Distributed Learning in Swarm Systems: A Case Study

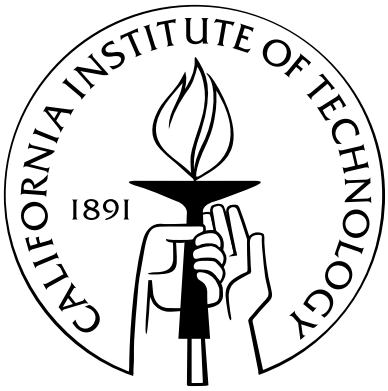
Thesis by

Ling Li

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science



California Institute of Technology

Pasadena, California

2002

(Submitted May 31, 2002)

Acknowledgements

It is my great pleasure to thank my advisor Professor Yaser S. Abu-Mostafa for his help, support, and guidance throughout my research and studies.

Dr. Alcherio Martinoli was invaluable in getting me started and providing many helpful and insightful suggestions along the way. Discussions with him were always fruitful and enjoyable.

I would like to acknowledge Lavanya Reddy and Eric Tuttle for having implemented a first version of the Δ-method used in this thesis.

I owe a lot to Alexander Nicholson, who helped me in many aspects of daily life and research, especially for proofreading all my slides and writing on this work.

My fellow members of the Learning Systems Group have provided me with many valuable discussions and helpful suggestions for my work. In addition to those already mentioned, my thanks go to Amir Atiya, Igor Bargatin, Dustin Boswell, Gentian Buzi, Carl Gold, Nathan Gray, Malik Magdon-Ismail, and Amrit Pratap.

Finally, I thank my family, especially my wife and parents-in-law, for their endless love and support.

This work was supported by the Caltech Center for Neuromorphic Systems Engineering, a National Science Foundation supported Engineering Research Center, under NSF Cooperative Agreement EEC-9402726.

Abstract

This thesis investigates several learning issues in swarm systems under a case study—the stick pulling experiment. This is a strictly collaborative problem where collaboration between non-communicating robots is required to complete the task. We base our experiments on a probabilistic model which is faithful in simulating experiments with real robots. We extend the systematic search with early stopping and get the optimal performances of fully heterogeneous teams consisting of 2–6 robots.

By integrating learning ability into individual robots, the whole team can adapt according to environmental changes and can maintain a near-optimal performance. We test several learning algorithms, including adaptive line search and Q-learning. We find, for this case study, that learning algorithms which directly search for optimal parameters work much better than those based on reward estimation.

Compared with the optimal performance obtained from the systematic search, the learned performance is a bit lower on average. We discuss several issues that may hinder learning from finding the optimal parameters, such as different reinforcement, noise, and adaptability. Our experiments show that, though learning cannot lead to optimal performance, it does enhance adaptability and stability of the whole team. As an untested hypothesis, we conjecture that any learning model can only achieve a trade-off between optimality and adaptability.

Though the team is initially homogeneous, specialization is observed after learning. Our results show that policies allowing specialization achieve in general similar or better performances than policies forcing homogeneity. We develop ad hoc methods to measure the specialization, and find that a measure of specialization is sub-linear to the number of robots.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In the last few years, there has been increased interest in *swarm systems* consisting of multiple autonomous agents. Such systems can exhibit complex behavior which appears to transcend the abilities of the relatively simple constituent individuals. Perhaps the most striking examples are from nature: social insect colonies are able to build sophisticated structures and regulate the activities of millions of individuals by endowing each individual with simple rules. According to environmental changes, a colony can adjust its behavior through assigning different numbers of insects to different tasks or adjusting the behavior of individual insects. Scalability, flexibility and robustness are three main advantages for such swarm systems (Bonabeau et al. 1999).

Researchers motivated by such observations have tried to extract ideas, models and philosophies underlying natural swarm systems and apply them to artificial problems (e.g., optimization problems such as the notable traveling salesman problem (Dorigo and Gambardella 1997; Bonabeau et al. 2000)), and have had great success, even in business (Bonabeau and Meyer 2001).

When applying rules extracted from natural systems to artificial problems, the difference between the natural systems and artificial problems essentially requires different control parameters to be used. Learning, as an automatic way to adjust control parameters, is used to adapt rules to new problems and to improve the performance. Learning also serves as a way to adapt to a changing environment.

1.1 Swarm Systems

A social insect colony usually consists of millions of individuals. Though each individual has very limited ability, the whole colony is able to do many sophisticated jobs without a centralized control mechanism. In fact, social insects work autonomously, and their teamwork is essentially self-organized. Coordination between individuals arises from different interactions between insects or between insects and environment (the stigmergic mechanisms). Although these interactions might be primitive, as a whole they result in efficient solutions to difficult problems such as finding the shortest route to a food source among myriad possible paths.

The collective behavior that emerges from social insects (Bonabeau et al. 1999), as well as swarming, flocking, herding, and shoaling phenomena in vertebrates (Parrish and Hamner 1997), has been dubbed *swarm intelligence*.

Artificial swarm systems based on swarm intelligence consist of relatively simple autonomous agents. They are truly distributed, self-organized, and inherently scalable since there is no global control or communication mechanism. The agents are designed to be simple and interchangeable, and may be dynamically added or removed without explicit reorganization, making the collective system highly flexible and fault tolerant.

Swarm systems can be homogeneous or heterogeneous. A *homogeneous team* consists of physically identical agents with the same hardware and software capabilities. A *heterogeneous team* may differentiate in several ways: at the hardware level, at the (controller) software level, or simply because each agent has a different identifier. Heterogeneity can be hardwired or plastic, that is, a homogeneous team can become heterogeneous if the environmental constraints prompt this.

1.2 Distributed Learning

The learning issue in swam systems is how each agent can adapt its individual behavior (update its strategy) so that the whole system can "maximize" the overall per-

formance under a changing environment (including the change of number of agents). We call this *distributed learning* since it is the learning process happening at the individual level under partial information about the global performance. There are three main challenges specific to distributed learning:

- The environment that each agent can sense is only a small part of the overall system. Though the agents may have a complete view of the whole system by directly exchanging information, full communication is expensive in terms of both production cost and energy consumption, and is infeasible for systems consist of thousands or millions of agents. Thus, the learning of each agent is usually conducted with only *partial information*.

- Secondly, the environment, including the number of agents, may vary from time to time. Thus, the optimal strategy for an agent is not fixed. Each agent needs to adjust its behavior according to the *changing environment*. In addition, changes resulted from other robots can also require a strategy change for one robot. Thus a collective experiment is intrinsically highly dynamic.

- The third challenge is the so-called *credit assignment problem* (Versino and Gambardella 1997). Since the team performance is the result from all the agents, it is usually hard for an agent to know the impact of its strategy change on the overall performance. The situation is even worse when every agent changes its strategy from time to time.

Besides these challenges, the time-delayed reward which is common in reinforcement learning also brings difficulty to learning.

1.3 Overview

This thesis presents our work on distributed learning in swarm systems with a case study—the stick pulling experiment. All experiments in this thesis are carried out with a probabilistic model which faithfully simulates experiments with real robots.

Chapter 2 introduces the experiment as well as the probabilistic model. Our implementation of the probabilistic model and a faster way to systematically search for optimal control parameters are also presented.

Several learning algorithms, such as adaptive line search and Q-learning, and team performances by using the algorithms, are collected in Chapter 3. Since we can not get the optimal performances via learning, we investigate several issues related to learning in Chapter 4, such as the role of noise affecting the reinforcement in learning, and the contribution of learning on adaptability and optimality.

In Chapter 5 we try to measure the degree of specialization, which is a common feature in social societies that leads to better performance. Results from learning are compared with those from random tests to show that learning contributes something unique to the team diversity.

Chapter 2

The Stick Pulling Experiment

Martinoli and Mondada (1995) and successively Ijspeert et al. (2001) investigated collaboration in teams of reactive, non-communicating robots engaged in a stick pulling experiment. The swarm system they studied is relatively simple in that there is only one adjustable control parameter for each robot. The task pursued in their experiment requires the collaboration of two robots. It is the strictly collaborative nature that makes this experiment interesting for investigation with distributed learning.

The stick pulling experiment and models for analyzing the experiment at different levels are briefly described in this chapter. A systematic search was used by Ijspeert et al. to find the optimal parameters for homogeneous teams and very simple heterogeneous teams. We extend their method with early stopping in this chapter and apply it to fully heterogeneous teams.

2.1 Experimental Setup

In the stick pulling experiment, several robots equipped with gripper turrets and proximity sensors search a circular arena and pull sticks out of holes in the ground (Figure 2.1). The length of a stick has been chosen so that a single robot is not capable of pulling a stick out of the ground on its own. Collaboration between two robots is therefore necessary for completing this task. Each robot is characterized by a *gripping time parameter* (GTP), which is the maximal length of time that a robot waits for the help of another robot while holding a stick.

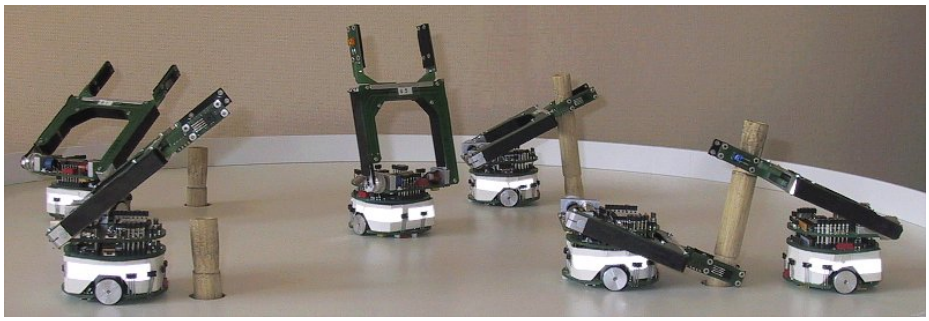


Figure 2.1: Physical set-up for the stick pulling experiment. Collaboration between two robots is necessary to pull a stick out of the ground.

The behavior of a robot is determined by a simple hand-coded program (Figure 2.2). The default behavior is searching for sticks, that is, wandering in the arena in a straight line until an object is detected by the frontal proximity sensors. If a stick is detected in the subsequent distinguishing procedure, the robot backtracks a few centimeters, grips the stick and pulls it up. During pulling, the robot can determine whether another robot is already gripping the same stick by measuring the speed of elevation of the gripper arm. If the elevation is fast, the robot assumes no other robot is holding the stick and we call such a grip a *grip1*. Otherwise the robot assumes that another robot is already holding that stick and therefore "braking" the elevation. Such a grip is called a *grip2*.

After a robot makes a grip1, two cases can occur: either a second robot helps the first one (we define this as a *successful collaboration*) or the GTP expires before any other robot can help and the first robot resumes the search for sticks in the arena. The specific values of GTPs play a crucial role in the overall *collaboration rate* (defined as the number of successful collaborations per unit time), which is the metric adopted in both previous papers (Martinoli and Mondada 1995; Ijspeert et al. 2001) and this one for measuring the team performance. To ensure the collaboration rate is reliably measured, experiments usually take a long time and a stick will be inserted back into the hole (by the experimenter) after it is completely pulled out by robots.

In (Ijspeert et al. 2001), teams of two to six Khepera robots were used in a circular arena (80 cm in diameter) delimited by a white wall. Four holes situated at

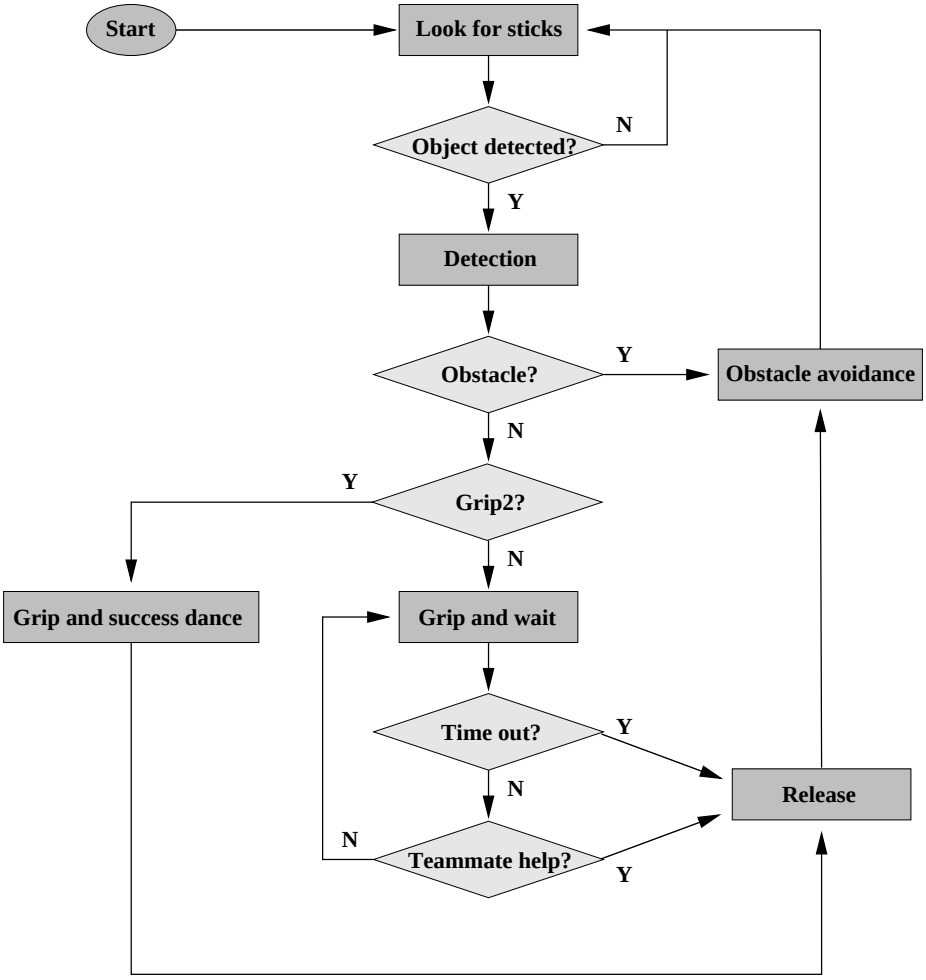


Figure 2.2: Flowchart of the robots' controller.

the corners of a square with 30 cm edges contained sticks (15 cm in length and 1.6 cm in diameter) which, in their lowest position, sticked 5 cm out of the ground. The same setting is used in this thesis unless noted otherwise.

On average, the collaboration rate is a function of the environmental setting (including the arena size, the number of sticks, and usually the number of robots) and the *GTP set* (which is the set of GTPs of all robots in the arena). When the environment is fixed, the team performance is a function of the GTP set. Note that when the team is homogeneous, the GTP set is determined by a single scalar.

2.2 Microscopic and Macroscopic Models

One of the main difficulties in designing efficient robotic teams is to predict how the team performance is affected by the hardware and software characteristics of the constituent individuals. This is particularly difficult for a large number of robots controlled in a fully distributed way. As experimenting with real robots is expensive and time-consuming, embodied or sensor-based simulators that simulate as realistically as possible the behavioral characteristics of the robots and the environment are usually helpful in, for instance, determining the optimal number of robots or the optimal control parameters for a robotic team, though these types of simulators still suffer from the long time needed for simulation.

Martinoli et al. (1999b; 1999a) introduced a novel microscopic probabilistic model which describes the experiment as a series of stochastic events. Ijspeert et al. extended the model to the stick pulling problem. An event in the probabilistic model corresponds to some diamond in Figure 2.2 and a state corresponds to some rectangle. For example, "object detected" is an event transiting a robot from the "look for sticks" state to the "detection" state. The probabilistic nature of the stick pulling experiment is captured by probabilistic events with probabilities based on simple geometrical considerations and systematic experiments with one or two real robots.

Since the model does not compute the details of the robots' trajectories and sensory information, it has been proven to be four or five orders of magnitude faster than real robot experiments. Experiments with real robots and Webots[i] also showed that this model is able to deliver both qualitatively and quantitatively accurate predictions (Ijspeert et al. 2001).

