Crucial Factors Affecting Decentralized Multirobot Learning in an Object Manipulation Task

Poj Tangamchit [1] John M. Dolan [2] Pradeep K. Khosla [1,2]
E-mail: poj@andrew.cmu.edu jmd@cs.cmu.edu pkk@cs.cmu.edu
Dept. of Electrical and Computer Engineering [1], The Robotics Institute [2]
Carnegie Mellon University, 5000 Forbes Ave. Pittsburgh, PA 15213, USA

Abstract

Decentralized multirobot learning refers to the use of multiple learning entities to achieve the optimal solution for the overall robot system. We demonstrate that single-robot learning theory can be successfully used with multirobot systems, but with certain conditions. The success and the effectiveness of this method are potentially affected by various factors that we classify into two groups: the nature of the robots and the nature of the learning entities. Incorrect setup of these factors may lead to undesirable results. In this paper, we methodically test the effect of varying four common factors (reward scope, learning algorithms, diversity of robots, and number of robots) in a decentralized multirobot system, first in simulation and then on real robots. The results show that two of these factors, reward scope and learning algorithm, if set up incorrectly, can prevent optimal, cooperative solutions.

1. Introduction

Reinforcement learning has been successfully used in a single robot in order to make it learn traveling through mazes and other tasks. Popular learning algorithms, such as Q-learning and TD(λ), have proven to be effective methods for making a robot adapt itself to the environment.

The advance of technology makes computer parts cheaper and more effective. This makes the use of several robots more affordable. Using multiple robots to accomplish a task is quite common. Multirobot systems potentially have the following five major advantages over a single robot: task capability, improved system performance, distributed sensing, distributed action, and high robustness. In [3], we proposed the dynamic task selection mechanism, which is a multirobot group architecture that promotes robustness. However, it still lacks an efficient task allocation mechanism. This is the reason why we consider using reinforcement learning in our multirobot system. Reinforcement learning in multirobot systems is a new research area. It is somewhat different from distributed artificial intelligence (DAI) and multi-agent systems (MAS) due to the embodiment of real robots and the unsynchronized nature of distributed learning entities. We have found few publications directly implementing multirobot learning.

Some researchers have successfully used single-robot learning theory (methods such as Q-learning) in multirobot systems. Most of this work modified the learning in some way, such as special rewards, heuristics or subgoals. However, we believe that single-robot theory/methods can also be used in multirobot systems, but with certain adjustments. In this paper, we test several factors that potentially have an effect on the final outcome of the learning results. We call these factors "environmental factors". This paper continues the work in [4], which tested the effect of different factors on a surveillance problem. This problem, called the multirobot patrolling problem, requires cooperation from all robots to guard an area. Performance in the multirobot patrolling problem depends on the placement of robots and paths they take. This paper is the test of the same factors but with a different type of task. This paper uses the puck-collecting task as a testbed problem, which we consider as a prototype for the object manipulation tasks. We first did the experiments in simulation and then verified some cases with real robots.

The rest of this paper is organized as follows. Section 2 describes previous work. Section 3 gives our approach by first explaining the taxonomy of multirobot learning and the learning algorithm that we used. It then describes the puck-collecting problem and the four factors that we investigated. Section 4 gives experimental results from simulation and on real robots. Section 5 presents a discussion of how the factors under consideration affect learning, and section 6 gives conclusions.

2. Previous Work

Unlike learning in single-robot systems, reinforcement learning in decentralized multirobot systems is a new research area and has not been systematically studied. Nevertheless, some aspects and theory of single-robot learning can be applied to multirobot systems. For example, some reinforcement learning algorithms [9] are still suitable for use in multirobot systems, as is Sutton's Dyna architecture [12]. Dyna reuses training data and creates a hypothetical world in order to use the training data to the fullest extent. There are examples of single-robot learning methods being applied to multirobot systems with some modifications. An example is Mataric's work [1]. With the use of progress estimators, she successfully implemented a single-robot learning method on each robot and achieved a good result as a team.

Despite the partial applicability of single-robot learning, multirobot systems have unique features that introduce additional considerations. Our previous work [4] is an attempt to test and find the factors that have effects on the usage of single-robot methods in a multirobot system. We used a surveillance problem as a testbed. In this paper, we followed along the same line as our previous work but instead used an object manipulation problem as a testbed. The idea is to test the same factors on different types of tasks in order to examine the effect of these factors in general. For example, one of the factors is different learning value functions for learning algorithms. We discovered that discounted-reward-based Q-learning [5], although effective for single robots, cannot produce cooperation, for which an average-rewards-based scheme such as Monte Carlo learning [12] should be used. Another factor is the diversity of multirobot teams, which was first investigated by Balch [6]. He showed that diversity can have an impact on the performance of robot teams in some types of tasks. Inspired by his work, we included diversity among the factors whose impact on learning performance was tested, using a hybrid, rather than Balch's purely reactive architecture.

