Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 11436–11443

Learning Locally, Communicating Globally: Reinforcement Learning of Multi-robot Task Allocation for Cooperative Transport

Kazuki Shibata * **Tomohiko Jimbo** ** **Tadashi Odashima** **
Keisuke Takeshita ** **Takamitsu Matsubara** ***

* *Applied Mathematics Research-Domain, Toyota Central R&D Labs., Inc., 41-1, Yokomichi, Nagakute, Aichi 480-1192, Japan (e-mail: kshibata@mosk.tytlabs.co.jp).*
** *R-Frontier Division, Frontier Research Center, Toyota Motor Corporation, 1, Toyota-cho, Toyota, Aichi 471-8571, Japan*
*** *Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology, Nara 630-0192, Japan*

Abstract: We consider task allocation for multi-object transport using a multi-robot system, in which each robot selects one object among multiple objects with different and unknown weights. The existing centralized methods assume the number of robots and tasks to be fixed, which is inapplicable to scenarios that differ from the learning environment. Meanwhile, the existing distributed methods limit the minimum number of robots and tasks to a constant value, making them applicable to various numbers of robots and tasks. However, they cannot transport an object whose weight exceeds the load capacity of robots observing the object. To make it applicable to various numbers of robots and objects with different and unknown weights, we propose a framework using multi-agent reinforcement learning for task allocation. First, we introduce a structured policy model consisting of 1) predesigned dynamic task priorities with global communication and 2) a neural network-based distributed policy model that determines the timing for coordination. The distributed policy builds consensus on the high-priority object under local observations and selects cooperative or independent actions. Then, the policy is optimized by multi-agent reinforcement learning through trial and error. This structured policy of local learning and global communication makes our framework applicable to various numbers of robots and objects with different and unknown weights, as demonstrated by simulations.

Keywords: Networked robotic systems, Multi-agent systems, Consensus, Decentralized control, Decentralized Control and Systems

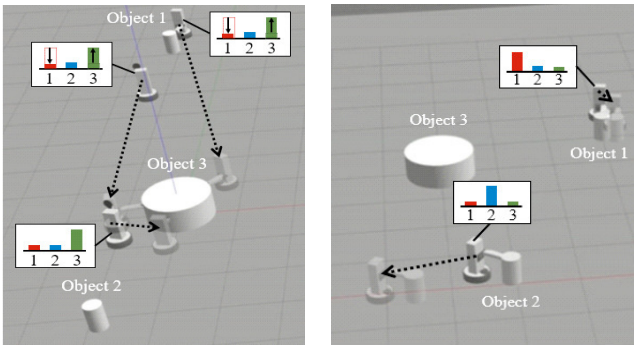
1. INTRODUCTION

In recent years, multi-robot transport has attracted attention in robotics for various applications such as delivery services, factory logistics, and search and rescue. To transport multiple objects over large areas, a team of robots can outperform a single robot in terms of load capacity, time efficiency, and robustness to individual robot failures. Unlike single-robot transport, multi-robot transport involves task allocation and cooperative manipulation. Each robot should select an object to transport multiple objects efficiently. Moreover, force control is required when various robots cooperate to transport a common object to its desired position (Culbertson and Schwager (2018)).

We consider task allocation for multi-object transport using a multi-robot system. In this study, a task corresponds to an object. The existing studies on multi-robot task allocation have adopted deterministic optimization methods (Liu and Shell (2011); Sabattini et al. (2017)) or auction methods (Braquet and Bakolas (2021)) under the assumption that the number of robots to execute each task is available. However, these assumptions are not always realistic. For instance, by using a camera, it may be possible to obtain information on the shape of an object; however, it is challenging to obtain the number of robots required to transport it. In this case, the assumption does not hold.

We explore multi-agent reinforcement learning (MARL) for multi-object transport using a multi-robot system. Each robot selects one object among multiple objects with different and unknown weights. In this study, the number of environmental objects and robots are constant during the training phase but variable during the execution phase. The objective is to transport all the objects to the desired positions as quickly as possible. The existing centralized methods assume the number of robots and tasks to be fixed (Qie et al. (2019); Niwa et al. (2022)), which is inapplicable to the scenarios in which the number of robots and tasks differs from the learning environment. Meanwhile, the existing distributed methods adopt local observation by limiting the minimum number of robots and tasks to a constant value, making them applicable to

10.1016/j.ifacol.2023.10.431



(a) Cooperative action (b) Independent action

Fig. 1. Multi-object transport using a multi-robot system. (a) Robots perform cooperative actions by building a consensus on the high-priority object when they cannot move the object. (b) Robots perform independent actions when they can move the selected objects.

various numbers of robots and tasks (Hsu et al. (2021)). However, they cannot transport an object whose weight exceeds the load capacity of robots observing the object.

