Crucial Factors Affecting Cooperative Multirobot Learning

Poj Tangamchit [1]
E-mail: poj@andrew.cmu.edu
Dept. of Control System and Instrumentation Engineering [1]
King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand

John M. Dolan [3]
jmd@cs.cmu.edu

Pradeep K. Khosla [2,3]
pkk@ece.cmu.edu
Dept. of Electrical and Computer Engineering [2], The Robotics Institute [3]
Carnegie Mellon University, 5000 Forbes Ave. Pittsburgh, PA 15213, USA

Abstract

Cooperative decentralized multirobot learning refers to the use of multiple learning entities to learn optimal solutions for an overall multirobot system. We demonstrate that traditional single-robot learning theory can be successfully used with multirobot systems, but only under certain conditions. The success and the effectiveness of single-robot learning algorithms in multirobot systems are potentially affected by various factors that we classify into two groups: the nature of the robots and the nature of the learning. Incorrect set-up of these factors may lead to undesirable results. In this paper, we systematically test the effect of varying five common factors (model of the value function, reward scope, delay of global information, diversity of robots' capabilities, and number of robots) in decentralized multirobot learning experiments, first in simulation and then on real robots. The results show that three of these factors (model of the value function, reward scope, and delay of global information), if set up incorrectly, can prevent robots from learning optimal, cooperative solutions.

Keywords: multirobot learning, distributed reinforcement learning, cooperative robots

1. Introduction

Reinforcement learning [21] provides robots with adaptability to fit the environment without prior knowledge of the world model. We focus our attention on fully decentralized multirobot systems because of their potential for robustness. Robustness is an important property in multirobot systems, especially for military-based tasks, among which is included our testbed task, the multirobot patrolling problem. Decentralized multirobot systems have high robustness because their control does not depend on a single entity. Reinforcement learning is a mechanism added to robots in order to enable them to adapt to their environment. We call reinforcement learning applied to decentralized multirobot systems "cooperative decentralized multirobot learning". This type of learning uses single-robot learning algorithms on each robot in a multirobot system and considers the resultant learning performance of the whole group. Each learning entity runs independently and asynchronously from the others. In this paper, we study the effect of different multirobot system parameters on the performance of cooperative decentralized multirobot learning.

Most decentralized multirobot learning implementations use single-robot learning algorithms [4],[16],[26]. However, when learning is applied to multirobot systems, there are additional issues that appear to require consideration. Some examples are how rewards are distributed among robots, and how information is synchronized among robots. These issues and their effects have not been systematically investigated. Although single-robot learning has been extensively studied [1],[21],[25], there have not been many publications about multirobot learning to date [8][4],[16]. Mataric [16] introduced progress estimators to provide additional rewards to robots with respect to the progress of tasks, and showed that they can help speed up learning and improve performance by performing multirobot learning experiments on foraging, flocking and docking tasks. Moreover, she reported that it was difficult for robots to learn to perform tasks in a real-world environment without using progress estimators [16]. Balch [4][5] investigated the performance of robot teams with different levels of diversity on three types of tasks: foraging, robot soccer, and cooperative movement. In his learning experiments, he used shaped reinforcement rewards, which are combinations of various rewards given to robots for different behaviors. For example, in his experiment with a foraging task, the objective was to use robots for gathering food into a bin. The robots received rewards for picking up food, delivering food, and moving toward

the bin with food. They also received punishments for dropping food, colliding with other robots, and moving away from the bin while carrying food. All the implementations of multirobot learning mentioned above require the use of human intelligence to design special rewards in order to guide robots to achieve the goal. The disadvantage of these special rewards is that they force the robots to learn corresponding behaviors that may not promote the original objective of the mission.

Traditional robot learning provides rewards only when robots reach the goal. For example, in robot soccer, a robot should get a reward only when it scores a goal, not for other actions. We discovered that the success of traditional learning in multirobot systems depends on the configuration of such "environmental factors" as robot architecture, learning algorithms, and rewards. For systematic study, we made a taxonomy of environmental factors and used it as an index of possible configurations that can be varied in multirobot learning. In our experiments, we tested the effect of varying these factors on the capability of multirobot learning to achieve the optimal result. The experiments indicate that traditional learning can achieve optimal solutions for the prototypical multirobot patrolling and puck-collecting tasks, but only with appropriate settings for the environmental factors.

This paper is structured as follows. Section 2 discusses previous work in multirobot learning. Section 3 describes our approach to multirobot learning and the multirobot patrolling problem. This section starts with a taxonomy of factors potentially relevant to multirobot learning performance. Section 3.2 then discusses the details of the multirobot patrolling problem. The details of each environmental factor tested are discussed in section 3.3. Section 4 contains the experimental details and results from simulations and summarizes our results from previous experiments with another testbed problem. Section 5 presents experimental results from real robots. Section 6 discusses the results and section 7 provides conclusions.

2. Previous Work

Cao et al. [8] extensively surveyed research in multirobot systems, dividing it into five areas, of which multirobot learning is one. The main focus in multirobot research to date has been multirobot architecture because it is the first step when implementing a multirobot system. Researchers have designed architectures to exploit special properties of multirobot systems, such as robustness and self-reconfiguration. This work includes CEBOT [10], a self-reconfigurable multirobot system composed of coupling robots, and ALLIANCE [18], an architecture based on behavior-based modules [7] with the addition of motivational behaviors used to determine the high-level plan of the system. This work provided researchers in the field with a framework for multirobot implementation and the inclusion of additional properties, such as multirobot learning. However, each architecture is different in the features it supports and its implementation details. There is no architecture that handles all individual users' requirements. As a result, we created a method called dynamic task selection [22], which uses a hybrid robot architecture and a task-sharing concept. Both features enable robots to learn and make decisions in task-level programming [17], which supports the concept of robot cooperation.