3. Approach

We begin our approach by discussing the details of multirobot architecture and learning algorithms. We then systematically study the effect of environmental factors on the learning. First, we make a taxonomy of these factors based on the earlier work of Balch [6] and Dudek et al. [7]. Next, we talk about the puck-collecting problem, which is the problem testbed for our experiments. Then, each of the four environmental factors are discussed in detail. We then present the experiments and results from both the simulations and the real robots.

3.1. Robot Architecture and Group Architecture

When building a multirobot system, one must make two architecture decisions: one for the individual robots, and one for the group. The individual robot architecture choice spans the continuum from the reactive to the deliberative. A deliberative architecture plans actions in detail based on a world model. A reactive architecture is a simple mapping of sensor inputs to actions. It does not keep a world model or plans. A hybrid architecture is a mix of the reactive and deliberative architectures. There are two types of group architectures: centralized and decentralized. A centralized architecture employs a central unit to control the operation of the whole system. The central unit can be a separate entity or can be one of the robots in the group. The duty of the central unit is to get data from all robots, plan actions of the whole group, and send commands back to the robots. Therefore, all controls depend on the central unit. A decentralized architecture lacks such a central unit. Each robot has to individually plan and control itself. Our architecture, the dynamic task selection [3], has a hybrid robot architecture and a decentralized group architecture.

3.2. Learning Algorithms

We use distributed learning entities that run asynchronously on each robot. There are two learning algorithms investigated in this paper: Q-learning, which has a discounted-reward value function, and the Monte Carlo algorithm (MC), which has an average-reward value function. Q learning is designed to optimize a robot policy (π) that is based on cumulative discounted rewards (V^{π}). The cumulative discounted reward is the sum of rewards that a robot expects to receive after entering into a particular state. The discount factor (γ) makes rewards that are received in the future fade over time.

$$V^{\pi}(t) = r_t + \gamma\ r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

where $0 < \gamma < 1$

Q learning defines an evaluation function *Q(s,a)*. This function is the maximum cumulative discounted reward that can be achieved by starting from state *s* and applying action *a* as the first action. Using Q learning, robots learn and update the Q value by the following equation:

$$Q(s,a) \longleftarrow r(s,a) + \gamma \max_{a'} Q(s',a')$$

where s' and a' are the next state and the next possible action.

The second learning algorithm tested was the Monte Carlo algorithm (MC). It uses probability theory to estimate the value of actions from experience. Monte Carlo learning is used in episodic tasks. The algorithm traces the states that have been visited until the end of an episode. It then gives credits to those states according to rewards that the robots receive. There are two versions of Monte Carlo learning: first-visit MC and every-visit MC. First-visit MC records average rewards after the first visit to each state. Every-visit averages all rewards after every visit to each state. The first-visit MC algorithm looks like the following.

Q(s,a) ← arbitrary % Q(s,a) is an average reward after the first visit in state s, action a
π(s) ← arbitrary % π (s) is the policy and decision at state s
Rewards(s,a) ← Empty list

Repeat Forever:
- Generate an episode using π
- For each pair s,a appearing in the episode:
  R ← reward following the first occurrence of s,a
  Append R to Rewards(s,a)
  Q(s,a) ← average(Rewards(s,a))
- For each s in the episode
  π(s) ← argmax_a Q(s,a)

We also use a modified ε-greedy method [12] to balance between learning an exploration for both algorithms. With the ε-greedy method, a robot will try to make random explorations with probability ε. For the modified version, the value of ε will decrease over time according to the learning progress and the amount of rewards that the robot gets.

3.3. Taxonomy of the environmental factors

Environmental factors are various characteristics that have to be chosen when researchers implement learning in multirobot systems. Due to the absence of guidelines for systematically specifying these factors, researchers currently require trial-and-error in choosing them until the desired results are achieved. A multirobot learning system has three main components: robots, learning algorithms and tasks. Our taxonomy is constructed based on the first two components. The nature and configuration of the robots entails the overall structure of the robot team. The nature of the learning entities involves the structure of the learning algorithms and rewards. We do not make a taxonomy of tasks because tasks are user-specific. Tasks can vary indefinitely with different aspects according to users' requirements. We instead plan to test the environmental factors on different types of multirobot tasks that we consider to be prototypes for general multirobot applications. In this paper, we conduct experiments with the puck-collecting problem, which we consider as a prototype for the more general object manipulation task.