To utilize the advantages of the centralized and distributed methods, we propose a framework using MARL for task allocation. The proposed framework first uses a structured policy model consisting of 1) predesigned dynamic task priorities with global communication and 2) a neural-network-based distributed policy model that determines the timing for coordination. The distributed policy reaches a consensus regarding high-priority tasks under local observations and selects cooperative or independent actions, as illustrated in Fig. 1. The policy is optimized by MARL through trial and error. This structured policy of local learning and global communication makes our framework applicable to various number of robots and objects with different and unknown weights. Although the present study has adopted global inter-agent communication and the validation is limited to homogeneous robots, our framework can reduce the transport time while transporting all the objects to the desired positions for various numbers of robots and objects compared to other methods via simulations.

The contributions of this study can be summarized as follows:

- We propose a learning framework using a structured policy model consisting of predesigned dynamic task priorities with global communication and a neural-network-based distributed policy model for multi-robot task allocation.
- Unlike the deterministic optimization and auction methods, our method does not require the number of robots to execute each task and can be applied to a wide range of task allocation problems.
- We confirm that our method can maintain the high performance for various numbers of robots and objects with different and unknown weights through multi-object transport simulations.

The remainder of this paper is organized as follows. Section 2 presents the related work on multi-robot task allocation. Section 3 describes the allocation problem for multi-object transport using a team of robots. Section 4 details the MARL and the proposed learning framework. Section 5 shows the effectiveness of our framework through multi-robot transport simulations. Finally, section 6 summarizes the study and provides directions for future work.

2. RELATED WORK

2.1 Deterministic optimization methods

Deterministic optimization formulates the task allocation problem as an optimization problem aimed at minimizing the total travel distance under constraints for the number of robots required for each task. These approaches have adopted various optimization techniques, such as the Hungarian algorithm (Kuhn (1955); Liu and Shell (2011)), integer linear programming (Sabattini et al. (2017)), and mixed integer linear programming (Flushing et al. (2017)). Although these studies can guarantee optimality in terms of the total travel distance, most methods require prior information regarding the number of robots required for each task.

2.2 Distributed Metaheuristic Methods

Metaheuristic methods are inspired by the division of labor exhibited by social insects. A common approach has adopted threshold models (Theraulaz et al. (1998); Krieger and Billeter (2000)), in which each robot selects a task under local observations using an activation threshold and a stimulus associated with each task. Although these methods can handle varying numbers of robots and tasks, they may allocate unnecessary tasks to robots, thus reducing the time efficiency.

2.3 Auction Methods

Auction algorithms (Gerkey and Mataric (2004); Dias et al. (2006)) are common methods for multi-robot task allocation and have been studied via centralized and decentralized approaches. The centralized method (Kwasnica et al. (2005)) adopts the auctioneer, which collects the bids from the bidders, and allocates the highest bidder to the task. In contrast, Choi et al. (2009) propose a decentralized auction-based algorithm without the auctioneer. This method adopts a consensus algorithm to estimate the bids of other robots using local communication with other robots. Then, the robots allocate the task to the highest bidder using the estimated bids. However, their method focuses on the problem where a single robot can execute each task.

Braquet and Bakolas (2021) addressed the closest problem to our study, where each task requires multiple robots. Their method adopts the consensus algorithm similar to Choi et al. (2009), which estimates the list of selected tasks, the list of winning bids, and the list of completed allocations. Robots assign a task to the robot with the highest bid among the unassigned robots based on the list of completed allocations. However, their methods require a probability of completing each task, which is difficult to compute for objects with unknown weights.

2.4 MARL Methods

Recent studies (Qie et al. (2019); Niwa et al. (2022)) have addressed task allocation problems using MARL. These approaches formulate a task allocation problem using the Markov decision process and learn the optimal policies using a multi-agent deep deterministic policy gradient (MADDPG) (Lowe et al. (2017b)). However, these methods adopt centralized training assuming that the number of robots and tasks is constant, failing in scenarios with different numbers of robots and tasks. To address this problem, Hsu et al. (2021) proposed a distributed policy model, which can be applied to various numbers of robots and tasks. The trained policies are applicable to up to 1000 robots and 1000 tasks through multi-target tracking simulations. However, they cannot handle a situation where the number of robots required to execute a task exceeds the number of robots observing it.

Although the proposed framework uses distributed policies under local observations, it differs from the method (Hsu et al. (2021)) in that our method employs a structured policy model consisting of predesigned dynamic task priorities with global communication and a neural network-based distributed policy model. Therefore, robots can perform all the tasks efficiently even when the number of robots required to complete each task is different and unknown.

