MULTI-ROBOT COOPERATIVE OBJECT TRANSPORTATION USING DECENTRALIZED DEEP REINFORCEMENT LEARNING

A PREPRINT

Lin Zhang*
Department of Aerospace Engineering and Engineering Mechanics
University of Cincinnati
Cincinnati, OH 45221
lin.zhang@uc.edu

Hao Xiong
School of Mechanical Engineering and Automation
Harbin Institute of Technology Shenzhen, China 518055
xionghao@hit.edu.cn

Ou Ma
Department of Aerospace Engineering and Engineering Mechanics
University of Cincinnati
Cincinnati, OH 45221
maou@ucmail.uc.edu

Zhaokui Wang
School of Aerospace Engineering
Tsinghua University
Beijing, China 100084
wangzk@tsinghua.edu.cn

July 21, 2020

ABSTRACT

Object transportation could be a challenging problem for a single robot due to the oversize and/or overweight issues. A multi-robot system can take the advantage of increased driving power and more flexible configuration to solve such a problem. However, increased number of individuals also changed the dynamics of the system which makes control of a multi-robot system more complicated. Even worse, if the whole system is sitting on a centralized decision making unit, the data flow could be easily overloaded due to the upscaling of the system. In this research, we propose a decentralized control scheme on a multi-robot system with each individual equipped with a deep Q-network (DQN) controller to perform an oversized object transportation task. DQN is a deep reinforcement learning algorithm thus does not require the knowledge of system dynamics, instead, it enables the robots to learn appropriate control strategies through trial-and-error style interactions within the task environment. Since analogous controllers are distributed on the individuals, the computational bottleneck is avoided systematically. We demonstrate such a system in a scenario of carrying an oversized rod through a doorway by a two-robot team. The presented multi-robot system learns abstract features of the task and cooperative behaviors are observed. The decentralized DQN-style

*Intelligent Robotics and Autonomous System Lab, University of Cincinnati, Cincinnati, USA. Correspondence to: Lin Zhang<lin.zhang@uc.edu>
controller is showing strong robustness against uncertainties. In addition, We propose a universal metric to assess the cooperation quantitatively.

1 Introduction

In the world of humans, complex tasks require multiple persons to cooperate mentally and physically. For example, in the Space Shuttle STS-49 mission [1, 2], NASA originally planned to have only one astronaut to capture the slow-rotating satellite IntelSat but failed to accomplish the task. The task failed again the second day with two astronauts working together. The task was finally accomplished on the fifth day by three astronauts and one robot (the Canadarm) worked together. The mission set several records including maximum number of astronauts in space walk and longest hours in a single spacwalk [1]. The main reason such task requires multiple astronauts is that the size of the IntelSat is overwhelming for a single astronaut. We can easily imagine that a single robot is facing the same challenge when handling oversized objects. A multi-robot system (MRS) can achieve the goal but deploying such a system requires more advanced coordination and control strategies.

Oversized object transportation is a typical task that is usually composed with smaller subtasks which can be assigned to multiple robots simultaneously [3]. We assume that the robots are physically attached to the object, and transport is achieved by either pushing or pulling (or both) the object. Decentralized architecture is the natural way to control a MRS in such a task as it is more flexible and more scalable [4]. To solve such a problem, on the one hand, an individual robot has to guarantee accomplishment of its own assignment. On the other hand, the robots have to cooperate with each other to achieve the shared high-level goal. However, the variation of the system (e.g. number of the robots, moving obstacles, unpredictable perturbations, etc.) can still bring challenges toward management of the MRS. A leader/follower architecture is very popular in the past for it can plan and control on top of the leader robot’s well-studied dynamics model [5, 6]. Besides of that, researchers has proved that MRS with identical controllers equally distributed on individuals performs the task well [7]. However, treating all the members equally can results in a more complicated dynamics model of the integrated system.

Designing a good controller for a robot could be extremely challenging due to the complicated dynamics of the task. However, the performance of a controller can be easily evaluated through the more obvious success conditions. Instead of modeling dynamics of the system, reinforcement learning (RL) algorithms model the reward mechanisms which directly tells the goodness of a state that is the consequence of the controller’s outputs. Therefore, we can optimize the controller with the guide of the reward function to achieve the goal without knowing the dynamics model at all [8]. The rapidly developing deep learning technologies further boost the RL into deep reinforcement learning (DRL) which enables handling more representative data (e.g. image) and thus has been applied to many complex control problems [9]. DRL is also welcome by the researchers aiming at solving cooperative object transportation tasks with MRS [10–12].