The solid curves in Figure 2.3 illustrate the collaboration rate of a homogeneous team in the stick pulling experiment generated from the probabilistic model. The team exhibits quite different behaviors depending on the ratio between the number of robots and the number of sticks. When there are more robots than sticks, the collaboration rate monotonically increases with the GTP, until it reaches a plateau

[i]Webots is a sensor-based, embodied simulator (Michel 1998) and was also adopted in (Ijspeert et al. 2001) to investigate the stick pulling experiment.

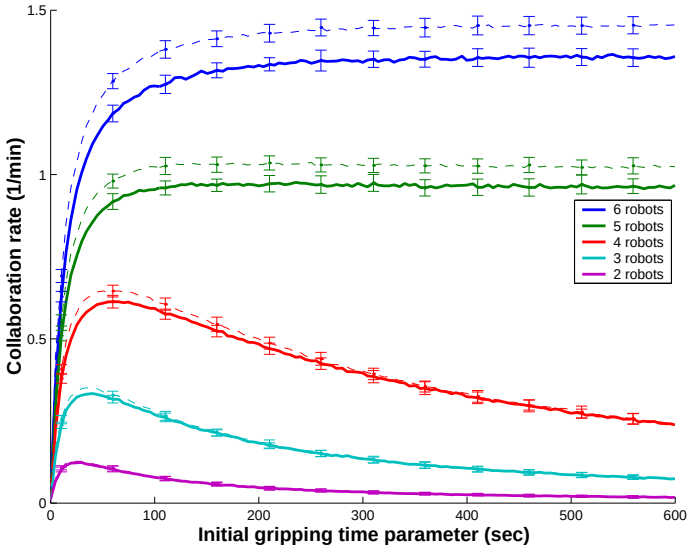


Figure 2.3: Collaboration rate as a function of GTP in homogeneous teams. Results from two different ways of handling stochastic events are shown. The solid curves are results from the event aligning technique and the dashed curves from the sorted event list technique. 100 simulations, each taking 1600 min, are run for every initial GTP from the set $\{5k\}_{k=0}^{120}$.

corresponding to the optimal collaboration rate for a homogeneous team. In other words, since there will always be some robot "free" to help, waiting a very long time is a good strategy for robots gripping sticks. However, when the number of robots is less than or equal to the number of sticks, waiting a very long time may incur "deadlock" (every robot holding a different stick and waiting for help but no one having the chance to help others) and thus becomes a very bad strategy. The optimal collaboration rate occurs at some small GTPs below 100 sec.

Lerman et al. (2001) presented a macroscopic analytical model of collaboration for the stick pulling experiment. They used a simplified state diagram including different states in a collaboration and used differential equations to describe the dynamics of the system (e.g., number of robots in different states). They reproduced the main qualitative conclusions of (Ijspeert et al. 2001).

Though analytical results seem inspiring, the macroscopic models have two major drawbacks that prevent them from being used for more complicated teams, such as fully heterogeneous teams. First, quantitatively correct macroscopic models based

uniquely on features of the individual agents are not always trivial to devise, in particular when agent-to-agent and agent-to-environment interactions are more complicated than simple elastic bounces. Second, these models intrinsically imply homogeneity by assuming that a certain number of agents can be clustered in a caste which in turn is represented by a set of differential or difference equations. If a heterogeneous team is treated as a collection of several homogeneous castes, the number of equations needed by a fully heterogenous team prohibits further analysis.

2.2.1 Handling Stochastic Events

To handle the stochastic events in the probabilistic model, the program Ijspeert et al. used checks every T time[ii] what events take place and then simulates them. In this way, every event is aligned to a time grid kT where k is some integer, which also implies that the time a robot stays in any state would be rounded to some multiple of T. For states lasting much longer than T, rounding does not have much impact to the simulated performance. However, when the time a robot spends in some state is comparable to T, the error caused by rounding may accumulate to significant error in the simulated team performance.

A better way to handle the stochastic events is to maintain an event list sorted ascendingly according to event time. At each iteration, the first event in the list is handled and new events caused by or affected by the current event are then inserted to the list or updated. Figure 2.3 shows the slightly different team performances from these two simulation techniques. The dashed curves generated from the sorted event list technique are a little higher than the solid ones generated by event aligning.

After a careful examination of Figure 7 in (Ijspeert et al. 2001), we found that, for two to four robots, the collaboration rate calculated by the probabilistic model is a little higher than that by the Webots simulator, while for six robots, the former is a little lower. Should the sorted event list technique be used, the collaboration rate would be consistently a little higher than that from the Webots. Remember that

[ii] T is the time a robot needs to patrol the smallest detection area of an object in the arena, which in this case study is the area to detect a stick.

parameters in the probabilistic model are determined based on simple geometrical considerations and systematic experiments with one or two real robots. We believe that after re-determining some parameters, the probabilistic model with the new technique will agree better with the Webots.

However, we keep using the event aligning technique for two reasons. First, adopting the same technique as (Ijspeert et al. 2001) makes the comparison of our work with their work easier. Second, the effect of learning should be independent from simulation techniques. If we observe performance promoted by learning with one technique, as long as the simulation model reflects the correct characteristics of the experiment, the same observation should also be seen with another simulation technique.

2.3 Systematic Search with Early Stopping

In Chapter 3, we will present distributed learning algorithms which are able to find near-optimal solutions with only local perception and adaptation. In order to compare the learned performance with the optimal one, we are interested in finding the *optimal GTP sets* that maximize the collaboration rate under a given environment. Note that this is different from the task of learning since every detail about the environment is invariant and known when we search for optimal GTP sets.

With the probabilistic model, we can systematically carry out simulations with different GTP sets and then find the best one. Depending on the size of the search space and the number of robots, such systematic investigation of the optimal GTP sets could be prohibitively time-consuming. Ijspeert et al. only investigated teams with one or two different GTPs, i.e., homogeneous teams and heterogeneous teams with two GTPs. For the heterogeneous case, they used 10 predefined GTPs to confine the search to a relatively small space. Their results showed that heterogeneity in GTPs could increase the collaboration rate.

We use more predefined GTPs in the systematic search and obtain optimal GTP sets for teams consisting of two to six robots. In the following text, we discuss the early stopping technique used in our systematic search.

2.3.1 Early Stopping

During the systematic search, every possible GTP set whose GTP values are from a predefined set $\mathcal{G}_s = \{5k\}_{k=1}^{20} \cup \{10k\}_{k=11}^{15} \cup \{175, 200, 250, 300, 400, 500, 750, 1000\}$ is tested. For each GTP set, we run 100 simulations and then calculate the mean and the standard deviation of the collaboration rate. The time needed for simulation is roughly

$$\binom{|\mathcal{G}_s| + n - 1}{n} R c_n T_s \ , \tag{2.1}$$

where $|\mathcal{G}_s| = 43$ is the number of possible choices for GTP, n is the number of robots, $R = 100$ is the number of simulations for each GTP set, T_s is the experiment time we want to simulate, and c_n is the ratio between the time used for simulation and the time simulated. Note that c_n depends on both the computing resource used and the environmental setting simulated, especially n.

It is tolerable to run all the simulations for two to four robots. For $n = 4$, $T_s = 160\,\text{min}$, the running time is roughly 7 hours using a 1G Hz Pentium machine. However, the running time is too long when $n > 4$. Though it is possible to decrease R or T_s, the stochastic nature of the experiment requires not-too-small R and T_s in order to estimate the average collaboration rate with some accuracy.

Another way to work around the problem is to decrease R for some GTP sets—stop investigating a GTP set as soon as statistically significant evidence has been collected to show that the set is not likely to be optimal. One intuitive way to implement this idea is as follows. During the systematic search, we keep recording the mean μ and standard deviation σ of the to-date best performance. If during the simulation of the current GTP set, at least 3 runs get a performance less than $\mu - 2\sigma$, we can reject this set. This idea is further formalized in Algorithm 2.1. For the intuitive way discussed above, $m = 3$ and $x = 2$ in the algorithm.

2.3.2 Probability of Wrong Rejection

Since a GTP set may be rejected earlier than it is fully tested for R runs, a natural question is how reliable such rejection is. That is, with what probability will a GTP

Algorithm 2.1 Systematic search with early stopping.

Parameter: R is the number of runs for each GTP set, x is a real number, and m is an integer.

Variable: G is the to-date best GTP set, and μ is the mean and σ is the standard deviation of the performance of G.

1. $\mu \leftarrow 0$, $\sigma \leftarrow 0$.
2. For each GTP set $\hat{G}$, do
   (a) Simulate until all R runs are done, or m runs get performance less than $\mu - x\sigma$.
   (b) If m runs get performance less than $\mu - x\sigma$, continue to the next GTP set.
   (c) Calculate the average performance $\hat{\mu}$ and standard deviation $\hat{\sigma}$.
   (d) If $\hat{u} > \mu$, do $\mu \leftarrow \hat{\mu}$, $\sigma \leftarrow \hat{\sigma}$, $G \leftarrow \hat{G}$.
3. Return μ and σ as the optimal performance and its standard deviation, and G as the optimal GTP set.

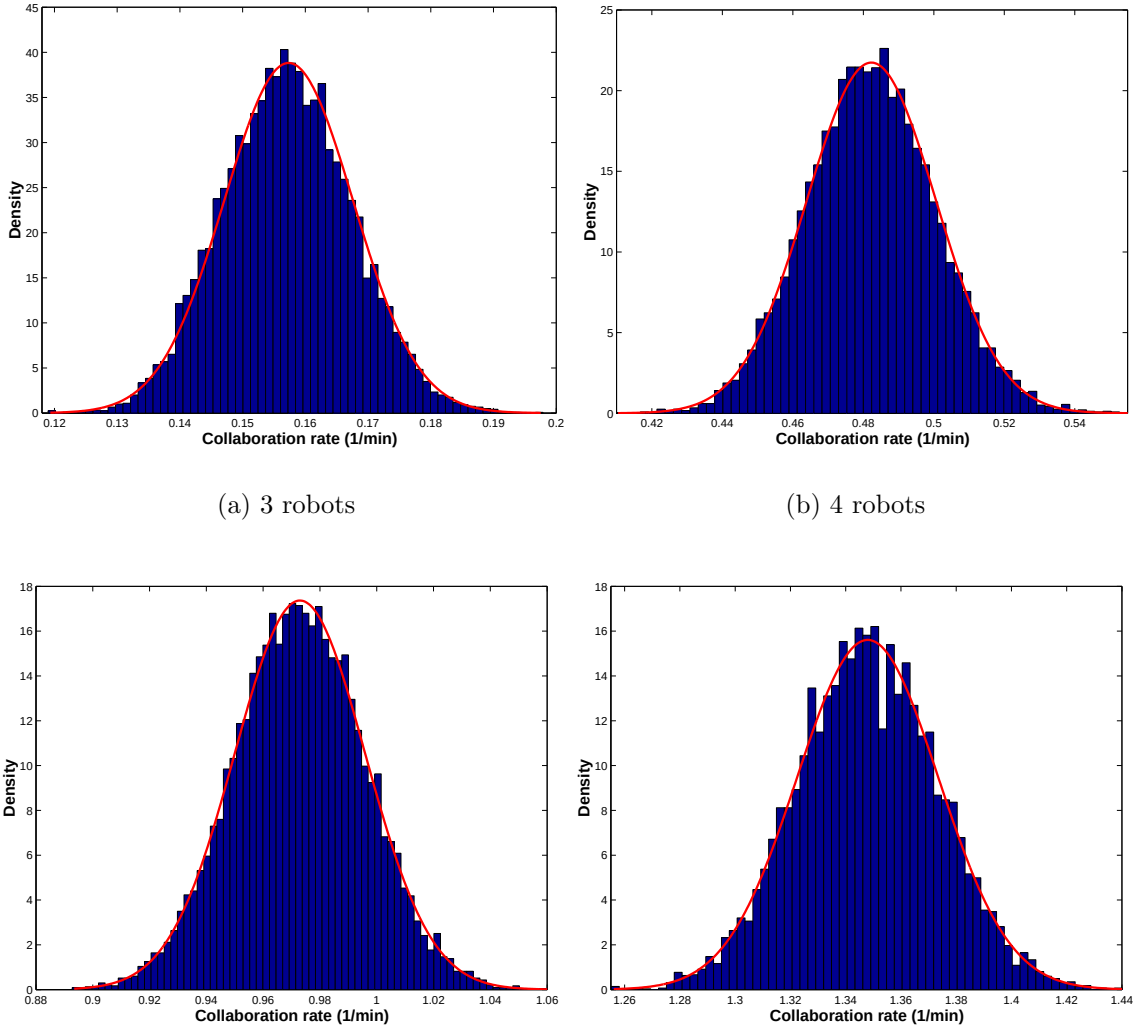
set whose collaboration rate is better than the to-date best one be rejected?

To answer this question, we should first know the possible distribution of collaboration rate. Figure 2.4 shows four typical histograms of team performances with fixed GTP sets, together with curves representing Gaussian distributions with mean and variance calculated from the performance distributions. We believe the performance with a fixed GTP set has Gaussian distribution.

Assume the performance of GTP set currently under testing also has standard deviation σ (a not-always-true but reasonable assumption). We want to test the hypothesis $\mathcal{H}$ that the mean performance of the current GTP set is at least μ. Under hypothesis $\mathcal{H}$, the probability that a single run has a collaboration rate less than $\mu - x\sigma$ is

$$e_1(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt \ . \tag{2.2}$$

Since a rejection is made if and only if at least m out of R runs have performances less than $\mu - x\sigma$, the probability that the the set is rejected when hypothesis $\mathcal{H}$ holds



(a) 3 robots

(b) 4 robots

(c) 5 robots

(d) 6 robots

Figure 2.4: Histograms of collaboration rate for fixed GTP sets. For each fixed GTP set, 10000 simulations are run to get the histogram of performance, and the bell-shaped curve is the Gaussian distribution with mean and variance calculated from the simulation performance. (a) 3 robots with GTP set $\{200, 300, 400\}$ (in seconds); (b) 4 homogeneous robots with GTP 200 sec; (c) 5 robots, with GTP set $\{100, 200, 300, 400, 500\}$ (in seconds); (d) 6 homogeneous robots with GTP 300 sec.

(thus a wrong rejection) is

$$e_{R,m}(x) = 1 - \sum_{i=0}^{m-1} \binom{R}{i} (1 - e_1(x))^{R-i} e_1(x)^i \ . \tag{2.3}$$

For $R = 100$, $m = 3$, and $x = 2$, we have $e_{100,3}(2) \approx 40\%$, which seems a little high. However, since many GTP sets have performances very close to the optimal one—for example, for a team of 4 robots, the 10^{th} best performance is within 99.3% of the optimal one—using $(m = 3, x = 2)$ is fine if we think that getting a near-optimal GTP set is acceptable.[iii] The relatively large standard deviation in performance also justifies the acceptance of a near-optimal GTP set.

A more conservative setting $(m = 3, x = 2.4)$ gives $e_{100,3}(2.4) < 4.96\%$, which takes much longer running time. Compared with the systematic search without early stopping, the speed-up is about a factor of 10.

2.3.3 Optimal GTP Sets

Table 2.1 lists the optimal GTP sets we get by using Algorithm 2.1 with $m = 3$ and

Team size	Optimal GTP set	Performance
2	$\{5, 750\}$	0.1942 ± 0.0282
3	$\{5, 250, 500\}$	0.4206 ± 0.0491
4	$\{5, 110, 750, 750\}$	0.6811 ± 0.0610
5	$\{35, 100, 300, 750, 750\}$	1.0018 ± 0.0675
6	$\{400, 400, 400, 500, 500, 1000\}$	1.3777 ± 0.0792

Table 2.1: Optimal GTP sets gotten from the systematic search with early stopping. $R = 100$, $m = 3$, and $x = 2.4$ in Algorithm 2.1.

$x = 2.4$. In Figure 2.5, we compare the optimal performances with homogeneous teams, heterogeneous teams with two GTPs, and fully heterogeneous teams. Since homogeneity is a special case of heterogeneity, it is expected that the optimal heterogeneous performance is always better than the optimal homogeneous performance. However, fully heterogeneity does not have much advantage over the simple heterogeneous case with two GTPs, though the former is consistently better. The reason may

[iii] The probability that the top 10 best GTP sets are all rejected is less than 1%.