3. PRELIMINARY

3.1 Problem Formulation

We consider a team of N robots. Each of these robots selects one object simultaneously among the M objects with different and unknown weights. In our setting, N and M are constant during the training phase but variable during the execution phase. The position of robot $i\,(i = 1, \cdots, N)$ is represented by $\boldsymbol{x}_i \in \mathbb{R}^2$. The position, velocity, and desired position of the object $l\,(l = 1, \cdots, M)$ are represented by $\boldsymbol{z}_l \in \mathbb{R}^2$, $\boldsymbol{v}_l \in \mathbb{R}^2$ and $\boldsymbol{z}_l^* \in \mathbb{R}^2$, respectively. Robot i has a local observation, including K nearest robots $j \in \mathcal{N}_i^{\mathrm{Robot}} := \{j_{i1}, \cdots, j_{iK}\}$ and K nearest objects $l \in \mathcal{N}_i^{\mathrm{Load}} := \{l_{i1}, \cdots, l_{iK}\}$, where K is constant. In this study, we simplify the transport problem such that the robots can move the object if the total load capacity of the robot within a certain distance from the object exceeds the mass of the object. The dynamics of the robot is a single integrator dynamics given by

$$\dot{\boldsymbol{x}}_i = \begin{cases} k_p(\boldsymbol{z}_{l_i^*} - \boldsymbol{x}_i) \text{ if } \|k_p(\boldsymbol{z}_{l_i^*} - \boldsymbol{x}_i)\| \leq v_i^{\max} \\ v_i^{\max} \dfrac{\boldsymbol{z}_{l_i^*} - \boldsymbol{x}_i}{\|\boldsymbol{z}_{l_i^*} - \boldsymbol{x}_i\|} \text{ otherwise,} \end{cases}$$

where k_p is a positive gain, $v_i^{\max}$ is the maximum speed of robot i and l_i^* is the selected object by robot i.

The objective is to transport all the objects to the desired positions as quickly as possible.

We made the following assumptions:

- Robots know M and N
- Robots know the current and desired positions of M objects
- Robots can communicate with other robots if necessary

3.2 MARL Settings for Multi-robot Task Allocation

To address the multi-robot task allocation problem for multi-object transport, we describe the MARL settings using a Markov decision process.

Let us denote the state, action, and observation of robot i $(i = 1, \cdots, N)$ as $\boldsymbol{s}_i$, $\boldsymbol{a}_i$, and $\boldsymbol{o}_i$, respectively. Robot i selects action $\boldsymbol{a}_i$ under local observation $\boldsymbol{o}_i$ including robots $j \in \mathcal{N}_i^{\mathrm{Robot}}$ and objects $l \in \mathcal{N}_i^{\mathrm{Load}}$. Action $\boldsymbol{a}_i$ includes a variable to compute the priority of objects $l \in \mathcal{N}_i^{\mathrm{Load}}$ and variables to determine communicating task priorities with other robots, as described in Section 4. Robot i updates the task priorities by computing the current actions $\boldsymbol{a}_i$, then selects the object with the highest priority among the M objects. After robot i moves to the selected object for a certain control period, $\boldsymbol{s}_i$ transitions to the next state $\boldsymbol{s}_i'$. Simultaneously, robot i receives reward r_t at every step t when moving the object or carrying it to the desired position. Robot i updates its policy by maximizing the expected reward $\mathbb{E}[R_t] = \mathbb{E}\left[\sum_{k=0}^{T-1} \gamma^k r_{t+k}\right]$, where $\gamma \in [0, 1]$ is a discount factor and T is the total number of steps per episode.

4. METHOD

In this section, we introduce the proposed MARL framework that can handle a varying number of robots and objects with different and unknown weights.

Fig. 2 shows the overview of the learning framework. Robot i has task priority $\boldsymbol{\phi}_i := \left[\phi_i^1, \cdots, \phi_i^M\right]^\top \in \mathbb{R}^M$, where $\phi_i^l \in [0, 1]$ is the priority of the lth object possessed by robot i. Robot i updates the priority of the neighboring object $l \in \mathcal{N}_i^{\mathrm{Load}}$ under local observation $\boldsymbol{o}_i = [\boldsymbol{x}_i, \phi_i^{l_{i1}}, \cdots, \phi_i^{l_{iK}}, \boldsymbol{x}_{j_{i1}}, \phi_{j_{i1}}^{l_{i1}}, \cdots, \phi_{j_{i1}}^{l_{iK}}, \cdots, \boldsymbol{x}_{j_{iK}}, \phi_{j_{iK}}^{l_{i1}}, \cdots, \phi_{j_{iK}}^{l_{iK}}, \boldsymbol{z}_{l_{i1}}, \boldsymbol{v}_{l_{i1}}, \boldsymbol{z}_{l_{i1}}^*, \cdots, \boldsymbol{z}_{l_{iK}}, \boldsymbol{v}_{l_{iK}}, \boldsymbol{z}_{l_{iK}}^*]$ using $\boldsymbol{c}_i = [c_i^1, \cdots, c_i^K]^\top \in \mathbb{R}^K$, where $c_i^l \in [0, 1]$ is the reference value of ϕ_i^l. The reference value $\boldsymbol{c}_i$ is computed by policy π_i. Limiting the minimum number of robots and objects to a constant value makes the policy applicable to varying numbers of robots and objects. However, this policy cannot transport an object whose weight exceeds the load capacity of robots observing the object because it cannot update the priorities of the object $l \notin \mathcal{N}_i^{\mathrm{Load}}$.

