock occurs when robots are unable to cooperatively carry the selected object. (c) Robots employ deadlock avoidance strategies and cooperate with each other to carry heavy objects. (d) After the introduction of three additional robots into the system, the robots cooperatively carry the object that was previously unable to be carried.

Recently, several approaches based on multi-agent reinforcement learning (MARL) [13] have been proposed [14]. However, MARL remains challenging because of the partial observability of each robot and the simultaneous learning of policies for each robot [13], [15]; therefore, the approach involving centralized training and decentralized execution (CTDE) is often used [16], [17]. The CTDE approach has demonstrated scalability for systems with various numbers of robots that need to perform diverse tasks [14], [18]. Furthermore, cooperative actions can be performed by learning communication between robots [14], [19].

Naturally, there are objects that cannot be transported (infeasible tasks) by all robots (Fig. 1 (b)), and there may be no prior information about them. In conventional methods [1], the cost and task completion probability for each task must be specified explicitly. In addition, when the method [14] based on MARL is applied to task allocation in an environment in which infeasible tasks exist, all robots are connected to the infeasible tasks, causing a stoppage (deadlock) in the robot movements.

To avoid deadlocks, it is necessary to exclude infeasible

[1]The authors are with R-Frontier Division, Frontier Research Center, Toyota Motor Corporation, 1, Toyota-cho, Toyota, Aichi, Japan {yuma_shida, tadashi_odashima}@mail.toyota.co.jp
[2]Tomohiko Jimbo is with Toyota Central R&D LABS., inc., Aichi, 480-1192, Japan
[3]Takamitsu Matsubara is with Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology, Nara, Japan takam-m@is.naist.jp

tasks from task allocation. However, when additional cooperative robots are introduced, the tasks become feasible, as shown in Fig. 1 (d), and it is desired to release the exclusion. Therefore, a mechanism is required to temporarily exclude the allocation of tasks that have already been allocated.

In this paper, we propose a framework for dynamic task allocation that enables a multi-robot system to continue executing tasks even when there are infeasible tasks among the transportation tasks. Specifically, a cloud server stores the task experiences in a scalable manner with respect to the number of robots and broadcasts them to the entire robot system. In our proposed method, each robot learns the exclusion level for each object based on the task experience and other information. The task priorities [14] are reset using an output gate based on the task experience and exclusion levels. Consequently, individual transportation, cooperative transportation, and the temporary exclusion of tasks that were considered as infeasible were achieved. Finally, we validated the performance of our proposed method in terms of success rate and transportation time by performing numerical experiments with a larger number of robots and objects compared with the training experiments, as well as with untrained objects and a varying number of robots.

The contributions of this study are as follows:

- We propose a task alloation framework with infeasible tasks comprising task experiences broadcasted from the cloud server and task exlusion levels learned by structured policy models.
- The proposed framework differs from conventional MRTA approaches that require the specification of the cost and task completion probability, in that it can temporarily exclude infeasible tasks without prior information until additional robots are introduced.
- We confirm that the proposed method successfully completes feasible tasks while excluding infeasible tasks, even in numerical experiments that differ from the training experiments, including an episode where additional robots are introduced within the episode.

The remainder of this paper is structured as follows: Section II presents related work, Section III explains the problem setting, Section IV introduces the proposed method, Section V evaluates the performance of the proposed method through numerical experiments, and Section VI concludes the paper.

II. Related Work

A. Combinatorial Optimization

Task allocation can be formulated as a combinatorial optimization problem that aims to minimize the total travel distance, and can be solved using methods such as the Hungarian algorithm [7] or integer linear programming [8]. However, combinatorial optimization is an NP-hard problem. Moreover, efficient solution methods are needed for problems with a large number of tasks and robots. Therefore, Q-learning based reinforcement learning [20] and clustering methods [21] combined with optimization have been proposed in recent years.

These methods require that the necessary resources (number of robots) are provided for each task. However, this information is not always available in advance. In addition, the task may not be executable with the current resources.

B. Bio-inspired Approach

Metaheuristic methods [22]–[24], which were inspired by biological systems and natural processes, such as the division of labor in social insects, have been used to solve MRTA problems [6]. A commonly used approach is the threshold model [22], [23], in which each robot selects tasks using activation thresholds and a stimulus associated with each task based on local information. These methods are flexible and can be adapted to different conditions with varying number of robots and tasks. However, infeasible tasks may be allocated to robots, which can lead to decreased efficiency.

C. Auction Approach

The auction algorithm is a commonly used method for task allocation among multi-robots [1], and has been studied using both centralized and decentralized approaches. With the centralized approach [25], an auctioneer allocates tasks to bidders. In contrast, Choi et al. (2009) [11] proposed a decentralized auction-based algorithm without an auctioneer. With this method, tasks are allocated based on a consensus algorithm that involves local communication among bidders (robots). However, their method focused on problems in which a single robot could execute each task.

