Bar-Ilan University
Department of Computer Science

Reinforcement Learning of Multi-Robot Coordination Based on Resource Spending Velocity

by

Dan Erusalimchik

Advisor: Prof. Gal Kaminka

Submitted in partial fulfillment of the requirements for the Master's degree
in the department of Computer Science

Ramat-Gan, Israel
April 2009

This work was carried out under the supervision of

Prof. Gal A. Kaminka

Department of Computer Science, Bar-Ilan University.

Abstract

Multi-robot systems researchers have been investigating adaptive coordination methods for improving spatial coordination in teams. Such methods utilize learning to improve selection of the coordination method, given the dynamic changes in density of the robots. Unfortunately, while their empirical success is evident, none of these methods has been understood in the context of existing formal work on multi-robot learning. This paper presents a reinforcement-learning approach to coordination algorithm selection, which is not only shown to work well in experiments, but is also analytically grounded. We present a reward function (*Effectiveness Index*, EI), that reduces time and resources spent coordinating, and maximizes the time between conflicts that require coordination. It does this by measuring *the resource-spending velocity*. We empirically show its successful use in stateless reinforcement learning, in several domains, including robots in virtual worlds, simulated robots, and physical AIBO robots executing foraging. In addition, we analytically explore the reasons that EI works well. We show that under some assumptions, spatial coordination opportunities can be modeled as matrix games in which the payoffs to the robots are unknown, but are directly a function of EI estimates. The use of reinforcement learning leads to robots maximizing their EI rewards in equilibrium. We then apply the EI reward function in full multi-state reinforcement learning, and demonstrate that it can be used in settings requiring tight coordination between the robots. This work is a step towards bridging the gap between the theoretical study of interactions, and their use in multi-robot coordination.

Acknowledgments

First of all, I wish to express my gratitude to my advisor Prof. Gal A. Kaminka, for believing in my abilities and giving me opportunity to be a part of the MAVERICK lab. He not only has acquainted me with a robotics, directed my activity and gave invaluable advices and ideas for my research, but also has helped me to understand many basic scientific principles.

I wish to thank my parents Erusalimchiku Vladimir and Galina. Without their love and support, I could achieve nothing that I have. Since childhood, they imparted to me the love of knowledge and always helped me with my inventions and undertakings.

I wish to thank all of students who prepared for exams with me, and who helped me with the solutions of various technical problems, which I have faced during the execution of experiments: Alon Levi, for its councils about computer administration, Ari Yakir for help with the experiments in the virtual environment, and Dov Miron and Shai Shlomai, for help with the real robots experiments. I thank all of the MAVERICK lab members for their friendly support.

I would like to also express special gratitude to Profs. Sarit Kraus, Peter Stone, and Mike Bowling for valuable advice on the theoretical part of my research. I especially thank Prof. Moti Schneider who is the first person who acquainted me with Artificial Intelligence, the science to which I am glad to make a small contribution in this thesis.

This research was supported in part by Israeli Science Foundation (ISF) grant #1357/07.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multi-robot systems researchers have been investigating communication-less coordination methods for improving spatial coordination in teams [14, 35, 34, 15]. Such methods attempt to resolve spatial conflicts between team-members, e.g., by dynamic setting of right-of-way priorities [39, 43], constrained avoidance of conflict areas [6], territorial separation [36, 12, 21], or role-based priorities [29]. It is accepted that no one method is always best [13, 11, 34]; rather, the best method depends on the density of the robots and other dynamically-changing settings [34]. All methods reach a point where adding robots to the group (i.e., increasing the density of the robots in space) reduces overall (not just marginal) productivity [36, 35]. However, this point differs between methods [34, 11], and between application contexts. Some approaches to this challenge include use of communications, and methods utilizing additional sensed information (e.g., the position of others around the robot).

A different promising approach to this challenge utilizes algorithm selection methods to dynamically select the best non-communicating coordination method, given the changing settings of the robots. For instance, Toledo and Jennings [11] propose a method based on reinforcement learning, where fixed coordination methods are switched to accommodate dynamic changes to the environment. More recently, Rosenfeld et al. [34] advocated allowing each robot to individually switch coordination methods to reduce its own estimated resource costs. They have shown that this results in combinations of heterogeneous coordination methods, that are significantly better than any single method selected uniformly. In general, all of these adaptive coordination methods have demonstrated much success in multiple domains of interest.

This thesis makes several distinct contributions. First, it presents a novel reward function, called *Effectiveness Index* (EI), which is used in a stateless reinforcement-learning approach to coordination algorithm selection in multi-robot tasks. The key idea in EI is to reduce time and resources spent coordinating, and maximize the time between conflicts that require coordination. It does this by measuring *the resource-spending velocity* (the resource "burn rate"). The use of reinforcement learning minimizes this velocity. One nice feature of EI is that it does not require any knowledge of the task involved, and is thus domain-independent. We empirically show that EI succeeds in improving multi-robot coordination in several domains, including robots in virtual worlds, simulated robots, and physical AIBO robots executing foraging. In particular, use of EI is shown to improve on results from previous techniques.

As an additional contribution, we analytically explore the reasons and assumptions underlying the success of EI. We formalize the experiment domains as extensive-form games. We show that under some assumptions, these games can be modeled as matrix games in which the payoffs to the robots are unknown, but are directly a function of EI estimates. The use of reinforcement learning leads to robots maximizing their EI rewards in equilibrium. We believe that this work represents an important step towards bridging the gap between the theoretical study of interactions (via game theory), and their use to explain and inform multi-robot coordination.

Finally, we explore the use of EI in more complex settings. First, we show that EI can be used in a state-based policy with the familiar Q-Learning algorithm. We demonstrate its use in highly-constrained settings, where the coordination between agents is critical to the success of the team as a whole. In these settings, the use of EI in multi-state policies is shown to be superior to its use in a stateless method discussed earlier in the thesis, for foraging. Second, we show that EI can be used to indirectly learn the appropriate conditions for its own use, e.g., the conditions under-which a conflict is declared (which allows a coordination method to be selected). This overcomes the need for separate learning of the conditions under-which EI is to be used. Third, we apply EI in a commercial virtual environment, where synthetic agents learn to select different routes to a target location, so as to minimize arrival times. The surprising result of this application is that EI-based learning works well, even though the coordination method is fixed, and it is the route travel durations that are being learned (i.e., a task-oriented selection).

This thesis is organized as follows. Chapter 2 discusses background and related work. Chapter 3 introduces the Effectiveness Index (EI) measure and relevant notation. Chapter 4 discusses the use of EI in stateless reinforcement learning. This chapter leaves two immediate challenges open, which we cover in other chapters: (i) A theoretical explanation as to why stateless EI work in multi-agent reinforcement learning (Chapter 5); and (ii) How can EI be used in more complex settings, involving multiple states and parametrized conditions on its use (Chapter 6). Finally, Chapter 7 concludes and presents directions for future work.

Chapter 2

Background and Related Work

Our work is related to several lines of investigations in multi-agent and multi-robot systems research. Some of our work in this thesis is evaluated and contrasted with other work on *multi-robot foraging,* a standard multi-robot coordination problem, discussed in Section 2.1. However, we generalize our results beyond foraging, and indeed show that it fits into the area of multi-agent reinforcement learning and game-theory (Section 2.2).

2.1 Multi-Robot Coordination and Foraging

Foraging is a canonical task in multi-robot systems, where multiple robots are all situated within a common work-area. Their task is to pick up small objects (*pucks*) spread through the work area, and bring them to a goal location (typically at the center of the work area). They continue in the process of searching for pucks, and bringing them to the goal location, until time runs out; the number of pucks is typically not known in advance. Scoring is team-based: Robots are evaluated based on the total number of pucks which have been successfully brought to the goal location. In most variants of multi-robot foraging, robots are not allowed to explicitly communicate with each other.

Many investigations have utilized multi-robot foraging as a test problem with which to evaluate coordination methods that do not utilize communications; when communication possible, other methods apply (see, e.g., [33]. Communication-less coordination is inherent to multi-robot foraging, as robots tend to collide with others when leaving and entering the goal area, and when searching through the environment [35, 36]. Indeed, a number of non-adaptive coordination methods have been published and demonstrated in multi-robot foraging; a few of these are used extensively in this thesis.

The *noise* method is described by Balch and Arkin in [6]. When using this method, a virtual repulsion force is projected into the movements of a robot when it is about to collide, to make it change its heading. Some noise is injected as well, to prevent being stuck in a local minimum. The *aggression* method was proposed by Vaughan et al. [39]. It works by having robots who are about to collide stop in their tracks, and randomly pick either "meek" or "aggressive" behaviors. These would cause the robots to back away (meek) for a period of time or attempt to move forward ("aggressive"). As this is done with every action cycle, the probability of actual collision is very low. Finally, the the *repel* method causes colliding robots to back away for a given amount of time, essentially reversing their course for the period. Repel is essentially an enforced "meek" behavior selection for all robots; it is described in [34].

It has been repeatedly shown that no one coordination method is best for foraging [35, 36], or indeed for other tasks requiring coordination [13]; rather, the effectiveness of the method is dependent on group size. However, different methods stop being effective with different group sizes [34, 11]. Thus several researchers have described ways of adapting the coordination method to the group size.

Fontan and Matarić [36, 12] have proposed a method of coordination that attempts to prevent collisions from taking place, by allocating robots to different regions. A *bucket brigade* algorithm is used to transfer pucks from one region to the next, until the puck is delivered to the goal region. The allocation of robots to regions is adaptive, in the sense of taking the number of robots into account. To maintain coordination, the robots require knowing of the liveness of other robots, in contrast to our work. Similar ideas of territorial division have been developed for other tasks, such as area patrolling [10, 3], and coverage [21, 16, 20]. Later work in foraging, by Ostergaard et al. [29], has examined ways of adapting the bucket-brigade to the number of robots possible, without explicit knowledge of the liveness of robots. They thus adapt a particular coordination method to the scale of the team. In contrast, our work has tackled both adapting a specific method, as well as choosing between methods.

A different approach, closely related to ours, is based on coordination algorithm selection. Rosenfeld et al. [34, 32] presented a method that adapts the selection of coordination methods by multi-robot teams, to the dynamic settings in which team-members find themselves. The method relies on measuring the resources expended on coordination, using a measure called Combined Coordination Cost (*CCC*); however, it ignores the gains accumulated from long periods of no coordination needs, in contrast to our work. Similarly to our work, the adaptation is stateless, i.e., has no mapping from world state to actions/methods. Instead, the CCC is estimated at any given point, and once it passes pre-learned (learned offline) thresholds, it causes dynamic re-selection of the coordination methods by each individual robot, attempting to minimize the CCC. In contrast, all our learning and adaption is done on-line.

Interference [15, 14] is a closely related measure to CCC, and can be seen as a special case of it: It measures the amount of time spent on coordination, but not other resource usage costs (as in CCC) or frequency of the need to coordinate (as in our work). Goldberg and Matarić [15] use it to evaluate and compare different organizational solutions for foraging, i.e., different task-execution methods: In the first, all robots are homogeneous, and play the same role (this is the same organization we evaluate in this thesis, and also studied by Rosenfeld et al. [34]); in the second, robots are divided into different behavioral groups ("castes"), where the behavior of robots are decided by the group they belong to (similar in principle to [36]); in the final organizational solution, all robots are identical, but explicitly cooperate with each other by stopping their motions whenever one of them picks a puck. Goldberg and Matarić have shown that the first method is superior to others, and has the smallest interference value. In contrast, we focus on only one task-execution method (corresponding to the first, winning, solution they present), and study the use of alternative coordination methods within it.

Following up on earlier work on the aggression method [39], Zuluaga and Vaughan [43] have shown that choosing aggression level proportional to the robot's task investment can produce better overall system performance compared to aggression chosen at random. Thus biasing the random selection of the "aggressive" behavior to the effort already spent on bringing the puck home improves overall performance. This result is compatible with our findings. However, our work on Effectiveness Index relies solely on task-independent resource measurement.

2.2 Multi-Agent Reinforcement Learning

Since Sutton and Barto's introduction of reinforcement learning (RL) [37], there have been, of course, numerous investigations of reinforcement learning in a wide variety of settings and forms. A majority of these has focused on single-robot learning, which is beyond the scope of this paper. In general, RL is used to learn a policy, i.e., a mapping between states and actions to take when the robot is in these states. This is done by relying on a reward function, which is given to the RL algorithm.

Most investigations of reinforcement learning in multi-robot settings have focused on improving the learning mechanisms (e.g., modifying the basic Q-learning algorithm), and utilized task-specific reward functions. We briefly discuss these below. Two recent surveys are provided in [42, 17].

Matarić [28] discusses three techniques for using rewards in multi-robot Q-learning: A local performance-based reward (each robot receiving reward for its own performance, and per its own goals), a global performance-based reward (all robots receive reward based on achievement of team goals), and a heuristic strategy referred to as shaped reinforcement. Shaped reinforcement, which was developed by Mataric, provides a heuristic function that combines rewards based on local rewards, global rewards and coordination interference of the robots. Balch [7] reports on using reinforcement learning in individual robot behavior selection. His work uses a two-layer architecture: The lowest layer is a reactive motor schema (similar to the potential fields), which implement atomic behaviors. The selection of schemas is made at the upper level, using a Q-Learning mechanism. The rewards for the selection were carefully selected for each domain and application, in contrast to our work. Indeed, in other work, Balch [4] discusses considerations for task-dependent reward functions for reinforcement learning in multi-robot settings. Balch shows that the choice of reward function influences the behavioral diversity, and group performance in a variety of tasks, including foraging and soccer.

Kok and Vlassis [24] discuss a technique for propagating rewards among cooperative robots, based on the structure of the dependence between the robots. However, they too assume that the reward function is given as part of the task.

In contrast to Balch's [7], and Kok and Vlassis's [24] work, we explore a domain-independent reward function, based on minimizing resource use, and use them in selecting between coordination methods, rather than task behaviors.

Maes and Brooks[26] describe an algorithm which allows a behavior-based single robot to learn to move by a correct coordination of behaviors that control each of the robot's six legs. This learning is based on the perception of a positive and a negative feedback and according to the philosophy that the architecture is fully distributed. Each behavior in all six legs searches for correlation between action and positive feedback (i.e. appropriate action) and tries to understand what conditions are robust enough for the action (i.e. maximize probability of a positive feedback and minimize probability of negative). Although this work demonstrate significant results in coordinated behaviors, the algorithm relies on essentially instantaneous communications between the behaviors, for their coordination (and feedback). However, when in multi-robot settings, one cannot assume perfect, instantaneous communications. Our proposed work complements Maes and Brooks' work: While they focus on distributed reinforcement learning based on a given set of rewards, we focus on general reward functions that may be useful for re-reinforcement learning.