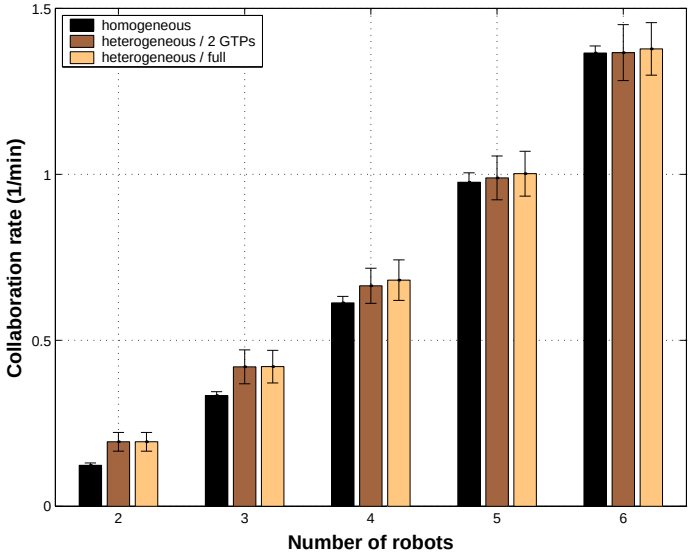


Figure 2.5: Optimal performances of homogeneous and heterogeneous teams. The optimal performances of homogeneous teams are calculated by the original systematic search and those of heterogeneous teams are calculated by the systematic search with early stopping.

be intrinsically rooted in the task constraint, i.e., the requirement of the collaboration between two robots.

If in order to pull a stick, three robots are required (e.g., longer sticks are used), optimal teams may consists three different types of robots: (1) robots with large GTPs are specialized for initiating the grip, (2) robots with median GTPs specialized for helping the first robots, and (3) robots with small GTPs specialized for completing grips. Under this task constraint, a fully heterogeneous team may be much better than a team with only two GTPs.

Chapter 3

Learning Methods

One major advantage of systems consisting of multiple agents over those represented by a single agent is their flexibility to allocate in time and space different numbers of individuals to a given task or several sub-tasks (Bonabeau et al. 1999). However, sometimes this characteristic is not enough to generate an optimal collective response under certain environmental conditions. By combining collective flexibility with individual adaptivity we can further enhance the robustness of the collective behavior and obtain high system efficiency under a wider spectrum of environmental conditions. In other words, since the parameter "number of individuals" of a swarm system could be nonlinearly correlated with parameters characterizing the individual behavior, being flexible in both parameter spaces allows us to optimize the overall performance of the team, or optimize one dimension after having satisfied constraints on the other.

Ideally, we would like to have rules implemented on an individual agent, exclusively based on its local perception, that evaluate how individual behavioral parameters should change as a function of, for instance, the density of teammates. However, designing such rules from scratch is difficult and usually requires at least a previous systematic study in simulation in order to understand the macroscopic dependence of the system dynamics on the microscopic changes. Machine learning methods represent an effective alternative for finding out these rules or their parameters.

In this chapter, we will look at several learning algorithms and their effects on GTP sets and team performances. The procedure used to assess a learning algorithm consists of a training phase and a test phase. At the beginning of the training phase,

the team is homogeneous and initialized with an *initial GTP*. The learning algorithm is then carried out with the simulation going on and changes the GTP of every robot. After the 1600 min (in simulation time) training phase, the GTP set is fixed and a test phase with the same length as the training phase is used to calculate the performance of the learned GTP set, which we call the *learned performance*. This procedure is repeated 100 times for each initial GTP from the set $\{5k\}_{k=0}^{120}$. The average learned performance is compared to the performance without learning—the performance with the initial GTP fixed.

The basic scenario we study is individual learning without communication. Each robot adapts its GTP autonomously using the *local reinforcement*, which is the rate of its successful collaborations (regardless of whether a robot was the first or the second in gripping the stick).

3.1 Memoryless Adaptation

Intuitively, if a robot times out during a grip1, that is, no other robots come to help, it should increase its GTP so that next time it will wait longer. In contrast, if a robot gets the help from some other robot and completes a successful grip, it may decrease its GTP since the current value may be longer than necessary. Based on this idea of finding "proper GTPs" from the experience of timeout, we try the algorithm in Algorithm 3.1. Note that all constants in the algorithm are ad hoc. Since any adaptations made have nothing to do with the history, we name this method *memoryless adaptation*.

Algorithm 3.1 Memoryless adaptation. GTP is updated after a time-out or a successful collaboration.

After grip1, two outcomes may happen:

1. If the GTP expires, GTP $\leftarrow 1.3\text{GTP} + 1$;
2. If some other robot comes to help, calculate the time w spent on waiting and GTP $\leftarrow \max\{0.75\text{GTP}, 1.2w, 5\}$;

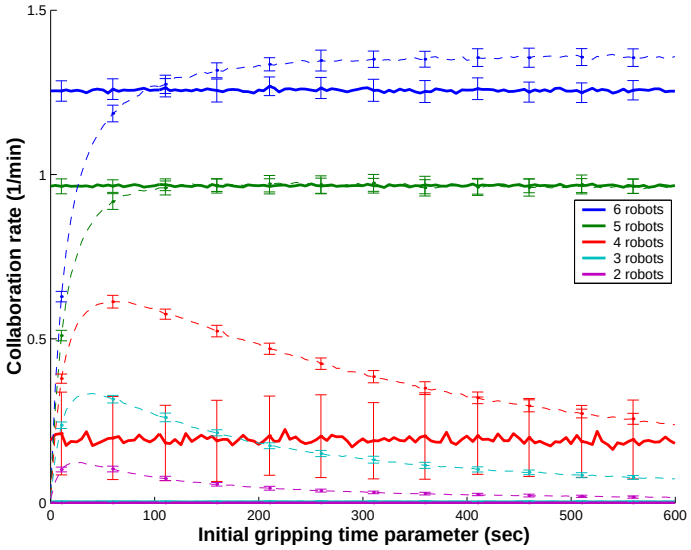
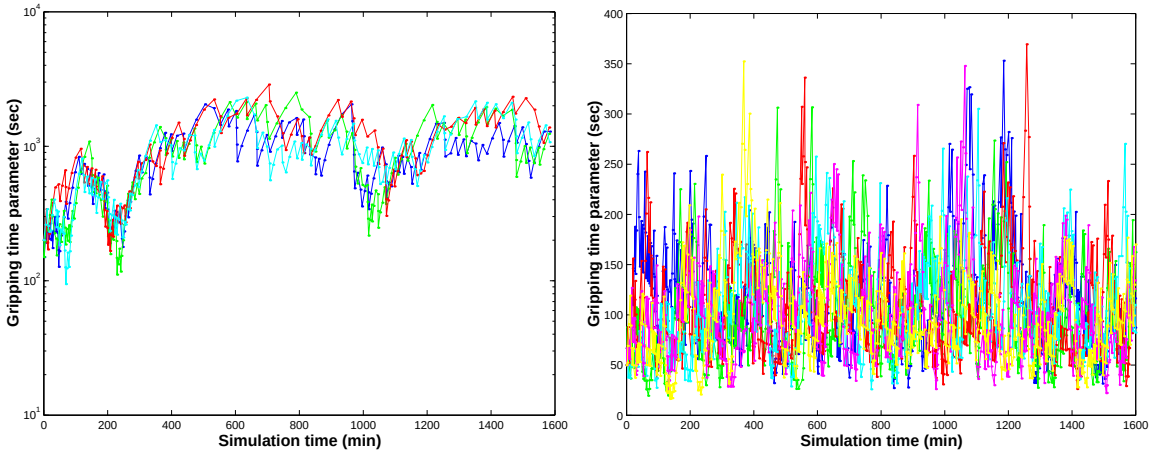


Figure 3.1: Performances with the memoryless adaptation algorithm. Solid curves are learned performances in the test phase while adaptation is used in the training phase. Dashed curves are performances of homogeneous teams with fixed GTPs. Note that the solid curves for two and three robots almost overlap with the GTP axis.

The collaboration rates after learning are drawn in Figure 3.1. As we could tell, this naive algorithm suffers from several aspects. First, the strategy does not connect GTP sets directly to team performances. For example, when the number of robots is less than or equal to the number of sticks, large GTPs are more likely to cause deadlock and should be avoided. Meanwhile, since robots do not "care" about the performance, they tend to increase their GTPs when a time-out happens, which under this situation increases the possibility of time-out. Thus GTPs are getting larger and larger and the collaboration rate becomes very small. Second, the GTPs change all the time and do not converge to some "optimal" set. Figure 3.2 depicts the GTP curves for a single run of 4 robots and another single run of 6 robots. GTPs oscillate violently, and for the team of 4 robots, they seem to be homogeneous.

As a second attempt, we consider a completely different strategy. A time-out means lack of robots helping others, which should lead to a decrease of GTP, i.e., the robot decides to help others; a successful grip means there are enough robots helping others, and the GTP should be increased. Both this strategy and the previous one could be correct under some situations. However, the lack of memory (some historical



(a) 4 robots, 200 sec

(b) 6 robots, 50 sec

Figure 3.2: GTP curves with the memoryless adaptation algorithm. Robots are initially homogeneous. The curves are from single runs with (a) 4 robots with initial GTP 200 sec, and (b) 6 robots with initial GTP 50 sec.

information) prevents a robot from choosing the right strategy.

3.2 Adaptive Line Search

For a single robot, its GTP is the only parameter and the searching for a good GTP is one-dimensional. If we assume the environment and the GTPs of the other robots are fixed, the team performance becomes a one-input function and the optimal performance could be determined by searching in that dimension.

Note that the assumption we just made is unrealistic.[i] Integrated with learning, the other robots also update their GTPs at their own paces. However, if we assume the performance function changes gradually with time and take the line search as a dynamic process, the line search technique could still be used to find the optimal GTP. That is, we need a mechanism to "forget" the outdated performance data and "reset" the search direction.

The learning principle we propose in this section is very similar to what a human

[i]If homogeneity is forced via an external supervisor or global communication, the whole team is represented by one GTP. Thus the search is still one-dimensional. However, the assumption does not hold in general.

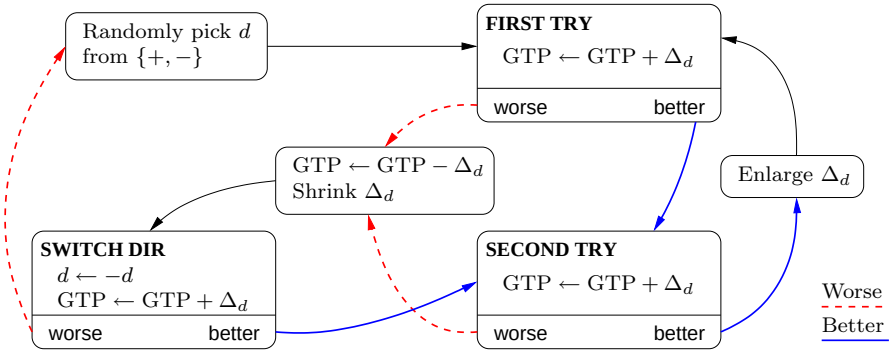


Figure 3.3: State diagram of the adaptive line search algorithm. GTP is adjusted according to the current state and the performance change (better or worse). Δ_d is the search step in direction d.

being would do in a partially (locally) known environment (Figure 3.3). The robot first tries to change its GTP in a randomly chosen direction (either increase it or decrease it). After the change, the robot keeps that GTP for a small period of time (default 10 min), which we call the *evaluation time*, and monitors the performance improvement. If the improvement is positive, the robot will continue in that direction; if negative, the robot will undo the last change and try the other direction.

In addition to adapting the GTP, the search step (Δ_d in the graph) also varies. When the same direction has been selected twice, the search step for that direction is increased in order to speed up search in that direction. When the performance observed oscillates, which implies that the current GTP is near the optimal one, the search step is accordingly decreased to stabilize the performance. Thus this method is called *adaptive line search*.

With slightly different updating rules, the three methods below are tested.

3.2.1 Δ-method

By adding a few details into Figure 3.3, we obtain the Δ-*method* described in Algorithm 3.2. The algorithm gets its name from the fact that, after each evaluation period, the GTP is changed by adding or subtracting some amount Δ_d. The inclusion of an evaluation period between states requires some adjustments to the algorithm, which explains the minor difference between Figure 3.3 and Algorithm 3.2.

Algorithm 3.2 Δ-method adaptive line search. Note that there is an evaluation period between each state so the algorithm is a little different from the state diagram in Figure 3.3.

Input: P_1 is the performance of this period.

Parameter: $\Delta_{\min} = 2\,\text{sec}$, $\Delta_{\max} = 60\,\text{sec}$, some coefficients such as 1.9.

Variable: State S, performance of last period P_0, step sizes Δ_+ and Δ_-, direction $d \in \{+, -\}$, number of successful GTP changes n.

The algorithm starts from the **INIT** state:

INIT: $\Delta_+ \leftarrow \Delta_{\max}$, $\Delta_- \leftarrow -\Delta_{\max}$, and $S \leftarrow$**SELECT**.

SELECT: Randomly select d from $\{+, -\}$; set GTP $\leftarrow$ GTP$+\Delta_d$, $P_0 \leftarrow P_1$, $n \leftarrow 0$, and $S \leftarrow$**TRY**.

TRY: If $P_1 > P_0$, $n \leftarrow n + 1$. If $n \geq 2$, set $\Delta_d \leftarrow 1.9\Delta_d$ and $n \leftarrow 0$. Set GTP $\leftarrow$ GTP$+\Delta_d$, $P_0 \leftarrow P_1$, and $S \leftarrow$**TRY**. Otherwise if $P_1 \leq P_0$, set GTP $\leftarrow$ GTP $- \Delta_d$, $\Delta_d \leftarrow 0.5\Delta_d$, $d \leftarrow -d$, GTP $\leftarrow$ GTP $+ \Delta_d$, and $S \leftarrow$**SWITCH**.

SWITCH: If $P_1 > P_0$, set GTP $\leftarrow$ GTP $+ \Delta_d$, $P_0 \leftarrow P_1$, and $S \leftarrow$**TRY**. Otherwise set GTP $\leftarrow$ GTP $- \Delta_d$, $\Delta_d \leftarrow 0.5\Delta_d$, and $S \leftarrow$**SELECT**.

The Δ_+ is always confined to $[\Delta_{\min}, \Delta_{\max}]$ and Δ_- is confined to $[-\Delta_{\max}, -\Delta_{\min}]$; the GTP is forced to be larger than or equal to 5.

The ratios in enlarging the step size (1.9) and shrinking the step size (0.5) are chosen in an ad hoc manner. We deliberately choose 1.9 so that one enlargement and one reduction will not return to the original search step but a little smaller ($1.9 \times 0.5 < 1$). This leads to a more conservative behavior.

Figure 3.4 shows that for some initial GTPs, the performance improves after learning, and for the other portion of initial GTPs, learning does not help much. Figure 3.5 gives typical GTP curves of two single runs. Though starting from a homogeneous team, robots adjust their GTPs and improve the performance with learning, and finally "stabilize" at usually different GTPs and form a heterogeneous team. This is consistent with (Ijspeert et al. 2001) where experiments showed that, when the number of robots is no more than the number of sticks, specialization should help. More about the specialization issue will be discussed in Chapter 5.