Learning is a mechanism that helps robots optimally choose tasks and divide their duty. Learning in decentralized multirobot systems has not been systematically studied. Mataric [16] and Balch [4] used single-robot learning algorithms to implement decentralized multirobot learning. This included the use of Q-learning [25], which is the most popular single-robot learning algorithm. However, Mataric [16] also indicated that single-robot learning in multirobot systems usually requires some modifications in order to achieve the best performance. She introduced progress estimators, a special type of reward that helps robots learn the task more efficiently. Balch [4] investigated team diversity and showed that it can have an impact on the learning performance of robot teams in some types of tasks. He used shaped reinforcement rewards, another modified reward system. Parker [19] introduced the L-ALLIANCE architecture, which integrates reinforcement learning into her multirobot group architecture. L-ALLIANCE adapts its parameters using reinforcement learning. This is different from other work, which uses reinforcement learning with action selection. Yanco [26] implemented learning on a group of robots to learn their communication language.

Previous work in multirobot learning such as Mataric's and Balch's has emphasized learning performance using specially designed reward systems. Our work differs from this in that we use a traditional, reward-at-goal reward system. Our rule for distributing rewards is clear and general for all tasks. Unlike Mataric and Balch, Parker and Yanco conducted learning experiments that emphasized simple problems that do not have

delayed rewards, such as following the leader. A group of multirobots performing these types of tasks can accomplish the goal with one action, and can thus be considered a reward-at-goal system. However, many practical cooperative multirobot tasks such as the multirobot patrolling problem and the puck-collecting problem involve delayed rewards which are only given once multiple robots have completed a task by performing multiple subtasks. This makes learning with delayed rewards complicated and requires the additional consideration of several factors.

3. Approach

We start by introducing a taxonomy of environmental factors (section 3.1) in order to make the study of these factors as systematic as possible. In section 3.2, we explain the multirobot patrolling problem, which we use as a testbed for our study. In section 3.3, we discuss the tested environmental factors in detail and how they are varied in the multirobot patrolling experiments.

3.1. Taxonomy of 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. For systematic study of these factors, we make a taxonomy of them using the work by Balch [6] and Dudek [9] as guidelines. 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 learning algorithms and rewards. We do not make a taxonomy of tasks because they are user-specific and can vary indefinitely according to users' requirements.

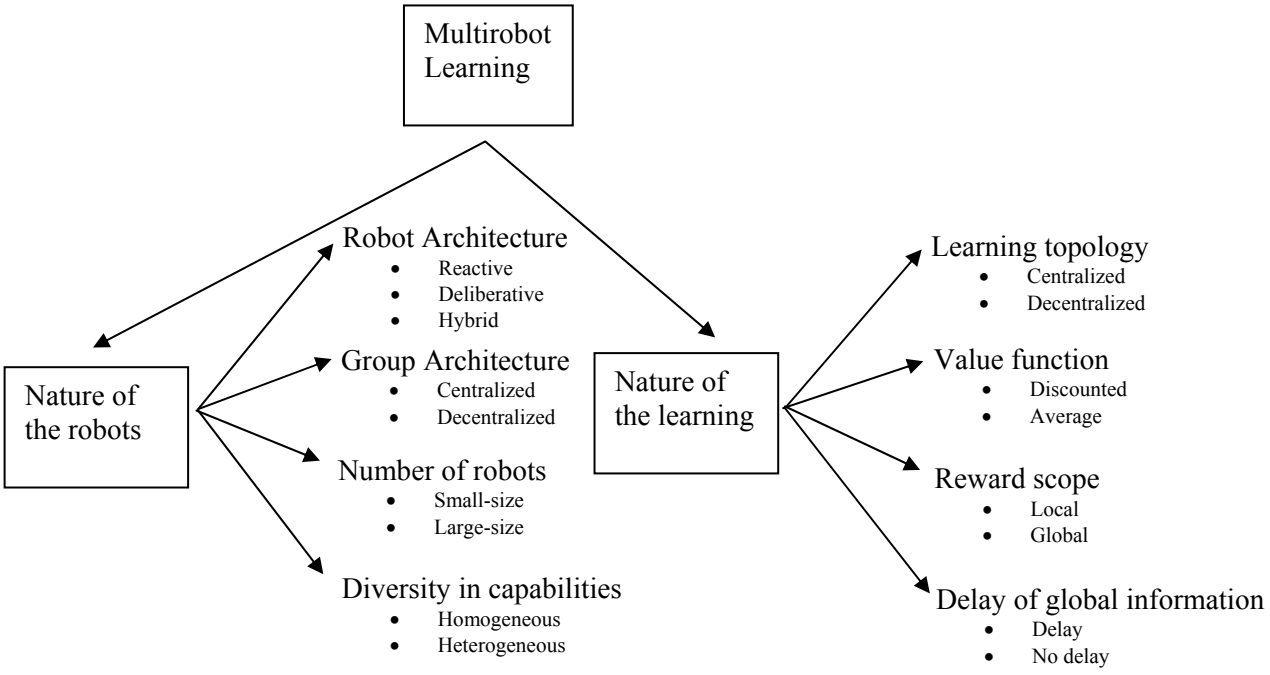

Figure 1 An Overview of Multirobot Learning Taxonomy

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), model of the value function (discounted / average), delay of global information (delay/non-delay) and reward scope (local / global).