In this paper, we present an MRS controlled by distributed DRL controllers performing cooperative object transportation task. We modified original DQN algorithm [9] to form two algorithms to train controllers in an MRS with homogeneous or heterogeneous robots. Specifically, we instantiate the MRS with two homogeneous mobile robots with each robot driven by a differential driving mechanism. The task is transporting a long rod from inside of a room to the outside of it. The most challenging part is the robots need to transport the rod through a narrow doorway. We employ DQN controllers to regulate behaviors of the robots by estimating the total reward of available actions at any given state then take the action with highest value. Due to the DQN controller’s fundamental mechanism of estimating values of actions given states, we propose to use absolute error of estimated state-action values between two robots to quantitatively measure how well they cooperate with each other. To our best knowledge, this is the first implementation of DRL algorithm in the similar context. The highlights of this research are:

- Controllers are distributed on individual robots. No command center involved;
- No dynamics modeling, no path planning. Controllers make decisions directly from the sensing data (end-to-end);
- The progress of gradually unified value estimations is observed through the newly proposed cooperation metric;

In the next section, we will review some related researches; we introduce our research methodology in section 3; the experiment results and analyses will be described in section 4. Stay tuned...
2 Related Work

More researchers began to be interested in solving the problem of cooperative object transportation using MRS from mid 90’s [5, 13–16]. Although the settings could be varied a lot from each others, there are three major strategies to configure such a problem: 1) pushing-only strategy; 2) grasping strategy; 3) caging strategy [3]. Our research adopt the second strategy as it keeps the complexity of the task that the robots have to coordinate with each others using different actions (pushing or pulling), while the robots are focusing on the transportation without considering the spatial configuration. While centralized control schemes are rarely reported, There exist research using such an organization [17]. Nevertheless, majority of the researchers adopt the distributed architectures, thus the following literatures are default with decentralized control schemes.

When solving MRS based cooperative object transportation problem with grasping strategy, leader/follower configuration is very popular [5, 6, 18–20]. In general, a leader robot is responsible for initiating and directing the transportation, while the follower robots coordinate their actions with respect to the leader’s guidance. It is true that the leader/follower architecture saves the cost of computation and communication, but this architecture sacrifices some flexibility of the follower robots. Under some complicated situations, involving a leader robot with superior capabilities does not make the problem easier.

Researchers investigated MRS with every individual playing the same role, such that the MRS can be more flexible in their tasks. A two-stage motion planning strategy was proposed in [21, 22] to help the MRS safely transport an object around dynamic obstacles. A research scenario that was closely analogous to ours was proposed in [23]. The researchers were able to control two omnidirectional mobile robots to transport an object through an narrow opening with decentralized sliding mode controllers. In this research, predefined trajectories and dynamics model of the MRS needs to be established beforehand. An MRS controlled with decentralized sliding mode controllers was reported in [7], which was able to transport arbitrary shaped objects without predefined trajectories shaped. However, dynamics model of the individual robot is still needed. Research of a decentralized adaptive control strategy were reported in [24], which enabled cooperative object transportation with two robotic arms. These researchers also investigated model predictive control (MPC) on this task [25]. Although adaptive control compensated uncertainties and MPC deal with unsolvable optimalities, the dynamics model and path planning routine cannot be saved. Our colleagues proposed a solution with fuzzy logic system which saves the complication of dynamics modelling and path planning [26]. However, the control strategy is only validated with point kinematics and point mass dynamics.

Due to the complicated dynamics in the cooperative object transportation task, researchers began to seek help from DRL. The most related research was introduced in [11] that two robots transporting an object through a narrow opening managed by a DQN style algorithm [27]. However, this research relied on a path planning algorithm and the function of the DQN algorithm was only to adjust the robots whenever pre-planned trajectory is not accessible. In [28], a multi-agent reinforcement learning algorithm was proposed to deal with hose transportation problem. The proposed algorithm was based on the original Q-learning, thus was not capable of continuous state inputs. A showcase of multiple robots carrying a long rod while preventing it to fall was demonstrated in [10]. This research was highlighted with controlling the MRS with taking continuous input data to resolve continuous output signals. Although the algorithm was more generalized, the goal and constraints were largely different from out study.

3 Methodology

3.1 System Configuration

The problem originates from a scenario that two persons carrying a big piece of furniture out of the room through a narrow door. We simplify and model the furniture transportation as two mobile robots linked by a solid rod moving out of a walled cell with an opening. We define the cell in a squared shape with dimensions of $10 \times 10$. A global coordinate system $\{X, Y, Z\}$ is fixed to the center of the room, where $X$ axis is pointing to the east, $Y$ axis is pointing to the north and $Z$ axis is determined by the right-hand rule. The opening is located on the south wall which is a doorway with width, $w = 2$ and depth, $d = 1$. The rod has length $l = w$. After attached to the robots with radius $r = 0.25$, the rod cannot be transported out when parallel to the room opening. We restrict the whole system with two linear degrees of freedom (DOF) and one rotational DOF all in the plane of $XY$. Three body reference frames: $\{(1)X, (1)Y, (1)Z\}$, $\{(2)X, (2)Y, (2)Z\}$ and $\{(s)X, (s)Y, (s)Z\}$ are attached to robot 1, robot 2 and the rod, respectively. The $(1)X$ and $(2)X$ are set to point toward the head of the robots, while $(s)X$ is set along the line between body frames of the robots pointing towards robot 1’s origin. $(1)Z$ axes are perpendicular to the $XY$ plane with same direction as $Z$ axis point to. Hence, $(1)Y$ axes can be determined by $(1)Z$ and $(1)Z$ axis and right hand rule. Coordinates of body frames origins under the global frame define positions of the robots ($\vec{r}_1$, $\vec{r}_2$) and the rod ($\vec{r}_3$). Our settings can be illustrated by Fig. 1.
3.2 Deep Reinforcement Learning Context