Braquet and Bakolas (2021) [12] addressed a problem similar to that in our research. It focuses on the cases in which multiple robots are required for each task. Their method employed a consensus algorithm that is similar to that of Choi et al. (2009) [11] to estimate the lists of selected tasks, winning bids, and completed allocations. The robots allocate tasks to the robot with the highest bid among the unallocated robots based on the list of completed allocations. However, this method requires the probability of completing each task, which becomes computationally challenging when dealing with objects for which the resources required for completion are unknown.

D. MARL Approach

Previous studies [26]–[31] have focused on task allocation problems using MARL. These methods formulate task allocation as a Markov decision process and learn policies using learning algorithms, such as MADDPG [16]. However, these methods are designed for a fixed number of agents and tasks, rendering them ineffective

under different conditions. To address this issue, policy models that only obtain neighboring agents and tasks have been utilized [14], [18]. In particular, a framework for task allocation in the presence of an unknown number of robots required for cooperative transportation was proposed in previous studies [14]. This proposed framework utilizes dynamic priorities for each task and global robot communication. However, if the tasks do not have sufficient resources (number of robots), the robots may encounter a deadlock scenario until the introduction of additional robots.

Our proposed framework, which is similar to the approach proposed by Shibata et al. (2022) [14], uses dynamic priorities, but in contrast to their method, it incorporates the learning of dynamic exclusion levels to exclude infeasible tasks. Dynamic exclusion levels are used in the output gate, which temporarily resets dynamic priorities. Therefore, if a robot encounters a deadlock with a specific object, the priority of that task is reset. In addition, when more cooperative robots are introduced, the reset for that object is released, enabling the robots to again be transported cooperatively.

III. Problem Setting

Consider transporting M objects of different weights located at $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_M \in \mathbb{R}^2$ with N robots to the corresponding goals $\boldsymbol{z}_1^*, \cdots, \boldsymbol{z}_M^* \in \mathbb{R}^2$. (Fig. 1). This belongs to single-task robot, multi-robot task, and instantaneous assignment (ST-MR-IA) problem that is categorized by Garkely et al. (2004). From a practical viewpoint, the weights of the objects are unknown and there may be objects that cannot be transported. In this study, the transport task aims to efficiently transport all feasible objects using all robots as quickly as possible.

The study focuses on the allocation of objects to robots. Robots move to reach the allocated objects in the shortest time possible. If the number of robots $|\mathcal{C}_l|$ connected to object l is greater than the weight of w_l, the object l is transportable: namely, supposedly robots are homogeneous. Here, $\mathcal{C}_l = \{i \in \{1, \cdots, N\} | \|\boldsymbol{x}_i - \boldsymbol{z}_l\|_2 \leq \delta\}$ is the set of robots connected to object l, $\boldsymbol{x}_i \in \mathbb{R}^2$ is the position of robot $i (= 1, \cdots, N)$, and the small positive constant δ is a distance threshold. When considering cases where the robots are heterogeneous, it is necessary to consider not only the number of connected robots $|\mathcal{C}_l|$ but also the transport performance of each robot.

This study made the following assumptions:

- The cloud server can always obtain the latest information (global information) about the robots and objects.
- When cooperating with other robots is necessary, each robot can obtain global information via the cloud server.

IV. Method

The weights of the objects are unknown, and there may be objects that cannot be transported; hence, the simple allocation of tasks to the robots in advance would result in a deadlock. Therefore, we propose the framework shown in Fig. 2. The framework consists of three parts for the object l; the cloud server stores the total number of robots connected to object l by time t as global information (called task experience E_l); each robot $i \in \{1, \cdots, N\}$ has a task exclusion level ζ_i^l for each object $l \in \{1, \cdots, M\}$ in addition to the task priority ϕ_i^l to temporarily reset the task priority using E_l for $l \in \{1, \cdots, M\}$ from the cloud server.

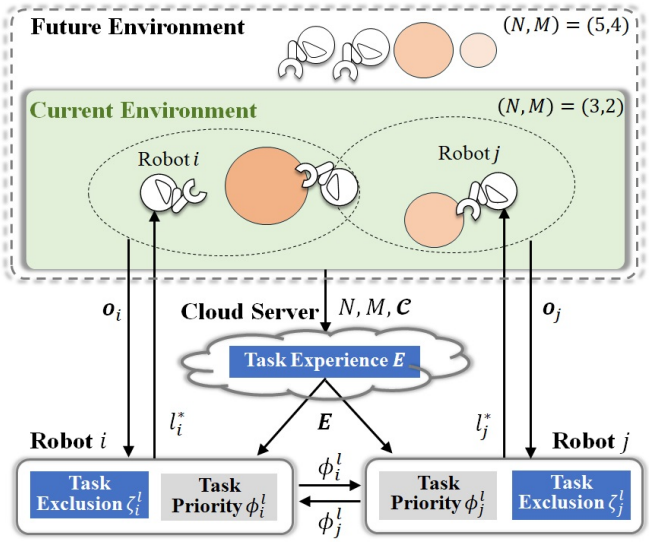

Fig. 2. Abstracted framework of the proposed dynamic task allocation, which includes handling infeasible tasks. Task experiences E that are scalable with the number of robots N, are broadcasted from the cloud server to each robot in order to learn task exclusion levels ζ_i and task priorities ϕ_i. Subsequently, robots shares their information between all robots when cooperating is necessary.