We focus our study on decentralized multirobot systems because of their potential for high robustness. Our robots have hybrid robot architectures for exploitation of reactive behaviors with high-level plans. Therefore, this paper presents multirobot learning with the following fixed parameters: 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, model of the value function, delay of global information, and reward scope.

3.2. The Multirobot Patrolling Problem

The multirobot patrolling problem requires the deployment of multiple mobile robots, each equipped with a camera, to guard a large area. The cameras are assumed to have limited range. To provide enough coverage, the robots have to move around and search for intruders. We use the security guard rule, which assigns checkpoints at important positions in the area that we want to guard. The robots then have to divide these checkpoints and patrol them in cycle as fast as possible. This principle can be applied to a complex layout like that in Figure 2 as long as the robots have a good navigation algorithm. Checkpoints can be located at closed corridors or positions where valuable resources are located.

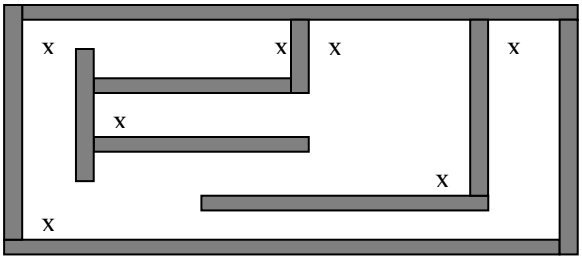

Figure 2 Multirobot Patrolling Problem on a Complex Layout

The system performance (time used) of the multirobot patrolling problem depends heavily on appropriate checkpoint division. We use reinforcement learning in order to make the robots automatically adjust themselves to the optimal task assignment without prior knowledge of the world model. Learning also handles changes in the environment, such as a change in the number of robots or checkpoints. Our problem set-up consists of a rectangular area with four checkpoints and two or three robots. In each cycle, the robots have to visit each checkpoint at least once. After all checkpoints are visited, a new cycle begins. Our work and previous work in multirobot learning [3],[14],[18] use simple tasks as experimental testbeds, such as foraging [3],[14] and box pushing [18]. The main reason is that we are initially more interested in validating learning theory than in implementing it on a complex task. Simple tasks are easy to observe and analyze. We chose the multirobot patrolling problem with a small number of checkpoints to be our testbed for that reason.

Due to the horizontal distance between points being greater than the vertical distance, the optimal solution is where one robot takes on two checkpoints on the left-hand side and the other robot takes on two checkpoints on the right-hand side.

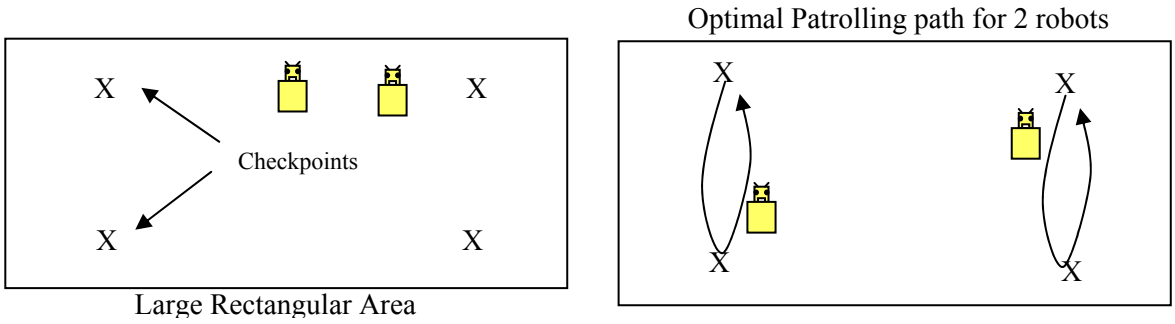

Figure 3: The Multirobot Patrolling Problem

3.3. Details of Tested Environmental Factors

In this paper, we investigate three factors related to the nature of the learning entities (model of the value function, reward scope, and delay of global information) and two factors related to the nature of the robots (number and diversity). Each factor varied is detailed below.

- *The model of the value functions of the learning algorithm*

The learning algorithm calculates the value functions for each robot's state and chooses the best action. The value functions are normally computed as total expected rewards after reaching each state. Due to the nature of reinforcement learning, which models a robot's actions into discrete steps, there are two methods for calculating the value functions. The first method uses cumulative discounted reward and the second method uses average reward. The cumulative discounted reward evaluates future rewards to be of less value than current rewards. For each step in the future, the expected reward gets discounted by a factor γ, as in the equation below.

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

R(t) is the total expected reward in the future after reaching this state. R(t) is equal to the sum of the reward in the current state plus discounted expected rewards in the future.