The rod transportation task can be viewed as a series of events. The task starts at time step \( t = 0 \) and ends at time step \( t = T \). Everything happens in between this period consists an episode. At any time step \( t \), we can describe a robot in the state of \( (i)_s \), where \( i \) is the index of the robot. Assume every robot in this system can perfectly sense poses and rotations of the rod and all the robots (include itself and teammates), we can define the state of robot \( i \) as:

\[
(i)_s = [\vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t, \vec{r}_t]
\]

where \( \vec{r} \)'s are velocity vectors of the robots and the rod. Each robot can take an action \( (i)_a \in \{\text{forward left, forward right, backward left, backward right}\} \) at any time step. Then, the system will transfer to the next step \( t + 1 \), and the state of robot \( i \) can be obtained as:

\[
(i)_{s_{t+1}} = [\vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}, \vec{r}_{t+1}]
\]

Together with the new state, the robot \( i \) receives a reward \( (i)r_{t+1} \) as defined in Eq. 1 which is the key to guarantee the robots can cooperate in this task. An individual will not receive any reward until both of them successfully escaped the room.

\[
(i)r((i)_s, (i)_a) = \begin{cases} 
1 & \text{if entire body of the rod is out of the room} \\
0 & \text{otherwise}
\end{cases}
\]  

However, rewards generated by such function will be too sparse that the learning process could be largely slow down due to this effect. In practice we extend Eq. 1 to the form as seen in Eq. 2 to guarantee non-zero reward can be received at any time step.

\[
(i)r((i)_s, (i)_a) = \begin{cases} 
400 & \text{if entire body of the rod is out of the room} \\
-100 & \text{if any robot hit wall} \\
-0.1 & \text{otherwise}
\end{cases}
\]  

The dense reward function in Eq. 2 is inspired by LunarLander environment from OpenAI Gym [29]. Instead of giving negative reward according to scale of control signals, we punish the robots from the perspective of time. Since we limit the horizon of an episode to be 1000 time steps, it is reasonable to give a large negative reward for the event of hitting the wall 1000 times larger than the routine time cost. In our case, if punishment for wall hitting larger than -100 (e.g. -1), the robots are possible to stuck at a local optimal which will lead them hitting the wall directly. Because they are more likely to receive smaller negative total reward for hitting the wall compare to struggling too long in the room but accumulate larger negative reward in the end.
In an episode, the total reward a robot receives by taking an action \( a_t \) at state \( s_t \) can be defined as
\[
R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k+1}(s_t, a_t)
\]
where \( \gamma \in [0, 1] \) is the discount rate which weighs future rewards less and less because of the nature of uncertainties. We can further describe the value of an action \( a_t \) taken at state \( s_t \) to be the expected total reward: \( Q(s_t, a_t) = \mathbb{E}[R_t] \).

We can call this value of state-action pair as Q value. Assume the behavior of a robot is determined by a control policy \( \pi(a|s) \), then the Q value of current step can be represented by the Q value of next time step according to the Bellman Expectation Equation as shown in Eq. 4 [8].
\[
Q^\pi(s_t, a_t) = r_{t+1} + \gamma \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1}|s_t, a_t) \sum_{a_{t+1} \in A} \pi(a_{t+1}|s_{t+1})Q^\pi(s_{t+1}, a_{t+1})
\]
where \( \mathbb{P} \) is the dynamics model (here we use a probability model) which governs the transition between two consecutive time steps. Typically, we need to model the dynamics, \( \mathbb{P} \), such that we can make plans for the robots then control it to stick to the plan. However, obtaining the dynamics model becomes more challenging as the task getting more complex. Reinforcement learning (RL) methods seek to solve the problem bypassing the dynamics model to only focus on optimizing the control policy through trial-and-error style interactions. The interactions has one objective that is maximizing the expected total reward at any given state, thus the dynamics model can be safely ignored. In the MRS cooperative object transportation problem, dynamics of the system is hard to be modeled. Therefore, we propose to use RL method to solve such a problem.