The proposed framework introduces dynamic task priorities with global communication and a neural network-based distributed policy model. The distributed policy computes communication inputs $\alpha_i \in [0, 1]$ and $\beta_i \in [0, 1]$ under local observations, where α_i is the parameter by which the robot i receives task priority from other robots, and β_i is the parameter by which the robot i sends $\boldsymbol{\phi}_i$ to other robots. If robots communicate the task priority with other robots, the dynamic task priority makes the agents establish a consensus on the high-priority object and select cooperative actions. Otherwise, robots select independent actions. Therefore, robots can transport all objects efficiently without knowing the number of robots required to transport objects. Robot i selects the object l_i^*, which has the highest priority among M objects. Then, the policy is optimized by MARL through trial and error.

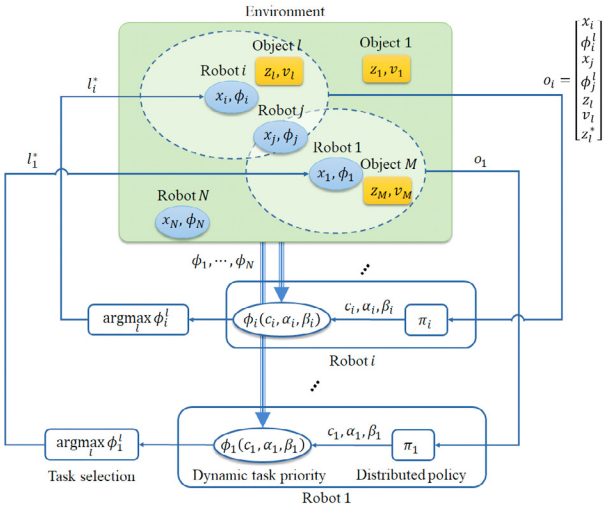


Fig. 2. Overview of learning framework. Robot i updates task priorities of the neighboring objects using $\boldsymbol{c}_i$ while building consensus on the high-priority object using α_i and β_i according to the distributed policy π_i under local observations $\boldsymbol{o}_i$. Robot i selects the object l_i^* which has the highest priority among M objects.

4.1 Dynamic Task Priority with Global Communication

This subsection introduces the dynamic task priority with global communication to select an object among various candidates.

We design the dynamic task priority such that the robot i can update ϕ_i^l ($l \in \mathcal{N}_i^{\text{Load}}$) according to its policy while updating ϕ_i^l ($l \notin \mathcal{N}_i^{\text{Load}}$) using the priorities of the N robots. In this case, the robots should balance cooperative and independent actions to transport all the objects efficiently. To this end, we design the dynamic task priority of object l for robot i given by

$$\dot{\phi}_i^l = \begin{cases} k_\phi(c_i^l - \phi_i^l) + \sigma_i \sum_{j=1}^{N} d_j k_\phi(\phi_j^l - \phi_i^l) \text{ if } l \in \mathcal{N}_i^{\text{Load}} \\ \sigma_i \sum_{j=1}^{N} d_j k_\phi(\phi_j^l - \phi_i^l) \text{ otherwise,} \end{cases} \tag{1}$$

where $k_\phi > 0$, d_i and σ_i are equal to 0 or 1. We introduced the first-order linear time-delay system to avoid the occurrences of chattering, where the robots travel back and forth between different objects.

In (1), $k_\phi(c_i^l - \phi_i^l)$ induces an independent action while $k_\phi(\phi_j^l - \phi_i^l)$ induces a cooperative action. If $\sigma_i = 1$ and $d_j = 1$, $k_\phi(\phi_j^l - \phi_i^l)$ makes ϕ_i^l asymptotically converge to ϕ_j^l, establishing consensus on the task priority. Otherwise, $k_\phi(c_i^l - \phi_i^l)$ makes ϕ_i^l asymptotically converge to c_i^l according to its own policy. The distributed policy calculates σ_i and d_i to reach a consensus on the high-priority object as well as $\boldsymbol{c}_i$ under local observations.

4.2 Distributed Policy Model

We introduce a distributed policy model under local observations $\boldsymbol{o}_i$ given by

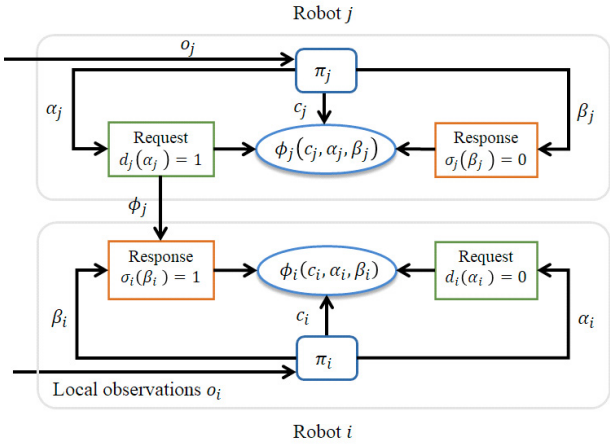


Fig. 3. Example of the communication of the task priority using the proposed distributed policy under local observation. When $d_j(\alpha_j) = 1$ and $\sigma_i(\beta_i) = 1$, robot i receives $\boldsymbol{\phi}_j$ transmitted by robot j.

$$\boldsymbol{a}_i = \left[\boldsymbol{c}_i^\top, \alpha_i, \beta_i\right]^\top = \pi_i(\boldsymbol{o}_i) \tag{2}$$

where π_i is computed by a deep neural network, as described in subsection 4.5. Robot i determines the reference values of ϕ_i^l using c_i^l for K local objects while maintaining the priority of the M objects.