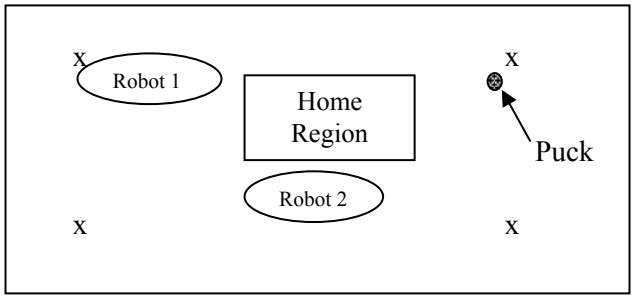
The nature of the robots involves robot architecture (reactive / deliberative / hybrid), group architecture (centralized / decentralized), number of robots (small-size / large-size), and diversity in capabilities (heterogeneous / homogeneous).

The nature of the learning involves learning entities (centralized / decentralized), learning algorithms (Q-learning / Monte Carlo), and reward scope (local / global).

This paper presents multirobot learning with fixed parameters as follows: hybrid robot architecture, decentralized group architecture, and decentralized learning entities. This paper tests the effect of varying the following parameters: number of robots, diversity in capabilities, learning algorithms, and reward scope.

3.4. The Puck-Collecting Problem

Our test problem is the puck-collecting problem, consisting of two robots and a rectangular field. Pucks are distributed randomly at four predefined points at the corners of the field. The robots have short-range puck detection so that they have to move close enough to a puck in order to see it. The robots' task is to investigate and find a puck around these points. There is a home region in the middle of the field with a bin inside. The robots have to move all pucks to the home region and deposit them in the bin. Both robots can sense a puck, pick up a puck, or drop a puck. The first robot (Robot1) can move to and investigate around the points, or it can move to the home region and deposit a puck. The second robot (Robot2) is restricted to move anywhere but staying at the home region. However, it can still sense a puck, pick up a puck, or deposit a puck in the bin. Depositing a puck in the bin is time-consuming for the first robot, but it is easy for the second robot. Therefore, although the second robot cannot move around, it can play an important role by depositing pucks in the bin. The optimal complete sequence is that the first robot picks up a puck, comes back to the home region, and drops the puck. Then, the second robot picks up the puck, and deposits it in the bin.



State =
{At?, HavePuck, SensePuck}

Action =
{Goto?, PickPuck, DropPuck, Dump,
DoNothing}

Figure 1 The Puck-Collecting Problem

The optimal solution sequence is shown below. First, both robots start at the home region. Second, Robot1 moves to the point on the upper right hand corner. Third, Robot1 picks up a puck. Fourth, Robot1 comes back to the home region. Fifth, Robot1 drops the puck to the floor. Sixth, Robot2 picks up the puck. Seventh, Robot2 dumps the puck into the home region.

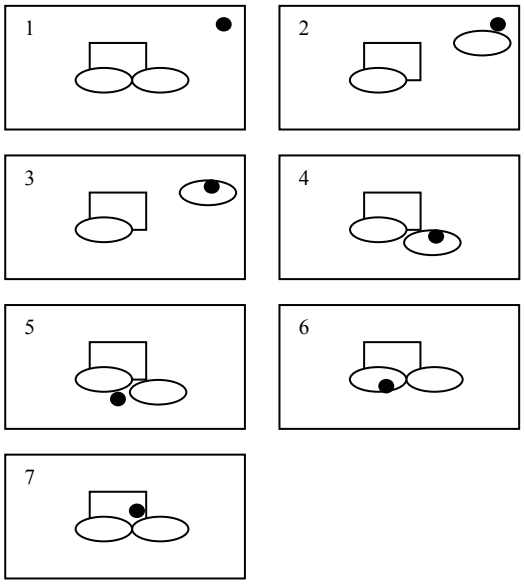


Figure 2 The Optimal Solution of the Problem

Action	Reward
Move	(Time used in second * -100) – 10
PickPuck	(Time used in second * -100) – 10
DropPuck	(Time used in second * -100) – 10
Dump	10000 for Robot 1 30000 for Robot 2 -10 if unsuccessful
Wait	-10

Parameter values of rewards and costs are shown in the table above. All robot actions result in negative rewards (cost) except depositing a puck, which gives a big positive reward because it is the final goal. The cost for picking, dropping, and dumping a puck are proportional to the actual time used for each action plus an overhead of 10 units.