The second method uses the average of the current and all future rewards as a value function. This method views rewards of equal value as having the same merit regardless of time. The expected total reward in the future is computed as follows.

$$R(t) = \lim_{n\to\infty}\left(\frac{1}{n}(r_t + r_{t+1} + r_{t+2} + \ldots + r_{t+n})\right) = \lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=0}^{n} r_{t+i}\right)$$

The total expected reward in the equation is computed as a sum of rewards in the future divided by the number of steps *n*. There is no discount factor involved in the equation. The limit notation assumes a continuous unlimited time, so that the expected reward can be calculated up to a point far in the future. However, in a real implementation, the tasks take limited time. This type of task is called episodic and ends after a certain condition, such as reaching the goal, is met. The other type of task is called non-episodic, in which robots learn the environment continuously. All of our learning experiments in this paper are episodic. The final goal is well defined, and the learning re-starts after the goal has been reached. In episodic tasks, we omit the limit notation in the equation. The total expected reward becomes as follows:

$$R(t) = \frac{1}{n}(r_t + r_{t+1} + r_{t+2} + \ldots + r_{t+n}) = \frac{1}{n}\sum_{i=0}^{n} r_{t+i}$$

In our experiments, we use Q-learning [25] to implement the discounted-reward value function scheme and use Monte Carlo Learning [21] to implement the average-reward value function scheme. Learning algorithms that are based on the same value function and same environmental factor set-ups will produce the

same final result, but may be different in terms of convergence speed. This is because all existing learning algorithms guarantee convergence to the individual optimal solution.

- *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. There are two reward schemes 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 reward or punishment together, as a team. The reward (cost) used in the patrolling problem is the time needed to travel from one checkpoint to another checkpoint. The implementation of local/global reward is common to all kinds of tasks. A global reward scheme, which shares rewards among robots, can be implemented by broadcasting all rewards generated within each robot to the teammates. The transmitted rewards are then added up to the internal reward for the current action. A local reward scheme, on the other hand, keeps all rewards internally in each robot without broadcasting.

- *Delay of Global Information*

In decentralized systems, communication among robots is a key mechanism to make the system efficient and consistent. The information about the world that robots exchange among themselves is called global information. In multirobot learning, the state of each robot is dependent not only on its own state information, but also on information from its teammates. Communication is required in order to synchronize the world state among all robots. Generally, a robot will tell its teammates when there is a change in the world. In the patrolling problem, a robot has to tell its teammates about which checkpoint it chooses to visit next. A delay in this information is created by broadcasting the message after the robot has already reached that checkpoint.

- *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. In the patrolling problem, the robots' main action is moving to checkpoints. Therefore, we can create a difference in capability by giving each robot different permissions for going to checkpoints. For example, one robot can have permission to go to all checkpoints while another robot only has permission to go to checkpoints on the right-hand side. In our experiment on a heterogeneous team, the first robot can go to all checkpoints but the second robot is prevented from going to the checkpoint in the upper right-hand corner.

- *Number of Robots*

We classify this factor into two types: small-size (2-3 robots) and large-size (>20 robots). The large-size group is outside the scope of our study because we have a limited number of robots. Moreover, our simulation program processes sensor data, robot control and interactions among robots in real time (update every 0.1 seconds). The number of interactions grows exponentially with the number of robots. The learning entities use the Dyna architecture [21], which uses a lot of computation power. All of these requirements make the simulation of a large-size group intractable. 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 algorithm. We tested this factor by adding both robots and checkpoints and then restarting the learning to see if the division of labor has changed. In our experiment, one robot and two more checkpoints were added.

4. Simulation and Results

4.1 Effect of Environment Factors in the Multirobot Patrolling Problem

Our simulation was written in Visual C++. The robots have two predefined low-level behaviors: move to a point and avoid obstacles. The robots are assumed to be equipped with GPS, sonar sensors and a communication device. Each robot runs on a separate thread to emulate the asynchronous timing of the real world. Learning entities are implemented on each robot independently. The communication channel is used for

sending global rewards and the world state update. The multirobot patrolling problem is modeled as a Markov Decision Process (MDP). The state of each robot is computed from the following parameters:

State = {last checkpoint visited, checkpoint already visited}

The set of possible actions for each robot was as follows:

Actions = {move to point n, wait and do nothing}

The state of each robot is composed of information about checkpoints that have been visited. This is global data. We synchronize this data using a message/mailbox mechanism. Each robot sends an update message when it visits a new checkpoint. Also, each robot periodically checks its mailbox to synchronize global data. For patrolling with 4 checkpoints, there are 64 (2^4x4) states and 5 (4+1) actions. The state is dependent on which checkpoints have already been visited (2^4 possible combinations) and which checkpoint the robot visited last (4). The possible actions are going to any of the points (4) plus the "wait" command (1). The reward system is straightforward. We give a reward to the robots once when the goal is achieved. This is called the reward-at-goal method. In the multirobot patrolling problem, the goal is accomplished when all checkpoints are visited. At that time, a big positive reward is given to the robots. We also give negative rewards that are considered as the cost of movement from one point to another. The movement cost is calculated as the time (in seconds) needed to travel between points, and can fluctuate due to possible collisions among robots. When the robot chooses to wait and do nothing, it receives a small cost due to overhead. The reward table is presented below.

Action	Reward
Move	(Time used in second × (-10)) - 10
Wait	-10
All points are visited	2000

Each robot uses a modified version of the ε-greedy algorithm [21]. The original ε-greedy algorithm chooses the best action with fixed probability ε and chooses a random action (for random exploration) with fixed probability (1- ε). The modified version has a variable value of ε, which is inversely proportional to the amount of the cumulative reward in each learning epoch. At the beginning, the value of ε is set to be equal to the minimum limit (MinReward), in which the probability of choosing a random action is one. As learning goes on, the robots get rewards, and the cumulative reward increases. If the value of ε is equal to or greater than the maximum limit (MaxReward), the probability of choosing random action will be zero. The value of the maximum limit (MaxReward) will be continuously decreased stepwise when there is no improvement in the cumulative reward. When MaxReward is decreased to be equal to MinReward, the probability of choosing a random action will be zero for all values of the cumulative reward. This makes the learning result become stable at the end. Another consideration is that we plan to implement learning on real robots. We cannot have a long learning time because of the power requirements and the strain on human attention. We therefore use Sutton's Dyna architecture [21] to help speed up the learning. Its basic idea is that most learning algorithms do not use training samples to their full extent. Dyna keeps a record of training samples and reuses them. In other words, it uses past training samples to create a hypothetical world and learn from this world. There are therefore two sources of learning: one from new training samples and one from the hypothetical world.