An interesting technique, Learning Momentum (LM), was demonstrated by Clark, Arkin and Rome [30] for a single robot, and extended for multi-robots by Lee and Arkin [25]. The main idea of LM is behavior's weight's modification. Weights are adjusted by a gradient descent method during the learning process and then the behavior manager is fusing behaviors accordingly to them. All of the behaviors, which LM is working with, have different goals, and thus the technique learns to select between competing goals. This is the main difference from our approach, which is intended for selecting one best behavior from a set of different behaviors with a same goal.

Excelente-Toledo and Jennings [11] propose a mechanism for selecting between coordination methods, based on their effectiveness and importance. They define a number of general characteristics of coordination methods, including the conditions (and cost for achieving them) for the application of each method, the cost of the algorithm, and their likelihood of success. Each of these characteristics manually receives a qualitative grade (high, medium, low), during an offline evaluation period. During run-time, the cost of each coordination method (with the additional cost of achieving its application conditions), and the likelihood of success are used as the basis for selection. Similarly to this work, we utilize the concepts of method costs and success, though the process is automated, and measures these factors quantitatively *on-line*. Reinforcement learning is used as the basis for coordination method selection.

Hogg and Jennings [18] examine economic strategies balancing individual and cooperative goals. They utilize models of the other robots and methods of coordination to adjust the sociability of an robot. They utilize one-stage Q-learning for learning models of other robots, to improve predictions of coordination results. In contrast, we present a reward function that can be utilized to learn to individually select between coordination methods, without modeling the other robots.

Kapetanakis and Kudenko [22] present the FMQ learning algorithm. This algorithm is intended for coordination learning in one-stage MDP games. FMQ is a modified regular Q-Learning method for one-stage games and this modification is based on the Boltzmann strategy. They then examine how an robot that uses FMQ learning technique may influence other robot's effectiveness of learning, when the latter uses a simple Q-learning algorithm [23]. This method does not use communication or monitoring of the other robots' actions, but is based on the assumption that all of the robots are getting the same rewards. The reward functions are assumed to be given. In contrast, we focus on the reward functions to be used, rather than the learning algorithm.

Wolpert et al. [41, 40] developed the COIN reinforcement-learning framework. Each agent's reward function is based on *wonderful life utility*, the difference between the group utility with the agent, and without it. Similarly to Wolpert et al., our study focuses on the reward function, rather than the learning algorithm; and similarly, we focus on functions that are *aligned* with global group utility. However, our work differs in several ways. First, we distinguish utility due to coordination, from utility due to task execution. Second, our reward function involves also the time spent coordinating and time spent executing the task. Finally, we contribute in this paper a game-theoretic perspective on multi-robot tasks.

Tumer [38] continued developing the concept of aligned reward functions, and discusses methodology for defining reward functions for reinforcement learning in multi-robot settings. The properties necessarily for these functions require independence of individual rewards, and alignment with the global utility of the group. Tumer presents two individual reward functions that are consistent with these requirements. utility functions presented in the paper. Both of them contain part of global utility, and thus require full observation or communications with other robots, in contrast to our work.

Agogino and Tumer [1, 2] construct fitness functions for evolutionary adaptation of a multi-component (e.g., multi-robot) system. This approach emphasizes (i) the need for correlation (alignment) between individual fitness of agent and the general fitness function of the entire system success; as well as (ii) independence of the individual functions of the actions of other agents (which affects the speed of adaptation). Agogino and Tumer provide several domain-dependent functions for the solution of a specific problem (searching for Points Of Interest). Our work differs in several points: First, we did not apply our EI calculations to evolutionary learning (adaptation); second, we distinguish the task-oriented and coordination components of the global function.

Chapter 3

Effectiveness Index (EI)

We first cast the problem of selecting coordination algorithms as a reinforcement learning problem (Section 3.1). We then introduce the effective index (EI) reward function in Section 3.2.

3.1 Coordination Algorithm Selection

Coordination prevents and resolves conflicts among robots in a multi-robot system. Such conflicts can emerge as results for shared resource (e.g., space), or as a result of violation of joint decisions by team-members. Many coordination algorithms (protocols) have been proposed and explored by MRS researchers [12, 29, 36, 39]. Not one method is good for all cases and group sizes [34]. However, deciding on a coordination method for use is not a trivial task, as the effectiveness of coordination methods in a given context is not known in advance.

We focus here on loosely-coupled application scenarios where coordination is triggered by conflict situations, identified through some mechanism (we assume that such a mechanism exists, though it may differ between domains; most researchers simply use a pending collision as a trigger). Thus the normal routine of a robot's operation is to carry out its primary task, until it is interrupted by an occurring or potentially-occurring conflict with another robot, which must be resolved by a coordination algorithm. Each such interruption is called *a conflict event*. The event triggers a coordination algorithm to handle the conflict. Once it successfully finishes, the robots involved go back to their primary task. Such multi-robot scenarios include foraging, search and exploration, and deliveries.

Let $A = \{a_1 \ldots, a_i, \ldots, a_N\}, 1 \leq i \leq N$ be a group of N robots, cooperating on a group task that started at time 0 (arbitrarily) lasts up-to time T (A starts working and stops working on the task together). We denote by $T_i = \{c_{i,j}\}, 0 \leq j \leq K_i$ the set of conflict events for robot i, where $c_{i,j}$ marks the time of the beginning of each conflict.

The time between the beginning of a conflict event j, and up until the next event, the interval $I_{i,j} = [c_{i,j}, c_{i,j+1})$, can be broken into two conceptual periods: The *active* interval $I^a_{i,j} = [c_{i,j}, t_{i,j})$ (for some $c_{i,j} < t_{i,j} < c_{i,j+1}$) in which the robot was actively investing resources in coordination, and the *passive* interval $I^p_{i,j} = [t_{i,j}, c_{i,j+1})$ in which the robot no longer requires investing in coordination; from its perspective the conflict event has been successfully handled, and it is back to carrying out its task. By definition $I_{i,j} = I^a_{i,j} + I^p_{i,j}$. We define the *total active time* as $I^a = \sum_i \sum_j I^a_{i,j}$ and the *total passive time* as $I^p = \sum_i \sum_j I^p_{i,j}$.

Our research focuses on a case where the robot has a nonempty set M of coordination algorithms to select from. The choice of a specific coordination method $\alpha \in M$ for a given conflict event $c_{i,j}$ may effect the active and passive intervals $I^a_{i,j}, I^p_{i,j}$ (and possibly, other conflicts; see next section). To denote this dependency we use $I^a_{i,j}(\alpha)$,$I^p_{i,j}(\alpha)$ as active and passive intervals (respectively), due to using coordination method α. Figure 3.1 illustrates this notation.

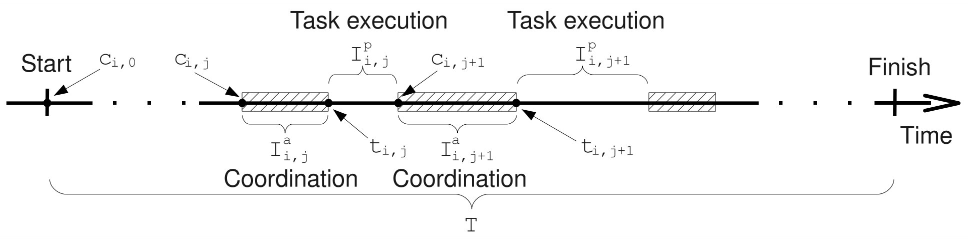


Figure 3.1: Illustration of task time-line, from the robots' perspective. Task execution is occasionally interrupted by the requirement to spend resources on coordination.

We define the problem of coordination algorithm selection in terms of reinforcement learning. We assume each robot tries to maximize its own reward by selecting a coordination method α.

3.2 Effectiveness Index

We call the proposed general reward for coordination *Effectiveness Index* (EI). Its domain independence is based on its using three intrinsic (rather than extrinsic) factors in its computation; these factors depend only on internal computation or measurement, rather than environment responses.

3.2.1 The cost of coordinating. The first factor we consider is the cost of internal resources (other than time) used by the chosen method. This is especially important in physical robots, where battery life and power are a concern. We argue that such internal estimate of resource usage is practical:

- First, some resource usage is directly measurable. For instance, energy consumption during coordinated movement (e.g., when getting out of a possible collision) or communications (when communicating to avoid a collision) is directly measurable in robots, by accessing the battery device before and after using the coordination algorithm.

- Second, resource usage may sometimes be analytically computed. For instance, given a the basic resource cost of a unit of transmission, the cost of using a specific protocol may be analytically computed (as it is tied directly to its communication complexity in bits).

We denote by C_i^C the total cost of coordination, of robot i. It can be broken into the costs spent on resolving all conflicts $C_i^C = \sum_j C_{i,j}^C$. $C_{i,j}^C$ is similar to other measures suggested previously, but excludes the cost of time and resources spent before the conflict (unlike [34]), and is limited to only considering individual intrinsic resources (unlike [43]).

Let us use $cost_i(\alpha, t)$ to denote the costs due to using coordination method $\alpha \in M$ at any time t, by robot i, during the lifetime of the task. $cost_i(\alpha, t)$ is a function mapping the selection of a coordination method α at time t to the cost of applying it, in terms of resources available to the robot. We expect this cost to vary with the chosen method α (as different methods have different costs of application), and with time (as applying the method at a time when robots are about to collide is different from applying it when the robots are far apart). The function is not necessarily known to us a-priori (and indeed, in this research, is not).

Using the function $cost_i(\alpha, t)$ we define the $C^C_{i,j}$ of a particular event of robot i at time $c_{i,j}$:

$$\begin{aligned} C^C_{i,j}(\alpha) &= \int_{c_{i,j}}^{t_{i,j}} cost_i(\alpha, t)\ \mathrm{d}t + \int_{t_{i,j}}^{c_{i,j+1}} cost_i(\alpha, t)\ \mathrm{d}t \\ &= \int_{c_{i,j}}^{t_{i,j}} cost_i(\alpha, t)\ \mathrm{d}t \end{aligned} \tag{3.1}$$

$C^C_{i,j}$ is defined as the cost of applying the coordination algorithm during the active interval $[c_{i,j}, t_{i,j})$ and the passive interval $[t_{i,j}, c_{i,j+1})$. However, the coordination costs during the passive interval are zero by definition.

3.2.2 The time spent coordinating. The main goal of a coordination algorithm is to reach a (joint) decision that allows all involved robots to continue their primary activity. Therefore, the sooner the robot returns to its main task, the less time is spent on coordination, and likely, the robot can finish its task more quickly. Thus, smaller I^a_i is better. Note that this is true regardless of the use of other resources (which are measured by C^C_i). Even if somehow other resources were free, effective coordination would minimize conflict-resolution time.

We thus define the *Active Coordination Cost* (ACC) function for robot i and method α at time $c_{i,j}$, that considers the *active time* in the calculation of coordination resources cost:

$$ACC_{i,j}(\alpha) \equiv I^a_{i,j}(\alpha) + C^C_{i,j}(\alpha) \tag{3.2}$$

3.2.3 The frequency of coordinating. If there are frequent interruptions to the robot's task in order to coordinate, even if short-lived and inexpensive, this would delay the robot. We assume (and the results show) that good coordination decisions lead to long durations of non-interrupted work by the robot. Therefore, the frequency of coordination method's use is not less important than the time spent on conflict resolving. Thus, larger $I^p_{i,j}$ is better.

We thus want to balance the total active coordination cost $ACC_i = \sum_j ACC_{i,j}$ against the frequency of coordination. We want to balance short-lived, infrequent calls to an expensive coordination method against somewhat more frequent calls to a cheaper coordination method.

We therefore define the Effectiveness Index of robot i, of conflict j, due to using coordination method $\alpha \in M$ as follows:

$$EI_{i,j}(\alpha) \equiv \frac{ACC_{i,j}(\alpha)}{I^a_{i,j}(\alpha) + I^p_{i,j}(\alpha)} = \frac{I^a_{i,j}(\alpha) + C^C_{i,j}(\alpha)}{I^a_{i,j}(\alpha) + I^p_{i,j}(\alpha)} \tag{3.3}$$

That is, the effectiveness index (EI) of a coordination method α during this event is the velocity by which it spends resources during its execution, amortized by how long a period in which no conflict occurs. Since greater EI signifies greater costs, we typically put a negation sign in front of the EI, to signify that greater velocity is worse; we seek to minimize resource spending velocity.

Chapter 4

Stateless Reinforcement Learning Using EI

We now turn to evaluate the use of EI in reinforcement learning settings. This chapter introduces the use of EI in stateless reinforcement learning, where there EI is used as the basis for the reward associated with selecting a coordination method. The stateless reinforcement learning algorithm is introduced in Section 4.1. We then turn to survey experiment results in multiple domains, supporting the use of stateless EI in multi-robot team tasks.

In Sections 4.2 and 4.3, we use EI in a canonical multi-robot systems task, called foraging. We show that the use of EI leads to improved performance in two independent foraging settings, in simulation (Section 4.2) and with real robots (Section 4.3). We then explore the use of EI in settings somewhat different than previously described. In Section 6.3 presents the use of EI in settings where coordination is implicit in the selection of domain actions, rather than explicit coordination methods.

4.1 EI in stateless Q-Learning

In this paper we use the simple single-state Q-learning algorithm (Algorithm 4.1) to estimate the EI values from the robot's individual perspective. The term *stateless* applies here, as environment or robot states in which coordination methods are selected are not distinguished. In other words, there is no distinction between the different states a robot might be in when selecting a specific coordination method α. Rather, the reward ($-EI$) collected for using α is associated with it no matter where and when it was selected.

The learning algorithm we use is based on the following Q-Learning equation:

$$Q_t(a) = Q_{t-1}(a) + \rho(R_t(a) - \gamma Q_{t-1}(a))$$

where ρ is the learning speed factor, and γ is a factor of discounting. The algorithm uses a constant exploration rate β, for simplicity; there are of course complex—and more sophisticated—methods of changing the exploration rate dynamically, but they are not our focus in this paper. While single agent Q-learning is known to face difficulties in some multi-robot settings [8, 42, 17], we show experimentally that even this simple algorithm can converges to a useful result. We discuss its successes and failures extensively in the next sections. Chapter 5 explores the success of the learning mechanism in much greater detail. In Algorithm 4.1 below, we use the following variables.

- $Q[\alpha]$ is a table with the learned EI for all coordination methods $\alpha \in M$.
- $SolvingTime$ is the time when the conflict ends ($t_{i,j}$).
- $BgnCycleTime$ is the time when the conflict is started ($c_{i,j}$).
- $EndCycleTime$ is the time when the previous conflict cycle ends ($c_{i,j-1}$).
- $Cost$ is a cost spent resolving the conflict ($ACC_{i,j}$).
- $Counter$ keeps track of conflicts (j).