A. Task Experience

The cloud server stores the task experience, for object l,

$$E_l(t) = \int_0^t \left(\frac{\left(1 - \sigma_0(|\boldsymbol{v}_l(\tau)|)\right) \cdot |\mathcal{C}_l(\tau)|}{N} \right)^{\kappa} d\tau \quad (1)$$

where $\boldsymbol{v}_l \in \mathbb{R}^2$ represents the velocity of object l, σ_0 is the step function with a threshold of 0, and κ is a positive constant. Note that E_l does not increase in t while the object is moving ($|\boldsymbol{v}_l(t)| \neq 0$). Furthermore, to make the value scalable to the number of robots, it is normalized by N.

B. Dynamic Task Exclusion

To obtain exclusion levels that are scalable with the number of objects M, robot i utilizes a partial observation $\boldsymbol{o}_i$:

$$\begin{aligned} \boldsymbol{o}_i = [&\boldsymbol{x}_i^\top, \boldsymbol{\phi}_i^{l_{i1}}, \cdots, \boldsymbol{\phi}_i^{l_{iK}}, \\ &\boldsymbol{x}_{j_{i1}}^\top, \boldsymbol{\phi}_{j_{i1}}^{l_{i1}}, \cdots, \boldsymbol{\phi}_{j_{i1}}^{l_{iK}}, \cdots, \boldsymbol{x}_{j_{iK}}^\top, \boldsymbol{\phi}_{j_{iK}}^{l_{i1}}, \cdots, \boldsymbol{\phi}_{j_{iK}}^{l_{iK}}, \\ &\boldsymbol{z}_{l_{i1}}^\top, {\boldsymbol{z}_{l_{i1}}^*}^\top, \boldsymbol{v}_{l_{i1}}^\top, \cdots, \boldsymbol{z}_{l_{iK}}^\top, {\boldsymbol{z}_{l_{iK}}^*}^\top, \boldsymbol{v}_{l_{iK}}^\top, \\ &E_{l_{i1}}, \cdots, E_{l_{iK}}]^\top, \end{aligned} \quad (2)$$

which contains $K(< M)$ nearest robots $j \in \mathcal{N}_i^{\text{Robot}} := \{j_{i1}, \cdots, j_{iK}\}$, objects $l \in \mathcal{N}_i^{\text{Load}} := \{l_{i1}, \cdots, l_{iK}\}$, and task experience E_l for the nearest objects. It is crucial for the MARL approach to restrict observation to partial ones within the CTDE. A limited number of nearest robots and objects may reduce the effectiveness of centralized training.

Robot i learns a static policy network model with the target exclusion level, for $l \in \mathcal{N}_i^{\text{Load}}$,

$$\zeta_i^{l^*} = \pi_i^{\zeta}(\boldsymbol{o_i}). \tag{3}$$

as output. For other objects, the current exclusion levels ζ_i^l are set. In other words, the target exclusion levels for all objects $l(= \{1, \cdots, M\})$ are set as follows:

$$d_i^l = \begin{cases} \zeta_i^{l^*}, & l \in \mathcal{N}_i^{\text{Load}} \\ \zeta_i^l, & \text{otherwise} \end{cases}.$$

The exclusion levels can be set in a decentralized execution manner using the partial observations.

Furthermore, for objects $l \notin \mathcal{N}_i^{\text{Load}}$ that are outside the $K(< M)$ nearest neighbors of robot i, a mechanism is introduced to share the exclusion levels with other robots via the cloud server at time when $\sigma(\beta_i)$. The timing is obtained by learning a static policy network model, which is denoted as follows:

$$\beta_i = \pi_i^{\beta}(o_i) \in [0, 1]. \tag{4}$$

The timing is also determined in a decentralized execution manner using the partial observations.

Based on the above, robot i updates the exclusion levels for all objects $l(= \{1, \cdots, M\})$ using a consensus protocol via the cloud server:

$$\dot{\zeta}_i^l = k_\zeta (d_i^l - \zeta_i^l) + k_\zeta \sigma(\beta_i) N (\max_j \zeta_j^l - \zeta_i^l). \tag{5}$$

Here, k_ζ is a positive constant,

$$\sigma(x) = \begin{cases} 1, & x \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

is a step function with a threshold of 0.5 for variable $x \in [0, 1]$.

C. Integration with Dynamic Task Priority

Each robot sequentially allocates objects using dynamic priorities [14].