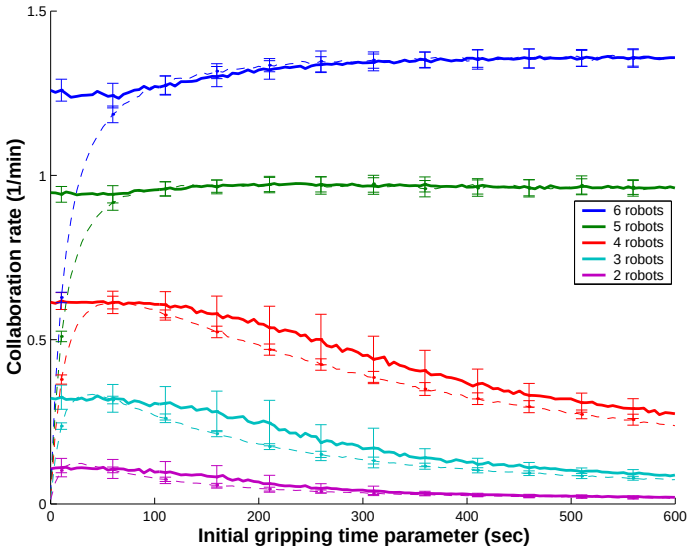
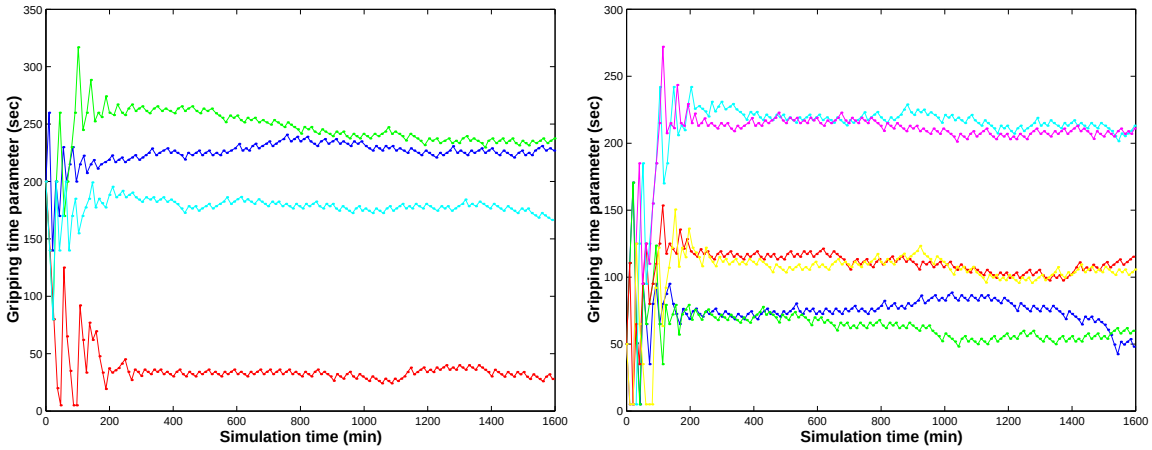


Figure 3.4: Performances with the Δ-method. Solid curves are learned performances with the Δ-method, and dashed curves are performances of homogeneous teams without adaptation.



(a) 4 robots, 200 sec

(b) 6 robots, 50 sec

Figure 3.5: GTP curves with the Δ-method. Robots are initially homogeneous. (a) 4 robots with initial GTP 200 sec; (b) 6 robots with initial GTP 50 sec.

3.2.2 %-method

In the diagram for the Δ-method, the change of GTP is linear. That is, after k updates, the change of GTP is no more than $k\Delta_{\max}$. However, as demonstrated in Figure 2.3, the collaboration rate is much more sensitive to small changes in the GTP when the GTP is "small" than when it is "large," though "small" and "large" are relatively vague here. Therefore, choosing the search step proportional to the absolute value of the GTP may produce a more effective search in the GTP space, particularly when the GTPs are large. This idea becomes the so-called %-*method* as described in Algorithm 3.3.

Figure 3.6 gives a much better performance compared with the result from the Δ-method. The GTP curves in Figure 3.7 show that though some GTPs still remain in the "small" range around the initial GTP, some are higher than 1000 sec.

Experiments with real robots may suggest not to use so large GTPs since waiting

---

Algorithm 3.3 %-method adaptive line search.

Input: P_1 is the performance of this period.

Parameter: $r^+_{\min} = 1.1$, $r^+_{\max} = 5$, $r^-_{\min} = 0.9$, $r^-_{\max} = 0.2$, some coefficients.

Variable: State S, performance of last period P_0, step ratios r_+ and r_-, direction $d \in \{+, -\}$, number of successful GTP changes n.

The algorithm starts from the **INIT** state:

INIT: $r_+ \leftarrow r^+_{\max}$, $r_- \leftarrow r^-_{\max}$, and $S \leftarrow$**SELECT**.

SELECT: Randomly select d from $\{+, -\}$; set GTP $\leftarrow$ GTP $\cdot\, r_d$, $P_0 \leftarrow P_1$, $n \leftarrow 0$, and $S \leftarrow$**TRY**.

TRY: If $P_1 > P_0$, $n \leftarrow n + 1$. If $n \geq 2$, set $r_d \leftarrow r_d + 0.3\,(r_d - 1)$ and $n \leftarrow 0$. Set GTP $\leftarrow$ GTP $\cdot\, r_d$, $P_0 \leftarrow P_1$, and $S \leftarrow$**TRY**. Otherwise if $P_1 \leq P_0$, set GTP $\leftarrow$ GTP$/r_d$, $r_d \leftarrow r_d - 0.5\,(r_d - 1)$, $d \leftarrow -d$, GTP $\leftarrow$ GTP $\cdot\, r_d$, and $S \leftarrow$**SWITCH**.

SWITCH: If $P_1 > P_0$, set GTP $\leftarrow$ GTP $\cdot\, r_d$, $P_0 \leftarrow P_1$, and $S \leftarrow$**TRY**. Otherwise set GTP $\leftarrow$ GTP$/r_d$, $r_d \leftarrow r_d - 0.5\,(r_d - 1)$, and $S \leftarrow$**SELECT**.

The r_+ is always confined to $\left[r^+_{\min}, r^+_{\max}\right]$ and r_- is confined to $\left[r^-_{\max}, r^-_{\min}\right]$; the GTP is forced to be larger than or equal to 5.

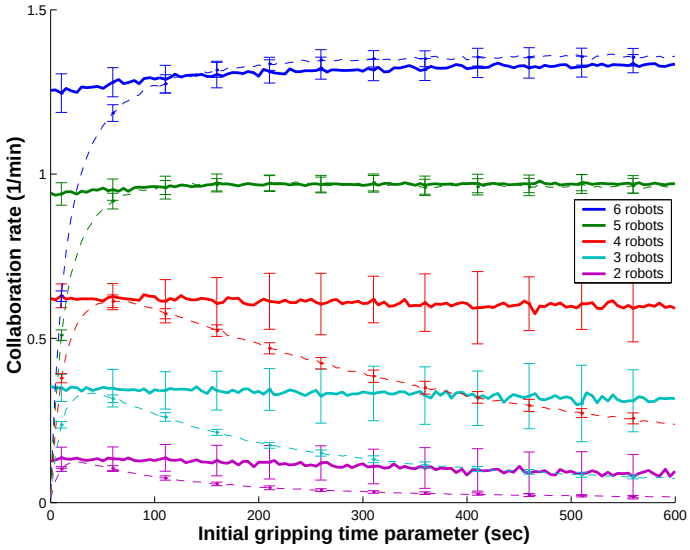
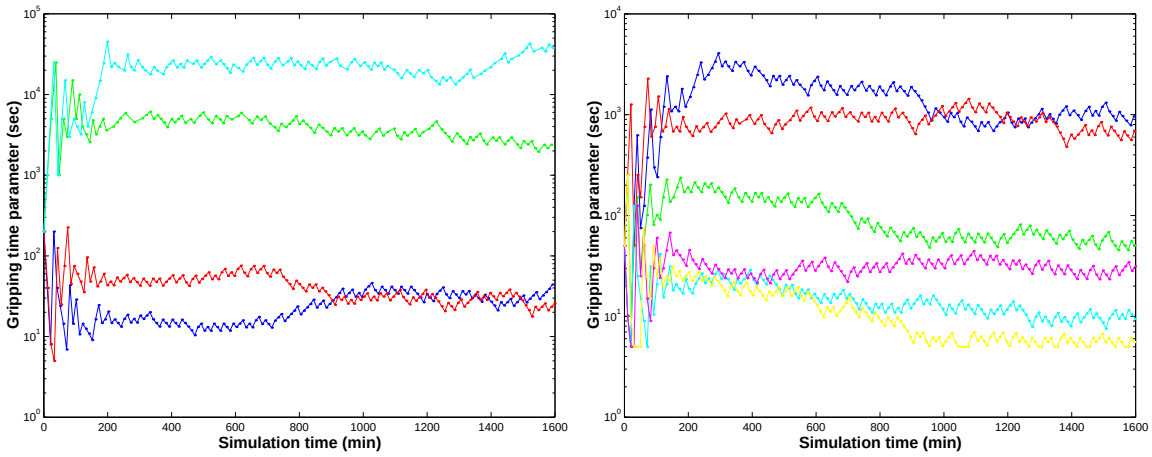


Figure 3.6: Performances with the %-method. Solid curves are learned performances with the %-method, and dashed curves are performances of homogeneous teams without adaptation.



(a) 4 robots, 200 sec

(b) 6 robots, 50 sec

Figure 3.7: GTP curves with the %-method. Robots are initially homogeneous. (a) 4 robots with initial GTP 200 sec; (b) 6 robots with initial GTP 50 sec.

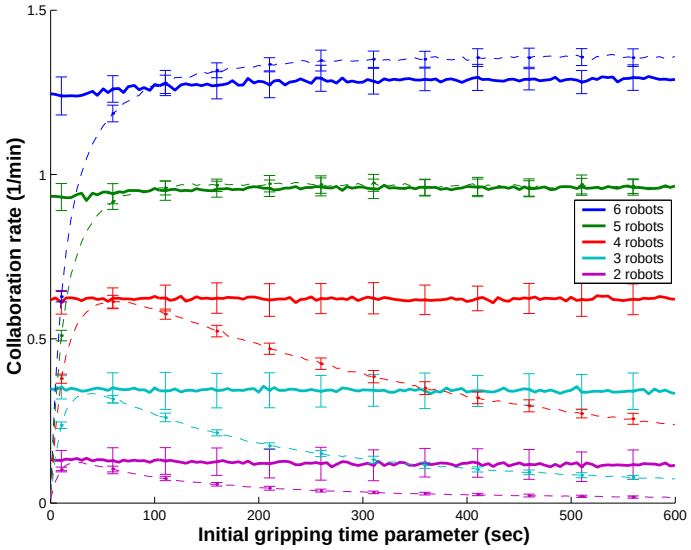


Figure 3.8: Performances with the %-method and the 1000 sec upper limit. Compared with Figure 3.6, the performance is improved (and the error bar is smaller) when there are no more robots than sticks, and is worse when the robots are more than the sticks.

such a long time is intolerable. To validate this in the probabilistic model, we add the 1000 sec upper limit of GTP to the algorithm in Algorithm 3.3 and obtain the results of Figure 3.8. Compared with the results from the one without upper limit, the performance is improved and the error bar is smaller when there are no more robots than sticks; and the performance is worse when there are more robots than sticks. These results imply that when the number of robots is larger than the number of sticks, very large GTPs are preferred even if they seem intolerable; on the other hand, when the number of robots is less than or equal to the number of sticks, bounded (thus relatively small) GTPs are favored. The underlying reason could be found in the explanation of Figure 2.3.

However, such properties would not be known beforehand, especially when the environment is complex. So we pretend that we do not know this effect, and will not apply the 1000 sec upper limit in future simulations.

3.2.3 Mix-method

Comparing Figure 3.4 with Figure 3.6, we can see that the Δ-method works well when the initial GTP is small, and the %-method works better when the initial GTP is large. A simple combination of these two methods (which we called the *mix-method*) yields the overall best performance (see Figure 3.9 and Figure 3.10). In the mix-method, each time the %-method and the Δ-method are carried out sequentially.

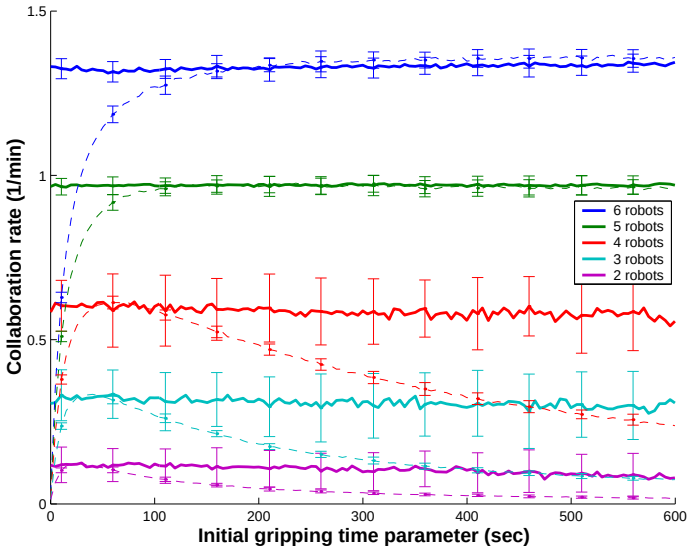
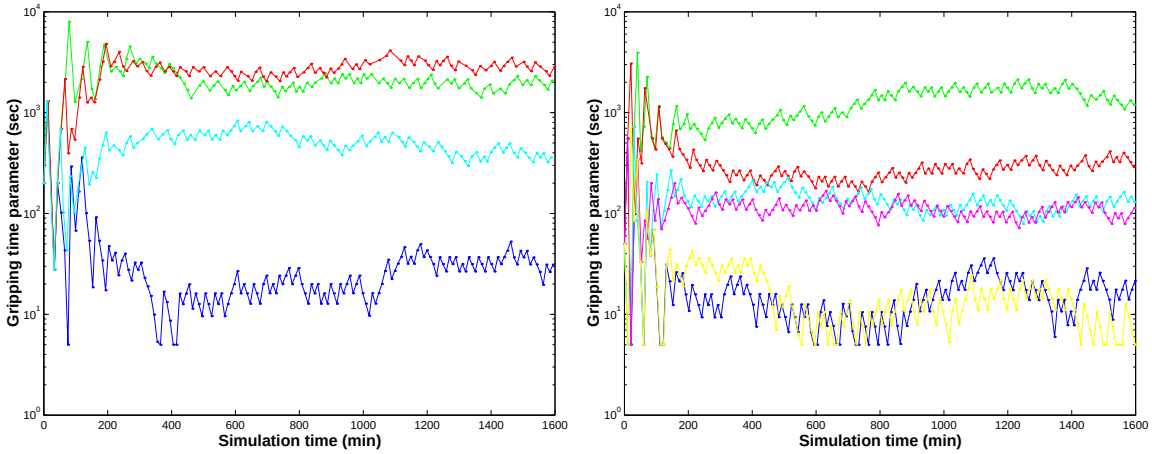


Figure 3.9: Performances with the mix-method. Solid curves are learned performances with the mix-method, and dashed curves are performances of homogeneous teams without adaptation.

3.3 Q-Learning

Reinforcement learning (Kaelbling et al. 1996) is the problem faced by an agent that has to learn behavior through trial-and-error interactions with a dynamic environment. In this section, we present our experiments with one special form of reinforcement learning, Q-learning (Watkins and Dayan 1992), applied to our case study.

In Q-learning, an agent tries an action (using Boltzmann exploration) at a particular state, and evaluates its consequences in terms of the immediate reward or



(a) 4 robots, 200 sec (b) 6 robots, 50 sec

Figure 3.10: GTP curves with the mix-method. Robots are initially homogeneous. (a) 4 robots with initial GTP 200 sec; (b) 6 robots with initial GTP 50 sec.

penalty it receives and its estimate of the value of the state to which it is taken. The basic form to update the Q value (the expected discounted reward) is

$$Q(s,a) := (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right] \quad , \tag{3.1}$$

where s' is the successive state of state s after action a, r is the reward or penalty of such action, α is the learning factor, and $0 \leq \gamma < 1$ is the discount factor. Watkins and Dayan (1992) proved that Q-learning converges under some conditions. After convergence, the optimal policy π^* is

$$\pi^*(s) = \arg\max_a Q(s,a) \quad . \tag{3.2}$$

Normal Q-learning uses discrete states and actions, and a look-up table to store and update the values of Q.

3.3.1 Settings

To apply Q-learning to the stick pulling experiment, we need to decide what the state s, the action a and the reward r are.