Using α_i and β_i in (2), request signal d_i and response signal σ_i are calculated by the event-triggered law (Baumann et al. (2018); Shibata et al. (2021)) given by

$$d_i(\alpha_i) = \begin{cases} 1, \text{ if } \alpha_i > 0.5 \ \& \ \|\boldsymbol{v}_{l_i^*}\|_2 = 0 \\ 0, \text{ otherwise} \end{cases}, \tag{3}$$

$$\sigma_i(\beta_i) = \begin{cases} 1, \text{ if } \beta_i > 0.5 \ \& \ \|\boldsymbol{v}_{l_i^*}\|_2 = 0 \\ 0, \text{ otherwise} \end{cases}, \tag{4}$$

where robot i can transmit and receive the priority when it cannot move the selected object l_i^*. Fig. 3 illustrates the communication of the task priority using our distributed policy under local observation. Using the triggering law in Eqs. (3) and (4), robot i can receive $\boldsymbol{\phi}_j$ transmitted by robot j and then reach consensus on the high-priority object using (1).

4.3 Object Selection

This subsection introduces the procedure for the selection of an object based on its priority. Robot i selects the object with the highest priority among M objects using $l_i^* = \arg\max_l \phi_i^l$. Moreover, we set the priority of the object that has reached close to the desired position using $\phi_i^l \leftarrow 0$, if $\|\boldsymbol{z}_l - \boldsymbol{z}_l^*\|_2 < \delta$, where $\delta > 0$ represents a threshold to determine whether the object reaches the desired position.

4.4 Reward Design

To transport all the objects to the desired positions as quickly as possible, we designed a reward function given by

$$r = \sum_{l=1}^{M} P_l + \lambda \sum_{l=1}^{M} \|\boldsymbol{v}_l\|_2, \tag{5}$$

where λ is a positive constant. $P_l = 1$ if $\|\boldsymbol{z}_l - \boldsymbol{z}_l^*\|_2 < \delta$; otherwise $P_l = 0$. The first term in (5) aims to transport all the objects to the desired position, while the second term aims to move as many objects as possible.

4.5 Policy Optimization

In this study, we optimized the multi-agent policies using multi-agent deep deterministic policy gradient (MADDPG) (Lowe et al. (2017b)), which is one of the deep actor-critic algorithms for multi-agent systems.

A common issue of MARL is that the learning becomes unstable as the number of unobservable agents increases. The MADDPG algorithm addressed this problem using a learning framework called "centralized training and decentralized execution." During training, the weight parameters of the critic networks are optimized through the Q-learning algorithm using the observations and actions of all the agents. At the same time, the weight parameters of the actor networks are optimized through a policy gradient method using its observations and actions. During execution, the actor networks compute actions under local observations. See Lowe et al. (2017b) for the details of the policy optimization steps.

5. SIMULATION

We conducted multi-object transport simulations using multiple robots to confirm the scalability and versatility of the proposed framework for various numbers of robots and objects and various proportions of heavy and light objects.

5.1 Simulation Setup

We show the simulation scenario in Fig. 4. We randomly generated the initial positions of the robots and objects in the region $Q := \{(x, y) \mid 2 \le x \le 8, 2 \le y \le 8\}$. The desired positions of the objects were evenly arranged on a circumference with a center and radius of $[5.0, 5.0]^\top$ and 4.0 m, respectively. We set the load capacity of the robot to 1 kg.

During training, we set $K = 2$, $N = 3$, and $M = 6$, while setting the object's mass to 1 or 3 kg with 50 % probability. To confirm the scalability of the algorithm, we evaluated $N \in \{3, 6\}$ and $M \in \{4, 6, 8, 10\}$.

We used the MADDPG code (Lowe et al. (2017a)) and set the simulation parameters as listed in Table 1.

We set $k_\phi = 0.2$ in (1) such that the priority changed according to a first-order delay with the time constant of 5.0 s, which was longer than the selection period. The threshold δ was set to 0.05 for the positions of the objects to be controlled within 0.05 m from the desired positions. We set the weight parameter λ in (5) to 3.0×10^1 such that transporting a different object obtained almost the same reward as locating an object to the desired position.

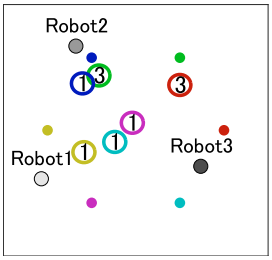


Fig. 4. Simulation scenario. The colored circles, dots, and numbers indicate the objects, their desired positions, and the number of robots required to transport the object, respectively. Robots should transport each object to the desired position with the same color.