4.2 Foraging in TeamBots Simulation

As previously described in Chapter 2, foraging is a canonical task in multi-robot systems research, and has been studied extensively. In this task robots locate target items (pucks) within a defined work area, and deliver them to a goal region.

Algorithm 1 Single-State EI-Based Adaptation

```
Require: $\beta \in [0, 1]$ – rate of exploration vs exploitation
Require: $\rho \in [0, 1]$ – learning speed factor
Require: $\gamma \in [0, 1]$ – learning discount factor
Require: $M$ – a set of coordination algorithms
Require: A way of measuring accumulating coordination resource costs
Require: A way of checking current time or step number in case of discrete system.
1:  $Count \leftarrow 0$
2:  for all $\alpha \in M$ do
3:    $Q[\alpha] \leftarrow \text{random}([0, 1])$
4:  while robot is active do
5:    WAIT FOR CONFLICT EVENT
6:    $Count \leftarrow Count + 1$
7:    $EndCycleTime \leftarrow \text{CurrentTime}()$
8:    if $Count > 1$ {Not First Conflict} then
9:      $t_a \leftarrow SolvingTime - BgnCycleTime$
10:     $t_p \leftarrow EndCycleTime - SolvingTime$
11:     $EI \leftarrow -\frac{t_a + Cost}{t_a + t_p}$
12:     $\alpha \leftarrow SelectedMethod$
13:     $Q[\alpha] \leftarrow Q[\alpha] + \rho \cdot (EI - \gamma \cdot Q[\alpha])$
14:   $BgnCycleTime \leftarrow EndCycleTime$
15:   if $\beta > \text{random}([0, 1])$ then
16:     $SelectedMethod \leftarrow \text{random}(M)$
17:   else
18:     $SelectedMethod \leftarrow \text{argmax}_{\alpha \in M} Q[\alpha]$
19:   EXECUTE $SelectedMethod$ {Record resource spending to $Cost$}
20:   $SolvingTime \leftarrow \text{CurrentTime}()$
```

Because there is typically a single goal region, and the pucks are spread throughout the work area, the goal area becomes congested as robots move in and out of it as they bring in new pucks, and leave to collect new ones, respectively. A number of coordination methods have been explored for foraging (see Chapter 2). Here we utilize only on a subset of methods as the basis for the use of EI.

We follow Rosenfeld et al.'s work [34]. We used the TeamBots simulator [5] to run experiments. Teambots simulated the activity of groups of Nomad N150 robots in a foraging area that measured approximately 5 by 5 meters. We used a total of 40 target pucks, 20 of which were stationary within the search area, and 20 moved randomly. For each group, we measured how many pucks were delivered to the goal region by groups of 3,5,15,25,35,39 robots within 10 and 20 minutes. We averaged the results of 16–30 trials in each group-size configuration with the robots being placed at random initial positions for each run. Thus, each experiment simulated for each method a total of about 100 trials of 10 and 20 minute intervals; this was done to evaluate the effect of training time on the use of the EI method.

We compare the EI method with three types of coordination methods described also in [34]: *Noise* (which essentially allows the robots to collide, but increases their motion uncertainty to try to escape collisions), *Aggression* [39], and *Repel*, in which robots move away (variable distance) to avoid an impending collision. We compare all of these to random coordination algorithm selection (RND), and to the method of Rosenfeld et al. (ACIM) [34].

We present the results of various configurations below, in Figures 4.1–4.12. In all, unless otherwise marked, the X axis marks the group size, and the Y axis marks the number of pucks collected.

Figure 4.1 shows that given no resource limitations, the EI method is as good as ACIM (and Repel) which provides the best results, though it has not used prior off-line learning. All methods were run for 20 minutes. Resources were not limited, and had no cost.

Figures 4.2–4.4 show the advantage of EI over ACIM when resource costs apply. Here, motion costs a unit of simulated fuel for each simulation tick. There is a total of only 500 units of fuel available for 20 simulation minutes; robots stop moving (*"die"*) as their fuel runs out.

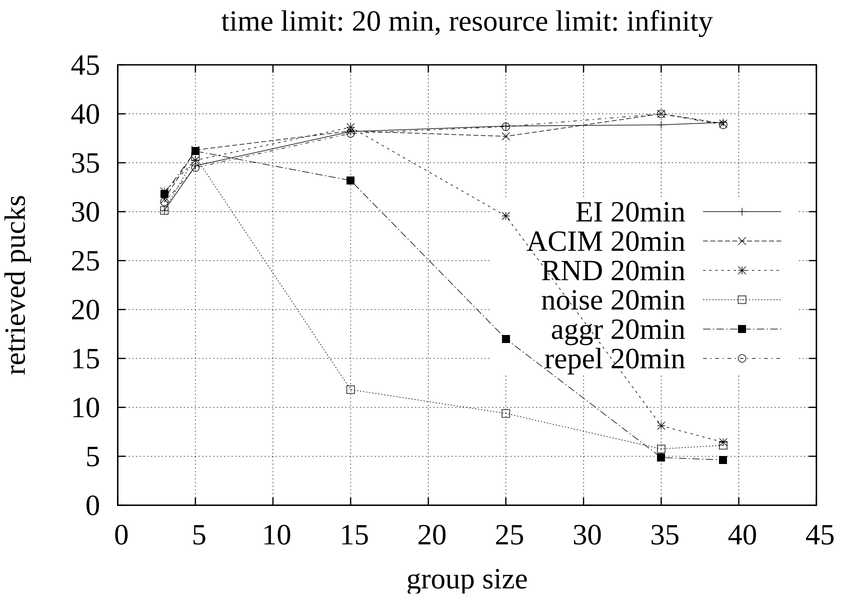


Figure 4.1: Simulated foraging: $T = 20$, no resource limits.

First, we contrast ACIM, RND, and EI under these severe constraints. The line marked *ACIM(t:1,f:0)* marks the performance of the ACIM when it was not trained off-line with these costs in effect, but rather assumed that fuel is free (i.e., as in the previous set of results). The *ACIM(t:.7,f:.3)* line shows the performance of ACIM when the CCC method assigns a non-negative weight to the cost of fuel. The lines *EI(no fuel)* and *EI(with fuel)* show the analogical performance of the EI methods when the fuel is not taken into account, or is taken into account. The performance of random coordination method selection is presented, to serve as a baseline. The figure shows that when ACIM takes fuel costs into account, it performs well. But when it does not, its performance is very low; indeed, it is lower than random. On the other hand, EI always performed well, with explicit knowledge of fuel costs or without.

Let us consider these results in finer-grain resolution. Figure 4.3 contrasts the performance of EI and RND, with the coordination methods they select from. The figure clearly shows the superior performance of EI with and without a-priori knowledge of fuel-constraints.

Figure 4.4 provides an additional perspective on the same situation. The figure

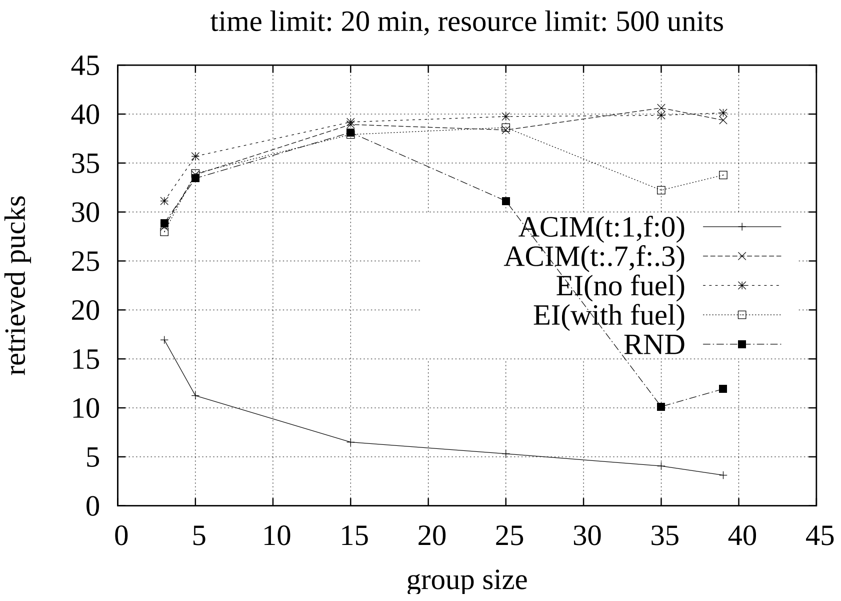


Figure 4.2: Simulated foraging: ACIM and EI, $T = 20$, fuel is limited to 500 units.

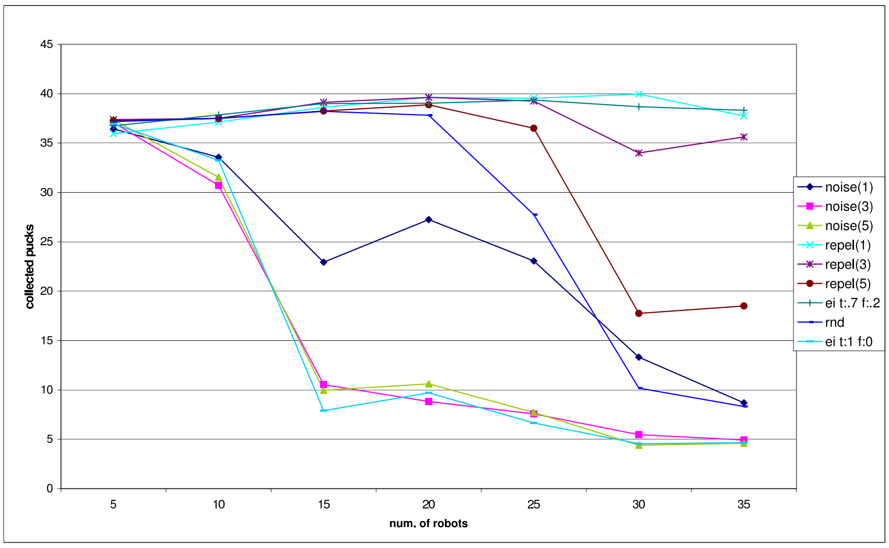


Figure 4.3: Simulated foraging: EI, RND and the base methods, $T = 20$, fuel is limited to 500 units.

shows the number of robots that remain active when time ran out. The two EI methods are the only ones that consistently leave all robots active as time runs out.

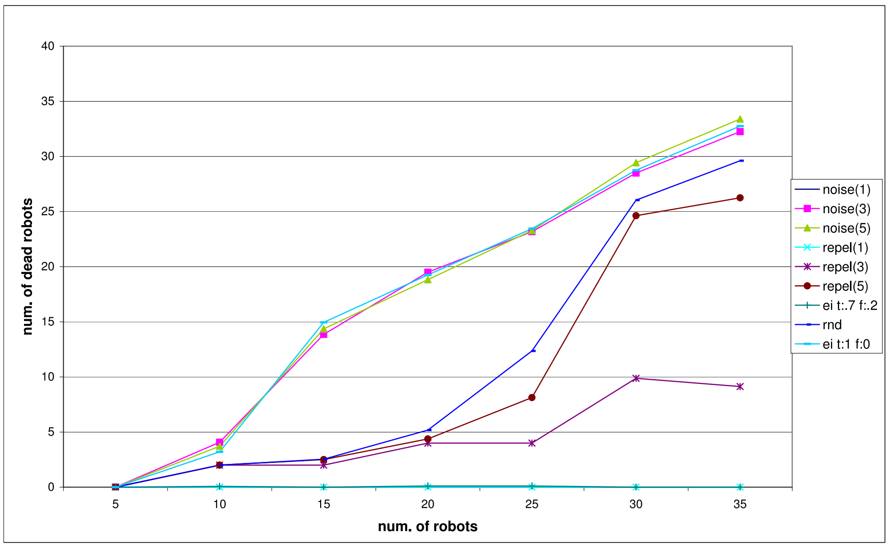


Figure 4.4: Simulated foraging: Number of inactive (dead) robots at $T = 20$, when fuel is limited to 500 units.

Figure 4.5 demonstrates that our use of EI is not limited to selecting between the specific methods described above. Here, we utilized it to selected between multiple variants of the repel method. Repel has a parameter which controls the distance (given in robot radii) to which robots backtrack before attempting to move to their goal location. We use EI to select between these variants.

One weakness of the method we propose is that it relies on an on-line training period, unlike Rosenfeld et al's ACIM method (which trains offline). Indeed, when we reduce the training period from 20 minutes to 10 minutes, the relative effectiveness of EI (compared to ACIM) is greatly reduced. Figures 4.6 and 4.7 show the effect of this reduction in training time, when fuel is unlimited, and when it is limited (respectively).

Given sufficient training time, the fact that EI adapts completely on-line gives it a significant advantage over offline learning methods, such as ACIM. In a final set of experiments, we evaluated the use of EI (contrasting with ACIM) when there is limited fuel, and the coordination methods are "leaky", i.e., they spend

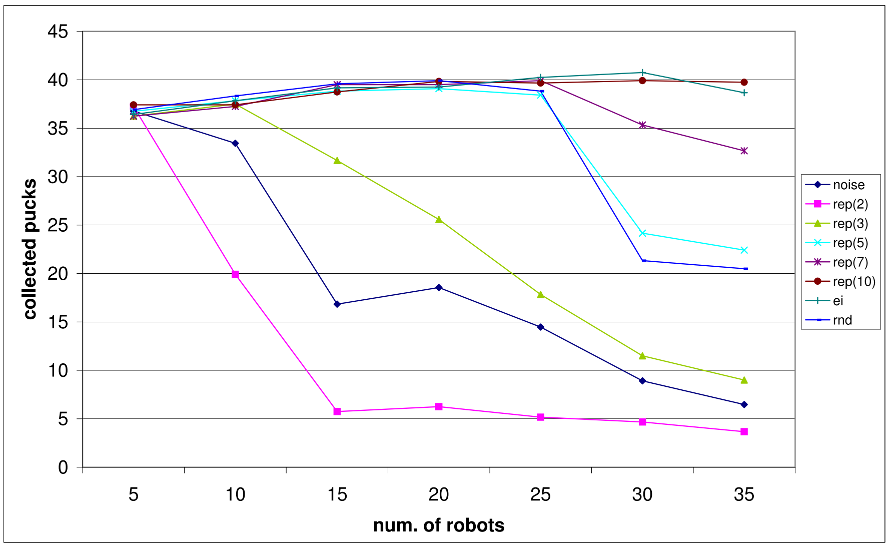


Figure 4.5: Simulated foraging: $T = 20$, Repel at distance of 2,3,5,7 and 10 robot radii. Fuel is unlimited.