For the state s, it may consist of statistics that a robot could access during the evaluation period[ii]

- `grip1`: number of grip1's made;
- `robot1`: number of robots detected while doing grip1;
- `robot2`: number of robots detected while doing grip2;
- `robot`: number of robots detected;
- `obst`: number of obstacles (robots, sticks, wall) run into.

These statistics are related to robot density, stick density, and/or frequency that other robots could come to help, thus may be useful for a robot to determine good GTPs.

An action could be the percentage to increase (if positive) or decrease (if negative) the GTP, i.e., a *relative action*. It could also be directly used as the GTP, i.e., *absolute action*. Accordingly, the reward (penalty) could also be relative or absolute—the *relative reward* is the logarithmic performance change related to the action and the *absolute reward* is the performance itself.

If discrete states and actions are used, we still have many choices with the granularity and distribution of the discrete states and actions. A finer granularity leads to more accurate observation of states and finer tuned actions, while the increasing number of states or actions requires longer time for Q-learning to converge.

The training phase is further divided into 2 parts. During the first part, robots are given initial GTPs from the set $\{10 + 50k\}_{k=0}^{11}$ and a high temperature in the Boltzmann exploration. This part tends to allow the robots to experience different GTP sets and obtain some initial Q values. The high temperature prevents the early part of learning from being stuck at local minima. The second part is similar to the

[ii] In order to normalize them, simulations are used to determine the ranges.

training phase with the adaptive line search. Robots are endowed with initial GTPs from the set $\{5k\}_{k=0}^{120}$ and a lower temperature. This part is designed to tune the Q values with more experiences and tries to converge to the optimal policy π^*. During each part, the same procedure is repeated 100 times.

3.3.2 Results

Let's first look at a typical result with discrete states and actions. Figure 3.11 shows the collaboration rate in the test phase. The pair (`grip1`, `robot1`) with 25 discrete values is used as the state, and absolute actions and rewards are used. The range for an action is $[4, 1000]$ and 8 uniformly distributed discrete values are used.

We also experiment with different granularities for the states and actions but have not observed significant difference from Figure 3.11. With finer granularity of actions, the average performance gets slightly better when the number of robots is 5 or 6, and slightly worse when the number is between 2 and 4.

When using selected GTPs ($\{5, 30, 50, 200, 500, 1000\}$) instead of uniformly distributed values for the actions, the performance (Figure 3.12) gets a lot better for 2

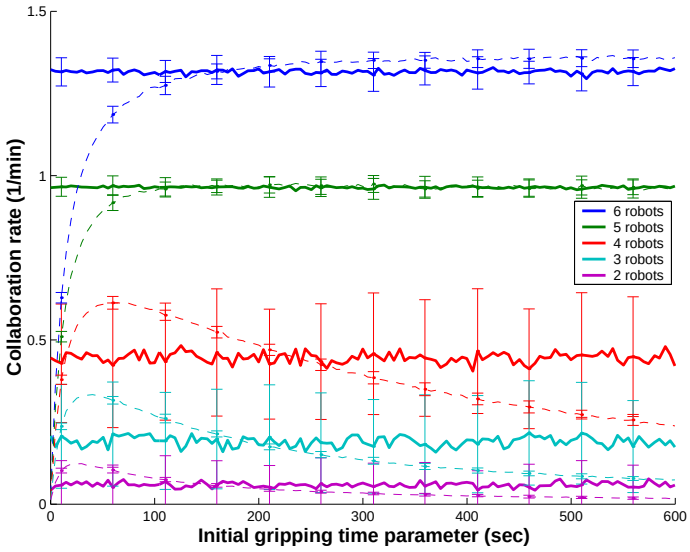


Figure 3.11: Performances with Q-learning (absolute actions and rewards). 8 uniformly distributed values from $[4, 1000]$ are used as the actions (directly used as the GTPs).

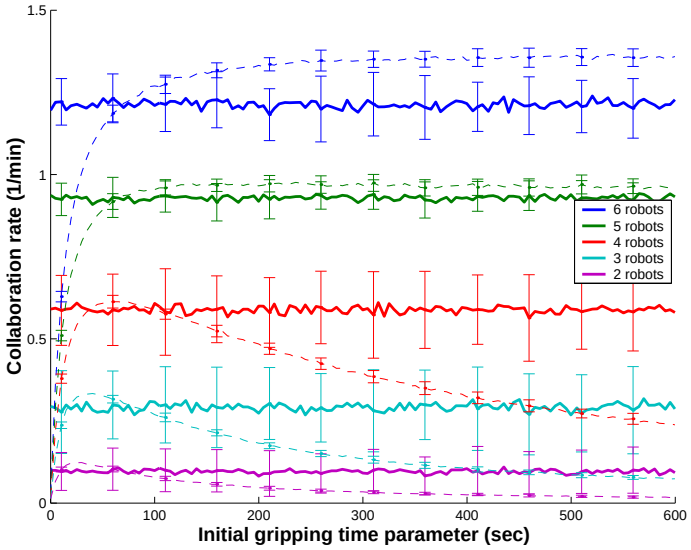


Figure 3.12: Performances with Q-learning (preset GTPs as actions). Selected GTPs ($\{5, 30, 50, 200, 500, 1000\}$) are used as actions.

to 4 robots but worse for 5 or 6 robots.

With relative reward set to

$$\log \frac{\text{performance}}{\text{previous performance}} \quad , \tag{3.3}$$

and relative action having 12 discrete values between -0.9 and 1.5, we get the results shown in Figure 3.13. Again, we have not observed much difference in the results when different granularities are used.

The same situation is observed when we used other state, action, and reward combinations.

Figure 3.14 gives GTP curves for two simulations using absolute action and relative action, respectively.

3.4 Discussion

Q-learning suffers from several drawbacks. First, Q-learning assumes full observation of the environment. When only partial information is available, it is not guaranteed to work. Second, while Q-learning is good at inferring time-delayed reward (through its

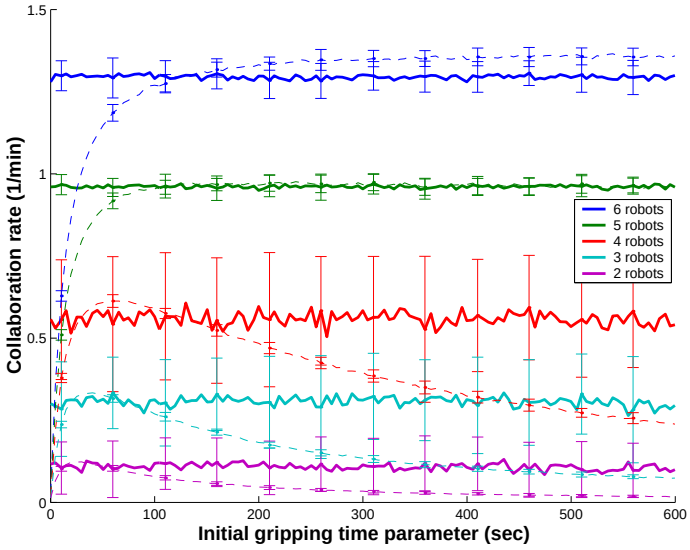
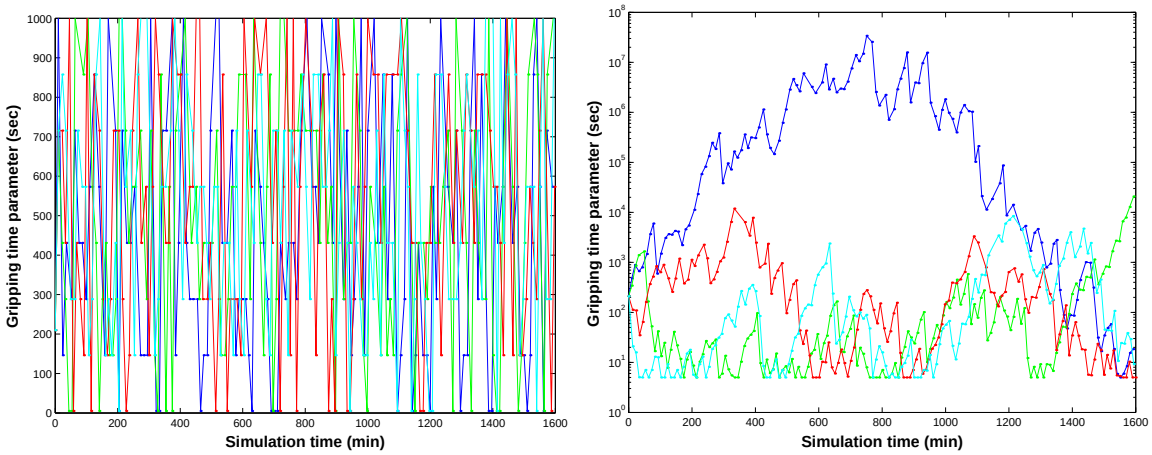


Figure 3.13: Performances with Q-learning (relative actions and rewards). 12 values from $[-0.9, 1.5]$ are used as relative actions.



(a) Absolute actions

(b) Relative actions

Figure 3.14: GTP curves with Q-learning. In both plots, 4 robots are initially homogeneous with GTP 210 sec. (a) Q-learning using absolute actions and rewards, as in Figure 3.11; (b) Q-learning using relative actions and rewards, as in Figure 3.13.

γ discount feature), it has no mechanism to deal with the credit assignment problem in a multi-agent space (Versino and Gambardella 1997). Third, since in the stick pulling experiment there is no communication, the team performance only depends on the environment and the current GTP set, Q-learning's strength in time-delayed reward is not exploited.

Similar observations were made in a case study of soccer teams (Salustowicz et al. 1998). Their simulation using several algorithms showed that direct search in policy space could offer advantages over evaluation function-based approaches, such as Q-learning.

However, if communication is introduced and robots share their policies and episodes, Q-learning may do a better job. Kelly et al. (1997) tested a reinforcement learning algorithm and showed that a robot that received the other robots' experiences learned more quickly and robustly than a robot not receiving such information, though the credit assignment problem is independent of whether policies are shared or not (Salustowicz et al. 1998). Note that the conclusion may change in the stick pulling experiment, in that the individual performance was optimized in (Kelly et al. 1997), while in the stick pulling experiment, we want to optimize the team performance.

Chapter 4

Towards Optimal Performances

Our results with learning (the overall best one is in Figure 3.9) show that learning helps a lot when starting from random GTPs. The average performance after learning is comparable to the optimal performance of homogeneous teams. However, compared with the optimal heterogeneous performance, there is still room left for improvement. As we will see in Section 4.1, if homogeneity is forced during learning, the learned performance becomes much worse than the optimal homogeneous one. Parker and Touzet (2000) also showed that, while the learning approach performs better than random, naive approaches, much improvement is still needed to match the results obtained from the hand-generated approach.

In this chapter, we try to understand the effect on learning of several issues, such as local and global reinforcement, noise in the reinforcement, and we try to find out what are the barriers between the learned solutions and the optimal solutions. We discuss why learning, a procedure that tries to achieve both optimality and adaptability, has to find a trade-off between these two goals, and cannot maximize both of them. We call this the problem of *trade-off between optimality and adaptability*.

4.1 Local and Global Reinforcement

In previous experiments, individual robots were reinforced locally and were asked to "learn" the optimal GTP set, which is essentially a global characteristic. While local reinforcement is more realistic for a swarm system, *global reinforcement*, which

usually implies a supervisor that measures and broadcasts the team performance to the individual robots, provides an interesting term of comparison. Using local reinforcement (or global reinforcement combined with homogeneity) is a way to bypass the credit assignment problem (Versino and Gambardella 1997) but it may not achieve the best team performance (Murciano et al. 1997), which is the quantity we want to improve. The global reinforcement has no such problem, but we have to face the credit assignment problem.

Figure 4.1 shows that performances using global reinforcement are comparable to those using local reinforcement (Figure 3.9), implying a good alignment between local and global reinforcement.

When global reinforcement is used, we can force homogeneity by allowing one robot to adapt and broadcasting the GTP of that robot to other robots. Without surprise, the learned performances (Figure 4.2) are lower than the performances produced by policies that allow heterogeneity since the task constraint prompts for specialization when there are no more robots than sticks. Another reason is that homogeneity is only a special case of heterogeneity.

We end this section with a chart (Figure 4.3) comparing performances we get from

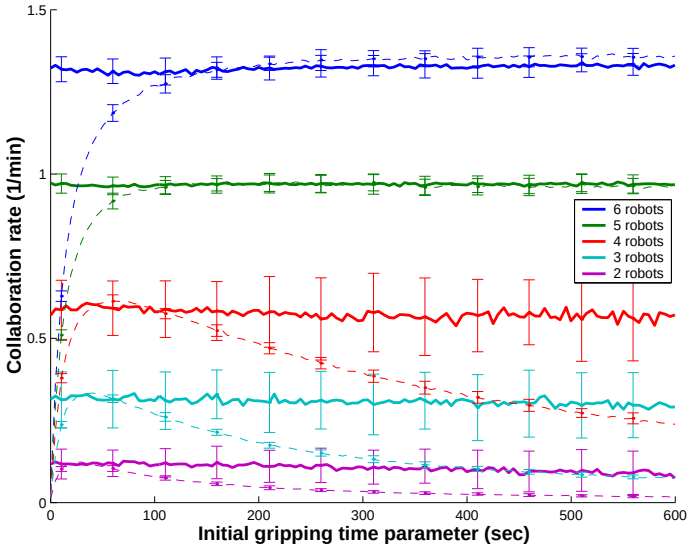


Figure 4.1: Performances of heterogeneous teams under global reinforcement. Solid curves are learned performances with the mix-method, and dashed curves are performances of homogeneous teams without adaptation.

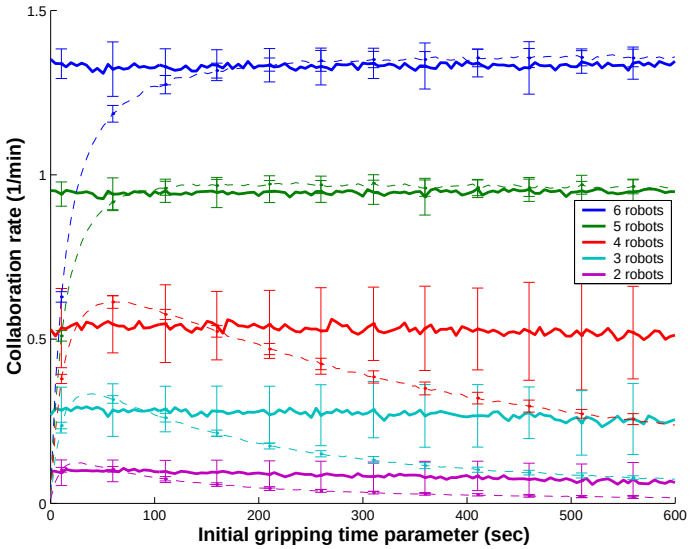


Figure 4.2: Performances of homogeneous teams under global reinforcement. Solid curves are learned performances with the mix-method, and dashed curves are performances of homogeneous teams without adaptation.

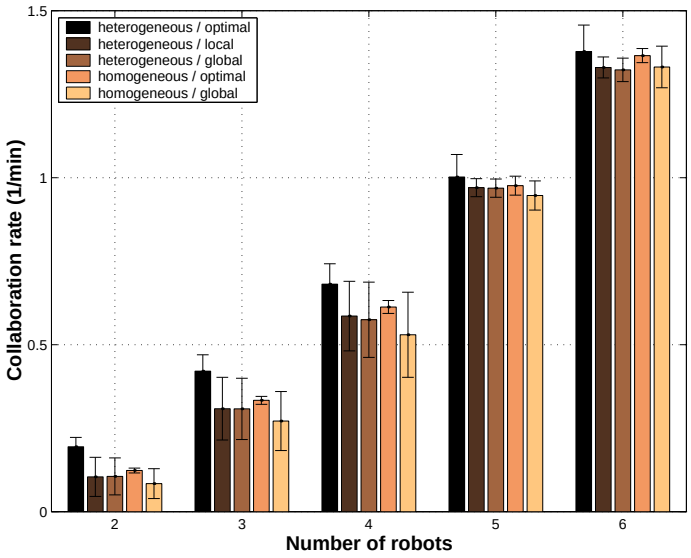


Figure 4.3: Performances under different reinforcement and team diversity. Performances are obtained with the systematic search (see Figure 2.5) and after learning under different reinforcement and team diversity.