The proposed method can select different strategies depending on the situation. To confirm this property, we conducted comparisons through the following methods:

- **Nearest**: Each robot selects the nearest object
- **One**: Each robot is randomly assigned an object from the M objects
- **Local**: Our method without dynamic task priority with global communication by setting $\sigma_i = 0$ in (1).
- **Nearest-one**: Each robot selects the object closest to its current position. When the robot does not move the object for a specific time t_s, the robot picks the same object as the robot, unable to carry the load for the longest time. We set $t_s = 1.0$ s for all the robots.
- **No-com**: **Local** method under local observations without the task priorities.
- **No-dynamics**: **No-com** method without the dynamics of the task priority by setting $\phi_i^l = c_i^l$ ($l \in \mathcal{N}_i^{\mathrm{Load}}$).

To evaluate our approach quantitatively, we used the following measures:

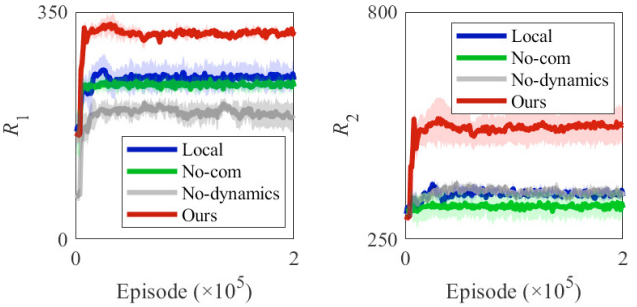
- Success rate (SR): The ratio of trials to 100 trials, in which robots can transport all the objects to the desired positions within 10 min. We considered 10 min for method **One** to achieve a 100 % success rate for various numbers of robots and objects.
- Transport time (TT) [s]: Average time required to move all the objects to the desired positions within 10 min.

5.2 Comparisons of Training Performance

We evaluated the effects of dynamic task priority and communication on the training performance by comparing

Table 1. Simulation parameters

Parameter	Value
Selection period [s]	1.0
Number of steps per episode	150
Number of episodes	2.0e5
Number of hidden layers (critic)	4
Number of hidden layers (actor)	4
Number of units per layer	64
Activation function of hidden layers	ReLU
Activation function of output layers (critic)	linear
Activation function of output layers (actor)	tanh
Discount factor	0.99
Batch size	1024



(a) First term (b) Second term

Fig. 5. Cumulative rewards of various evaluated methods.

our framework with methods **Local**, **No-com**, and **No-dynamics**. For each method, we repeated the training three times.

Fig. 5 shows the cumulative rewards of the first and second terms in (5), which are denoted as R_1 and R_2, respectively. When applying method **No-dynamics**, we confirmed the occurrences of chattering where the robots travel back and forth between different objects. As a result, this method made R_1 achieve smaller values compared to those in other methods.

Method **Local** achieves slightly higher R_1 and R_2 values than method **No-com**. Therefore, training the policy with the priority of neighboring robots improves the training performance. Moreover, the proposed framework achieves higher values than the other methods. These results indicate that the dynamic task priority with global communication in (1) has a greater impact on the training performance of our framework than the local communication of task priorities.

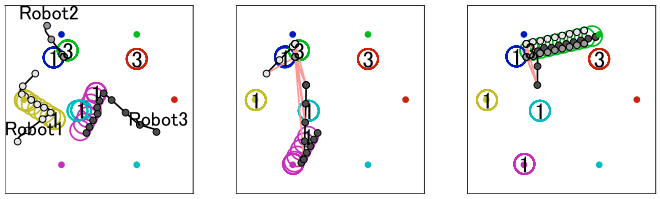
5.3 Emergence of Cooperative and Independent Actions

We confirmed the emergence of cooperative and independent actions when applying the proposed framework. We show trajectories and communication occurrences when applying the framework in Fig. 6.

At the initial stage, robot 2 cannot move the object, which requires three robots to transport, as shown in Fig. 6(a). To prevent this situation, robot 2 transmits its priority to other robots by setting $d_2 = 1.0$, as shown in Fig. 7(a). At the same time, robots 1 and 3 receive the priority by setting $\sigma_1 = 1.0$ and $\sigma_3 = 1.0$, as shown in Fig. 7(b). As a result, the task priority of the green object is the highest for the three robots, as shown in Fig. 7(c). Then, the robots select cooperative actions and transport the green object to the desired position, as shown in Fig. 6(c).

Finally, two objects remain to be transported by three robots, as shown in Fig. 6(e). While three robots transport the light-blue object, the priority of the blue object is the highest for robot 3, as shown in Fig. 7(c). Afterwards, robot 3 select independent actions and transport the blue object to the desired position, as shown in Fig. 6(f).

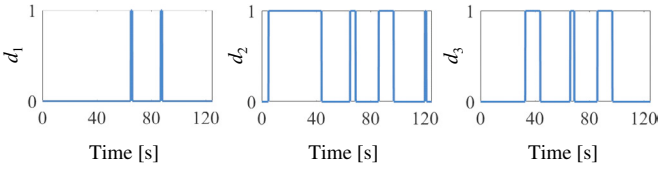
Overall, our framework can balance cooperative and independent actions using the communication signals and dynamic task priorities.