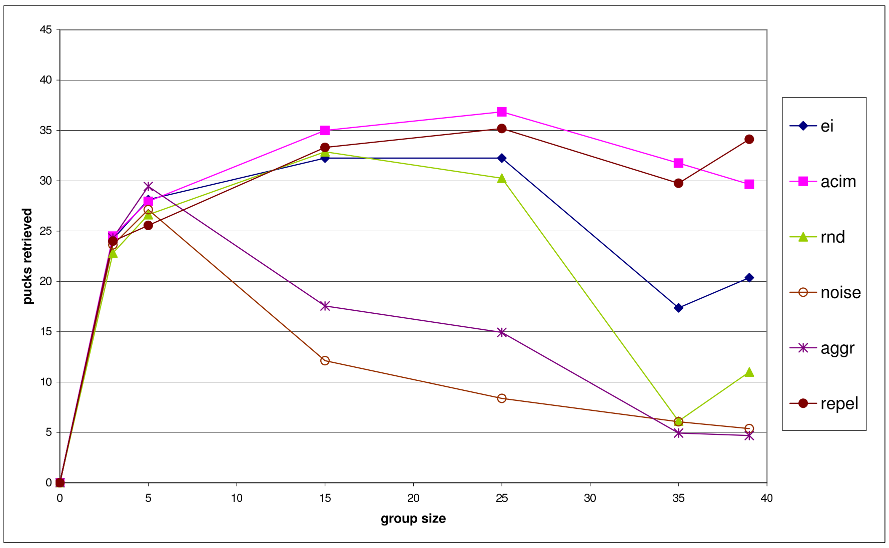


Figure 4.6: Simulated foraging: $T = 10$, Fuel is unlimited.

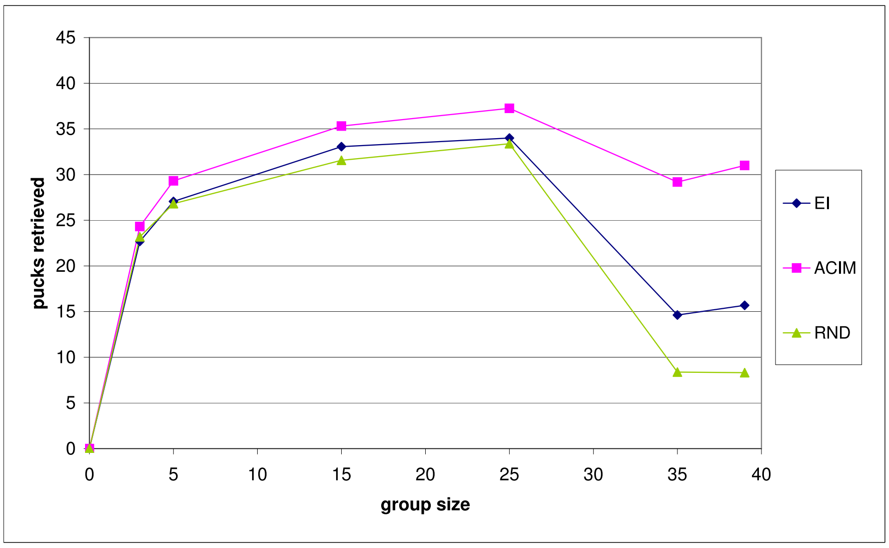


Figure 4.7: Simulated foraging: $T = 10$, Fuel is limited to 500 cycles.

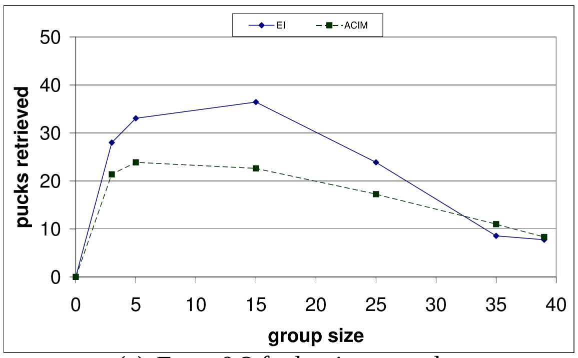
more fuel then they report to the method selection process.

Figures 4.8–4.10 show how ACIM and EI respond to unknown costs. In these figures, we show a comparison between ACIM and EI adaptation, where one of the base coordination methods (aggression, noise, repel, respectively) are "leaking fuel", i.e., they spend some fraction of a fuel unit more than advertised, with *every cycle* of operation. In all of these, the EI methods outperforms ACIM in almost all settings in group sizes, demonstrating its efficacy over the off-line-based method.

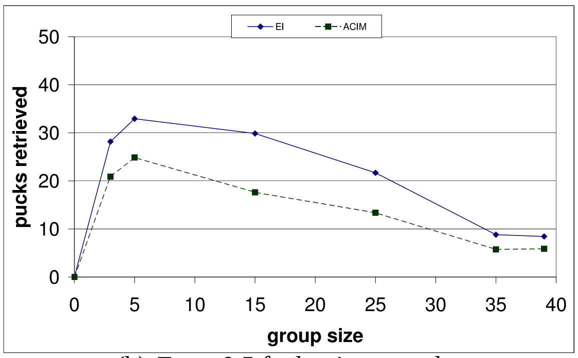
Figure 4.11 summarizes these findings. Here, both EI and ACIM take fuel costs into account, but the actual fuel costs are greater. EI provides significantly better performance in these settings (1-tailed t-test, $p = 0.0027$).

These results are further strengthened by examining the standard deviation of the number of collected pucks when using "leaky" methods. Figure 4.12 shows the average standard deviation of collected pucks, when the coordination method is spending 0.2, 0.5, and 1 fuel units more than it reports, per cycle. The X axis measures the group size. The Y axis measures the average standard deviation across the three conditions. The three sub-figures show the results separately for leaky aggression, noise, and repel. Each sub-figure contrasts ACIM and EI.

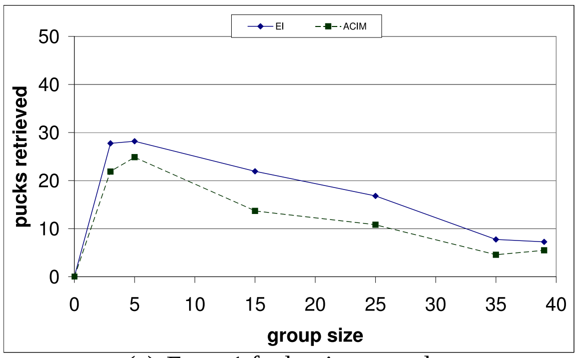
A lower standard deviation indicates reduced sensitivity to misreporting of the coordination methods on their resource spending, and shows that in fact the



(a) Extra 0.2 fuel unit per cycle

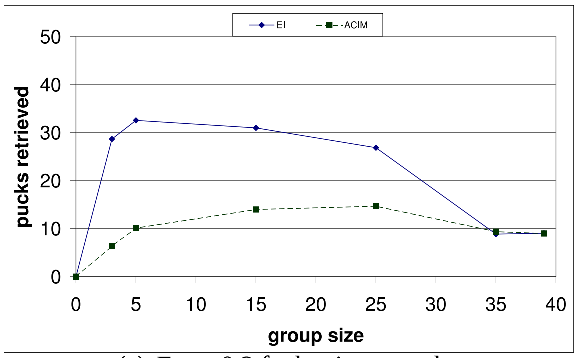


(b) Extra 0.5 fuel unit per cycle

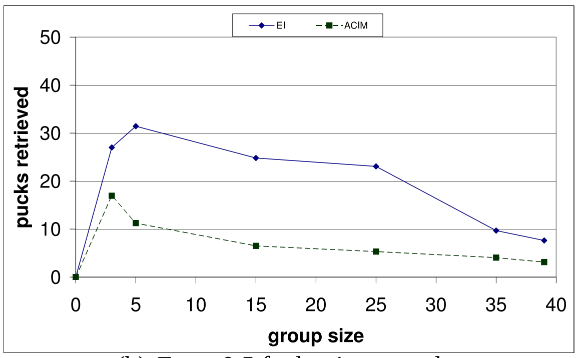


(c) Extra 1 fuel unit per cycle

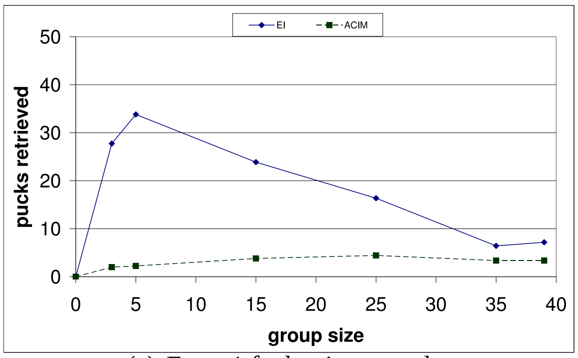
Figure 4.8: Simulated foraging: $T = 20$, Fuel is limited to 500 cycles. The aggression method is spending more than it claims. In all of these, EI was given a weight of 0.3 when calculating fuel costs.



(a) Extra 0.2 fuel unit per cycle

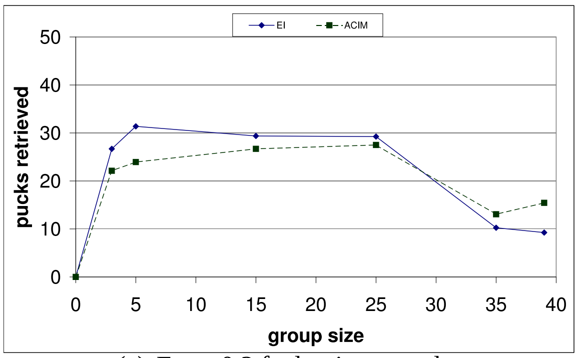


(b) Extra 0.5 fuel unit per cycle

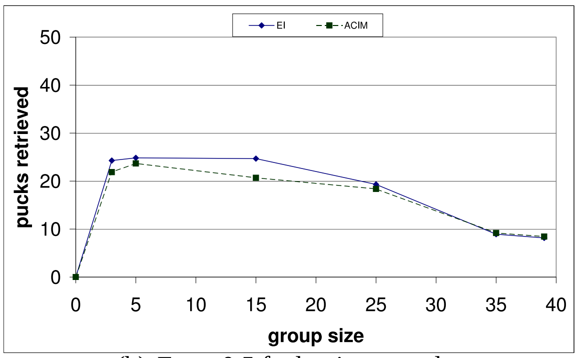


(c) Extra 1 fuel unit per cycle

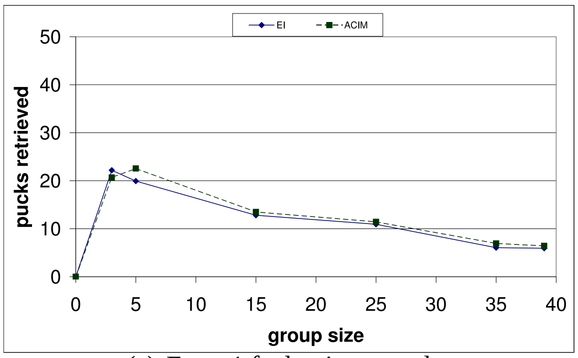
Figure 4.9: Simulated foraging: $T = 20$, Fuel is limited to 500 cycles. The noise method is spending more than it claims. In all of these, EI was given a weight of 0.3 when calculating fuel costs.



(a) Extra 0.2 fuel unit per cycle



(b) Extra 0.5 fuel unit per cycle



(c) Extra 1 fuel unit per cycle

Figure 4.10: Simulated foraging: $T = 20$, Fuel is limited to 500 cycles. The repel method is spending more than it claims. In all of these, EI was given a weight of 0.3 when calculating fuel costs.

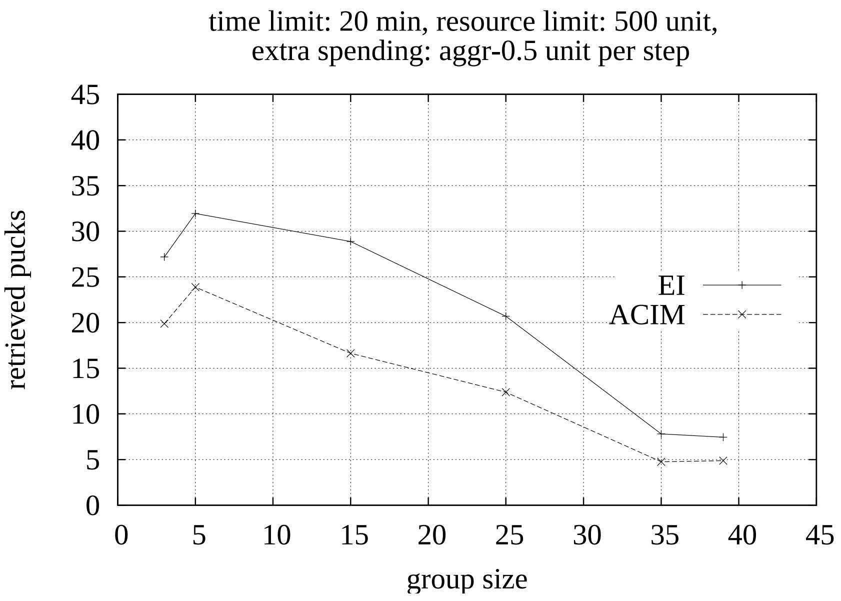
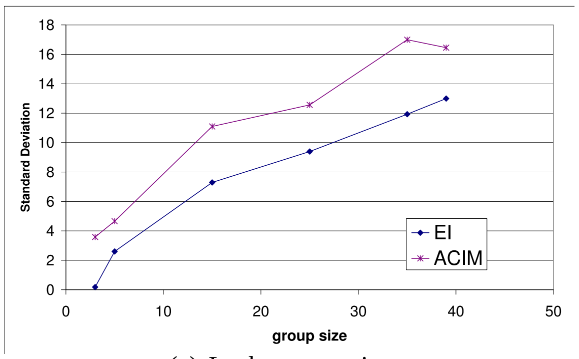
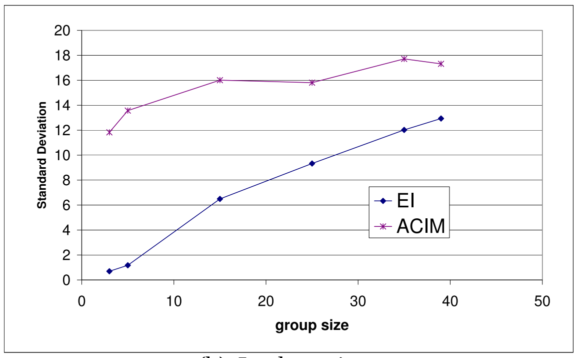


Figure 4.11: Simulated foraging: $T = 20$, resource cost unknown.