After randomly varying factors and running the experiments, we found two types of results: the optimal case where each robot takes two points in the same column, and the sub-optimal case where one robot or more gets stuck and always chooses to do nothing. The second case occurs because the robots learn that choosing the "wait" action repetitively will give it the highest total reward. We designate the first case "converge" and the second case "deadlock". Moreover, the effect of factors that create deadlock seems to be dominant regardless of the setting of other factors. For example, using a local reward scheme always creates deadlock regardless of the setting of other factors. There was also no change in results when the factors that did not create deadlock were simultaneously adjusted. This suggests that the effects of the five factors are independent of one another. The

results presented here are based on changes in each factor compared to the "standard case" of two robots, four checkpoints, Monte Carlo learning, global reward, a homogeneous team, and no delay in global information. 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 randomness of exploration (value of ε) reducing to zero. In addition to human observation, we recorded the amount of reward gained during the stable state on each run, and performed a statistical t-test [12] in order to confirm the optimality of the results. The test confirms a 95% confidence interval for optimality with different set-ups of the robots' capability and delay of global information Statistical tests of value function and reward scope were omitted because they cannot achieve a stable result, and a statistical test of the number of robots was omitted because the optimal paths are different. The table below summarizes the results.

Factors		No. times converged	Median time to converge (epochs)	Min time to converge (epochs)	Max time to converge (epochs)
Value function	Monte Carlo	10	145	130	185
	Q-learning	0	N/A	N/A	N/A
Reward Scope	Global	10	145	130	185
	Local	0	N/A	N/A	N/A
Delay of Global Info.	No Delay	10	145	130	185
	Delay	5	324	224	483
Diversity of robots' capability	Homogeneous	10	145	130	185
	Heterogeneous	10	98	83	124
Number of robots	1 robots/4 points	10	68	55	91
	2 robots/4 points	10	145	130	185
	3 robots/6 points	10	830	593	1143

The results were consistent across runs except for the delay of global information, which had 50% convergence and 50% deadlock. Increasing the learning time did not change this result. The reward scope had a great effect on the learning result, whereas robot diversity and the number of robots had no effect. Figure 4 shows the plot of total rewards in each epoch during the learning. The graph on the left is the reference case that has 2 robots, 4 checkpoints, homogeneous team, Monte Carlo learning (average-reward), global reward scheme, and perfect communication. The graph on the right is the case when we change to local reward scheme. These two graphs represent converge and deadlock cases that were discussed previously. The shapes of the graphs for other cases are similar to these two. For converge cases, there is a trend of improvement in total reward gained overtime. At the end, the graph reaches a stable point where the total reward reaches the maximum. For deadlock cases, the total reward gained in each epoch is quite random until the epoch in which the robots reach a deadlock situation.

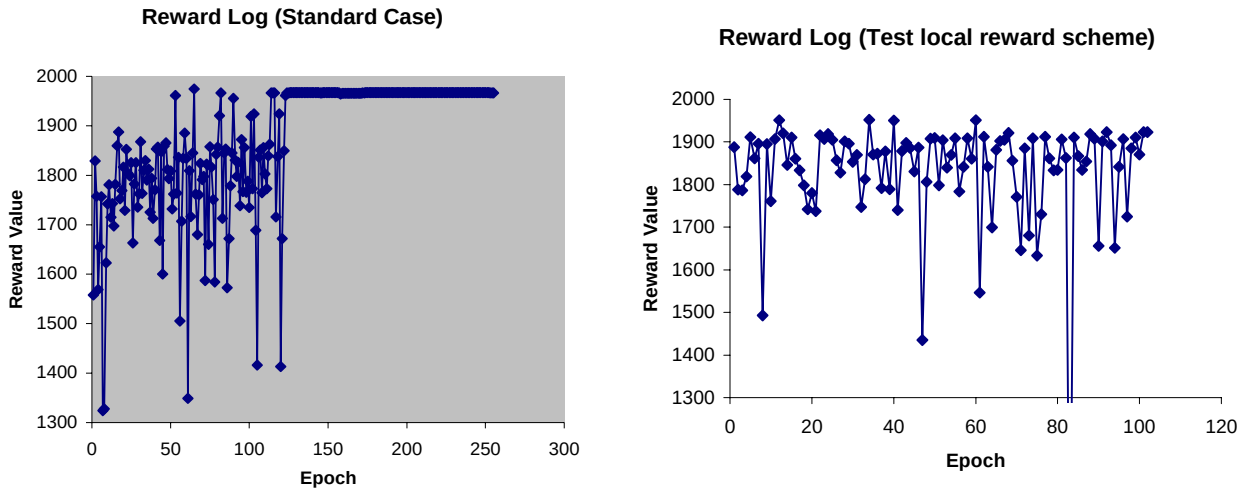

Figure 4 Total reward accumulated in each epoch during the learning (simulation)

The test of the last factor is to vary the number of robots and number of checkpoints. We have three cases: (1 robot, 4 checkpoints), (2 robots, 6 checkpoints), and (3 robots, 6 checkpoints). We also try randomly relocating the placement of checkpoints in the 2 robots/ 4 checkpoints case. The results show that decentralized learning, with the standard setting of other environmental factors, can obtain the optimal result for all cases. However, the time required to achieve the optimal results is different in each case. The 1-robot case takes the shortest time (because of least complexity), while the 3-robot case has the longest time. The results from all cases are shown below.