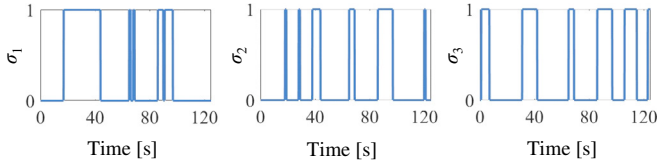
(a) 0 - 20 s (b) 20 - 40 s (c) 40 - 60 s

(d) 60 - 80 s (e) 80 - 100 s (f) 100 - 120 s

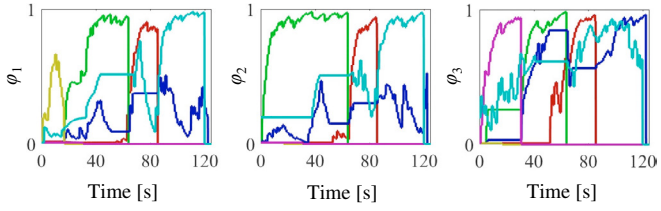
Fig. 6. Trajectories and communication occurrences. We offset the overlapping robot trajectories for clarity. The black and red lines show the trajectories of the robots and occurrences of priority communication.



(a) Request signals d_i



(b) Response signals σ_i



(c) Task priorities ϕ_i

Fig. 7. The time evolution of request signals, response signals and task priorities for three robots. The colors in Fig. 7(c) correspond to those of the objects, as shown in Fig. 6.

5.4 Scalability Analysis

We evaluated the success rate and transport time when using our framework and other methods for various numbers of robots and objects.

Table 2 shows the quantitative results for various numbers of robots and objects when applying each method. Methods **Nearest** and **Local** cannot achieve a 100 % success rate for various numbers of robots and objects. In contrast, **One**, **Nearest-one**, and **Ours** can achieve a 100 % success rate for various numbers of robots and objects.

Table 2. Quantitative results for various numbers of robots and objects

(N, M)	Metrics	Nearest	One	Local	Nearest -one	**Ours**
(3,4)	SR	0.59	**1.0**	0.93	**1.0**	**1.0**
	TT ($\times 10^2$)	**1.1**	1.4	1.3	1.2	1.2
(3,6)	SR	0.34	**1.0**	0.85	**1.0**	**1.0**
	TT ($\times 10^2$)	**1.7**	2.0	2.2	1.9	1.8
(3,8)	SR	0.17	**1.0**	0.67	**1.0**	**1.0**
	TT ($\times 10^2$)	**2.2**	2.7	2.8	2.5	2.4
(3,10)	SR	0.07	**1.0**	0.54	**1.0**	**1.0**
	TT ($\times 10^2$)	**2.5**	3.4	3.5	3.1	3.0
(6,4)	SR	0.97	**1.0**	**1.0**	**1.0**	**1.0**
	TT ($\times 10^2$)	0.99	1.3	0.91	0.88	**0.86**
(6,6)	SR	0.88	**1.0**	0.98	**1.0**	**1.0**
	TT ($\times 10^2$)	1.5	2.0	**1.4**	**1.4**	**1.4**
(6,8)	SR	0.82	**1.0**	0.88	**1.0**	**1.0**
	TT ($\times 10^2$)	1.9	2.7	1.9	1.9	**1.8**
(6,10)	SR	0.74	**1.0**	0.72	**1.0**	**1.0**
	TT ($\times 10^2$)	2.6	3.6	2.5	2.4	**2.2**

When applying Method **One**, the transport time is the longest because all the robots select the same object. To confirm the effectiveness of our method, we evaluated the average time t_a for carrying two or more objects simultaneously when applying each method to $(N, M) = (6, 10)$ for 100 trials. Note that the large t_a value indicates that the robots select more independent actions. Our method achieves $t_a = 6.0 \times 10^1$ while Method **Nearest-one** achieves $t_a = 4.9 \times 10^1$. The results indicate that our method can promote more independent actions compared to Method **Nearest-one**.

Overall, compared to other methods, our framework can reduce the transport time while transporting all the objects to the desired positions for various numbers of robots and objects.

5.5 Versatility Analysis by Varying Proportion of Heavy and Light Objects

Additionally, we verified the versatility of our framework by varying the proportion of heavy and light objects. We set $N = 6$ and $M = 10$ while setting the mass of the objects to 1 or 3 kg. We evaluated each method by generating 3 kg objects with probabilities of 0 %, 25 %, 50 %, 75 %, and 100 %.

Table 3 shows the quantitative results for various proportions of heavy and light objects. When applying methods **Nearest** and **Local**, the success rate becomes lower with the increasing proportion of heavy objects. In contrast, **One**, **Nearest-one**, and **Ours** can achieve a 100 % success rate for various proportions of heavy objects.