### 3.3 Multi-robot Deep Q-network

To success in this task, a robot has to employ an optimal control policy, \( \pi^* \) that maximizes the expected total reward at a given state, \( s_t \) by taking an optimal action, \( a_t \) governed by \( \pi^* \). Then Eq. 4 becomes Eq. 5 according to Bellman Optimality Equation.
\[
Q^\pi^*(s_t, a_t) = r_{t+1} + \gamma \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1}|s_t, a_t) \max_{a_{t+1}} Q^\pi^*(s_{t+1}, a_{t+1})
\]

Q-learning provides a straightforward way to iteratively optimize tabularized Q values without considering dynamics model, \( \mathbb{P} \) [27]. The Q values can be updated through Eq. 6 that
\[
Q(s_t, a_t) = Q(s_t, a_t) + \alpha (r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))
\]
where \( \alpha \) is the learning rate. Given the optimized Q values, a greedy policy \( a = \arg \max_a Q(s, a) \) usually works well to achieve the goal. Q-learning algorithm successfully gets rid of the restriction of dynamics model, but is suffered from tabularized states and actions. From previous section, our robot’s state space and action space has 18 and 4 dimensions, respectively. Discretize such state and action space may result in a giant table that requires forever to be converged. Hence, we can introduce a function approximator to take continuous states into account and serve the same role as the Q table in Q-learning algorithm. A popular type of function approximator is the neural networks (NN), and this is how Deep Q-network succeeded in the control tasks with image inputs [9]. We adopt NN with trainable weights, \( \theta \) to approximate Q function as the Q-net: \( Q(s, a; \theta) \). The Q-net needs to approximate the expected total reward as can be computed in Eq. 3. So, we need define a loss function which can tell the difference between the current Q values and the expected total rewards as Eq. 7 shows.
\[
\mathcal{L}(\theta) = \left[ r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta) - Q(s_t, a_t; \theta) \right]^2
\]
Differentiating Eq. 7 with respect to \( \theta \), we obtain the gradient \( \nabla_\theta \mathcal{L} \). Hence we can use stochastic gradient decent to update \( \theta \) and optimize the loss function.

The robots in this task are set to be homogeneous, thus only one DQN controller needs to be trained with integration of all the robots’ experience. The individuals in the MRS can actually accelerate the training by collecting more data in every time step. A little modification on the original DQN algorithm leads to our multi-robot DQN algorithm for homogeneous MRS as shown in Alg. 1. Although the number of robots in this research is limited to two, this algorithm can be scalable to more robots as needed. As opposed to the original DQN algorithm, we implement Double DQN (DDQN) trick to compute target Q values so that the algorithm can suppress the over-optimistic estimations toward the target Q values [30]. We also extend Alg. 1 to a heterogeneous version (Alg. 2) so that an individual in an MRS will be trained by self collected data only. As homogeneous MRS is just a special case of heterogeneous MRS, Alg. 2 can adapt to the homogeneous MRS without any problem.
Algorithm 1 Homogeneous MRS DQN

Require: Initialize replay memory $D$
Require: Initialize active Q-net with random weights: $\theta$
Require: Initialize target Q-net with identical weights as in the active Q-net: $\bar{\theta} = \theta$

for episode = 1 to $M$ do
    Initialize pose of the MRS randomly
    Perform $\epsilon$ decay
    for step = 1 to $T$ do
        for robot = 1 to $N$ do
            Select a random action $i_a$ with probability $\epsilon$;
            otherwise, $(i_a)_{at} = \arg\max Q((i)s_t, (i)a_t; \theta)$
        end for
        Execute all $(i)a_t$'s, then observe next states: $(i)s_{t+1}$, receive rewards: $(i)r_{t+1}$.
        Store all transitions $(i)s_t, (i)a_t, (i)s_{t+1}, (i)r_{t+1})$ into $D$
        Sample random batch of transitions $(s_j, a_j, s_{j+1}, r_{j+1})$ from $D$
        Set target $y_j = r_{j+1}$ if episode terminates at $j + 1$
        otherwise, $y_j = r_{j+1} + \gamma Q(s_{j+1}, \arg\max a Q(s_{j+1}, a; \bar{\theta}))$
        Perform gradient decent on $\theta$
    end for
end for

3.4 Cooperation Metrics

As researchers lack of tools to quantitatively assess the cooperation among the individuals in an MRS, we propose to introduce the mean absolute error (MAE) of Q-values between any two robots to be the standard. The nature of the Q-value is the expected total reward as being illustrated by Eq. 4. At time step $t$, the expected total reward of robot $i$ can be represented by $\max_a Q(i)s_t, (i)a_t)$. The absolute error of Q-values at same moment $t$ between robot $i$ and robot $j$, $e_t = |Q(i)t - Q(j)t|$, represents how different these two robots evaluate same moment. Using the MAE of Q-values, we introduce the novel metric $\Delta Q$ to assess cooperation between any couple of robots (Eq. 8).

$$ (ij)\Delta Q = \frac{1}{T} \sum_{t=0}^{T} |Q(i)t - Q(j)t| $$

3.5 Experiment Configurations

Design of the two-robot system is as Fig. 5 illustrated in Appendix B. For each individual robot, five rigid parts with basic geometries are included (chassis, left wheel, right wheel, caster wheel, hat). The rod are attached to the robots through their hats which can be freely rotated with respect to their chassis.