3.5. Details of Tested Environmental Factors

In this paper, we investigate two factors from the nature of the learning entities (reward scope and learning algorithms) and two factors from the nature and configuration of the robots (number and diversity). The factors in the nature and configuration of the robots that we fix are the hybrid robot architecture and the decentralized group architecture. The factors in the nature of the learning entities that we fix are the decentralized learning entities. Each factor varied is detailed below.

3.5.1. Reward Scope

Rewards are an important component of reinforcement learning. A reward is given to a robot when it does something good, e.g., reaching the goal. We classified rewards in a similar way to data of robots. Robots carry two types of data: local data, which are the data used and kept private within each robot, and global data, which are the data shared among robots by the synchronization mechanism. Based on the same principle, there are two reward scopes in multirobot learning: local and global. A local reward scheme keeps rewards within each robot individually, whereas a global reward scheme broadcasts rewards generated within each robot to all other teammates. Therefore, with a global reward scheme, robots receive rewards and punishment together, as a team. A local reward scheme is a straightforward method that is used in single-robot learning. The learning occurs independently in each robot without information exchange. A global reward scheme, on the other hand, needs to share rewards among robots. It can be implemented by broadcasting all rewards generated within the robot to the teammates. The transmitted rewards are then added up to the internal reward for the current action.

3.5.2. Learning Algorithms

Two learning algorithms were tested: Q-learning, which is based on cumulative discounted reward, and Monte Carlo learning, which is based on average reward.

3.5.3. Diversity of Robots' Capabilities

In a robot team, robots can have the same or different capabilities. This property is referred to as the diversity of a robot team. Teams consisting of robots with the same capabilities are termed homogeneous. Teams consisting of robots with different capabilities are termed heterogeneous. The impact of diversity on robot teams was first investigated by Balch [6]. Using a reactive robot architecture, he showed that diversity is beneficial in some types of tasks, but unsuited to others. This inspired our motivation to investigate diversity on our multirobot system, which has a hybrid robot architecture. In the puck-collecting problem, the original setup has one robot with an ability to move around and one robot disabled to stay at the home region. This is a heterogeneous team. The purpose of this setup is to test whether the learning algorithm can learn the best cooperative strategy among the robots. For a diversity test, we created a homogeneous team by using two robots with the same capability and reward setup. Both can move to all points and can dump a puck with the same reward value.

3.5.4. Number of Robots

We classify this factor into two types: small-size (2-3 robots) and large-size (>20 robots). Although we do not test the large-size case, we vary the number of robots within the small-size range in order to test the scalability of the learning algorithms. We tested this factor by adding another robot that also has the capability to move around the points. In total, there are three robots: two are identical and one is disabled to stay at the home region.

4. Experiments and Results

4.1. Robot Physical

We used Pioneer robots models P2DX and P2AT from Activmedia Corp. (see Figure 3). Each robot is equipped with an onboard PC104 running Linux (for computing and learning), sonar sensors (for reactive obstacles avoidance), and a wireless LAN 801.11b card (for communication). We used an overhead camera with a Cognachrome vision board to detect robot position (act as a GPS). Each robot was marked with two color blobs to determine position and orientation. Figure 3 shows one robot marked with two red blobs and the other robot marked with two blue blobs. The two blobs have different sizes. The blobs on the front of the robots are smaller than the blobs on the back. We used the blob area (pixel

count from the camera) to differentiate between the front and the back blobs. Then, we used the position of the two blobs to calculate the position and orientation of each robot. Each robot is programmed with fixed low-level reactive behavior, such as avoiding obstacles, avoiding other robots, moving to a point and manipulating a puck.

Figure 3 Two Pioneer Robots on the field

We used off-the-shelf components to build an easy passive gripper attached to the front of each robot. The grippers were made of copper rods bending in a fork-like shape. The puck was made of two compact discs connected together with spacers (see Figure 4). The puck was marked with an orange color for camera detection.

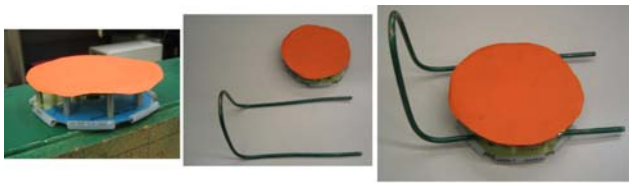


Figure 4 The Puck and The Passive Gripper

With this gripper, the robot can pick up a puck by moving directly toward it. When the puck is in place, the robot can move forward and turn without losing the puck. When the robot wants to drop the puck, it simply moves backward.