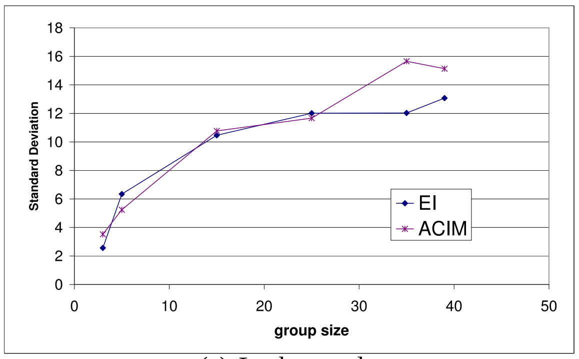
EI-based method is extremely robust to resource spending measurement.



(a) Leaky aggression.



(b) Leaky noise.



(c) Leaky repel.

Figure 4.12: Simulated foraging: The average standard deviation of the number of collected pucks, when $T = 20$, fuel is limited to 500 cycles, and the methods spend 0.2, 0.5, and 1 additional fuel unit for every cycle of operation. In all of these, EI and ACIM were given a weight of 0.3 when calculating fuel costs.

4.3 Foraging in AIBO Robots

We have also utilized EI-based adaptation in foraging experiments with real robots. The application to physical robots presents a number of challenges compared to the application in simulation:

- First, learning times are considerably shorter, due to the limited battery power available. Thus this evaluates the ability of EI to converge to useful values quickly.

- Second, the number of robots is smaller, and thus effects that are due to the number of robots might not come into play. In our experiments, we used three robots (see experiment setup description below).

- Third, there is considerable uncertainty in motion and sensing which might cause robots to misbehave, compared to their simulated counterparts. For instance, robots might become entangled, or may try to use coordination with a fixed obstacle (which they cannot recognize); robots may also face difficulties in leading a puck to the goal location.

The experiment setup is shown in Figure 4.13. Three Sony AIBO robots were placed within a boxed arena, measuring 2m by 2m, and containing four pucks (red aluminum cans, with some additional weight). Each robot is equipped with a camera, for recognizing color, and infra-red sensors that allow it to measure distance (up to 90cm) in the direction of the head. The robots were allowed up to 10 minutes to collect the pucks, by bringing them to the goal area in one of the corners (marked blue). Every puck that was brought to the goal area was physically removed by the experimenters and moved outside of the arena. This ensured that pucks in the goal area did not confuse the robots.

We implemented three coordination method: Two basic coordination methods, *Noise* and *Repel* (described above), and the stateless Q-learning method using the EI reward. Due to the extensive training period required (well beyond battery life), we did not carry out a comparison with Rosenfeld et al.'s ACIM method. We ran ten trials with each of the three methods. However, due to technical failures, some of the data was destroyed and so we were left with eight trials of Noise, nine of Repel, and ten of EI.

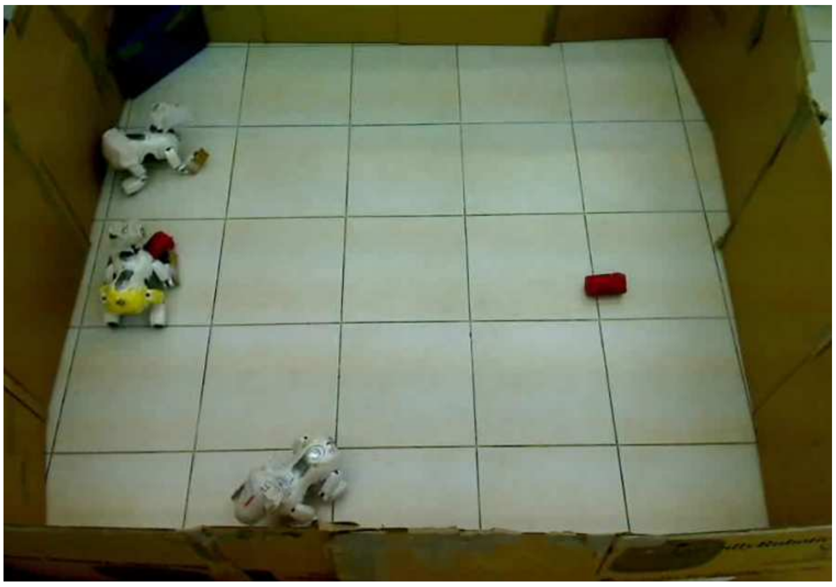


Figure 4.13: Three Sony AIBO robots executing a foraging task in our laboratory. The goal location is in the top left corner. Every puck collected was taken out of the arena.

We faced several challenges in applying EI to the robots. First, we found that the time-limit was not sufficient to allow EI to train. We thus allowed preliminary learning to take place, for approximately 15 minutes. The EI values at the end of this period (which were not optimal) were used as the initial values for the EI trials. Each of the ten trials started with these initial Q table values, and the Q updates continued from this point.

Second, the robots cannot detect conflicts with certainty. For instance, a robot bumping into the walled side of the arena would detect a conflict. Moreover, some collisions between robots cannot be detected, due to their limited sensing capabilities. We solved this by allowing the operator to initiate conflicts by a fixed procedure.

Finally, we found that sometimes robots failed catastrophically (i.e., suffered hardware shutoff). So as to not bias the trials, we measured the average time per puck retrieved. This allowed us to compare runs of different lengths (each up to 10 minutes or the first robot failing catastrophically).

We contrasted the performance of the three groups (Noise, Repel, and EI). Figure 4.14 shows the median number of pucks collected per minute by each of the three methods. The X axis shows the three methods. The Y axis measures the median number of pucks collected.

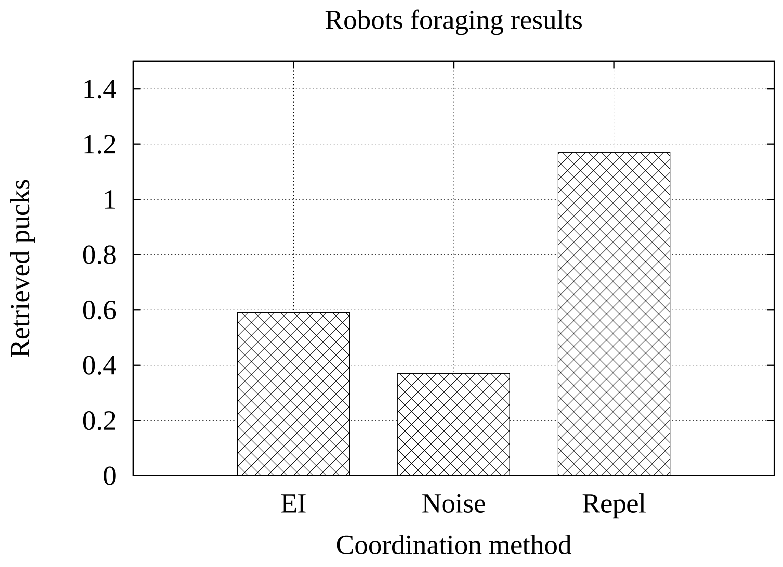


Figure 4.14: AIBO foraging: Pucks collected per minute (median). Higher result is better.

We found that Repel (selected by all three robots) is the best technique. The EI method did better than Noise, but did not reach the results of Repel. To some degree, this is to be expected, because the EI algorithm utilized constant exploration rate (up to 19% of the conflicts of each robot). Thus even under the best of conditions, the EI runs are expected to worse than the best performing method.

We see the same trend in Figure 4.15, which shows the average number of conflicts in the different groups. Here the X axis again shows the three methods. The Y axis measures the mean number of conflicts. We see that the number of conflicts in learning is between Repel and Noise.

To show that indeed the fixed exploration rate had a significant contribution to the results, we also examine the EI-based rankings of the noise and repel methods (i.e., whether the EI values ultimately prefer repel or noise). Figure 4.16 shows the average EI values that were achieved at the end of each run. The X axis shows the robot in question. The Y axis shows negative EI values ($-EI$), thus the 0 line is at the top. We remind the reader that our goal is to minimize EI, i.e., prefer smaller

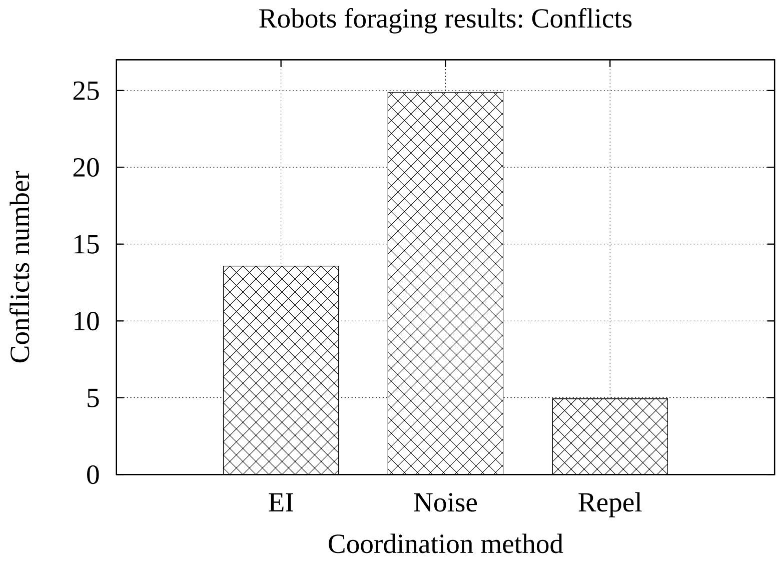


Figure 4.15: AIBO foraging: Mean number of conflicts. Lower result is better.

negative results. Thus a higher results is better. For each robot, we see two bars: One for the EI value of Repel, and one for Noise. We see that in all three robots, the EI values learned for Repel are better (lower). Thus left to choose based on the EI values, all robots would have chosen the Repel method (the optimal choice).

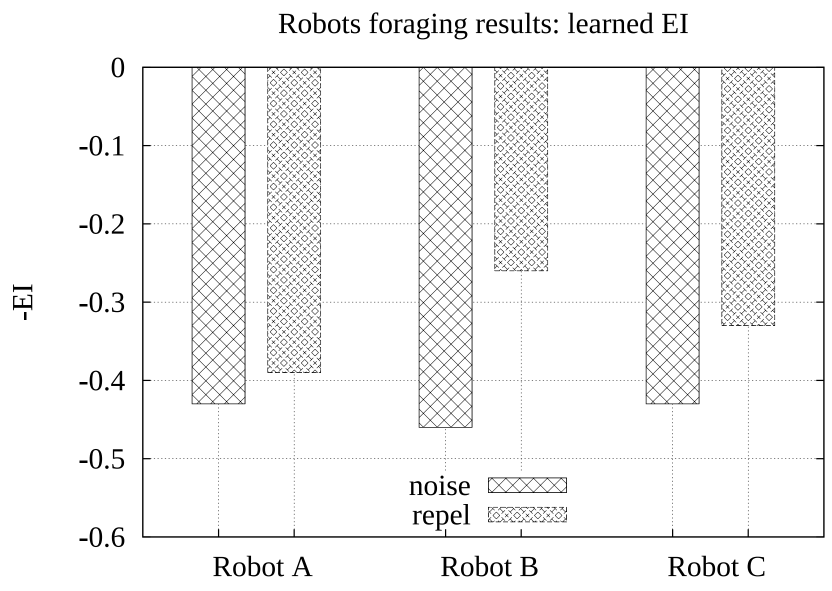


Figure 4.16: AIBO foraging: Negative EI values for Noise and Repel for each of the three robots (higher is better).

Chapter 5

Why EI Works? And Why it Does Not?

We now turn to discuss the use of EI as a reward function, from an analytical perspective. We are interested in exploring the conditions under-which we expect EI to be effective. There are common themes that run through all the tasks in which EI has been successful: (i) loose coordination between the robots (i.e., only occasional need for spatial coordination); (ii) a cooperative task (the robots seek to maximize group utility); and (iii) a fixed time (deadline) for completing the task. We refer to these tasks as *LCT tasks* (Loose-coordination, Cooperative, Timed tasks).

For instance, in foraging, we see that robots execute their individual roles (seeking pucks and retrieving them) essentially without any a-priori coordination. When they become too close to each other, they need to spatially coordinate. The robot all contribute to the team goal, of maximizing the number of pucks retrieved. Moreover, they have limited time to do this. Incidentally, they also have finite number of pucks, which break some of the assumptions we make below. We shall come back to this.

Computing optimal plans of execution for tasks such as foraging is purely a theoretical exercise in the current state of the art. In practice, determining detailed trajectories for multiple robots in continuous space, with all of the uncertainties involved (e.g., pucks slipping from robots' grips, motion and sensing uncertainty), is infeasible. Much more so, when we add the a-priori selection of coordination methods in different points in time. We therefore seek alternative models with which to analytically explore LCT tasks.

5.1 LCT Tasks as Extensive-Form Games

We utilize game theory to analyze LCT tasks. As we have already noted, each individual robot's perspective is that its task execution is occasionally interrupted, requiring the application of some coordination method in order to resolve a spatial conflict, to get back to task execution. Starting in this subsection, we will make a series of simplifying assumptions and analysis steps that will show that it might be possible to view LCT tasks from a game theoretic perspective. Our objective is therefore to contribute an analytical first step towards a formal understanding of why EI works in real-world LCT tasks.

For the initial part this discussion, we assume for simplicity that we limit ourselves to two robots. This is a strong assumption, as in actuality, most often LCT tasks often involve more than two robots. We address this assumption later in this section. In particular, we show that the convergence of EI learning is assured in particular in cases where (many) more than two robots make up the group.

We make an additional assumption, which is not as strong, that conflicts always involve two robots only, and that they are both aware of it, and they both enter the conflict at the same time. This assumption often holds in practice, since when one robot's sensors detect a conflict, most often so does the other robot's.

At first glance, it may seem possible to model LCT tasks as a series of single-shot games (i.e., repeating games), where in each game the actions available to each robot consist of the coordination methods available to it. The joint selection of methods by the two robots creates a combination of methods which solves the conflict (at least temporarily). The payoffs for the two robots include the pucks collected in the time between games, minus the cost of resources (including time) spent making and executing the selected methods. The fact that there exists a time limit to the LCT task in question can be modeled as a given finite horizon.

However, finite-horizon repeating games are not a good model for LCT tasks. In particular, the methods selected by the robots in one point in time affect the payoffs (and costs) at a later point in time. First, the choice of coordination methods at time t affects the time of the next conflict. One coordination method may be very costly, yet reduce the likelihood that the robots get into conflict again; another method may be cheap, but cause the robots to come into conflict often. Second, the robots change the environment in which they operate during the time they are carrying out their tasks, and thus change future payoffs. For instance, robots collect pucks during their task execution time, and often collect those nearest the goal area first. Thus their payoff (in terms of pucks collected) from games later in the sequence is lower than from games earlier on.