Similar to the exclusion levels, to achieve scalable learning with a number of objects M, robot i sets the target priorities of all objects $l(= \{1, \cdots, M\})$ as follows:

$$c_i^l = \begin{cases} \phi_i^{l^*}, & l \in \mathcal{N}_i^{\text{Load}} \\ \phi_i^l, & \text{otherwise} \end{cases},$$

where ϕ_i^l is the current priority of robot i for object l; in addition, robot i learns the target priorities $\phi_i^{l^*}$ for $K(< M)$ neighboring objects $l \in \mathcal{N}_i^{\text{Load}}$ using a policy network model as follows:

$$\phi_i^{l^*} = \pi_i^{\phi}(o_i). \tag{6}$$

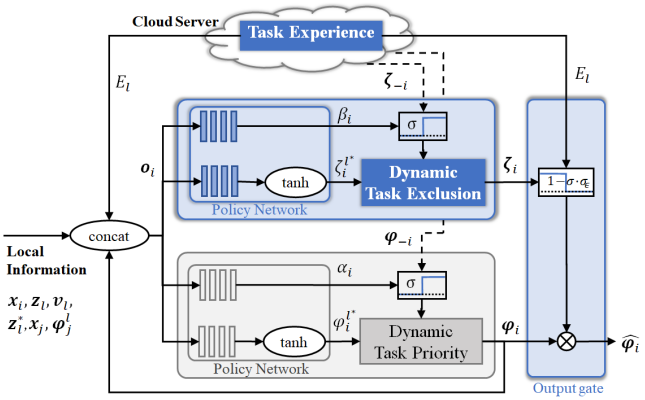

Fig. 3. Block diagram of multi-robot task allocation with infeasible tasks. The output gate plays a role in integrating dynamic task exclusion and dynamic task priority.

Cooperation with robots that are not located nearby is required for handling heavy objects. Therefore, similar to the exclusion level, when the required timing is $\sigma(\alpha_i)$, robot i updates the priorities for all objects $l(= \{1, \cdots, M\})$ using a consensus protocol via a cloud server:

$$\dot{\phi}_i^l = k_\phi (c_i^l - \phi_i^l) + k_\phi \sigma(\alpha_i) \sum_{j=1}^{N} (\phi_j^l - \phi_i^l). \tag{7}$$

Here, k_ϕ denotes a positive constant. In addition, the timing for sharing priorities with other robots is obtained by learning a static policy network as follows:

$$\alpha_i = \pi_i^{\alpha}(o_i) \in [0, 1]. \tag{8}$$

The exclusion level ζ_i^l in Equation (5) is integrated with priority ϕ_i^l in Equation (7) using the output gate, as shown in Fig. 3. The integration is expressed as follows:

$$\hat{\phi}_i^l = \left(1 - \sigma(\zeta_i^l) \cdot \sigma_\epsilon(E_l)\right) \phi_i^l. \tag{9}$$

Here, σ_ϵ is a step function with the threshold of a small positive constant ϵ. If object l satisfies $\zeta_i^l \geq 0.5$ and $E_l \geq \epsilon$, transportation is determined to be infeasible with the current resources, and its priority is reset.

Finally, robot i selects the object with the highest priority among priorities $\hat{\phi}_i^l$ $(l = 1, \cdots, M)$ after the output gate in Equation (9):

$$l_i^* = \underset{l \in \{1, 2, \cdots, M\}}{\text{argmax}} \ \hat{\phi}_i^l.$$

Furthermore, the priorities of the objects that have reached their goals or are being transported by other robots are set to 0. In addition, if an object is already allocated and is being transported, the update of all priorities is stopped.

D. Policy Optimization

As shown in Fig. 3, a static policy network model consisting of Equation (3), (4), (6), and (8), for the robot i,

$$\boldsymbol{a}_i = \pi_i(\boldsymbol{o}_i). \tag{10}$$

is learned. Here,

$$\boldsymbol{a}_i = \left[{\phi_i^{l_{i1}}}^*, \cdots, {\phi_i^{l_{iK}}}^*, \alpha_i, {\zeta_i^{l_{i1}}}^*, \cdots, {\zeta_i^{l_{iK}}}^*, \beta_i\right]^\top.$$

In this study, the MADDPG algorithm [16] was used to perform the centralized training of the policy network model of Equation (10). The algorithm used the MARL settings using Markov decision processes (partially observable Markov games), i.e., states of all robots, objects and the task experience for all objects are denoted as

$$\begin{aligned}\boldsymbol{s} \;=\; [&\boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_N^\top,\\ &\boldsymbol{\phi}_1^1, \cdots, \boldsymbol{\phi}_1^M, \cdots, \boldsymbol{\phi}_N^1, \cdots, \boldsymbol{\phi}_N^M,\\ &\boldsymbol{z}_1^\top, \cdots, \boldsymbol{z}_M^\top, \boldsymbol{v}_1^\top, \cdots, \boldsymbol{v}_M^\top,\\ &E_1, \cdots, E_M]^\top,\end{aligned}$$