4.2. Simulation and Results

Our simulation was written in Visual C++. The robots had predefined low-level behaviors: move to a point, pick up a puck, drop a puck, dump, and wait. The robots were assumed to be equipped with GPS, sonar sensors, a passive gripper and a communication channel. Learning entities were implemented on each robot independently. Each robot and the environment ran on separate threads in order to simulate the asynchronous timing of the real world. Also, there is some chance that the robots will miss the puck when they try to pick it up.

Due to the number of varied factors, we made the tests more systematic by first finding a standard case which provides the optimal result (robots cooperate). Then, we varied the factors one by one and made comparison to the standard case. The standard case had 2 robots, Monte Carlo learning, a heterogeneous team, and a global reward scheme. After randomly varying factors and running the experiments, we found two types of final results: the optimal case where both robots cooperate, and the greedy case where Robot1 does all actions by itself. In the optimal case, Robot1 passes the puck to Robot2, who is good at dumping. In the greedy case, Robot1 does not pass the puck but instead dumps the puck by itself. Moreover, we found that the effect of factors that create the greedy case are dominant regardless of the setting of other factors. For example, the use of a local reward scheme always makes Robot1 greedy no matter what other factors are. This suggests that the effects of the four factors are independent of one another. Because the learning performs random exploration, we ran the experiment multiple (10) times for each case to ensure the consistency of the results. Each run ended when the learning reached a stable state defined by the point when the ε value of the modified ε-greedy method reduces to zero (meaning no more exploration). The results presented below are based on changes in each factor compared to the standard case.

- Single robot case

The purpose of the single robot case is to test the integrity of the simulation and the learning algorithm. We use only one robot to patrol the four checkpoints. If a puck is found, the robot should learn to carry it back the home region and dump it.

The results show that both Q-learning and Monte Carlo learning can achieve the optimal result where the robot goes straight to the puck, picks it up, comes back to the home region, and dumps it. Q-learning has slightly better speed than Monte Carlo learning. The plot of cumulative rewards on each epoch using the Monte Carlo learning is shown in Figure 5.

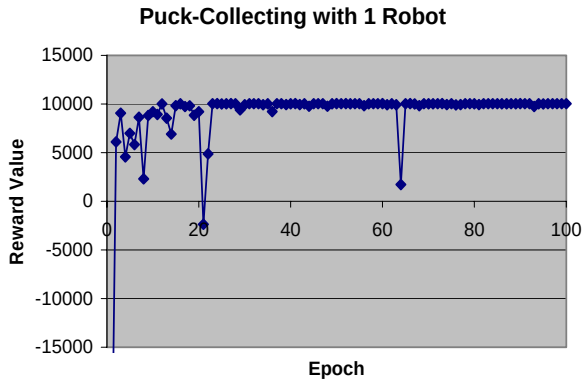


Figure 5 Puck-Collecting 1 Robot (Simulation)

- Standard case (Cooperation achieved)

The standard case designates the case when the robots achieve the optimal solution. This is when Robot1 picks up a puck and hands over to Robot2, which is better at dumping a puck. The parameters setup is determined by trial and error until the optimal result is achieved. The standard case has 2 robots, 4 checkpoints, heterogeneous team, Monte Carlo learning (average-reward), and a global reward scheme. The plot of rewards is shown in Figure 6.

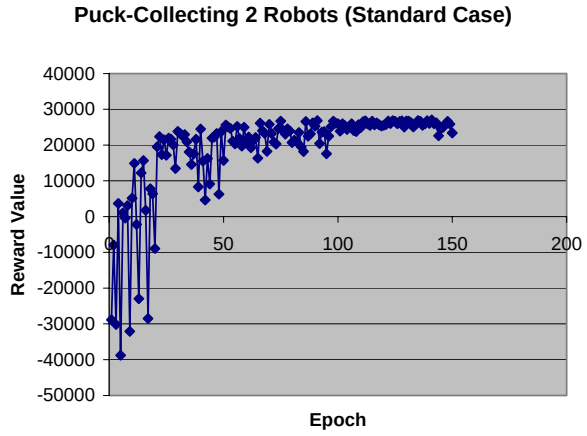


Figure 6 Puck-Collecting (standard case, simulation)

- Varying learning algorithm value function

We compare Q-learning and Monte Carlo learning in this experiment. In the reference case, we found that using Monte Carlo learning with the right setup can achieve the optimal solution. However, when using Q-learning with the same setup, we discovered that the robots cannot achieve the optimal solution. The learning algorithm can achieve a stable result but not the optimal one. The final results show Robot1 doing the tasks all by itself (pick up and dump) without passing the puck to Robot2. The results suggest that Q-learning creates a greedy strategy on the robots. It makes each robot want to be the one who get the big reward. Therefore, the level of the total reward as a team will be less than that of the standard case.