We thus utilize a model of LCT tasks as extensive-form games. The initial node of the game tree lies at the time of the first conflict, $c_{i,1}$, and the choices of the first robot at this time lead to children of this node. As the two robots act simultaneously, these children also occur at time $c_{i,1}$. Also, note that the selections of the robots are not observable to each other[1]. An illustration of the game tree appears in Figure 5.1.

Following each simultaneous choice of methods by the robots, the chosen combination of coordination methods is executed (during coordination time $I^a_{i,j}$), and this is followed by a period of task execution $I^p_{i,j}$. The game ends when total time T runs out. The payoffs to the robots are then given as the number of pucks retrieved, minus the cost of resources spent on the task. Terminal nodes may appear anywhere in the game tree, as some selections of the robots lead to less conflicts, and thus greater opportunity for task execution.

Under ideal—and purely theoretical conditions—the robots would know the payoffs awaiting them in each terminal node, and would thus be able to, in principle, compute a game-playing strategy that would maximize the team's utility. To do this, the robots would need to know the times spent resolving conflicts and executing the task, and would also need to know (in advance) the gains achieved during each task-execution period. Even ignoring the gains, and assuming that maximizing task-execution time $\sum_i \sum_j I^p_{i,j}$ is sufficient, the robots would be required to know all conflict resolution times in advance. This is clearly impractical,

[1]This is true in all communication-less coordination methods, which are used in most previous work [39, 34]. When used with communication-based coordination method, this restriction may be removed. It might also be possible to relax this restriction if robots could infer each others' choices post-factum.

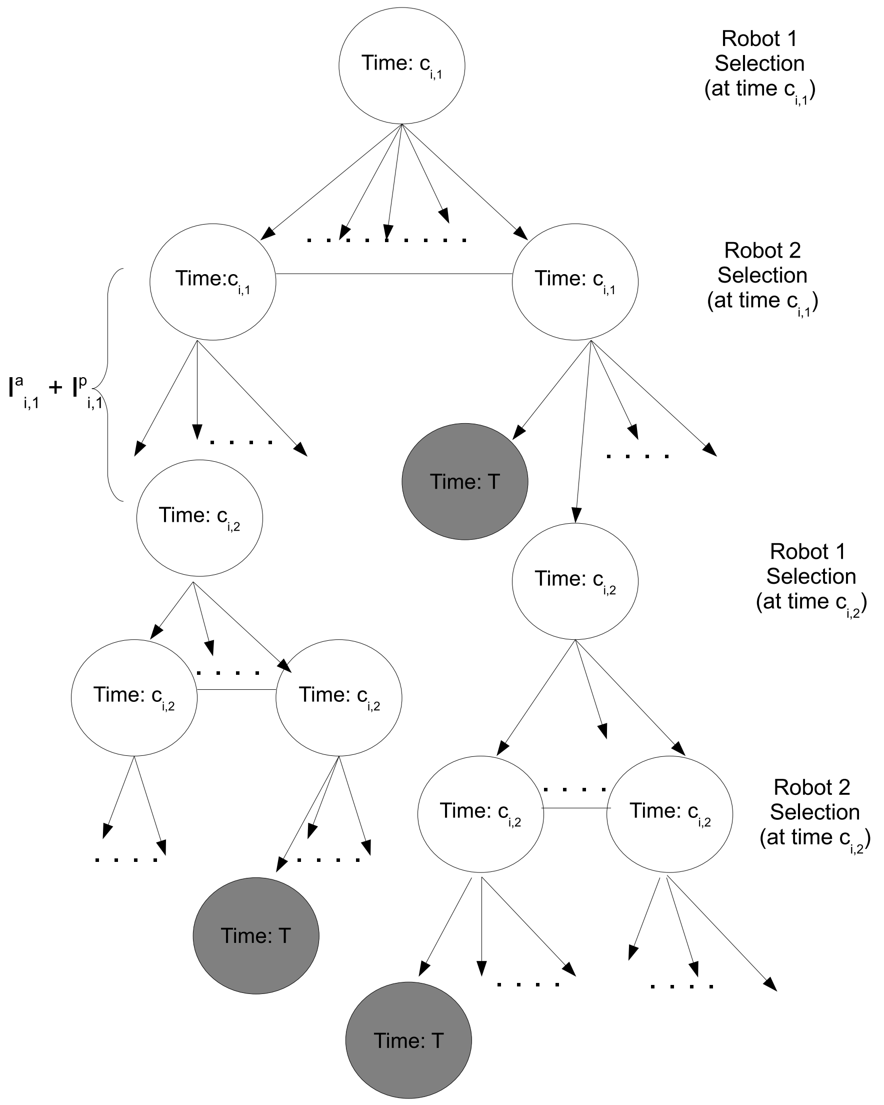


Figure 5.1: An illustration of the extensive-form game tree for an LCT task. Conflict times are denoted in the nodes. Terminal nodes (total time=T) are dark. Note that the second conflict $c_{i,2}$ may occur at different absolute times depending on the choices of the robots at time $c_{i,1}$.

as it requires predicting in advance all possible conflicts and their durations and effects. And the sheer size of the game tree (there are hundreds of conflicts in a typical foraging task, as presented in the previous section) makes learning it a difficult task at best. We are not aware of any method capable of learning the terminal payoffs or node-associated durations and effects for the type of domains we study in this paper.

5.2 Modeling LCT Tasks as a Matrix Game

We thus make a simplifying assumption, that all effects of coordination method selections remain fixed, regardless of where they occur. In other words, we assume that the joint execution of a specific combination of selected coordination methods will always cost the same (in time and resources), regardless of the time in which the conflict occurred. Moreover, the assumption also implies that we assume that the task-execution time (and associated gains)—which depends on the methods selected—will also remain fixed. We state this formally:

Assumption 1. Let α be a coordination method, selected by robot i. We assume that for any $0 \leq j, k \leq K_i$, the following hold:

$$I^a_{i,j}(\alpha) = I^a_{i,k}(\alpha), \qquad I^p_{i,j}(\alpha) = I^p_{i,k}(\alpha), \qquad C^C_{i,j}(\alpha) = C^C_{i,k}(\alpha)$$

This strong assumption achieves a key reduction in the complexity of the model, but gets us farther from the reality of LCT multi-robot tasks. However, the resulting model provides an intuition as to why and when EI works. In Section 5.4 we examine the assumptions of the model and their relation to the reality of the experiments.

The duration of coordination method execution (I^a_i), and the duration of the subsequent conflict-free task-execution (I^p_i), are fixed; they now depend only on the method selected, rather than also on the time of the selection. Thus a path through the game tree can now be compressed. For each combination of selected coordination method, we can simply multiply the costs and gains from using this combination, by the number of conflicts that will take place if it is selected.

Thus we can reduce the game tree into a matrix game, where $K_{i,j}$ is the number of conflicts occurring within total time T that results from the first robot selecting α_i, and the second robot selecting α_j. $U_{i,j}$ is the utility gained from this choice. This utility is defined as:

$$\begin{aligned} U_{i,j} \equiv \ & [gain(I_i^p(\alpha_i) + gain(I_j^p(\alpha_j))] \\ & - [C_i^C(\alpha_i) + C_j^C(\alpha_j)] \end{aligned} \tag{5.1}$$

where we use (for robot i) the notation $gain(I_i^p(\alpha_i))$ to denote the gains achieved by robot i during the task execution time $I_i^p(\alpha_i)$. Note that we treat these gains as being a function of a time duration only, rather than the method α, which only affect the time duration. Underlying this is an assumption that the coordination method choice affect utility (e.g., the pucks acquired) only indirectly, by affecting the time available for task execution. We assume further that gains monotonically increase with time. Maximizing the time available, maximizes the gains.

Table 5.1 is an example matrix game for two robots, each selecting between two coordination methods. Note however that in general, there are N robots and $|M|$ methods available to each.

	α_1^2	α_2^2
α_1^1	$K_{1,1}U_{1,1}$	$K_{1,2}U_{1,2}$
α_2^1	$K_{2,1}U_{2,1}$	$K_{2,2}U_{2,2}$

Table 5.1: LCT task as a matrix game, reduced from the LCT game tree by Assumption 1. Entries hold team payoffs.

Note that the robots do not have access to the selections of the other robots, and thus for them, the game matrix does not have a single common payoff, but individual payoffs. These are represented in each cell by rewriting $K_{i,j}U_{i,j}$ as $K_{i,j}u_i(\alpha_i), K_{i,j}u_j(\alpha_j)$, where

$$u_k(\alpha_k) \equiv gain(I_k^p(\alpha_k)) - C_k^C(\alpha_k).$$

This results in the revised matrix game appearing in Table 5.2.

The number of conflicts $K_{i,j}$ is really the total time T, divided by the duration of each conflict cycle, i.e., $I^a + I^p$. Thus the individual payoff entries for robot l

	α_1^2	α_2^2
α_1^1	$K_{1,1}^1 u_1(\alpha_1^1), K_{1,1}^2 u_1(\alpha_1^2)$	$K_{1,2}^1 u_1(\alpha_1^1), K_{1,2}^2 u_2(\alpha_2^2)$
α_2^1	$K_{2,1}^1 u_2(\alpha_2^1), K_{2,1}^2 u_1(\alpha_1^2)$	$K_{2,2}^1 u_2(\alpha_2^1), K_{2,2}^2 u_2(\alpha_2^2)$

Table 5.2: An example LCT task as a matrix game, with individual payoffs.

selecting method α_k can be rewritten as $\frac{T}{I_l^a(\alpha_k)+I_l^p(\alpha_k)} u_l$.

Let us now consider these individual payoffs. The payoff for an individual robot l that selected α is:

$$\frac{T[g(I_l^p(\alpha)) - c(I_l^a(\alpha))]}{I_l^a(\alpha) + I_l^p(\alpha)} \propto \frac{g(I_l^p(\alpha)) - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \tag{5.2}$$

$$\propto \frac{I_l^p(\alpha) - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \tag{5.3}$$

These two steps require some explanation. First, of course, since for all entries in the matrix T is constant, dividing by T maintains the proportionality. The second step is key to the EI heuristic. It holds only under certain restrictions on the nature of the function $gain()$, but we believe these restrictions hold for many gain functions in practice. For instance, the step holds whenever $gain()$ is linear with a coefficient greater than 1. Now:

$$\frac{I_l^p(\alpha) - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} = \frac{I_l^p(\alpha) + [I_l^a(\alpha) - I_l^a(\alpha)] - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \tag{5.4}$$

$$= \frac{[I_l^p(\alpha) + I_l^a(\alpha)] - [I_l^a(\alpha) + c(I_l^a(\alpha))]}{I_l^a(\alpha) + I_l^p(\alpha)} \tag{5.5}$$

$$= \frac{I_l^p(\alpha) + I_l^a(\alpha)}{I_l^p(\alpha) + I_l^a(\alpha)} - \frac{I_l^a(\alpha) + c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \tag{5.6}$$

$$= 1 - EI_l(\alpha) \tag{5.7}$$

$$\propto -EI_l(\alpha) \tag{5.8}$$

Thus the game matrix is in fact equivalent to the following matrix (Table 5.3). Here, each robot seeks to minimize its own individual EI payoff (maximize its -EI payoff). If robots minimize their individual EI payoffs, and assuming that their equilibrium is Hicks optimal (i.e., the sum of payoffs is maximal), then solving this game matrix is equivalent to maximizing group utility.

	α_1^2	α_2^2
α_1^1	$-EI_1(\alpha_1^1), -EI_2(\alpha_1^2)$	$-EI_1(\alpha_1^1), -EI_2(\alpha_2^2)$
α_2^1	$-EI_1(\alpha_2^1), -EI_2(\alpha_1^2)$	$-EI_2(\alpha_2^1), -EI(\alpha_2^2)$

Table 5.3: LCT task as an EI matrix game.

5.3 Learning Payoffs in LCT Matrix Games

Unfortunately, when the robots first begin their task, they do not know the payoffs, and thus rely on the reinforcement learning framework to converge to appropriate EI values. Of course, it is known that Q-learning does not, in the general case, converge to equilibrium in 2-player repeated games [8, 42, 17]. However, there are a number of features that hold for the EI game matrix *in the domains we study*, which makes the specific situation special.

First, the game matrix is theoretically symmetric. Because robots are homogeneous, a combination of coordination methods $\langle \alpha_1, \alpha_2 \rangle$ will yield the same payoffs as $\langle \alpha_2, \alpha_1 \rangle$.

Second, we know that for the specific game settings, one combination yields optimal payoffs (in the sense that the sum of robot payoffs is optimal). Although it is now accepted that no one coordination method is always best in all settings, it is certainly the case that in a specific scenario (e.g., a specific conflict, a specific group size), a combination can be found which is best.

Third, the value of EI for the optimal individually-selected method α_j^1 can only decrease if the other robot does not select an optimal method α_k^2. Under normal conditions, the numerator of the EI value, $I_1^a(\alpha_j^1) + C^C(\alpha_j^1)$ is dependent only on the execution of α_j^1 by the robot. On the other hand, the denominator $I_1^a(\alpha_j^1) + I_1^p(\alpha_j^1)$ can only decrease (because the time to the next conflict, $I_1^p(\alpha_j^1)$ can only decrease, by definition). Thus, the EI value can only grow larger (i.e., $-EI$ grows smaller). Selection of the optimal EI values is thus dominant.

Finally, and most importantly, the games that take place here are *not* between two players. Rather, the process is more akin to randomized anonymous matching in economics and evolutionary game theory[2]. In this process, pairs of players are randomly selected, and they do not know their opponents' identity (and thus do not know whether they have met the same opponents before).

[2]We thank Sarit Kraus for this observation.

Indeed, this last quality is crucial in understanding why our use of EI works. It turns out that there exists work in economics that shows that under such settings, using simple reinforcement learning techniques (in our case, stateless Q-learning) causes *the population* to converge to Nash equilibrium, even if mixed [19]. Thus rather than having any individual agent converge to the mixed Nash equilibrium, the population as a whole converges to it, i.e., the number of agents selecting a specific policy is proportional to their target probabilities under the mixed Nash equilibrium.

In particular, Hopkins lists several conditions for this population convergence [19]. First, the game matrix must be symmetric [19, pg. 95], as is ours. Second, agents update their reward estimates using fictitious play or stimulus-response learning (which essentially corresponds to our stateless reinforcement learning approach) [19, pg. 101–102]. Finally, The agents randomly select their initial methods; in our case this is also true. Under these conditions, Hopkins shows that the population of the agents converges to an evolutionary-stable Nash equilibrium.