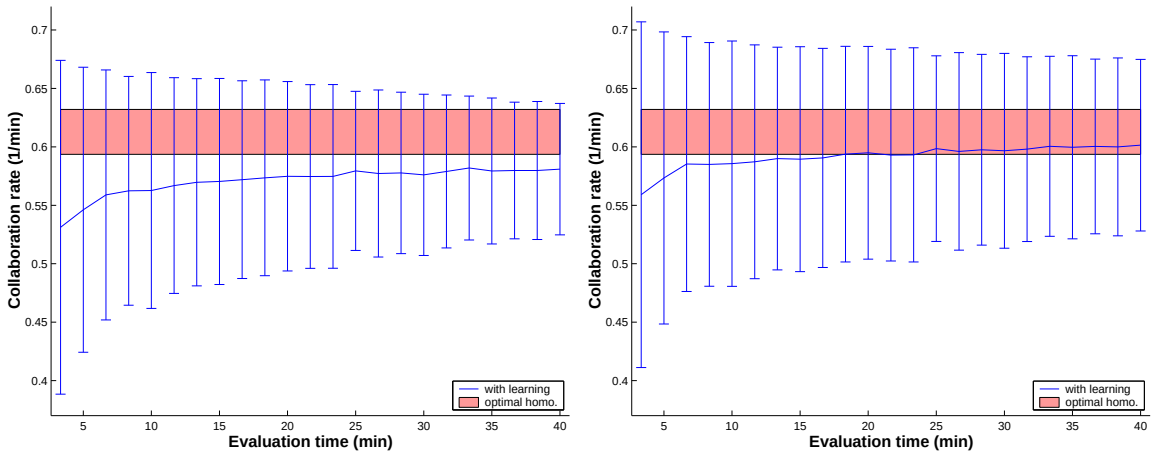
different reinforcement and different team diversity. It is observed that the learning can not achieve the optimal performance, no matter what kind of reinforcement is used or what team diversity is allowed. An explanation to the limitation of learning could be, for instance, that the noise affecting the reinforcement may prevent the learning algorithm from converging to a good final choice of the GTP. As we will see in the next two sections, noise from different sources plays an important role in learning.

4.2 Evaluation Time

The stochastic nature of the stick pulling experiment as well as the simulation model adds randomness, or noise, to the local reinforcement—the individual performance a robot gets during an evaluation period is volatile. This definitely makes learning difficult. We may decrease this randomness by increasing the evaluation time. Longer evaluation time provides more accurate and stable reinforcement, thus improves the learning, but the delay for a robot to react to environmental changes also becomes longer. This is the first time we meet the problem of the trade-off between optimality and adaptability.

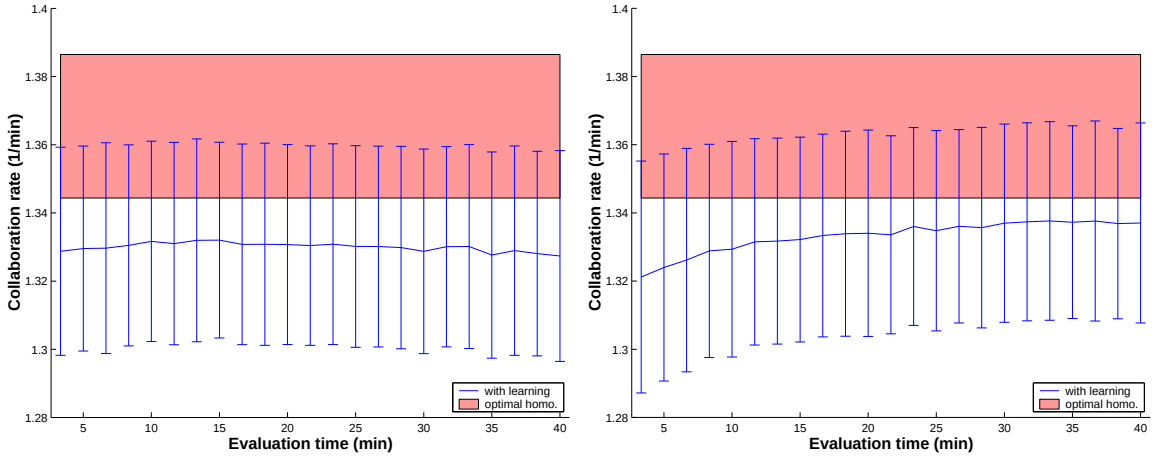
The simulation results (Figure 4.4 and 4.5) validate our judgment about the performance and the evaluation time. Long evaluation time improves both the quality of GTP sets learned (which is reflected in the increasing test phase performance) and the stability (which is reflected in the decreasing standard deviation). At the same time, longer evaluation time also requires longer training time for the learning to converge. With 6 robots and an extended 6400 min training phase, the performance resulted from long evaluation time ($\geq$ 25 min) is better than those with a normal 1600 min training phase.

However, even with long evaluation time, the difference between the learned performance and the optimal performance does not decrease significantly, implying either that the noise in the reinforcement is not the main barrier for learning, or that there are still significant sources of noise other than short evaluation time.



(a) 4 robots, training phase

(b) 4 robots, test phase



(c) 6 robots, training phase

(d) 6 robots, test phase

Figure 4.4: Performance vs. evaluation time. The rectangular region in each plot represents the range of the performance of the optimal homogeneous team (its height spans $\mu \pm \sigma$, where μ is the mean performance and σ is the standard deviation).

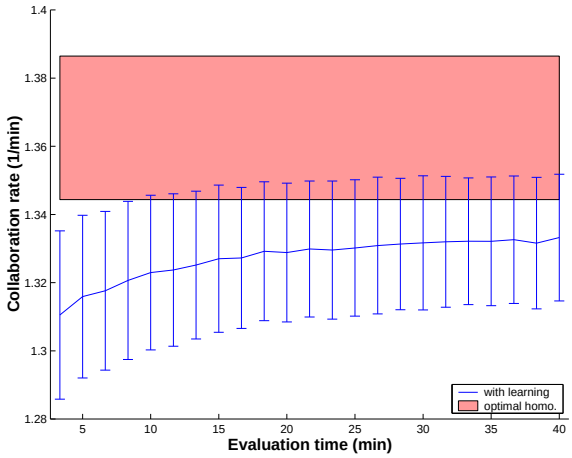


Figure 4.5: Performance vs. evaluation time (extended training phase). The curve is the performance of 6 robots in a 6400 min training phase. See also Figure 4.4.

4.3 Precomputed Performance

When a robot adapts its GTP, it has to wait until an evaluation period ends to obtain the new reinforcement associated with the new GTP value. This kind of delayed reward also hinders the learning of optimal GTP sets. Longer evaluation time leads to better evaluation of the reinforcement, but also introduces longer delay. Though it is impossible to completely eliminate the noise or the delay between action and reward in a real environment and in a collective scenario, doing so would be helpful in better understanding the effect of noise and delayed reward on learning.

One way to eliminate the noise and delay is to use the precomputed performance as the global reinforcement. That is, the performance of every possible GTP set has been computed (through simulation) beforehand. During the training phase, the current performance is thus obtained without delay by looking up the current GTP set in the precomputed table. The learned performance using the precomputed table should be the upper limit that the learning algorithm could achieve.

Since only for homogeneous teams we can run all the systematic simulations, we only test learning with the precomputed performance for a homogeneous team. Under this situation, the whole learning becomes a static one-dimensional line search without noise. It is therefore not surprising that the performance is much better (Figure 4.6).

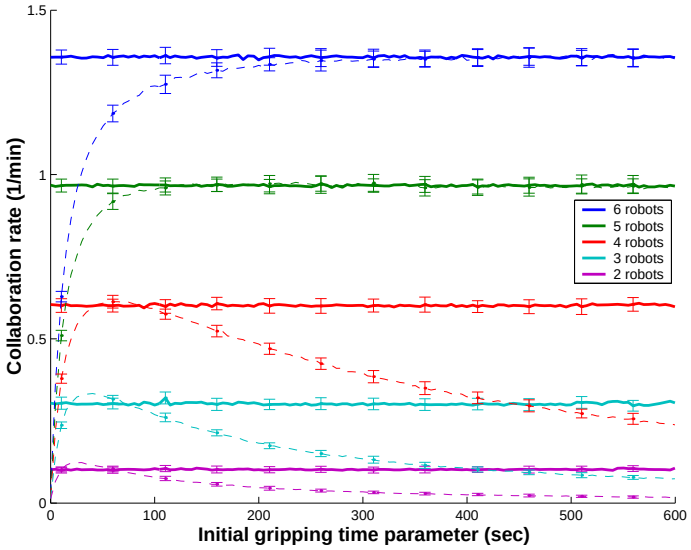


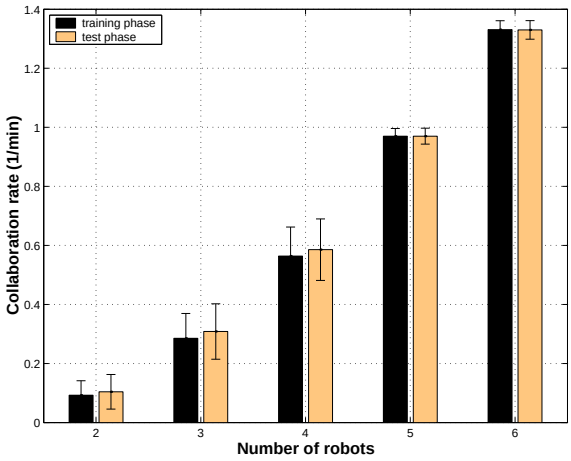
Figure 4.6: Performances of homogeneous teams with precomputed performance. The mix-method is used. See also Figure 4.2 for comparison.

When precomputed performance is used as the reinforcement, the role of the evaluation time becomes trivial—it only controls the frequency with which a robot updates its GTP. Different evaluation times (2 min, 10 min, and 40 min) lead to almost the same team performances, showing that the algorithm converges quickly under the no-noise reinforcement.

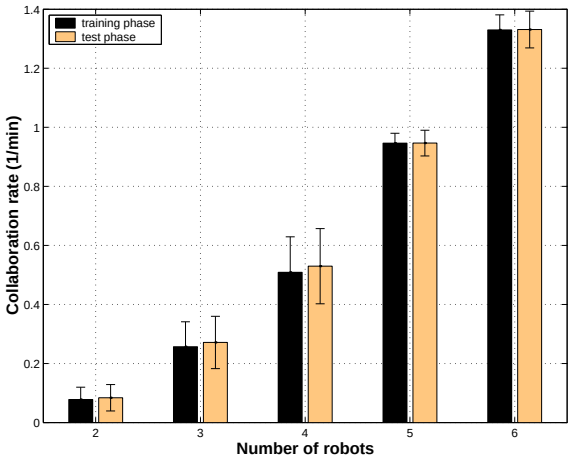
4.4 Training and Test Phases

We have observed different team behaviors in the training phase and the test phase (see Figure 4.4 for an example). The performance in the test phase is usually higher but with larger deviation. This observation is quite consistent among different learning methods, different reinforcement and team diversity (Figure 4.7).

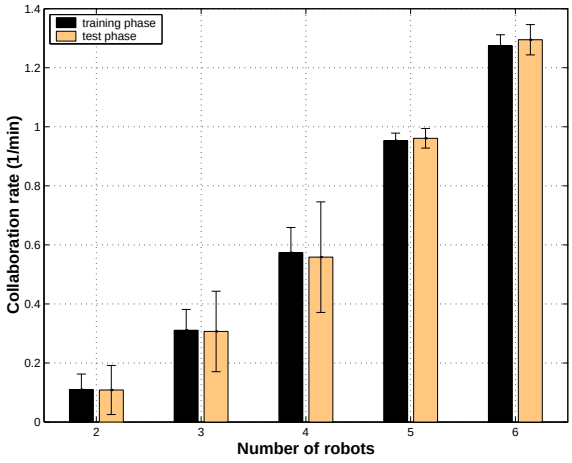
The only difference in the experimental setting between the two phases is that learning is disabled in the test phase. Remember that the stick pulling experiment is full of stochastic events. This observation implies that being able to learn has a twofold effect: (1) adapting to the stochastic events so that the performance is more stable; (2) adapting to "noisy" reinforcement even if this may divert the GTP set from the optimal one which is unknown to the team.



(a) Heterogeneous team (see Figure 3.9)



(b) Homogeneous team (see Figure 4.2)



(c) Heterogeneous team (see Figure 3.13)

Figure 4.7: Performance differences between the training phase and the test phase. The performance in the test phase is usually higher than that in the training phase. The deviation in the test phase is also larger. (a,b) Mix-method; (c) Q-learning with relative actions and rewards.

This phenomenon again relates to the trade-off between optimality and adaptability. From the viewpoint of a robot, it can not distinguish without delay whether a change in its "local world" is caused by the environmental change, or by some randomness. Time is needed for accumulating evidence to distinguish the two causes. That is, some information can only be obtained with delay. If the robot wants to catch up to the environmental change promptly, it has to use incomplete and sometimes wrong information. Thus the price of adaptability is some loss of optimality. This is the reason that, during the training phase, the team has smaller standard deviation while the average performance is lower than that in the test phase.

4.5 Multi-stage Test

To further investigate the issue of optimality and adaptability, we test the team performance under a changing environment. The basic form is that we break the simulation into several stages, and use different environmental settings (arena size, number of sticks, and number of robots) for different stages. We are particularly interested in how team performance changes according to environmental changes.

Environmental changes are simulated as realistically as possible—each robot, as a separate process in the simulation, could sense the changes without knowing it, and the learning process is not interrupted. That is, if possible, robots are not reset after an environmental change and all historical learning information is kept. Note that whenever a robot or a stick is removed from the arena, we need to check whether that robot is doing a grip1 or whether that stick is held by some robot in order to update the environment correctly.

4.5.1 Changing Stick Density

In this test, we change the stick density in the middle of the simulation by enlarging the arena and changing the number of sticks. The simulation starts at time 0 with the normal-size arena (80 cm in diameter), 4 sticks, and 6 robots. At time $\frac{1}{3}T_s$ (here $T_s = 4800$ min), the diameter of the arena is then increased to 100 cm and 4 other

sticks are added. At time $\frac{2}{3}T_s$, 4 sticks are removed from the arena but the size of the arena remains unchanged. Thus we have a larger arena with 4 sticks. The simulation ends at time T_s. We test the performance of 6 robots with and without learning.

When tested without learning, the robots are initialized with the optimal GTP set we got from the systematic search with early stopping (see Section 2.3), i.e., $\{400, 400, 400, 500, 500, 1000\}$. 3000 runs are carried out to get the average performance. The tests with learning are similar to those run in Chapter 3: 100 runs for each initial GTPs in set $\{5t\}_{t=0}^{120}$.

Figure 4.8 shows the average performance with the optimal GTP set and with learning. Note that the optimal GTP set is only optimal for the normal-size arena with 4 sticks. When the time reaches stage 2 (time between 1600 min and 3200 min), the ratio between the number of robots and the number of sticks changes from 1.5 to 0.75, producing a dramatic drop in the collaboration rate. However, robots with learning have a better position when the change happens and they continue increasing the advantage by adapting to the new environment. When the second change comes,

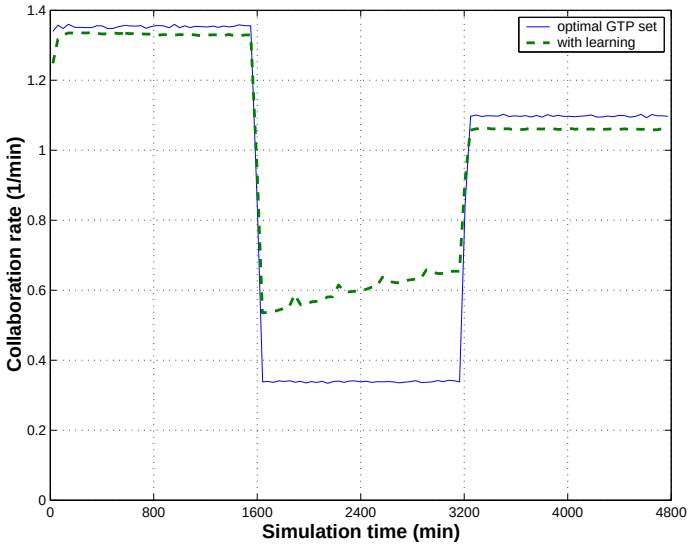


Figure 4.8: Performances under changing stick density. The solid curve is the average performance of a heterogeneous team with the optimal GTP set. 3000 runs are simulated. The dashed curve represents the average performance of a heterogeneous team with the mix-method. For each initial GTP from $\{5t\}_{t=0}^{120}$, 100 runs are simulated.