- Varying reward scope

When the reward scope was changed to local, the robots fail to achieve cooperation. Similar to the local reward case, the results show that the first robot will complete the mission all by itself without passing the puck to the other robot. The total reward within each robot will therefore be lower than the cooperative case. The result suggests that using a local scheme can prevent the robots from cooperating with each other. The plot of total reward in the first robot and the second robot are shown in Figure 7.

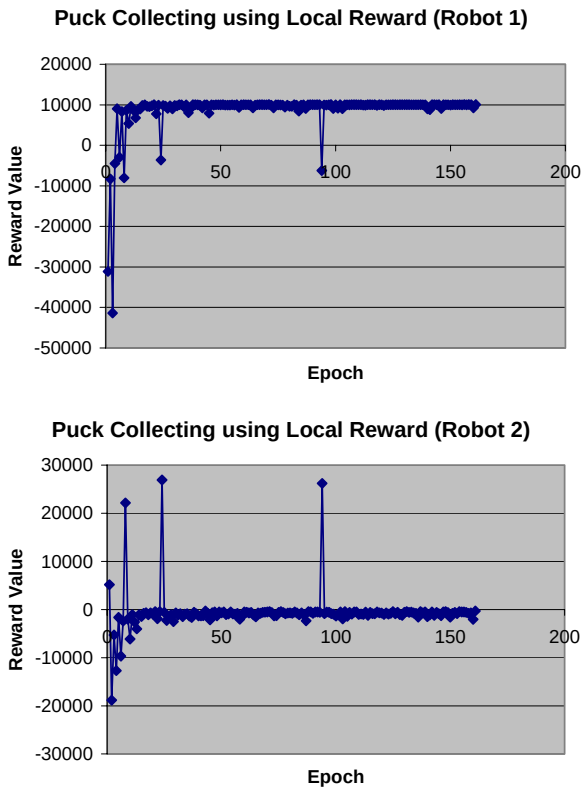


Figure 7 Puck-Collecting (local reward, simulation)

- Vary diversity in capabilities of robots

The reference case has two heterogeneous robots because we want to test the cooperative behavior that arises from learning. The purpose of having two robots with different capability is to see whether they can compensate each other's weaknesses. In this experiment, we instead use two identical robots. Both of them can go to all checkpoints, pick up a puck, drop a puck and dump a puck with the same costs. The results show that the learning can achieve the optimal solution where one of the robots does everything all by itself (there is no advantage to hand over the puck because the "dump" action costs the same). However, we found a small difference in the final results when using different reward schemes. For the global reward scheme, the final result shows one robot doing the pick-up – dump routine and the other robot staying out of the way to minimize collision. For the local reward scheme, both robots will compete to be the one who pick up the puck. The reason behind this behavior is that the global reward scheme distributes all rewards among robots. That means one robot can also get the big reward by letting its teammate do the job. In the local reward scheme, whoever gets the puck will get the big reward. Therefore, both of them have to compete for the puck.

- Varying number of robots

In this experiment, we added another robot that has full capability of reaching checkpoints. In total, there are two robots that can reach all checkpoints and one robot that only stays around the home region but is good at dumping a puck. The result indicates that all three robots can reach the optimal solution where one robot does nothing and one robot goes picking up a puck to hand over to the disabled robot. The learning time also increases because there are more robots and there are more states to be explored. The plot of reward is similar to that of the standard case.

The table below shows the summary of the results.

Factors		Median learn time (epoch)	Min	Max
Learning Algorithm	Monte Carlo	66	59	90
	Q-learning	50	44	73
Reward Scope	Global	66	59	90
	Local	112	64	127
Diversity of robots' capability	Heterogeneous	66	59	90
	Homogeneous	92	85	133
Number of robots	2 robots	66	59	90
	3 robots	110	94	173

4.3. Real Robot Experiments and Results

After simulations, we verified some important cases with real robots. All parameters and rewards are the same with simulation except the decrease rate of randomness in the ε-greedy method. We set the decrease rate faster in order to minimize the learning time. Other than that, we followed along the same line as in simulation: starting with 1 robot case, finding a standard case, and experimenting with the factors.

- One robot case

We started with one robot case, which we can easily observe the operation of the robot and calibrate the position of the passive gripper. The plot of reward log is shown below.