There remains the question of whether indeed we can guarantee that agents converge to the maximal team-payoff Nash equilibrium, if more than one equilibrium exists. We again turn to economics literature, which shows that for coordination games—including even the difficult Prisoner's Dilemma game—agents in repeated randomized matching settings tend to converge to the Pareto-efficient solution [9, 31]. However, these works typically assume public knowledge of some kind, which is absent in our domain. Thus we cannot conclude definitely that the use of stateless EI reinforcement-learning will necessarily converge to the group-optimal solution (the maximal group utility). This question remains, unfortunately, open.

5.4 Revisiting the EI Experiments

Armed with the analytically-motivated intuition as to why EI works, we now go back to re-examine the experiment results. In general, there are of course differences between the analytical intuitions and assumptions and the use of EI in a reinforcement learning context: (i) the values learned are approximations of the

EI values, which cannot be known with certainty; (ii) the assumptions allowing reduction of the LCT extensive-form game tree to a game matrix do not hold in practice; and (iii) even the assumptions underlying the extensive-form game tree (e.g., that robots start their conflict at the same time, or that their gains depend only on time available for task execution) are incorrect. We examine specific lessons below.

We begin with the teambots simulation experiments, where EI was highly successful, and was also demonstrated to be robust to unknown costs. Despite the fact that the domain cannot be reduced to the matrix game form, it turns out that some of the assumptions are approximately satisfied, which explain the success of EI here.

First, the fact that about half the pucks moved randomly helped spread them around the arena even after many pucks were collected. Thus the gains expected later in the task were closer to the gains at the beginning to the task, than it would have been had all pucks been immobile (in which case pucks closer to base are collected first, resulting in higher productivity in the beginning).

Second, the size of the arena, compared to the size of the robots, was such that the robots did not need to converge to one optimal combination of selection methods: Different zones in the arena required different combinations. In principle, this should have challenged the approach, as the stateless learning algorithm cannot reason about the robots being in different states (zones). However, as the robots moved between areas fairly slowly, they were able to adapt to the conditions in new zones, essentially forgetting earlier EI values. This is a benefit of the stateless algorithm.

Finally, in these simulation experiments, the number of robots was fairly large. Thus the conditions for convergence to the Nash equilibrium [19] apply.

The situation is different in the experiments with the real AIBO robots. The limited training time and the limited number of robots (only three) raises questions as to the applicability of Hopkins' work to this domain. Indeed, the application of EI stateless algorithms here is less successful than in the simulated foraging domain.

Another issue with the AIBO robot experiments is the use of the fixed exploration rate. The choice of a fixed exploration rate can hurt performance of the algorithm, as is clearly seen in the results of the AIBO foraging experiments. Because robots *must* explore, they are sometimes forced to act against their better knowledge, and thus reduce performance. But this did not affect the results in the simulation domain, where EI often gave the best results of all methods. We believe that this is due to the size of the arena, which created different zones as discussed above. Here exploration was very useful, to enable implicit transition between states. In contrast, in the AIBO experiments, the size of the arena was so small, that density remained fixed throughout the arena, and exploration eventually lead to reduced results.

Chapter 6

Advanced Use of EI in Reinforcement Learning

The single-state reinforcement learning framework in which we have utilized EI in previous chapters leaves several open questions, which we address in this chapter. First, the use of EI seems to depend critically on the times that define the scope of conflicts (the active and passive times of a cycle). We address this issue in Section 6.1, in which we show that the EI framework can also learn to select *when* to declare a conflict.. Then, in Section 6.2 we present the use of EI in a full state-based Q-learning implementation, i.e., moving away from the simplified stateless algorithm presented earlier, to more common implementations of Q-learning, in which learning occurs in multiple states. We show that this can result in very significant improvements to team performance Finally, in Section 6.3 we show preliminary results demonstrating the successful use of EI-based learning of selecting task actions, rather than coordination methods.

6.1 Learning When to Declare a Conflict

The physical calculation of an EI value depends directly on the decision that a conflict has occurred. The decision to declare a conflict begins the active portion of the cycle, in which a coordination method is selected for execution. The conflict ends when the coordination method finishes execution. Up until now, we have assumed a fixed procedure was given, and focused on the use of EI within the reinforcement learning framework.

However, the decision to declare a conflict can have significant impact on the use of a coordination method. A fixed coordination method α may be more or less successful, depending on when it is called into action. For instance, in examining spatial coordination in robots, a conflict state occurs when two (or more) robots are too close. But the distance in which the robots decide on a conflict may vary. Thus the repeal coordination method, for instance, which moves the robot away for a fixed distance, may cause the robot to end up at a different location, depending on the initial distance between the robots (the conflict decision).

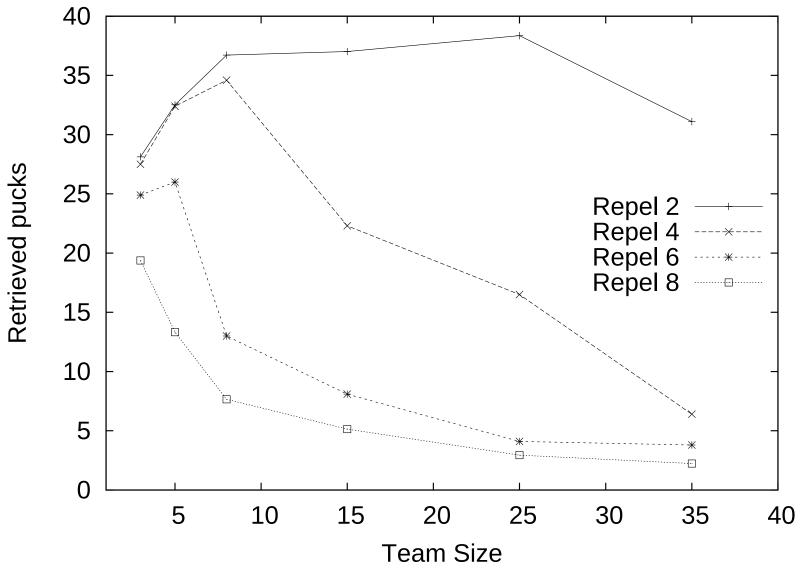
Indeed, Figure 6.1 shows the effects of variable conflict definition in the simulated foraging domain. Here, the distance between the robots, which is used to declare a conflict state, is varied between the default two (2) robot radii (used in earlier chapters), and eight (8). In the figures, the X axis measures the number of robots in the team. The Y axis shows the number of pucks collected by a team using only the repeal method (Figure 6.1-a) or the aggression method (Figure 6.1-b). Each line corresponds to a different conflict radius definition. The figures clearly show that different radii significantly change the results of the foraging.

Thus the definition of a conflict can change the behavior of a team. Given a new domain in which we want to utilize the EI method, we require a way that adapts also the conflict threshold, as well as selection of the coordination method.

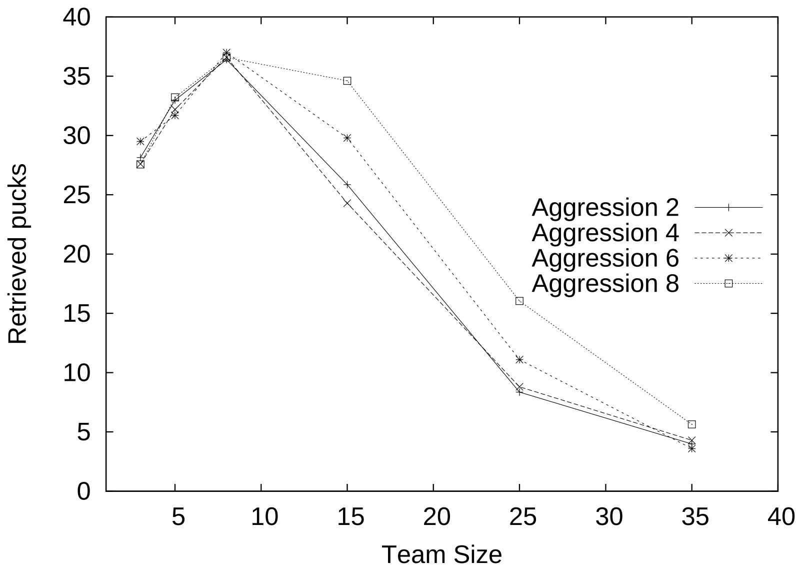
We propose to do this by folding the conflict distance definitions into the methods, creating multiple methods that involve different distances. Let M be the set of coordination methods, and R be the set of possible conflict definitions. For each method $\alpha \in M$ and conflict radius $r \in R$, we create a method α_r, which involves the α method at conflict distance r. We then use reinforcement learning with EI, as before, but allow the EI to select from the space of $|R| \times |M|$ methods.

Figure 6.2 shows the results of experiments using this approach in the simulated foraging domain. As before, the X axis measures the number of robots in the team. The Y axis measures the number of pucks collected. The figure contrasts several methods and distances:

Repel 2, Repel 8, Aggression 2, Aggression 8. These are fixed methods, which



(a) Robots using repeal.



(b) Robots using aggression.

Figure 6.1: Team performance in the simulated foraging domain. The distance defining a conflict between two nearby robots is varied from 2 to 8.

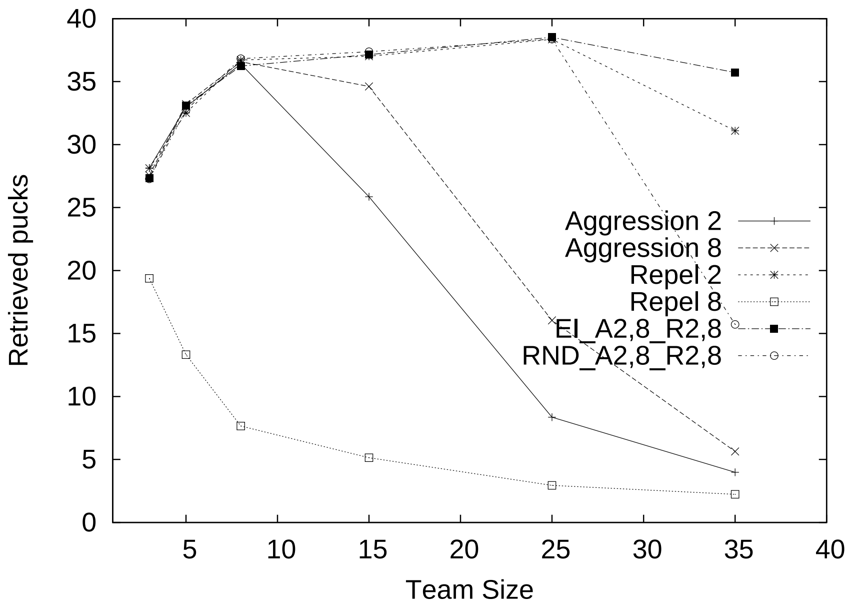


Figure 6.2: Team performance in the simulated foraging domain, using EI to learn conflict declaration parameters.

utilize repel or aggression (respectively), with a conflict distance of 2 robot radii or 8 robot radii. Thus here, $M = \{$ Repel, Aggression$\}$, and $R = 2, 8$.

EI_A2,8_R2,8. The result of using EI-based single-state reinforcement learning, selecting from the pool of four static combination methods $M \times R =$ $\{$Repel 2, Repel 8, Aggression 2, Aggression 8$\}$.

RND_A2,8_R2,8. The result of using random selection over the space of four static combination methods $M \times R = \{$Repel 2, Repel 8, Aggression 2, Aggression 8$\}$.

The different curves in Figure 6.2 shows the results of using repeal and aggression (with fixed conflict distances, at 2 and 8 robot radii), as well as the use of EI or random selection between the combination methods. The results demonstrate the efficacy of the method in learning methods that also incorporate conflict timing parameters.

6.2 Using EI in Reinforcement Learning with Multiple States

We now move away from the simple single-state Q-learning algorithm (Algorithm 4.1), and explore the use of EI in more familiar Q-learning settings, where the environment states are taken into account. Here, the reward ($-EI$) collected for using a specific coordination method a is associated with the robot being in a particular environment state s.

The learning algorithm we use is based on the following Q-Learning equation:

$$Q_t(s,a) = Q_{t-1}(s,a) + \rho(R_t(s,a) - \gamma Q_{t-1}(s,a))$$

where ρ is the learning speed factor, and γ is a factor of discounting. As before, the algorithm uses a constant exploration rate β.

Two versions of the MSL (Multi-State Learning) algorithm are possible. In one, the estimated EI values are learned for conflicts that end and begin in the same state (we call this variant MSL$_x$). In this variant of the algorithm, an agent that goes into a conflict in a specific state s and resolves it, will continue measuring time in the conflict's associated passive interval until it returns to the state s. Any other conflicts will be tracked separately. In the other, simpler variant, conflicts that begin in a state may end anywhere (we call this variant MSL$_1$). This is a more natural extension to the stateless reinforcement learning algorithm we have seen before, as only one conflict is tracked at any given moment.

We use the following notation in the algorithm:

- $Q[s, \alpha]$ is a table with learned EI for all states and actions, where $s \in S$ (S is the set of environment states), and $\alpha \in M$.

- cci is the Current Conflict Index, i.e, a specific conflict for which we are currently measuring time and costs. In the MSL$_x$ variant, it is used to keep track of multiple conflicts for which information is tracked, and thus takes on values from the set of states S. In the MSL$_1$ variant, it is set to a constant null value (which represents the fact that MSL$_1$ treats a conflict's active and passive intervals beginning at a state s and ending at any state).

- $SelectedMethod[cci]$ is a method that currently selected for solving conflict associated with cci.
- $ConflictState[cci]$ is a state where conflict associated with cci is started.
- $SolvingTime[cci]$ is a time moment when conflict associated with cci has been solved.
- $BgnCycleTime[cci]$ is a time moment when conflict associated with cci is started.
- $EndCycleTime[cci]$ is a time moment when previous conflict cycle associated with cci is ended (e.i. new conflict associated with cci is started).
- $Cost[cci]$ is a cost spent for solving conflict associated with cci (i.e. this is Active Coordination Cost).
- $Counter[cci]$ is counter of conflicts associated with cci.

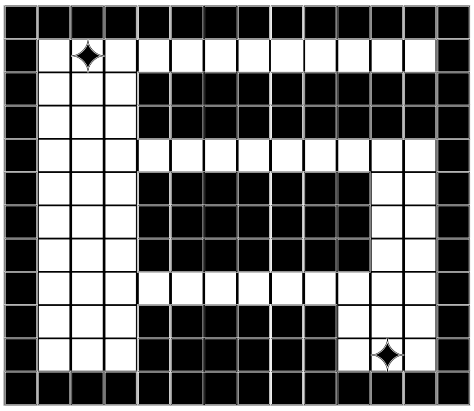
To evaluate the use of the two MSL variants, we applied them in a challenging domain, involving discrete maze-like environments, in which narrow corridors require agents to coordinate their movements to reach from one location to the next. Figure 6.3 shows the two test environments. In one, there are two goal locations; in the other, five. All agents within the environments (up to 20) can move in the 4 basic directions (north, south, east, west) in each time step. Each agent randomly picks a goal location and attempts to move to it; once the goal is reached, the agent picks another goal. Since agents cannot pass through each other, nor through walls, they require coordination in navigating in the narrow corridors of the mazes.