and for any time series of states and actions, $\boldsymbol{s}(\tau)$ and $\boldsymbol{a}_i(\tau)$ for all $\tau \in \{t+1, \cdots, 0\}$ and i, the state transition probability satisfies Markov property as follows:

$$\begin{aligned}&\mathbb{P}[\boldsymbol{s}(t+1) \mid \boldsymbol{s}(t), \boldsymbol{a_1}(t), \cdots, \boldsymbol{a_N}(t)]\\ &= \mathbb{P}[\boldsymbol{s}(t{+}1) \mid \boldsymbol{s}(\tau), \boldsymbol{a_1}(\tau), \cdots, \boldsymbol{a_N}(\tau), \forall\tau \in \{t, \cdots, 0\}].\end{aligned}$$

The goal positions were not included into the states, the same goes for the task exclusion levels. The initial positions of all robots and objects are determined randomly by the uniform distribution, and the goal positions $\boldsymbol{z}_l^*$ are fixed.

We use the reward for transporting objects as follows:

$$r = \sum_{l=1}^{M}(P_l + \lambda\|\boldsymbol{v}_l\|_2),$$

that is same to [14]. Where P_l is 1 if $\|\boldsymbol{z}_l - \boldsymbol{z}_l^*\|_2 \leq \delta$, otherwise P_l is 0. λ is positive constant.

V. Numerical Experiment

In this section, we present the results of numerical experiments on multiple object transportation tasks. The performance of the proposed method was evaluated under different settings by varying the number of robots and objects. The performance was also compared with three ablation studies that removed either the dynamic exclusion or E inputs, or both, from the proposed method.

A. Simulation Setup

Starting positions of the participants were randomly selected, and the target positions of each object were fixed for each simulation (Fig. 4). The carrying capacity of each robot was set to 1. By the inclusion of N robots cooperating to transport the same object, it becomes possible to transport an object with a weight of N.

In the training experiments (Training in Table I), $N(=3)$ robots and $M(=6)$ objects were used. The weights of the objects were set to three types, including objects that cannot be transported by N robots, with weights $w_l \in \{1, N, N+1\}$. Among the six objects, three objects had a weight of $N+1$, whereas the remaining three objects were randomly chosen from weights of 1 and N with a 50 % probability.

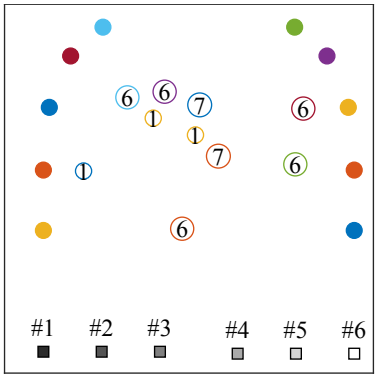

Fig. 4. Numerical environments with $N = 6$ robots and $M = 10$ objects. Each square represents a robot, each colored circle represents an object, and each colored dot represents a goal of transport. The number inside each circle represents the number of robots needed to transport the object, and the number of each square represents the robot's ID.

We used the MADDPG code [32] and set the simulation parameters listed in Table II. In the proposed method, the numbers of neighboring robots and objects for each robot were set to $K = 2$. The other parameters were set as follows: $\kappa = 10$, $\epsilon = 1$, $k_\phi = 0.2$, $k_\zeta = 0.2$, $\lambda = 60$. These parameters were determined manually throughout trial and error.

The validation experiments (refer to validations 1 and 2 in Table I) used $N(=6)$ robots and $M(=10)$ objects. Among these, two objects have an untransportable weight of $w_l = N+1(=7)$, whereas the remaining eight objects have weights of either $\{1, 3\}$ or $\{1, 6\}$. The proportion of objects with weights of 3 or 6 was selected using $\{0, 50, 100\ \%\}$ probabilities, and 100 episodes were run, with each episode consisting of 1,000 steps.

To evaluate the scalability and versatility of the proposed method, we used the following two criteria:

- Success rate: The percentage of episodes in which all objects reached their goals within the total number of steps.
- Transportation time: The average number of steps (seconds) until all transportable objects reached their goals for the episodes in which they succeed in transportation.

TABLE I
Number of objects and robots in the experiments

Experiment	Robot N	Object M	Object weight				
			1	3	4	6	7
Training	3	6	[0 3]	[0 3]	3	0	0
Validation 1	6	10	[0 8]	[0 8]	2	0	0
Validation 2	6	10	[0 8]	0	0	[0 8]	2
Validation 3	$3 \rightarrow 6$	10	1	4	0	5	0

TABLE II

Simulation parameters.