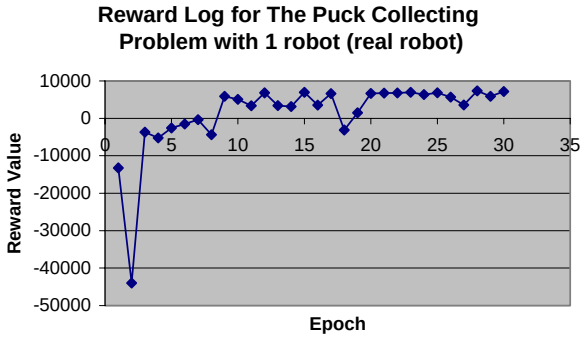


Figure 8 Puck-Collecting 1 Robot (real robot)

The learning converges quite fast (around epoch 20th).

- Standard case (Cooperation achieved)

We try similar parameter setting obtained from the simulation to be the standard case. The standard case has 2 robots, 2 checkpoints, a homogeneous team, Monte Carlo learning (average-reward), and a global reward scheme. The real

robots are able to learn cooperation using this parameter setup. The plot of rewards is shown below.

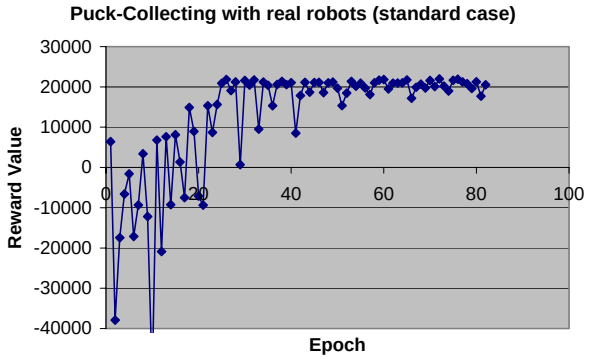


Figure 9 Puck-Collecting Standard Case (real robots)

- Varying reward scope

By changing the reward scope from global to local, the robots fail to achieve cooperation. Similar to simulations, the result shows that Robot1 will do everything by itself without passing the puck to Robot2.

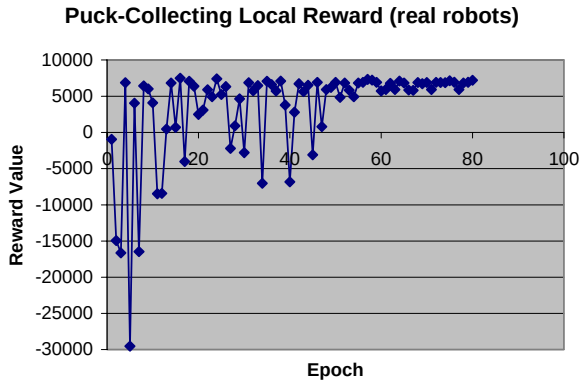


Figure 10 Puck-Collecting (local reward, real robots)

- Varying the learning algorithm

We tested Q-learning with the real robots in the puck-collecting problem. The results show that the robots fail to achieve cooperation. The robot that can move around will be the one who does all tasks by itself.

5. Discussion

The results indicate that the setup of learning algorithms and the reward scope can affect the final results of learning. In this section, we analyze the effect of each of the four factors tested.

The results indicate that Q-learning, which is based on discounted reward, fails to achieve cooperation while Monte Carlo learning, which is based on average reward succeeds. This phenomenon is best described by an example. Consider the example of two robots with a sequential task. The task consists of two parts in strict order (similar to the puck-collecting problem). Only after the first part is finished can the second part begin. Rewards are given to the robots at the end of the second part. Both robots use a global reward scheme.

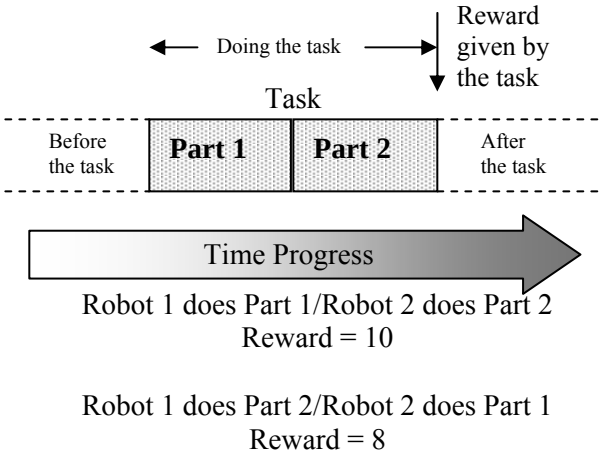


Figure 11 Time Frame in Continuous Subtasks

We assume that Robot 1 is more suited to do Part 1 than Robot 2. In the best case, Robot 1 chooses Part 1 and Robot 2 chooses Part 2, which will provide a reward of 10 units. The other case is when Robot 1 chooses Part 2 and Robot 2 chooses Part 1, which will provide a reward of 8 units. Suppose the length of Part 2 is three time-steps and the discount factor (γ) is 0.9. In the first case, Robot 1 chooses Part 1 and gets a reward three time-steps later of $10*(0.9)^3 = 7.3$. Robot 2 chooses Part 2 and gets a full 10-unit reward immediately. In the second case, Robot 2 chooses Part 1 and gets a reward three time-steps later of $8*(0.9)^3 = 5.8$. Robot 1 chooses