The multibody dynamics is taken care by Open Dynamics Engine (ODE) [31], which is integrated in Gazebo simulation software [32]. To enable interactions between the MRS and the simulated task environment, we first design robot models and environment model, then add a Python API for MRS to interact with the environment. Lastly, we can implement out RL algorithms through Tensorflow (an end-to-end open source platform for machine learning) [33]. The architecture of the software interface can be seen in Fig. 2. The details of system modeling, software API construction and training scripts are all open-sourced and can be found at https://github.com/IRASatUC/two_loggers.

The DQN in this research were all constructed with two fully connected hidden layers with 256 weights in each layer. Hyper-parameters we applied in the training were listed in Table 4 in Appendix C. The trainings were performed on a desktop computer with an AMD Threadripper 1900X CPU and an Nvidia GeForce GTX 1080Ti GPU. A GPU is not necessary for such task.
4 Results and Analyses

4.1 Training Performance Improvement

We employ averaged total reward as the metric to assess performance improvement during training. Assume the MRS receives total reward $R_m$ at episode $m$, then the averaged total reward at this episode can be expressed as

$$\bar{R}_m = \frac{1}{m} \sum_{i=1}^{m} R_i$$

The training procedures of using both homogeneous algorithm and heterogeneous counterparts were recorded and compared in Fig. 3. In addition, we trained a centralized controller using original DQN algorithm which can be served as the baseline which is also presented in Fig. 3. All these three types of training lasted 30000 episodes. We can notice that homogeneous DQN training performed slightly better than the heterogeneous one. The decentralized DQN trainings apparently outperforms centralized DQN training for they starting to success earlier but also achieved much higher averaged total reward at the 30000 episode milestone. Considering centralized DQN controller needs to handle 16 actions (compare 4 actions for decentralized setting), it is reasonable that the learning speed slowed down and the training process was less stable. The total amount of interactions(time steps) happened during homogeneous DQN training was about $5.1 \times 10^6$, the training of heterogeneous DQN took more than $5.8 \times 10^6$ interactions, while training of centralized DQN took more than $8 \times 10^6$ interactions to finish 30000 episodes. This result suggests that decentralized architecture is more superior than centralized counterpart. Employing homogeneous robots in an MRS is the most efficient configuration when training decentralized DQN controllers.

4.2 DQN Controller Performance Analysis

We evaluated the performance of the controllers by running 1000 trials with the MRS randomly initiated in the cell. The homogeneous training algorithm resulted in a MRS with 0.966 success rate, the heterogeneous training algorithm produced a MRS with 0.955 success rate, and centralized DQN gave out an answer of 0.941 success rate. We also tested performance of the DQN controllers against uncertainties. By increasingly adding Gaussian noise to the states (inputs) of the controllers, we found the success rate of both MRSs degraded gradually. Table 1 shows the performance drop of the MRS against the increased noise level. Considering the success rate of untrained controller (analogous to taking random actions all the time) barely reached 0.001, the DQN controllers demonstrated reasonable robustness against input uncertainties. Especially when noise level was controlled under $\mathcal{N}(0, 0.1)$, the performance of decentralized DQN were not affected at all.
The uncertainties can happen on the output end of the controllers, thus we also tested the controllers' tolerance on the randomness of control signals. The results can be seen in Table 2 that the performance degradation was slowly building up along the gradually increased output randomness. Even with 50% chance that a controller will take a random action, all the controllers were still having success rate over 0.8. When output randomness was under 10%, there was no performance drop observed in all three categories. When output randomness increased to 20%, decentralized DQN controllers can still maintain their performance, whereas the performance of centralized controller began to degrade.
4.3 Quantitative Cooperation Assessment

As a highlight of this research, we proposed a metric, $\Delta Q$ (described in 3.4), to assess the cooperation between two individuals in an MRS. Table 3 shows result from 1000 test trials with MRS randomly initiated in the cell. Under the assessment of $\Delta Q$, controllers trained by homogeneous DQN algorithm demonstrated best cooperation with minimum diverse when estimating total reward at the same moment. The two controllers were having larger disagreement if trained by heterogeneous DQN algorithm. During training, we saved DQN models every $10^6$ interactions, so that changed behaviors of the robots can always be tracked by loading previously saved models. We also compared metric $\Delta Q$ at different stage of training. Both algorithms exhibited larger $\Delta Q$s when trained with $10^6$ interactions data. Both algorithms showed a trend of decreasing $\Delta Q$s along the increasing training episodes, which indicates both algorithms can shape the value systems in-between the controllers to a more unified form.