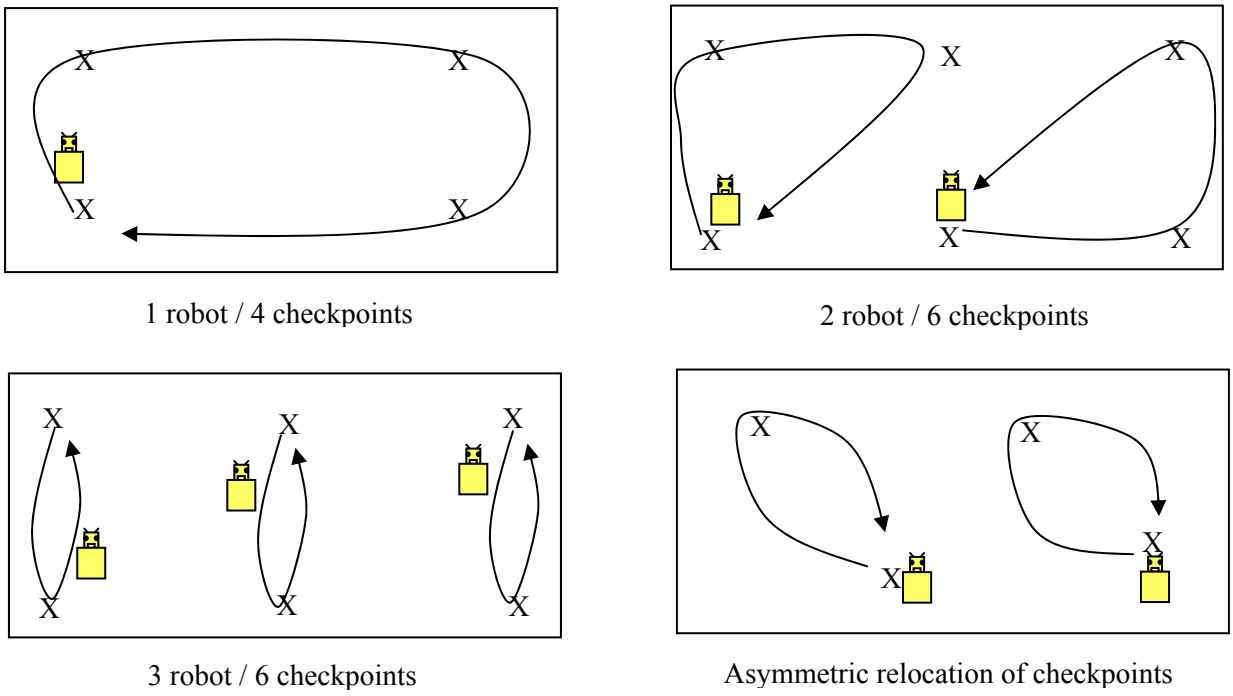

Figure 5: Optimal Patrolling Paths for Various Robots/Checkpoints Configurations

4.2 Effect of Environment Factors in an Object Manipulation Task

The experiments described use the multirobot patrolling problem as a testbed. The multirobot patrolling problem has its performance depending on placement and positions of all robots in the team. This problem is considered as a robot-based problem according to Balch's taxonomy [6]. In this section, we summarize our test of the same environmental factors with another type of task, the puck-collecting problem [24]. The puck-collecting problem is a test of cooperative scenario for two robots in gathering pucks and putting them into a bin. The task is designed in a way that the optimal result requires contributions from both robots. The first robot, which has more agility, searches for pucks and bring them back to the bin. The second robot, which has superior skill in manipulating a puck, then picks up the puck and puts it into the bin. Both robots have to learn this solution in order to achieve the optimal result. The reward scheme is the same as that in the multirobot patrolling problem. Each robot will get a negative reward proportional to the time used for all actions. However, when it puts a puck in into the bin, the robot will get a big positive reward.

We tested four environment factors: number of robots, diversity of robots' capability, model of the value function, and reward scope in both simulations and real robots. The results indicate that number of robots and diversity of robots' capabilities do not have an effect on the learning results. However, using the discounted model of value function and local reward scope prevents learning from achieve the optimal cooperative results. The use of these two factors creates a "greedy strategy", in which each robot ignores the team objective and emphasizes its utility. This is the same result as in the multirobot patrolling problem.

5. Real Robot Experiments

As a general rule, robot simulations need to be confirmed on real robots due to unexpected or unmodeled incidents or behaviors, such as noise in sensors. We verified our simulation by porting the same learning algorithm onto Pioneer robots from Activmedia Corp. equipped with an onboard PC104, sonar sensors, and a wireless LAN card. We used an overhead camera to detect robot position (act as a GPS).

The results from real robot experiments were qualitatively the same as those in simulation. The optimal solution was found with the standard setting of the environmental factors, as each robot took two points in the same column. Additionally, deadlock occurred when we used a local reward scheme and when we added the delay of global information.