Parameter	Value
Sampling period [s]	1.0
Number of steps per episode	300
Number of episodes	50,000
Number of hidden layers (critic)	4
Number of hidden layers (actor)	4
Activation function (hidden)	RelU
Activation function (output, critic)	tanh
Activation function (output, actor)	linear
Optimizer (critic)	Adam
Optimizer (actor)	SGD
Discount factor	0.99
Batch size	1024

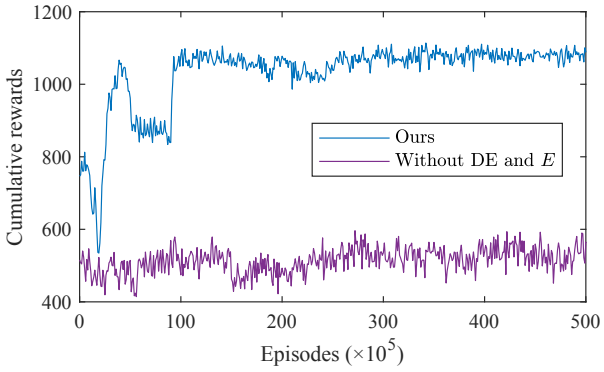

Fig. 5. Cumulative rewards of our framework using DP: Dynamic Priority and DE: Dynamic Exclusion, and E: task experience of objects.

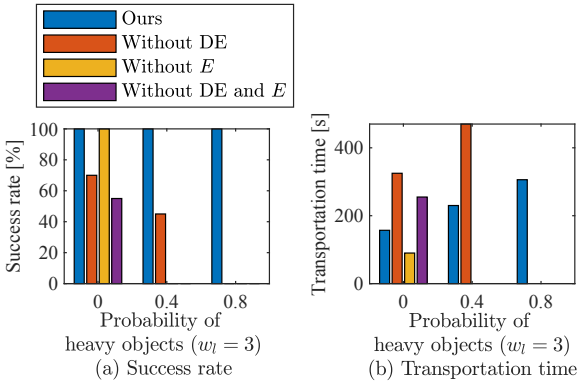

Fig. 6. Validation results of each learned distributed policy with 6 robots and 10 objects with weights of $w_l \in \{1, 3, 4\}$. Probability of untransportable objects, $w_l = 4$, is 0.2.

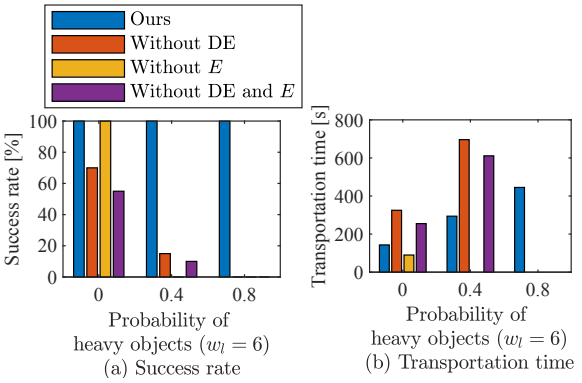

Fig. 7. Validation results of each learned distributed policy with 6 robots and 10 objects with weights of $w_l \in \{1, 6, 7\}$. Probability of untransportable objects, $w_l = 7$, is 0.2.

Furthermore, in an additional validation experiment (refer to Validation 3 in Table I), the number of robots was varied from 3 to 6 during each episode to evaluate the effectiveness of the proposed method.

B. Training Performance

The training performance of the proposed method was evaluated based on the average cumulative reward over 3 training runs. The results were compared with those of a method that used only the dynamic priority mechanism [14] without DE and E_l, as shown in Fig. 5. Here, DE in the figure refers to dynamic exclusion. The method without DE and E_l is identical to the framework introduced in Shibata et al. (2022) [14].

The average cumulative reward of the method using only the dynamic priority mechanism [14] was low because the robots became stuck in a deadlock while gathering around objects that could not be transported.

However, the proposed method achieved high cumulative rewards and did not cause deadlocks. It is believed that the output gate in Equation (9), which is controlled by the dynamic exclusion in Equation (5), effectively avoids deadlock.

C. Scalability and Versatility

Fig. 6 shows the validation results obtained when increasing the number of robots and objects compared to the learning phase, with the weights of the objects remaining the same. For comparison, the results excluding the dynamic exclusion (DE) and E inputs from the proposed method are shown in red and yellow, respectively. When using only the dynamic priority mechanism [14] without DE and E, there were cases in which objects could not be transported because of deadlock occurrences with infeasible objects, resulting in a success rate below 100 %. However, the proposed method achieved successful transportation in shorter times than the other methods under all conditions. Furthermore, the proposed method reduced the transportation time when there were fewer heavy objects. This indicates that both individual and cooperative transportation occurred simultaneously, depending on the composition of the objects. These results demonstrated the scalability of the proposed method.

Fig. 7 shows the validation results obtained when increasing the number of robots and objects compared to the learning phase, including the case where there are unlearned weight values for the objects. Even for objects with unlearned weights (= 6), the proposed method successfully completed the transport tasks. This demonstrates the versatility of the proposed method.