![Image](https://via.placeholder.com/150)

| Model version | Homogeneous | Heterogeneous |
|---------------|-------------|---------------|
| $10^6$        | 11.72 ± 4.55 | 19.39 ± 9.70 |
| final         | 3.77 ± 2.20  | 7.83 ± 3.06  |

We sampled the MRS’s performance from 9 different initial conditions. Metrics of $time\ consumed$, $travel\ distance$ and $\Delta Q$ were extracted from the case-wise experiments which can be found in Table 5. We compared performance of controllers trained by homogeneous, heterogeneous and centralized DQN algorithms. While homogeneous DQN algorithm trained controllers having significantly small gaps when evaluating states at the same moment, they were slightly outperformed heterogeneous DQN and centralized DQN trained controllers in the manners of $time\ consumed$ and $distance\ traveled$. Trajectories of these samples were recorded as Fig. 6 illustrates. An obvious pattern can be observed is that the robot initially more close to the exit is more likely leading the MRS in the end. However, when the two robots were equally set against the exit, we observed that robot 1 led the MRS in most cases.

4.4 Other Findings and Discussions

We sampled the MRS’s performance from 9 different initial conditions. Metrics of $time\ consumed$, $travel\ distance$ and $\Delta Q$ were extracted from the case-wise experiments which can be found in Table 5. We compared performance of controllers trained by homogeneous, heterogeneous and centralized DQN algorithms. While homogeneous DQN algorithm trained controllers having significantly small gaps when evaluating states at the same moment, they were slightly outperformed heterogeneous DQN and centralized DQN trained controllers in the manners of $time\ consumed$ and $distance\ traveled$. Trajectories of these samples were recorded as Fig. 6 illustrates. An obvious pattern can be observed is that the robot initially more close to the exit is more likely leading the MRS in the end. However, when the two robots were equally set against the exit, we observed that robot 1 led the MRS in most cases.

After all, DQN algorithm is not something new. Researchers proposed many more advanced DRL algorithms since the birth of DQN. A major draw back of DQN is it cannot deal with continuous action space. Deep deterministic policy gradient (DDPG) is one of the successors of DQN, which introduced a policy network to produce continuous actions [34]. Proximal policy optimization (PPO) is one of the families of policy gradient algorithms, which can optimize continuous policy through guaranteed improvement without stepping too far to avoid collapsed performance [35]. We have tested both algorithms in the presented task, but neither of them induced positive result so far. DDPG’s convergence were extremely brittle during training, and both DDPG and PPO were having hard time to balance between exploration and exploitation.
(a) Homogeneous DQN algorithm trained case

(b) Heterogeneous DQN algorithm trained case

Figure 4: Sampled trajectories and cooperation assessment
5 Conclusion

In this research, we introduced decentralized DQN controllers in an MRS to solve an object transportation task cooperatively. The controllers learned how to behave and cooperate from scratch. Given state of the MRS, the DQN controllers can output discretized control signals directly without knowing dynamics nor planning path. Two training algorithms were proposed to adapt homogeneous and heterogeneous MRS, respectively. Both algorithms were able to train well-performed DQN controllers on homogeneous robots to solve the task with high success rate. The decentralized architecture was proved to be more efficient than the centralized counterpart. The DQN controllers were proved to be robust against small to medium level uncertainties. More importantly, a novel and universal metric was proposed in this research that can quantitatively assess cooperation between robots in an MRS. With the encouraging results, we are now more confident with the potential of deep reinforcement learning (DRL) type controllers in multi-robot systems (MRSs).

In the future, we would like to continue digging potentials of MRS with more advanced DRL algorithms, and settings. As an off-policy DRL algorithm, DQN is theoretically more data-efficient compare to the on-policy algorithms. The training process in this research still took more than $5 \times 10^6$ interactions or 10 days. We look forward to improve the efficiency of training with more tweaks, and this is essential to bring a DRL-type controller to a robot in the real world. The sensing data in current research is pulled out directly from the simulation software, which is not likely to be accessible in real applications. More than that, current sensing data cannot deal with constantly changing obstacles (e.g. the doorway is randomly placed). Hence, we are preparing to upgrade the current robots to be equipped with cameras and Lidars to adapt an upgraded task environment with more dynamic objects.