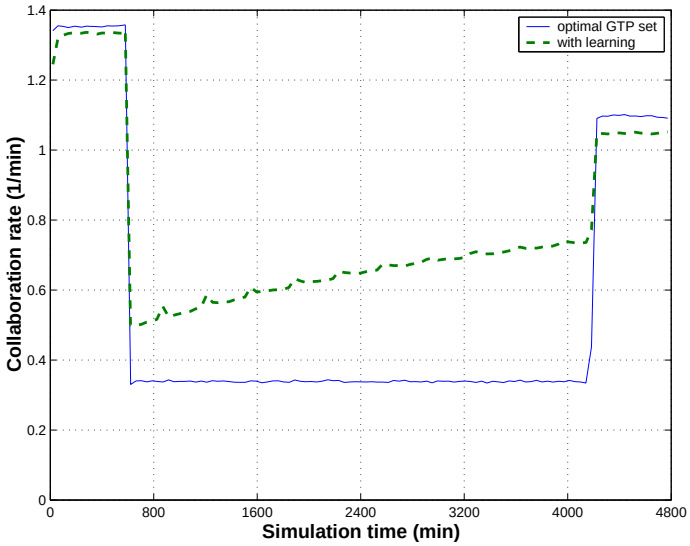


Figure 4.9: Performances under changing stick density (longer stage 2). See Figure 4.8 for details.

the optimal GTP set seems working well in a larger arena[i] and the performance of the team without learning surpasses that of the team with learning. However, the difference is not significant.

With a longer stage 2 (Figure 4.9), the team with learning definitely improves its performance more during that stage.

4.5.2 Adding/Removing Robots

In this test, we still start from the normal arena, 4 sticks and 6 robots. At time $\frac{1}{8}T_s$, two randomly chosen robots are removed from the arena and then at time $\frac{7}{8}T_s$, one of the robots previously removed is added back to the arena. The test terminates at time T_s. For the test without learning, the optimal GTP set used in the previous test for 6 robots is also used here.

Figure 4.10 gives the two performance curves. Again, the team with learning does a bit worse than the team with optimal GTP set during the first stage, recovers a lot during the second stage, and does almost as well in the third stage.

[i]Since the number of robots and the number of sticks are the same for stage 1 and stage 3, it is reasonable that the optimal GTP set for stage 1 is also (near-)optimal in stage 3. The lower performance is due to the longer searching time in a larger arena.

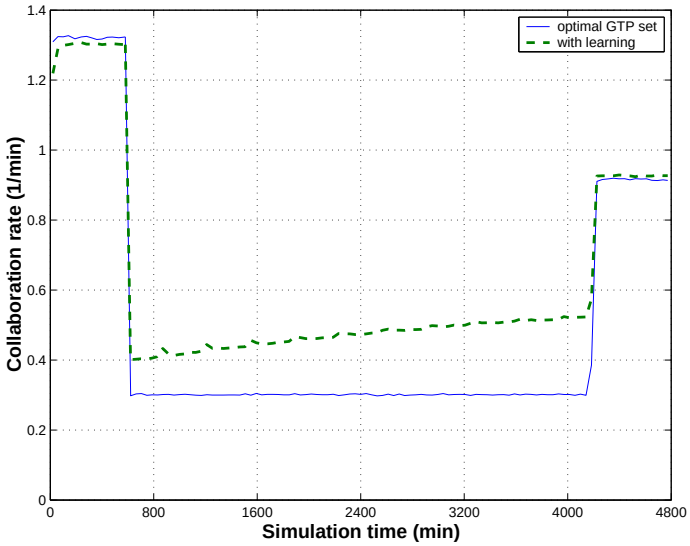


Figure 4.10: Performances under different numbers of robots. See Figure 4.8 for details.

4.6 Discussion

In this chapter, we investigated learning issues under different configurations. Experiments with local and global reinforcement showed that, in this case study, the two kinds of reinforcement align well and have similar functionality in learning. Allowing heterogeneity in learning leads to much better performance than forcing homogeneity even if the same reinforcement (the global one) is used for every robot. Randomness or noise in the reinforcement seems to prevent the learning algorithm from finding good final choices of GTPs. However, the stochastic nature of the experiment decides that even with global perception, the noise can not be 100% eliminated. Learning has to accept the existence of the noise as a fact. To distinguish noise from environmental changes (both can cause the reinforcement to change), time is needed for accumulating evidence and thus the robots cannot promptly adapt to environmental changes.

Finally we come to realize that a learning algorithm can only achieve a trade-off between optimality and adaptability, but cannot maximize both. Under the condition of partial perception and noisy, delayed reinforcement, better adaptability means incomplete or even wrong information will be used in learning and thus leads to worse

optimality. Conversely, better optimality requires more time to collect information and thus results in longer delay in catching up to the environmental change.

If we define *adaptability* as the time of convergence, *optimality* as the difference between the learned solution and the optimal one, and *environmental complexity* as a quantity estimating the effect of partial perception, delayed reward, and noise, we may get an inequality like the Heisenberg's uncertainty principle in physics:

$$\text{Adaptability} \times \text{Optimality} \geq \text{Environmental Complexity}. \tag{4.1}$$

This is just a conjecture; we have not found any theoretical basis for it.

Chapter 5

Measuring Specialization

In many experiments we conducted, it was observed that the robots became specialized after learning. For example, Figure 3.5 shows the GTP curves of several robots in two different runs using the Δ-method. All robots in a team started with the same GTP. As the experiment progressed, their GTPs diversified and formed several clusters in the GTP space.

These results, together with the inferior performance we got from global reinforcement and homogeneous teams (e.g., Figure 4.2), are consistent with the results of systematic experiments reported in (Ijspeert et al. 2001) which show that, when the number of robots is less than or equal to the number of sticks, specialization is helpful. In addition to the stick pulling experiment, specialization (a situation somewhere between homogeneity and full heterogeneity) has proven to be effective for solving several tasks (see for instance (Murciano et al. 1997; Campos et al. 2001)).

However, it is not enough to say "specialization emerges when learning is introduced." We want to know how "well" or to what degree the teammates become specialized. Heterogeneity will also appear when the robots randomly update their parameters, which we won't call specialization. Hence we also want to convince ourselves that solutions from learning are different from random ones.

This chapter presents our effort in quantifying specialization which is basically a qualitative notation. The question we try to tackle with is: Giving a GTP set, what is the degree of specialization in that set?

Consider again Figure 4.2 where six robots seem to form three clusters, with GTPs

around 50 sec, 100 sec, and 200 sec, respectively. The number of clusters in the GTP space looks like a good indication of how much the robots have been specialized. If there is only one cluster, i.e., all robots have almost the same GTP value, they do not specialize; if there are several clusters, the robots are regarded specialized according to different roles and localities.

However, finding the "right" number of clusters for a data set is often ill-posed (Smyth 1996). Depending on different criteria, one number may or may not be better than another number. In the following sections, two ad hoc algorithms are tested with two different GTP spaces (linear and logarithmic spaces).

5.1 Greedy Algorithm

For the first try, we test a very simple algorithm (Algorithm 5.1) which utilizes the fact that GTP is one-dimensional. The algorithm starts from the smallest GTP in the set and tries to include in the same cluster as many GTPs as possible while still maintaining the intra-cluster distance no more than a given parameter $d_{\max}$. Corresponding to ideas underneath the Δ-method and the %-method, the GTP set can be directly fed into the algorithm, or its logarithm can be used. For the former case, we say the *linear GTP space* is used, and for the latter case, we say the *logarithmic GTP space* is used.

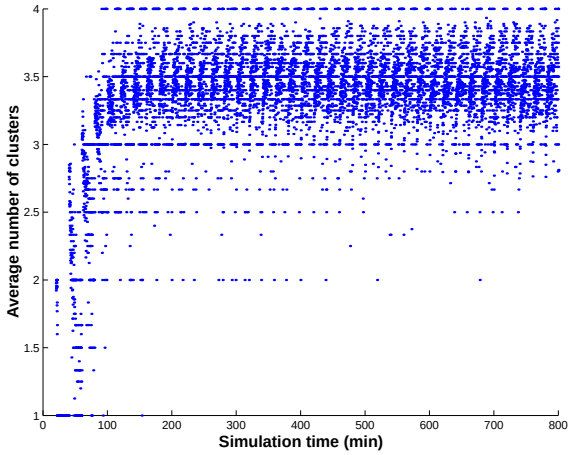
Algorithm 5.1 Greedy algorithm to find the number of clusters.

Input: A GTP set G, $n = |G|$ is the number of robots.

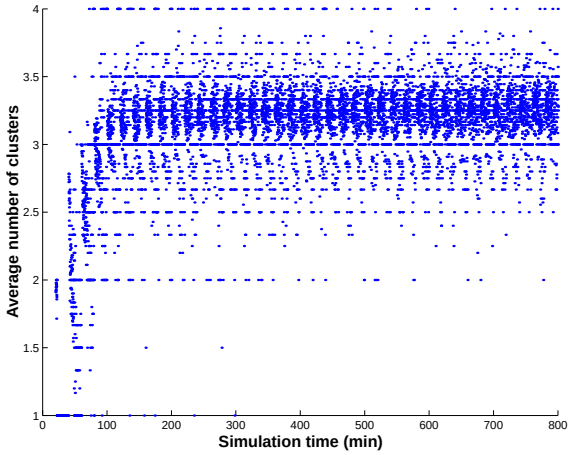
Parameter: $d_{\max}$ is the maximum intra-cluster distance.

1. Sort G so that $G = \{g_1, g_2, \ldots, g_n\}$ and $g_i \leq g_{i+1}$ for $1 \leq i < n$.
2. $m \leftarrow 1$, $i \leftarrow 1$, $j \leftarrow 1$.
3. Increase j until $j > n$ or $g_j - g_i > d_{\max}$.
4. If $j > n$, return m as the number of clusters; otherwise, $i \leftarrow j$, $m \leftarrow m + 1$, go to 3.

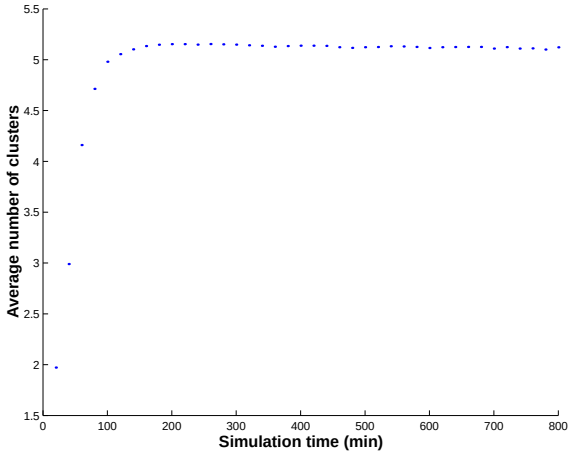
Figure 5.1 demonstrates the output of this greedy algorithm on four and six robots using the mix-method, in both linear GTP space and logarithmic GTP space. Apparently the number of clusters computed in the linear GTP space is higher than that computed in the logarithmic GTP space. The ratio between the number of clusters and the number of robots is close to 1, which is intuitively "too large" since by human



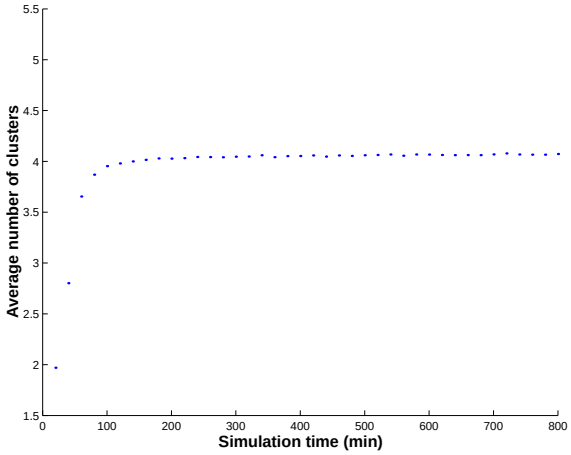
(a) Linear with $d_{\max} = 100$, 4 robots



(b) Logarithm with $d_{\max} = 0.7$, 4 robots



(c) Linear with $d_{\max} = 100$, 6 robots

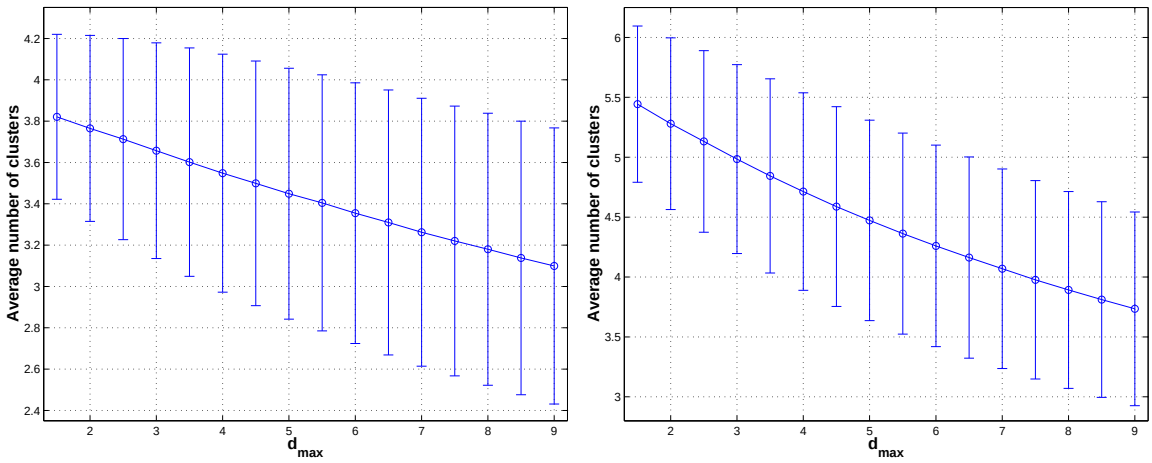


(d) Logarithm with $d_{\max} = 0.7$, 6 robots

Figure 5.1: Number of clusters vs. simulation time (greedy algorithm). Robots are initially homogeneous. The number of clusters calculated by the greedy algorithm is averaged over 50 runs for each GTP from $\{5k\}_{k=0}^{120}$. The mix-method is used as the learning algorithm.

judgment from several runs (e.g., Figure 3.5) the ratio is roughly one half.[i] This is mostly due to the fact that both the performance and the logarithm are less sensitive to GTP changes when GTP is high.

While the choice between linear space and logarithmic space depends on the problem, we still have to choose a good intra-cluster distance $d_{\max}$ for this simple algorithm. It turns out that even in the logarithmic space, the result of the algorithm depends heavily on the choice of $d_{\max}$ (Figure 5.2) and a "good" choice, if there exists such a choice for all numbers of robots, is very subjective.



(a) Logarithmic space, 4 robots

(b) Logarithmic space, 6 robots

Figure 5.2: Results of the greedy algorithm depend heavily on $d_{\max}$. With larger intra-cluster distance $d_{\max}$, the algorithm gives smaller number of clusters.

5.2 Best-Fit Algorithm

In order to more objectively measure the degree of specialization, we try to find the "best-fit" number of clusters using the expectation-maximization (E-M) algorithm (Dempster et al. 1977) as a subroutine. Illustrated in Algorithm 5.2, the basic idea of this algorithm is to try all possible numbers for clustering for a given GTP set. The best partitioning is then selected based on a heuristic criterion: a good partitioning

[i]The ratio calculated from results of the best-fit algorithm (we will soon come to that) is around 0.7.