The robots


The field


Converge


Deadlock (stuck)


Video clips of the converge sequence

Figure 6: Real Robot Experiment

6. Discussion

The results indicate that only the model of the value function, the reward scope, and the delay of global information affect the final results of learning. In this section, we analyze the effect of each of the five factors considered.

The model of the value function has a crucial impact on the learning results. Q-learning, which uses a discounted reward scheme, fails to achieve the optimal solution. The reason behind this failure comes from the nature of the discounted reward scheme. For most tasks, a big reward usually occurs after the goal is accomplished. This reward is often observed by only one robot. This robot will get a reward immediately. Using a global reward scheme, it will then take some time to propagate to other robots. Because of this delay, the other robots will get a smaller reward for their actions due to the cumulative discounted reward framework that discounts rewards over time. This phenomenon encourages each robot to be the one that completes the task (visit the last checkpoint). However, cooperation requires the robots to divide their duties fairly. If all robots compete for the same action, the system is unlikely to learn the best performance. As the results suggest, each robot chooses to wait for its teammate to do other tasks, so that it can be the one that visits the last checkpoint.

With a local reward scheme, each robot will be unwilling to do any task. An explanation of this result is as follows. Monte Carlo learning is episodic. It keeps counting the total reward accumulated until the end of each cycle and learns the whole series of actions needed to complete the job. In other words, the learning algorithm learns the series of actions, not just each individual action. With a local reward scheme, there is a possibility for a robot to avoid costs by doing nothing and have the job done by its teammates. When all robots learn this strategy, the final result will be that each robot is unwilling to do any task. This will make the robots greedy and finally lead to a deadlock. With a global reward scheme, a robot cannot avoid costs by doing nothing and have the job done. This is because the work done by its teammates will also generate a cost for itself. The best strategy for minimizing costs is therefore to cooperate and do the part of the job most suited to oneself. Therefore, global reward generates a fair division of labor and utilizes the capability of the team to the full extent. Global reward can therefore induce cooperation among robots.

The delay in global information also affects the learning. The result was sometimes convergence and sometimes deadlock. Delayed information has an effect equivalent to that of false information. It tells the robot that the robot is in one state when the robot is actually in another state. Therefore, the learning matches the states with the wrong actions and a wrong solution is created. Consider **Figure 7**, when there is no delay, the correct state-action pairs are S1-A1, S2-A2, and S3-A3. When there is a delay in global information, the state-action pairs will be S1-A2 and S2-A3, which are incorrect.

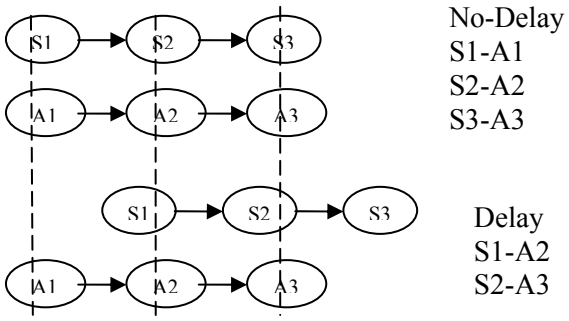

Figure 7: Delay of Global Information

When robots have diversity in capabilities, the learning algorithm on each robot will be able to find the optimal division of labor. In our experiment, the optimal solution is the same for both homogeneous and heterogeneous teams. For a heterogeneous team, the robot that cannot go to the upper right corner learns to patrol the two checkpoints on the left. The other robot compensates for the inability of the first robot by patrolling the checkpoints on the right.

The number of robots also has no effect on the final solution, but it slows the learning significantly. When there are more robots, there are more states and situations for the robots to learn. The learning time will grow exponentially. With 6 checkpoints, the number of states will be 2^6x6 = 384 states. Thus, a 50% increase in

checkpoints will result in about a 600% increase in the number of states to explore. The result shows that an increase in number of robots does not prevent the learning from achieving the optimal result. However, the multirobot patrolling problem does not scale well because it resembles a distributed version of the traveling salesman problem, which is NP–complete [20].

7. Conclusions

The effectiveness of multirobot learning in achieving optimal, cooperative solutions is potentially affected by various factors. We tested the effect of these factors by performing experiments on two types of tasks: the multirobot patrolling and the puck-collecting problems. Using a hybrid, decentralized group architecture, we demonstrated the sensitivity of learning performance to the model of the value function, reward scope and global information delay, and its insensitivity to the diversity of robots' capability and number of robots. A key insight of this paper is that learning algorithms that have discounted value function, such as the widely used Q-learning, generate a suboptimal result in cooperative multirobot learning. Local reward scope is another factor that prevents learning from achieving the optimal result. Therefore, an average reward-based value function and global reward scope must be used when seeking optimality in cooperative decentralized multirobot learning.

References

[1] R.C. Arkin, "Behavior-Based Robotics", MIT Press, Cambridge Massachusetts, 1998

[2] T. Balch and R.C. Arkin, "Communication in reactive multiagent robotic systems", Autonomous Robots, 1(1): 27-52, 1995.

[3] T. Balch and R.C. Arkin, "Cooperative Multiagent Robotic Systems, Artificial Intelligence and Mobile Robots , D. Kortenkamp, R.P. Bonasso, and R. Murphy (eds), MIT Press, 1998.

[4] T. Balch, "Reward and diversity in multirobot foraging", IJCAI-99 Workshop on Agents Learning About, From and With other Agents.

[5] T. Balch, "Behavioral diversity as multiagent cooperation", SPIE '99 Workshop on Multiagent Systems, Boston, 1999.