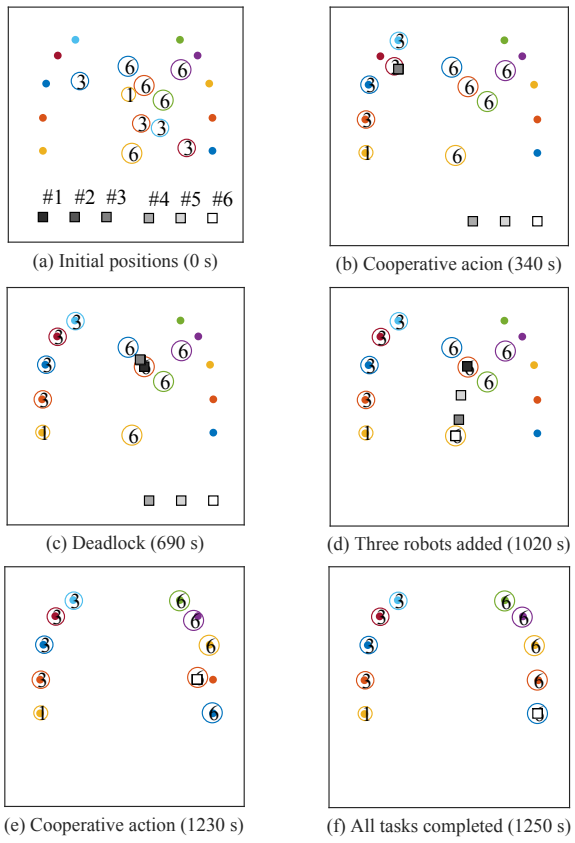

Fig. 8. Validation results of proposed method with 10 objects and after varying the number of robots from 3 to 6.

In Fig. 6 and 7, the policies obtained from the learning of methods removing either the DE or E input from the proposed method resulted in a significant decrease in the success rate. For the method without DE (shown by the red bars in the figures), even when the ratio of weights 3 to 6 is zero (i.e., all objects except for infeasible ones can be transported individually with weight 1), deadlock occurs because of the infeasible objects, and the success rate does not reach 100 %. For the method without E input (shown by the yellow bars in the figures), when the ratio of weights 3 to 6 was 0, the success rate reached 100 %. However, when heavy objects are transported cooperatively, the success rate decreases to 0 %. From these results, it can be inferred that the removal of DE results in the loss of deadlock avoidance functionality, and the removal of the E input resulted in the loss of cooperative transportation capability.

D. Effectiveness of Temporary Priority Exclusions

We confirmed the effect of temporarily resetting the priority using output gates through an experiment (Validation 3) in which robots were added within an episode. Fig. 8 shows the results sampled during a 2, 000 s episode. In Fig. 8 (a)–(c), 3 cooperating robots were able to transport objects weighing up to 3 before encountering a deadlock with objects weighing 6. In Fig. 8 (d)–(f), with 3 additional robots, all robots cooperate to transport all objects weighing 6.

VI. CONCLUSIONS

In this study, we propose a framework for dynamic task allocation in multiple transportation tasks where the weights of the objects are unknown and there are infeasible objects. In the proposed method, multi-robots sequentially select objects using a dynamic task allocation approach. To achieve scalability in terms of the number of robots and objects, we learn the timing for utilizing global information and the target priorities of neighboring tasks based on partial observations. We calculated dynamic priorities using an consensus protocol. We also constructed target exclusion levels and consensus protocols to temporarily reset priorities using an output gate, thereby avoiding deadlocks caused by infeasible objects. This construction enables task allocation strategies to be executable without prior consideration of the feasibility of the task. By performing numerical experiments, we confirmed the efficiency of task allocation through both individual and cooperative transportation, which can be achieved even with an increasing number of robots and objects (scalability), the ability to allocate tasks to unlearned objects (versatility), and the effectiveness of the temporary prevention of deadlocks while the number of robots was insufficient to execute the tasks.

Our proposed method assumes the use of a cloud server to obtain the required global information. In the future, we plan to implement the proposed method to validate its feasibility in real-world environments.

References

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," The International journal of robotics research, vol. 23, no. 9, pp. 939–954, 2004.

[2] N. Lissandrini, C. K. Verginis, P. Roque, A. Cenedese, and D. V. Dimarogonas, "Decentralized nonlinear mpc for robust cooperative manipulation by heterogeneous aerial-ground robots," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 1531–1536.

[3] F. Bertoncelli, F. Ruggiero, and L. Sabattini, "Characterization of grasp configurations for multi-robot object pushing," in 2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS). IEEE, 2021, pp. 38–46.

[4] Y. Liu, F. Zhang, P. Huang, and X. Zhang, "Analysis, planning and control for cooperative transportation of tethered multi-rotor uavs," Aerospace Science and Technology, vol. 113, p. 106673, 2021.

[5] M. Doakhan, M. Kabganian, and A. Azimi, "Cooperative payload transportation with real-time formation control of multi-quadrotors in the presence of uncertainty," Journal of the Franklin Institute, vol. 360, no. 2, pp. 1284–1307, 2023.