Algorithm 5.2 Best-fit algorithm to find the number of clusters.

Input: A GTP set G, $n = |G|$ is the number of robots.

1. For each $m = 1..n$, do
   (a) Use E-M algorithm to cluster G into m clusters;
   (b) Calculate the heuristic fitness f_m of such clustering.
2. Return $\arg\max_m f_m$ as the number of clusters.

of the GTP set should maximize the inter-cluster distances while minimizing the intra-cluster distances. We mathematically translate this heuristic as the following definition of fitness:

$$f_m = \langle \text{inter-cluster distance} \rangle - \langle \text{intra-cluster distance} \rangle \quad , \tag{5.1}$$

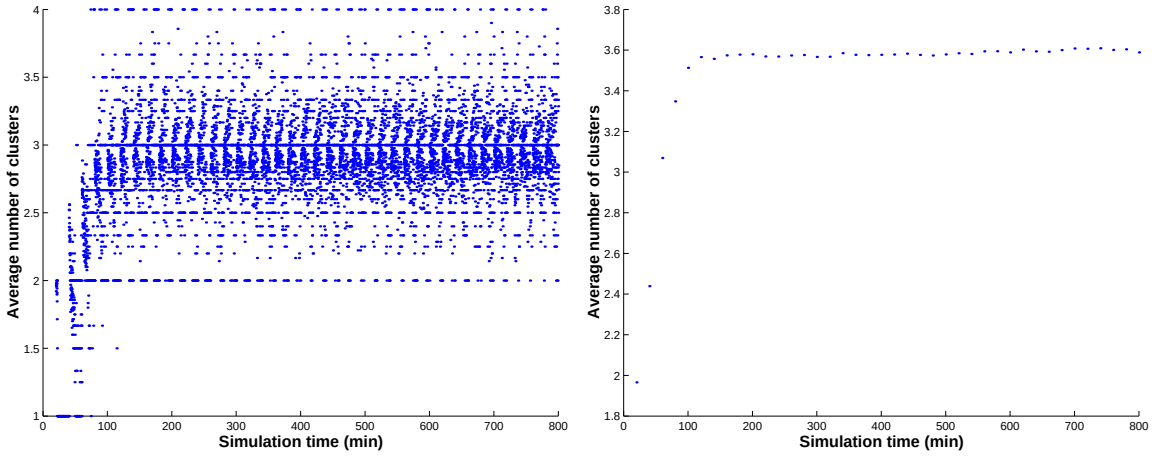
where $\langle \cdot \rangle$ denotes the average. As mentioned in previous sections, this is also an ad hoc criterion.

Figure 5.3 shows the best-fit algorithm is more conservative in estimating the number of clusters.

5.3 Sub-linearity

We apply the best-fit algorithm (Algorithm 5.2) to GTP sets obtained from the mix-method learning algorithm with local reinforcement. Figure 5.4 shows the average number of clusters after learning vs. total number of robots. With more and more robots, it is reasonable that the number of clusters also increases. However, if we normalize this number by the total number of robots, we obtain a clear saturation of the relative number of clusters per robot. Indeed, when the number of robots exceeds that of sticks, the need for specialization decreases and so does the degree of specialization.

As in Section 4.1, we also investigated heterogeneous teams combined with global



(a) Logarithm space, 4 robots

(b) Logarithm space, 6 robots

Figure 5.3: Number of clusters vs. simulation time (best-fit algorithm). Robots are initially homogeneous. The number of clusters calculated by the best-fit algorithm is averaged over 50 runs for each GTP from $\{5k\}_{k=0}^{120}$. The mix-method is used as the learning algorithm.

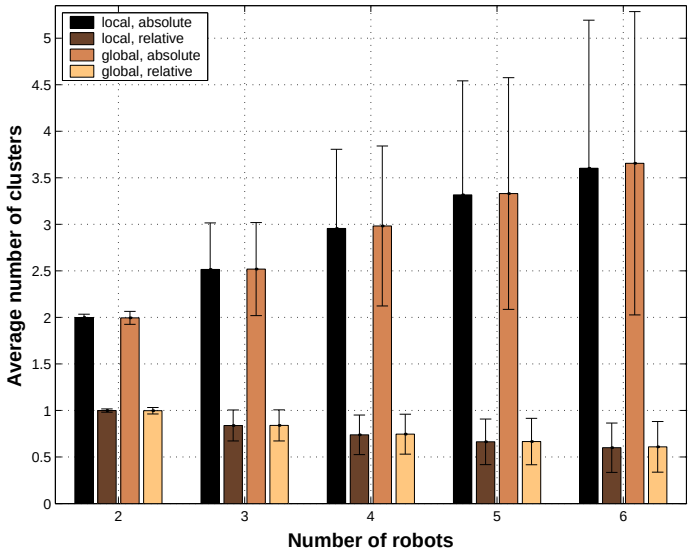


Figure 5.4: Number of clusters after learning under different reinforcement. The number of clusters (absolute value) and that per robot (relative value) are calculated by the best-fit algorithm.

reinforcement. Once again, we do not obtain any significant difference in the degree of specialization as a function of the reinforcement type.

5.4 Random Test

As a measure of specialization, the best-fit algorithm itself as well as the definition of the fitness (5.1) is quite arbitrary. We need to be convinced that this kind of measure indeed reveals *something* about the specialization resulting from learning. We decide to challenge the best-fit algorithm with GTP sets generated randomly instead of from learning.

Figure 5.5 clearly shows that the best-fit algorithm could distinguish the random input from the input after learning. When fed with randomly generated GTP sets, the algorithm gives higher output, no matter whether the linear or logarithmic space is used, indicating more scattered GTPs. Even though it results in robots with different GTPs, learning is likely to group several robots together, keeping the specialization

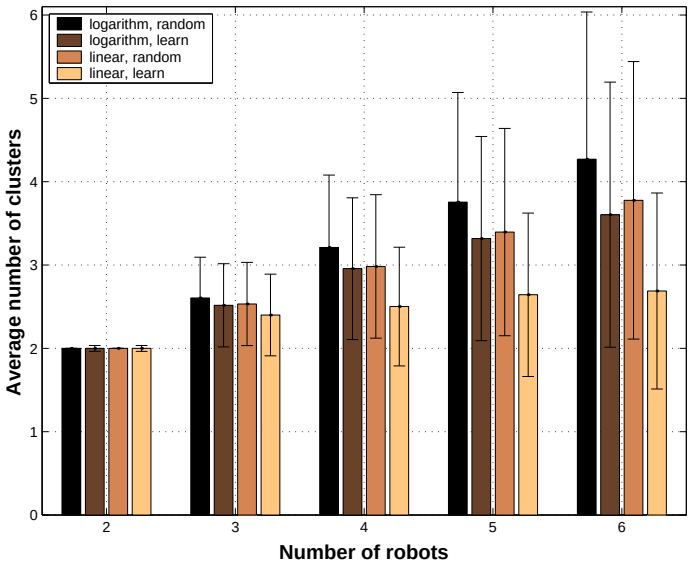


Figure 5.5: Number of clusters after learning and with random GTP sets. A random GTP set consists of GTPs randomly (uniformly) generated within $[5, 1000]$. For each team size, 10000 random sets are tested to get the error bars. Here the mix-method is used for learning with local reinforcement. 100 runs for each initial GTP in $\{5k\}_{k=0}^{120}$ are simulated. Obviously learning groups robots together while still maintaining some degree of specialization.

from full heterogeneity.

5.5 Discussion

The E-M algorithm in the best-fit algorithm could be replaced by any other clustering algorithm, such as ISODATA (Ball and Hall 1967) or k-means clustering (Moody and Darken 1989). The complexity of the best-fit algorithm depends on that of the underlying clustering algorithm and n, the number of the robots. Since every number between 1 and n will be tried as the number of clusters, the algorithm is not suitable for a large number of robots.

When applied to hierarchical clustering, deterministic annealing can find the "natural" number of clusters for any given β, which is a parameter related to the "temperature" (Rose 1991; Rose 1998). This is quite similar to what we want to do in this chapter. However, β is a free parameter which is hard to decide, causing similar problems as $d_{\max}$ in the greedy algorithm.

Clustering is similar to the mixture model (McLachlan and Basford 1987) in which samples of the mixture of several distributions with unknown parameters are given and the goal is to find the parameters. The problem of finding the "right" number of clusters (distributions) also exists when the number of underlying distributions is unknown.

In this case study, we may also adopt a non-integer as the number of clusters. It is reasonable, when some GTP sets are equally like to be two and three clusters, to assign the real number 2.5 to the "number of clusters," which is used as the degree of specialization. One fast (but untested) way to introduce real number of clusters in the best-fit algorithm is to return the weighted sum of m with the fitness as the weight.

Chapter 6

Conclusion

With the help of a microscopic probabilistic model, we investigated several issues on distributed learning in swarm systems with a concrete case study in collective robotics (the stick pulling experiment).

We tested several learning algorithms with local or global reinforcement and compared the learned performances with optimal ones obtained from the systematic search (with early stopping). While learning provides better solutions than random or naive approaches, it can only achieve near-optimal performance. Among algorithms we tested, the adaptive line search which directly searches for optimal parameters works much better than Q-learning which is based on reward estimation.

Specialization has consistently been observed after learning, though the teams were initially homogeneous and we never explicitly rewarded diversity. We found that learning policies which allowed teammates to specialize found an adequate diversity of the team and in general achieved similar or better performances than policies which forced homogeneity. Two ad hoc measures were designed to quantify the degree of specialization. Results showed that learning led to teams with specialization somewhere in the middle of homogeneity and total randomness. We also illustrated that the need for specialization decayed with more and more robots.

For a learning algorithm, optimality indicates how good the learned solution is and adaptability reflects how quick the team adapts to environmental changes. We proposed the problem of trade-off between optimality and adaptability. Several experiments were carried out in order to understand the effect of noise and time delay on

learning. We argued that a learning algorithm could only achieve a trade-off between optimality and adaptability, but could not maximize both.

Finally, in this specific case study, the emergent specialization and achieved team performances appear to be independent of the locality or globality of the reinforcement signal, probably due to the high alignment between both forms of reinforcement.

Although we have not tested our learning algorithms using real robots or realistic simulations, we believe that their validity is not limited to abstract agents since the simulation model is quite faithful in simulating experiments with real robots. Because there was no direct communication between robots, even though we only did experiments with two to six robots, our learning algorithms and some conclusions should still be valid to system with thousands of agents.

6.1 Future Directions

In the end of the thesis, let us consider future directions for research in both theory and application fields.

- We would like to further study the optimality and adaptability issue. Formalizing our conjecture of inequality (4.1) in some simple cases may still be a difficult job. We need to investigate quantitatively several variables, such as convergence time and learned solutions, under different learning strategies and reinforcement, and try to set up relationships between them.

- Some work on Internet congestion control (Low and Lapsley 1999; Low et al. 2002) established a model where global optimization can be solved by several local optimization problems. Though at an initial glance their method can not be applied to the distributed learning in swarm systems due to different system settings, further investigation is still needed.

- It would be interesting to evaluate and validate the same learning policies with real robots in order to investigate whether the solutions learned at the microscopic model level are also effective when implemented on real robots.

- As a way to reduce locality and share experiences, parameters, and policies, local communication helps to solve problems in distributed learning such as the credit assignment problem (Matarić 1998). We want to investigate different types of local communications and their effect on the team performance and specialization. Note that the geometric information (such as positions of robots) has to be supplied to microscopic models in order to simulate the local communication.

Bibliography

[Ball and Hall 1967] G. Ball and D. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, 1967.

[Bonabeau and Meyer 2001] E. Bonabeau and C. Meyer. Swarm intelligence: A whole new way to think about business. *Harvard Business Review*, 79(5):106–114, 2001.

[Bonabeau et al. 1999] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

[Bonabeau et al. 2000] E. Bonabeau, M. Dorigo, and G. Theraulaz. Inspiration for optimization from social insect behaviour. *Nature*, 406:39–42, 2000.

[Campos et al. 2001] M. Campos, E. Bonabeau, G. Théraulaz, and J.-L. Deneubourg. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8(2):83–94, 2001.

[Dempster et al. 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39(1):1–38, 1977.

[Dorigo and Gambardella 1997] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[Ijspeert et al. 2001] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella. Collaboration through the exploitation of local interactions in autonomous collec-

tive robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001.

[Kaelbling et al. 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kelly et al. 1997] I. D. Kelly, D. A. Keating, and K. Warwick. Mutual learning by autonomous mobile robots. In M. Beneder, ed., *Proceedings of the First Workshop on Teleoperation and Robotics, Applications in Science and Arts*, pp. 103–115. R. Oldenbourg, 1997.

[Lerman et al. 2001] K. Lerman, A. Galstyan, A. Martinoli, and A. J. Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, 7(4):375–393, 2001.

[Low and Lapsley 1999] S. H. Low and D. E. Lapsley. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–875, 1999.

[Low et al. 2002] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *IEEE Control Systems Magazine*, 22(1):28–43, 2002.

[Martinoli and Mondada 1995] A. Martinoli and F. Mondada. Collective and cooperative group behaviours: Biologically inspired experiments in robotics. In O. Khatib and J. K. Salisbury, eds., *Proceedings of the Fourth International Symposium on Experimental Robotics (ISER'95)*, 1995, vol. 223 of *Lecture Notes in Control and Information Sciences*, pp. 3–10. Springer-Verlag, Berlin, 1997.

[Martinoli et al. 1999a] A. Martinoli, A. J. Ijspeert, and L. M. Gambardella. A probabilistic model for understanding and comparing collective aggregation mechanisms. In D. Floreano, J.-D. Nicoud, and F. Mondada, eds., *Advances in Artificial Life*, vol. 1674 of *Lecture Notes in Artificial Intelligence*, pp. 575–584. Springer-Verlag, Berlin, 1999.

[Martinoli et al. 1999b] A. Martinoli, A. J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29(1):51–63, 1999.

[Matarić 1998] M. J. Matarić. Using communication to reduce locality in distributed multi-agent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):357–369, 1998.

[McLachlan and Basford 1987] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1987.

[Michel 1998] O. Michel. Webots: Symbiosis between virtual and real mobile robots. In J.-C. Heudin, ed., *Virtual Worlds*, vol. 1434 of *Lecture Notes in Artificial Intelligence*, pp. 254–263. Springer-Verlag, Berlin, 1998.

[Moody and Darken 1989] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[Murciano et al. 1997] A. Murciano, J. del R. Millán, and J. Zamora. Specialization in multi-agent systems through learning. *Biological Cybernetics*, 76(5):375–382, 1997.

[Parker and Touzet 2000] L. E. Parker and C. Touzet. Multi-robot learning in a cooperative observation task. In L. E. Parker, G. Bekey, and J. Barhen, eds., *Distributed Autonomous Robotic Systems 4*, pp. 391–401. Springer-Verlag, Berlin, 2000.

[Parrish and Hamner 1997] J. K. Parrish and W. M. Hamner, eds. *Animal Groups in Three Dimensions*. Cambridge University Press, New York, 1997.

[Rose 1991] K. Rose. *Deterministic Annealing, Clustering, And Optimization*. Ph.D. thesis, California Institute of Technology, Pasadena, California, 1991.

[Rose 1998] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[Salustowicz et al. 1998] R. P. Salustowicz, M. A. Wiering, and J. Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 33(2):263–282, 1998.

[Smyth 1996] P. Smyth. Clustering using Monte Carlo cross-validation. In E. Simoudis, J. Han, and U. M. Fayyad, eds., *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 126–133. AAAI Press, 1996.

[Versino and Gambardella 1997] C. Versino and L. M. Gambardella. Learning real team solutions. In G. Weiß, ed., *Distributed Artificial Intelligence Meets Machine Learning: Learning in Multi-Agent Environments*, vol. 1221 of *Lecture Notes in Artificial Intelligence*, pp. 40–61. Springer-Verlag, Berlin, 1997.

[Watkins and Dayan 1992] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.