[6] T. Balch, "Taxonomies of Multirobot Task and Reward", CMU-RI Technical Report

[7] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", MIT AI-LAB Technical Report 1985

[8] Y.U. Cao, A.S. Fukunaga and A.B. Khang, "Cooperative Mobile Robotics: Antecedents and Directions", Autonomous Robots, Kluwer Academic Publishers 1997

[9] G. Dudek, M.R.M Jenkin, E. Milios and D. Wilkes, "A Taxonomy for Multi-Agent Robotics", Autonomous Robots, 3:375-397 1996

[10] T. Fukuda, S. Nakagawa, Y. Kawauchi and M. Buss, "Self organizing robots based on cell structures-CEBOT" Proceedings of 1988 IEEE Intl. Workshop on Intelligent Robots and Systems (IROS'88)

[11] E. Gat, "Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots", Procedding AAAI-92 pp 809-815 1992

[12] J.L. Kenkel, "Introductory Statistics for Management and Economics", 4th edition, Wadsworth Publishing Company, Belmont California, 1996

[13] M.J. Mataric, "Issues and Approaches in the Design of Collective Autonomous Agents", Robotics and Autonomous Systems, 16(2-4), Dec 1995, 321-331.

[14] M.J. Mataric, "Reinforcement Learning in the Multi-Robot Domain", Autonomous Robots, 4(1), Mar 1997, 73-83.

[15] M.J. Mataric, "Using Communication to Reduce Locality in Distributed Multi-Agent Learning", Journal of Experimental and Theoretical Artificial Intelligence, special issue on Learning in DAI Systems, Gerhard Weiss, ed.,10(3), Jul-Sep, 1998, 357-369.

[16] M.J. Mataric, "Reward Functions for Accelerated Learning", Machine Learning: Proceedings of the Eleventh International Conference, William W. Cohen and Haym Hirsh, eds., Morgan Kaufmann Publishers, San Francisco, CA,1994, 181-189.

[17] P.J. McKerrow, “Introduction to Robotics” chapter 9, pp.483-489, Addison-Wesley Publishing Company 1991
[18] L.E. Parker, “ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation”, IEEE Transactions on Robotics and Automation, 14 (2), 1998.
[19] L.E. Parker, “L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behavior-Based Systems”, Advanced Robotics, Special Issue on Selected Papers from IROS '96, 11 (4) 1997: 305-322.
[20] S.J. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach”, Prentice Hall Series in Artificial Intelligence, Prentice Hall 1995
[21] R.S. Sutton and A.G. Barto, “Reinforcement Learning: An Introduction”, MIT Press, Cambridge MA, 1998
[22] P. Tangamchit, J.M. Dolan and P.K. Khosla, “Dynamic Task Selection: A Simple Structure for Multirobot System”, Distributed Autonomous Robotic Systems 2000 (DARS2000), Springer-Verlag 2000
[23] P. Tangamchit, J.M. Dolan and P.K. Khosla, “The Necessity of Average Rewards in Cooperative Multirobot Learning”, IEEE Int’l Conference on Robotics and Automation 2002 (ICRA’02)
[24] P. Tangamchit, J.M. Dolan and P.K. Khosla, “Crucial Factors Affecting Decentralized Multirobot Learning in an Object Manipulation Task”, ACM/SIGART Workshop on Agent Swarm Programming 2003 (WASP’03), Cleveland Ohio
[25] C.J.C.H. Watkins, "Learning from Delayed Rewards", Ph.D. thesis, Cambridge Univ., Cambridge, England 1989
[26] H. Yanco and L. Stein, “An Adaptive Communication Protocol for Cooperating Mobile Robots”, In Proceeding Simulation of Adaptive Behavior, pp.478-485, 1992

Poj Tangamchit received his B.Eng. (hon.) (1995) degree from King Mongkut’s University of Technology Thonburi, Bangkok, Thailand. He received his M.S. (1997) in Electrical, Computer and System Engineering from Rensselaer Polytechnic Institute and Ph.D. in Electrical and Computer Engineering (2003) from Carnegie Mellon University, USA. He is currently a faculty at the department of control and instrumentation at King Mongkut’s University of Technology Thonburi. His research interests include mobile robots, distributed robotic systems, machine learning, and multirobot learning.

John M. Dolan received his B.S. (1980) degree from Princeton University and his M.E. (1987) and Ph.D. (1991) from Carnegie Mellon University, all in mechanical engineering. He is currently a member of the research faculty at Carnegie Mellons Robotics Institute. He is the recipient of a Fulbright Scholarship for study at the Technical University of Munich, where he also worked at the German space agency (DLR). His research interests include distributed mobile robotics, sensor-based control of robotic and manufacturing systems, and man-machine interaction. His research has resulted in over 40 journal and conference publications and book contributions. Dr. Dolan is a member of the IEEE.

Pradeep Khosla received his B.Tech. degree (1980) from ITT, India, and MS (1984) and Ph.D. (1986) degrees in Electrical Engineering from Carnegie Mellon University (CMU). He is the Founding Director (1997-1999) of CMUs Institute for Complex Engineered Systems and is currently the Dowd Professor of Engineering and Robotics and Head of CMUs ECE Department. His research interests include collaborative and distributed design and manufacturing, agent-based architectures, and reconfigurable and distributed robotic systems. He has more than 200 journal articles, conference papers, and book contributions. Prof. Khosla is a Fellow of the IEEE.