[6] H. Chakraa, F. Guérin, E. Leclercq, and D. Lefebvre, "Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art," Robotics and Autonomous Systems, p. 104492, 2023.

[7] L. Liu and D. A. Shell, "Assessing optimal assignment under uncertainty: An interval-based algorithm," The International Journal of Robotics Research, vol. 30, no. 7, pp. 936–953, 2011.

[8] L. Sabattini, V. Digani, C. Secchi, and C. Fantuzzi, "Optimized simultaneous conflict-free task assignment and path planning for multi-agv systems," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 1083–1088.

[9] A. Kimmel and K. Bekris, "Decentralized multi-agent path selection using minimal information," in Distributed Autonomous Robotic Systems, N.-Y. Chong and Y.-J. Cho, Eds. Tokyo: Springer Japan, 2016, pp. 341–356.

[10] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," Proceedings of the IEEE, vol. 94, no. 7, pp. 1257–1270, 2006.

[11] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," IEEE transactions on robotics, vol. 25, no. 4, pp. 912–926, 2009.

[12] M. Braquet and E. Bakolas, "Greedy decentralized auction-based task allocation for multi-agent systems," IFAC-PapersOnLine, vol. 54, no. 20, pp. 675–680, 2021.

[13] Y. Yang and J. Wang, "An overview of multi-agent reinforcement learning from game theoretical perspective," arXiv preprint arXiv:2011.00583, 2020.

[14] K. Shibata, T. Jimbo, T. Odashima, K. Takeshita, and T. Matsubara, "Learning locally, communicating globally: Reinforcement learning of multi-robot task allocation for cooperative transport," IFAC-PapersOnLine, vol. 56, no. 2, pp. 11 436–11 443, 2023.

[15] M. Wen, J. Kuba, R. Lin, W. Zhang, Y. Wen, J. Wang, and Y. Yang, "Multi-agent reinforcement learning is a sequence modeling problem," in Advances in Neural Information Processing Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 16 509–16 521.

[16] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[17] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1, 2018.

[18] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable reinforcement learning policies for multi-agent control," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4785–4791, 2020.

[19] T. S. Dahl, M. J. Mataric, and G. S. Sukhatme, "Adaptive spatio-temporal organization in groups of robots," in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1. IEEE, 2002, pp. 1044–1049.

[20] R. J. Alitappeh and K. Jeddisaravi, "Multi-robot exploration in task allocation problem," Applied Intelligence, vol. 52, no. 2, pp. 2189–2211, 2022.

[21] F. Janati, F. Abdollahi, S. S. Ghidary, M. Jannatifar, J. Baltes, and S. Sadeghnejad, "Multi-robot task allocation using clustering method," in Robot Intelligence Technology and Applications 4: Results from the 4th International Conference on Robot Intelligence Technology and Applications. Springer, 2017, pp. 233–247.

[22] G. Theraulaz, E. Bonabeau, and J. Denuebourg, "Response threshold reinforcements and division of labour in insect societies," Proceedings of the Royal Society of London. Series B: Biological Sciences, vol. 265, no. 1393, pp. 327–332, 1998.

[23] M. J. Krieger and J.-B. Billeter, "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots," Robotics and Autonomous Systems, vol. 30, no. 1-2, pp. 65–84, 2000.

[24] C. J. Watkins and P. Dayan, "Q-learning," Machine learning, vol. 8, pp. 279–292, 1992.

[25] A. M. Kwasnica, J. O. Ledyard, D. Porter, and C. DeMartini, "A new and improved design for multiobject iterative auctions," Management science, vol. 51, no. 3, pp. 419–434, 2005.

[26] Y.-T. Tian, M. Yang, X.-Y. Qi, and Y.-M. Yang, "Multi-robot task allocation for fire-disaster response based on reinforcement learning," in 2009 International Conference on Machine Learning and Cybernetics, vol. 4. IEEE, 2009, pp. 2312–2317.

[27] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," Aerospace Science and Technology, vol. 92, pp. 588–594, 2019.

[28] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," IEEE access, vol. 7, pp. 39 974–39 982, 2019.

[29] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," IEEE access, vol. 7, pp. 146 264–146 272, 2019.

[30] H. Tang, A. Wang, F. Xue, J. Yang, and Y. Cao, "A novel hierarchical soft actor-critic algorithm for multi-logistics robots task allocation," Ieee Access, vol. 9, pp. 42 568–42 582, 2021.

[31] T. Niwa, K. Shibata, and T. Jimbo, "Multi-agent reinforcement learning and individuality analysis for cooperative transportation with obstacle removal," in Distributed Autonomous Robotic Systems: 15th International Symposium. Springer, 2022, pp. 202–213.

[32] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Maddpg algorithm," github. [Online]. Available:https://github.com/openai/maddpg [Accessed: 3-Nov-2021].