Method **One** increases the transport time compared with **Nearest-one** and our methods because all the robots select a common object regardless of its weight. Moreover, our framework achieves a lower transportation time than method **Nearest-one** for various proportions of heavy objects because our framework can promote more independent actions than Method **Nearest-one** as discussed in subsection 5.4.

Overall, compared to other methods, our framework can reduce the transport time while transporting all the objects to their desired positions when handling objects with various weights.

Table 3. Quantitative results for various proportions of heavy and light objects. P represents the proportion of heavy objects.

P	Metrics	Nearest	One	Local	Nearest -one	**Ours**
0.0	SR	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	TT ($\times 10^2$)	1.2	3.3	1.1	1.2	**1.0**
0.25	SR	0.94	**1.0**	0.98	**1.0**	**1.0**
	TT ($\times 10^2$)	1.9	3.5	1.7	1.8	**1.6**
0.5	SR	0.7	**1.0**	0.76	**1.0**	**1.0**
	TT ($\times 10^2$)	2.5	3.5	2.4	2.4	**2.2**
0.75	SR	0.6	**1.0**	0.36	**1.0**	**1.0**
	TT ($\times 10^2$)	3.1	3.7	3.2	2.9	**2.8**
1.0	SR	0.29	**1.0**	0.14	**1.0**	**1.0**
	TT ($\times 10^2$)	3.5	3.8	3.7	3.4	**3.3**

6. CONCLUSION

We propose a learning framework that can handle scenarios for various numbers of robots and objects with different and unknown weights. The distributed policy model builds consensus on the high-priority object under local observations, thus balancing the cooperative and independent actions. Therefore, compared to other methods, our framework can reduce the transport time while transporting all the objects to their desired positions for various numbers of robots and objects with different and unknown weights.

In the present study, we assume that each robot knows the positions of all the objects. Therefore, we may combine our framework with recurrent MARL models (Wang et al. (2020)) and confirm its effectiveness under partial observations with several unknown object positions. Furthermore, our framework requires global communication between robots. Therefore, we should decentralize the communication structure using techniques such as an attentional communication channel (Zhai et al. (2020)). In addition, we will investigate the influence of metaparameters such as k_ϕ, δ and λ on the learning performance.

In future work, we will validate our framework through experiments on real robots. In addition, we intend to apply our framework to allocation problems involving a team of heterogeneous robots or manipulation tasks in three dimensional environments.

REFERENCES

Baumann, D., Zhu, J.J., Martius, G., and Trimpe, S. (2018). Deep reinforcement learning for event-triggered control. In *2018 IEEE Conference on Decision and Control (CDC)*, 943–950.

Braquet, M. and Bakolas, E. (2021). Greedy decentralized auction-based task allocation for multi-agent systems. *IFAC-PapersOnLine*, 54(20), 675–680.

Choi, H.L., Brunet, L., and How, J.P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4), 912–926.

Culbertson, P. and Schwager, M. (2018). Decentralized adaptive control for collaborative manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 278–285.

Dias, M.B., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7), 1257–1270.

Flushing, E.F., Gambardella, L.M., and Di Caro, G.A. (2017). Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1861–1868. IEEE.

Gerkey, B.P. and Mataric, M.J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9), 939–954.

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

Krieger, M.J. and Billeter, J.B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1), 65–84.

Kuhn, H.W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97.

Kwasnica, A.M., Ledyard, J.O., Porter, D., and DeMartini, C. (2005). A new and improved design for multi-object iterative auctions. volume 51, 419–434.

Liu, L. and Shell, D.A. (2011). Assessing optimal assignment under uncertainty: An interval-based algorithm. *The International Journal of Robotics Research*, 30(7), 936–953.

Lowe, R., WU, Y., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. (2017a). Maddpg code. Github. [Online]. Available: `https://github.com/openai/maddpg` Accessed 3.11.2021.

Lowe, R., WU, Y., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. (2017b). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 30.

Niwa, T., Shibata, K., and Jimbo, T. (2022). Multi-agent reinforcement learning and individuality analysis for cooperative transportation with obstacle removal. In *Distributed Autonomous Robotic Systems*, 202–213. Springer International Publishing, Cham.

Qie, H., Shi, D., Shen, T., Xu, X., Li, Y., and Wang, L. (2019). Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE Access*, 7, 146264–146272.

Sabattini, L., Digani, V., Secchi, C., and Fantuzzi, C. (2017). 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)*, 1083–1088.

Shibata, K., Jimbo, T., and Matsubara, T. (2021). Deep reinforcement learning of event-triggered communication and control for multi-agent cooperative transport. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 8671–8677.

Theraulaz, G., Bonabeau, E., and Deneubourg, J.L. (1998). Response threshold reinforcement and division of labour in insect societies. *Proceedings: Biological Sciences*, 265(1393), 327–332.

Wang, R.E., Everett, M., and How, J.P. (2020). R-maddpg for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684*.

Zhai, Y., Ding, B., Liu, X., Jia, H., Zhao, Y., and Luo, J. (2020). Decentralized multi-robot collision avoidance in complex scenarios with selective communication. *IEEE Robotics and Automation Letters*, 6(4), 8379–8386.