References

[1] NASA. Space Shuttle STS-49.
[2] Richard W Orloff. Space Shuttle Mission STS-49 IntelSat VI (F-3) Repair Mission. Technical report, NASA, 1992.
[3] Elio Tuci, Muhanad HM Alkilabi, and Otar Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. Frontiers in Robotics and AI, 5:59, 2018.
[4] Zool Hilmi Ismail and Nohaidda Sariff. A survey and analysis of cooperative multi-agent robot systems: challenges and directions. In Applications of Mobile Robots. IntechOpen, 2018.
[5] Kazuhiro Kosuge and Tomohiro Oosumi. Decentralized control of multiple robots handling an object. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96, volume 1, pages 318–323. IEEE, 1996.
[6] Zijian Wang and Mac Schwager. Kinematic multi-robot manipulation with no communication using force feedback. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 427–432. IEEE, 2016.
[7] Hamed Farivarnejad, Sean Wilson, and Spring Berman. Decentralized sliding mode control for autonomous collective transport by multi-robot systems. In 2016 IEEE 55th Conference on Decision and Control (CDC), pages 1826–1833. IEEE, 2016.
[8] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.
[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.
[10] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In International Conference on Autonomous Agents and Multiagent Systems, pages 66–83. Springer, 2017.
[11] Sergey V Manko, Sekou AK Diane, Aleksey E Krivoshatskiy, Ilan D Margolin, and Evgeniya A Slepymina. Adaptive control of a multi-robot system for transportation of large-sized objects based on reinforcement learning. In 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), pages 923–927. IEEE, 2018.
[12] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Agent modeling as auxiliary task for deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 15, pages 31–37, 2019.
[13] C Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. Adaptive behavior, 2(2):189–218, 1993.
[14] Zhi-Dong Wang, Eiji Nakano, and Takuji Matsukawa. Cooperating multiple behavior-based robots for object manipulation. In *Distributed Autonomous Robotic Systems*, pages 371–382. Springer, 1994.

[15] Sandip Sen, Mahendra Sekaran, John Hale, et al. Learning to coordinate without sharing information. In *AAAI*, volume 94, pages 426–431, 1994.

[16] Russell G Brown and James S Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 562–568. IEEE, 1995.

[17] B Hichri, Louonis Adouane, J-C Fauroux, Youcef Mezouar, and Ioan Doroftei. Cooperative mobile robot control architecture for lifting and transportation of any shape payload. In *Distributed Autonomous Robotic Systems*, pages 177–191. Springer, 2016.

[18] Toni Machado, Tiago Malheiro, Sérgio Monteiro, Wolfram Erlhagen, and Estela Bicho. Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3111–3117. IEEE, 2016.

[19] Charalampos P Bechlioulis and Kostas J Kyriakopoulos. Collaborative multi-robot transportation in obstacle-cluttered environments via implicit communication. *Frontiers in Robotics and AI*, 5:90, 2018.

[20] Chun-Hui Lin, Shyh-Hau Wang, and Cheng-Jian Lin. Interval type-2 neural fuzzy controller-based navigation of cooperative load-carrying mobile robots in unknown environments. *Sensors*, 18(12):4181, 2018.

[21] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, 36(9):1000–1021, 2017.

[22] Javier Alonso-Mora, Eduardo Montijano, Tobias Nägeli, Otmar Hilliges, Mac Schwager, and Daniela Rus. Distributed multi-robot formation control in dynamic environments. *Autonomous Robots*, 43(5):1079–1100, 2019.

[23] A-N Ponce-Hinestroza, J-A Castro-Castro, H-I Guerrero-Reyes, Vicente Parra-Vega, and E Olguín-Díaz. Cooperative redundant omnidirectional mobile manipulators: Model-free decentralized integral sliding modes and passive velocity fields. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2375–2380. IEEE, 2016.

[24] Christos K Verginis, Matteo Mastellaro, and Dimos V Dimarogonas. Robust cooperative manipulation without force/torque measurements: Control design and experiments. *IEEE Transactions on Control Systems Technology*, 28(3):713–729, 2019.

[25] Alexandres Nikou, Christos Verginis, Shahab Heshmati-Alamdari, and Dimos V Dimarogonas. A nonlinear model predictive control scheme for cooperative manipulation with singularity and collision avoidance. In 2017 25th Mediterranean Conference on Control and Automation (MED), pages 707–712. IEEE, 2017.

[26] Yufeng Sun, Andrew Barth, and Ou Ma. An intelligent approach for a two-robot team to perform a cooperative task. In *AIAA SciTech 2020 Forum*, page 1116, 2020.

[27] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[28] Borja Fernandez-Gauna, Ismael Etxeberria-Agiriano, and Manuel Graña. Learning multirobot hose transportation and deployment by distributed round-robin q-learning. *PloS One*, 10(7):e0127129, 2015.

[29] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[30] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtyfirst AAAI conference on artificial intelligence*, 2016.

[31] Evan Drumwright, John Hsu, Nathan Koenig, and Dylan Shell. Extending open dynamics engine for robotics simulation. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 38–50. Springer, 2010.

[32] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[33] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *23rd {USENIX} symposium on operating systems design and implementation ({{OSDI} } 16)*, pages 265–283, 2016.

[34] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
A  Heterogeneous MRS Training Algorithm

Unlike the homogeneous counterpart, the training of heterogeneous MRS happens separately on each individual. Instead of sharing the data collected by all the members in the team, each robotic controller will be trained with data collected by itself.