Part 2 and gets an immediate reward of 8 units. The total reward of the first case is 7.3+10 = 17.3 and the total reward of the second case is 5.8+8 = 13.8. The second case seems inferior, but Robot 1 gets a bigger reward (8 instead of 7.3). Therefore, using the cumulative discount reward framework, Robot 1 will learn the second case, which is a selfish behavior.

Learning algorithms that are based on an average reward framework, such as the Monte Carlo algorithm, can solve this problem. With average rewards, it does not matter who gets the reward first, since the reward will not be discounted. The reward that each robot receives is the sum of all rewards divided by the number of time-steps. Therefore, all robots receive equal rewards. From the previous example, if the total number of time-steps is five, all robots receive an average reward of 10/5 = 2.0 units in the first case and 8/5 = 1.6 units in the second case.

With a local reward scheme, Robot1 is unwilling to pass the puck to Robot2 who is better at dumping. This is because, with the local reward scheme, the robot that dumps the puck will be the only one that gets the big reward and the other robot gets nothing. Therefore, this creates a competitive situation for both robots to be the one who dump the puck. Therefore, Robot1 learns not pass the puck to Robot2 and does everything by itself.

The diversity among robots does not have an effect on the final result. Our experiment shows that both homogeneous and heterogeneous teams can achieve the optimal solution. The learning algorithm can find the way to maximize the total reward in both cases. However, the experiment with homogeneous team attains higher total reward because the higher reward setup for Robot2.

The number of robots also has no effect on the final solution, but it requires more learning time. When there are more robots, there are more states and more situations for the robots to learn. This shows the scalability of the learning algorithm in our multirobot system without any modifications.

6. Conclusions

The effectiveness of multirobot learning in achieving optimal, cooperative solutions is potentially affected by various factors. In earlier work [4], we tested the effect of these factors with a surveillance task, in which the performance depends on the placement of robots. In the current work, we test the same factors with the puck-collecting problem, in which the performance depends on the interactions with pucks. Based on these two tasks, we found that the type of learning algorithm and the reward scope have a crucial effect on the learning results. They can create a greedy strategy which gives unacceptable team performance. We also found that the diversity of robots and the number of robots do not have effects on the final results although they can affect the learning speed.

References

[1] Mataric M.J., "Interaction and Intelligent Behavior", Ph.D. thesis, MIT EECS, 1994.

[2] Parker L.E., "Heterogeneous Multi-Robot Cooperation", Ph.D. thesis, MIT EECS, 1994.

[3] Tangamchit P., Dolan J.M. and Khosla P.K., "Dynamic Task Selection: A Simple Structure for Multirobot Systems", DARS 2000, pp.483-484.

[4] Tangamchit P., Dolan J.M. and Khosla P.K., "Crucial Factors Affecting Cooperative Multirobot Learning", to appear in IROS2003

[5] Watkins C.J.C.H., "Learning from Delayed Rewards", Ph.D. thesis, King's College, Cambridge, UK, 1989.

[6] Balch T., "Behavioral Diversity in Learning Robot Teams", Ph.D. thesis, Dept. of Computer Science, Georgia Tech., 1998.

[7] Dudek G., Jenkin M.R., Milios E. and Wilkes D., "A Taxonomy for Multi-Agent Robotics", Autonomous Robots 3 (4):375-397, December 1996, Kluwer Academic Publishers.

[8] Balch T., "Taxonomies of Multirobot Task and Reward", Technical Report Robotic Institute, CMU, 1998.

[9] Kaelbling L., Littman M. and Moore A., "Reinforcement Learning: A Survey", Journal of AI Research 4, pp.237-285, 1996.

[10] Tangamchit P., Dolan J.M. and Khosla P.K., "The Necessity of Average Reward in Cooperative Multirobot Learning", ICRA 2002.

[11] Fukuda T.,Kawauchi Y., "Cellular Robotics", pp. 745-782, Springer-Verlag 1993.

[12] Sutton R.S. and Barto A.G., "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA, 1998.