Algorithm 2  Heterogeneous MRS DQN training

\[
\text{for } robot = 1 \text{ to } N \text{ do}
\]

\[
\text{Initialize replay memory } (i)D
\]

\[
\text{Initialize active Q-net with random weights: } (i)\theta
\]

\[
\text{Initialize target Q-net with identical weights as in the active Q-net: } \bar{(i)}\theta = (i)\theta
\]

\[
\text{end for}
\]

\[
\text{for } episode = 1 \text{ to } M \text{ do}
\]

\[
\text{Initialize pose of the MRS randomly}
\]

\[
\text{Perform } \epsilon \text{ decay}
\]

\[
\text{for } step = 1 \text{ to } T \text{ do}
\]

\[
\text{for } robot = 1 \text{ to } N \text{ do}
\]

\[
\text{Select a random action } a_t \text{ with probability } \epsilon; \text{ otherwise, } (i)a_t = \arg \max Q((i)s_t, (i)a_t; \theta)
\]

\[
\text{end for}
\]

\[
\text{Execute all } (i)a_t\text{'s, then observe next states: } (i)s_{t+1}, \text{ receive rewards: } (i)r_{t+1}.
\]

\[
\text{for } robot = 1 \text{ to } N \text{ do}
\]

\[
\text{Store transition } ((i)s_t, (i)a_t, (i)s_{t+1}, (i)r_{t+1}) \text{ into } (i)D
\]

\[
\text{Sample random batch of transitions } ((i)s_j, (i)a_j, (i)s_{j+1}, (i)r_{j+1}) \text{ from } (i)D
\]

\[
\text{Set target } ((i)y_j = (i)r_{j+1} \text{ if episode terminates at } j + 1
\]

\[
\text{otherwise, } ((i)y_j = (i)r_{j+1} + \gamma Q((i)s_{j+1}, \arg \max_a Q((i)s_{j+1}, (i)a; \theta); \bar{(i)}\theta)
\]

\[
\text{Perform gradient descent on } (i)\theta
\]

\[
\text{Every } C \text{ steps, reset } (i)\bar{\theta} = (i)\theta
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

B  Robot Dimensions

C  DQN Hyper-parameters

Hyper-parameters used in DQN training are as follows:

| Hyper-parameter      | Value     | Description                                           |
|----------------------|-----------|------------------------------------------------------|
| batch size           | 8192      | Number of experiences sampled for one step of gradient descent optimization. |
| replay buffer size   | 10000000  | Maximum experiences can be stored in replay buffer. |
| update frequency     | 8000      | Target network will be updated to comply to the active Q-network at this frequency. |
| discount rate        | 0.99      | Used on future rewards to calculate the expected total reward. |
| learning rate        | 0.0001    | Used to control update step of the trainable weights in Q-networks. |
| initial exploration  | 1         | Initial value of action randomness.                   |
| final exploration    | 0.1       | action randomness after $\epsilon$ stopped decaying. |
| $\epsilon$ decay period | 2000  | Number of episodes needed for $\epsilon$ decaying from initial exploration to the final exploration |
| warm-up episodes     | 500       | Number of episodes run with completely random actions before Q-networks were optimized. |
D Case Study

We sampled 9 cases of transportations with specified initial conditions. The trajectories of the transportations using different controllers can be seen in Fig. 6 (from left to right: homogeneous, heterogeneous, centralized). Key metrics including total travel distance, time consumed and cooperation metrics: ΔQ were recorded. Since heterogeneous and centralized DQN trained controllers failed in the last case, the assessment on these two were not applicable.

| Case Index | Time Consumed   | Distance Traveled | ΔQ  |
|------------|-----------------|-------------------|-----|
|            | homo. | hete. | cent. | homo. | hete. | cent. | homo. | hete. |
| 1          | 253   | 269   | 381   | 25.62 | 25.71 | 32.13 | 3.33  | 8.85  |
| 2          | 233   | 285   | 230   | 24.42 | 27.28 | 24.97 | 2.54  | 6.82  |
| 3          | 265   | 252   | 252   | 25.38 | 25.62 | 26.35 | 2.54  | 6.82  |
| 4          | 174   | 233   | 196   | 18.93 | 22.61 | 20.38 | 3.59  | 6.41  |
| 5          | 178   | 186   | 177   | 18.92 | 18.91 | 19.12 | 2.32  | 9.04  |
| 6          | 161   | 155   | 151   | 15.24 | 15.16 | 15.22 | 3.34  | 8.68  |
| 7          | 147   | 141   | 147   | 14.19 | 12.94 | 14.89 | 5.96  | 21.90 |
| 8          | 147   | 199   | 237   | 15.30 | 15.76 | 18.35 | 3.08  | 9.79  |
| 9          | 180   | N/A   | N/A   | 10.88 | N/A   | N/A   | 10.11 | 15.85 |
Figure 6: Cooperative transportation cases (left: homogeneous; middle: heterogeneous; right: centralized)