In these two environments, we ran hundreds of repeated trials, each a thousand steps long. The exploration rate β was fixed at 0.2; the learning factor ρ was set to 0.8 by default (a second set of experiments varies this values); and the discount factor γ was set was to 1.0.

Figure 6.4 below shows the results. Here, the X axis marks the number of agents in the environment. The Y axis marks the number of goals achieved by the group of agents (the sum of their individual achievements). The figure shows a

Algorithm 2 Multi-State EI-Based Learning (MSL)

```
Require: $\beta \in [0, 1]$ – rate of exploration vs exploitation
Require: $\rho \in [0, 1]$ – learning speed factor
Require: $\gamma \in [0, 1]$ – learning discount factor
Require: $M$ – a set of coordination algorithms
Require: A way of measuring accumulating coordination resource costs
Require: A way of current state observation
Require: A way of checking current time or step number in case of discrete system.
 1: for all $cci \in$ members of $Count$ do
 2:    $Count[cci] \leftarrow 0$
 3: for all $(s, \alpha) \in S \times M$ do
 4:    $Q[s, \alpha] \leftarrow$ random$([0, 1])$
 5: $cci \leftarrow \varepsilon$ {This line for MSL$_1$ only}
 6: while robot is active do
 7:    WAIT FOR CONFLICT EVENT
 8:    $s \leftarrow$ ObsorveState$()$
 9:    $cci \leftarrow s$ {This line for for MSL$_x$ only}
10:    $Count[cci] \leftarrow Count[cci] + 1$
11:    $EndCycleTime[cci] \leftarrow$ CurrentTime$()$
12:    if $Count[cci] > 1$ {Not First Conflict for current state} then
13:       $t_a \leftarrow SolvingTime[cci] - BgnCycleTime[cci]$
14:       $t_p \leftarrow EndCycleTime[cci] - SolvingTime[cci]$
15:       $EI \leftarrow -\frac{t_a + Cost[cci]}{t_a + t_p}$
16:       $\omega \leftarrow ConflictState[cci]$
17:       $\alpha \leftarrow SelectedMethod[cci]$
18:       $Q[\omega, \alpha] \leftarrow Q[\omega, \alpha] + \rho \cdot (EI - \gamma \cdot Q[\omega, \alpha])$
19:    $BgnCycleTime[cci] \leftarrow EndCycleTime[cci]$
20:    $ConflictState[cci] \leftarrow s$
21:    if $\beta >$ random$([0, 1])$ then
22:       $SelectedMethod[cci] \leftarrow$ random$(M)$
23:    else
24:       $SelectedMethod[cci] \leftarrow$ argmax$_{\alpha \in M} Q[s, \alpha]$
25:    EXECUTE $SelectedMethod[cci]$ {Record resource spending to $Cost[cci]$}
26:    $SolvingTime[cci] \leftarrow$ CurrentTime$()$
```


(a) Maze with two goals.

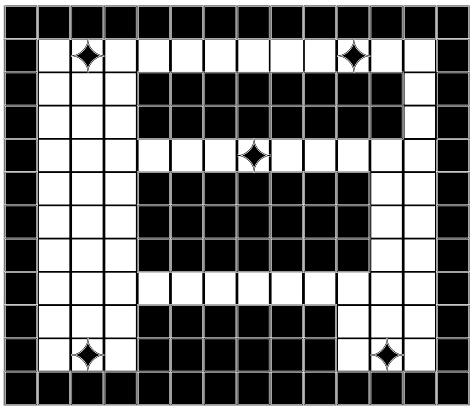

(b) Maze with five goals.

Figure 6.3: Discrete environment requiring coordination to navigate between goals. Filled squares denote walls and obstacles, four-point stars mark goal locations, empty/passable positions marked white.

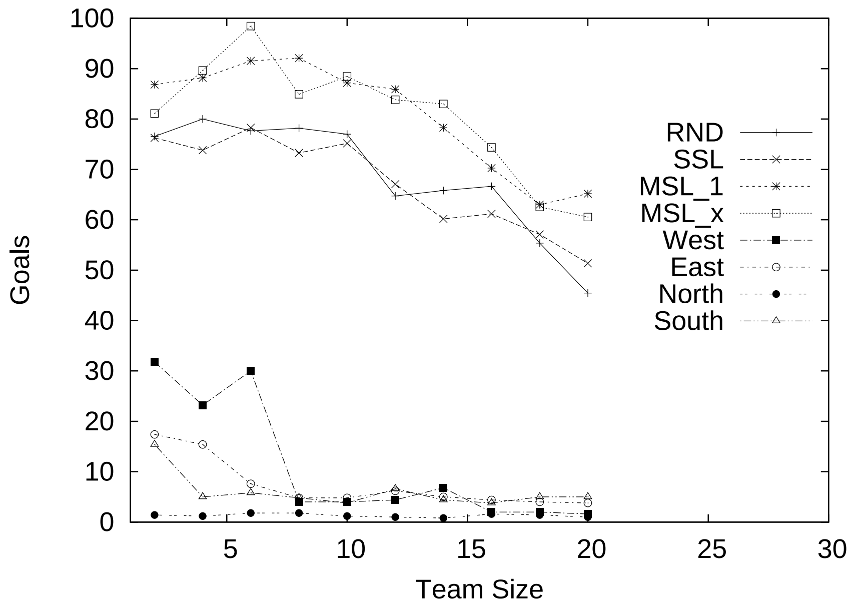
clear difference between the stateless reinforcement learning technique presented earlier (here, called SSL for Single-State Learning), and the multi-state reinforcement learning described in this section. Specifically, multi-state reinforcement learning in this domain proves extremely useful.

Since the discount factor has significant impact on the success of reinforcement learning algorithms, we also vary the discount factor in a final set of experiments, and reduce it from 1.0 to 0.2. The results are presented in Figure 6.5. The figure demonstrates that reducing the discount factor did not, in factor, change the results of the analysis, and indeed may indeed slight reduce performance.

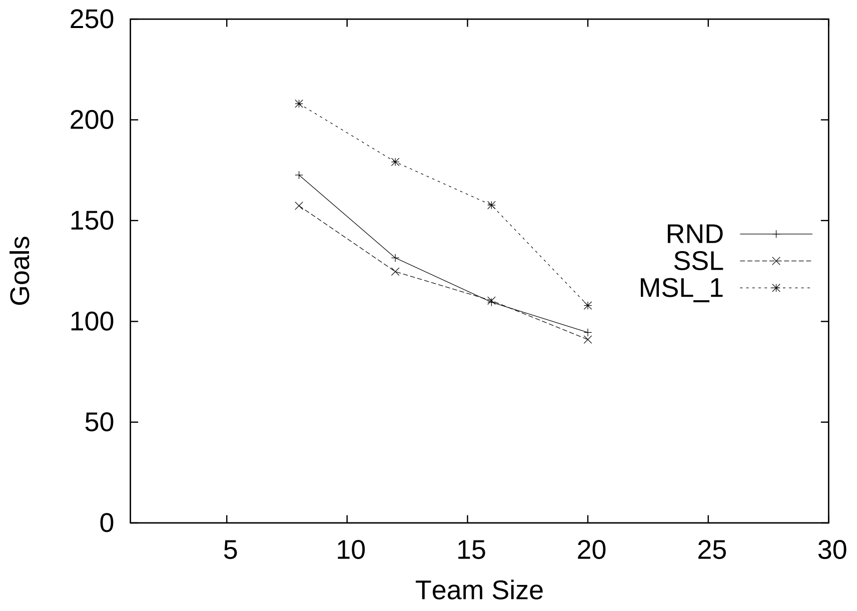
6.3 EI-Based Learning of Task-Oriented Decisions

Finally, we evaluated the use of EI with robots in virtual environments. Here, we utilized robots that operate in VR-Forces[27], a commercial high-fidelity simulator. Each robot controls a simulated entity in the environment, and must carry out its own path planning and decision-making. A bird's eye view of the experiment setup is shown in Figure 6.6.

Within this environment, we conducted experiments with four virtual robots, where the coordination was implicit, rather than explicit. All of the four robots



(a) Learning results after 1000 cycles, in the maze with two goals. Results are contrasted with the random-coordination algorithm, as well as with each of the fixed methods by itself.



(b) Learning results after 1000 cycles, in the maze with five goals.

Figure 6.4: Results from the maze experiments.

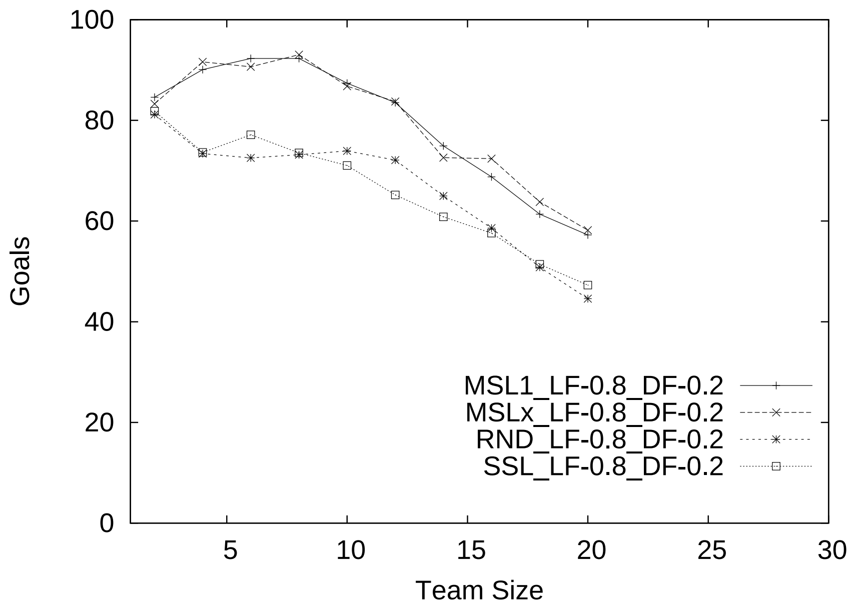


Figure 6.5: Learning results after 1000 cycles, in the maze with two goals.

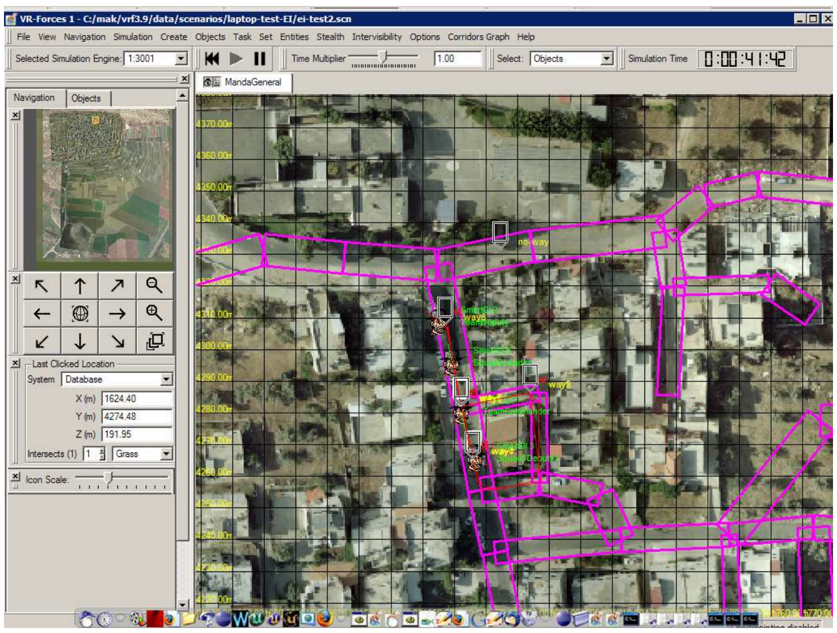

VR-Forces 1 - C:/mak/vrf3.9/data/scenarios/laptop-test-EI/ei-test2.scn
File View Navigation Simulation Create Objects Task Set Entities Stealth Intervisibility Options Corridors Graph Help
Selected Simulation Engine: 1:3001
Time Multiplier 1.00
Select: Objects
Simulation Time 0:00:41:42
MandaGeneral
Navigation Objects
Last Clicked Location
System Database
X (m) 1624.40
Y (m) 4274.48
Z (m) 191.95
Intersects (1) 1 Grass
Icon Scale:
no-way
way4

Figure 6.6: A screen shot of VR-Forces experiment setup. The screen shows a bird-eye view of the virtual environment, and the corridors in which the simulated entities (robots) travel.

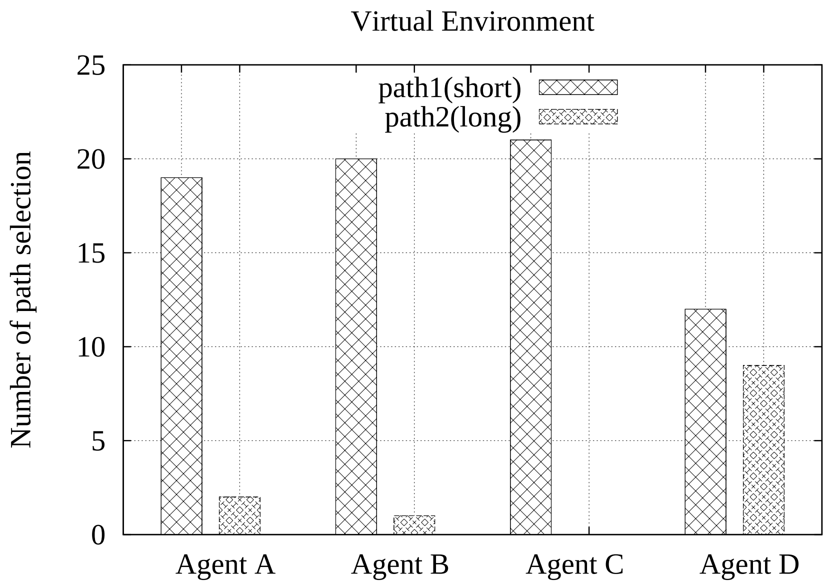


Figure 6.7: Results in the virtual environment domain.

had the goal of getting to a target location. They could do this through one of two paths, the first ($path1$) slightly shorter than the other ($path2$). Each path was built from 5–10 navigation points, which the robots go through in sequence to travel through the path. Actual travel times through the paths vary, and are not just a function of the path length: First, when robots move on the same path, they sometimes crowd the path and cause delays in moving on it (e.g., if robots collide or block others from reaching a navigation point); second, because this is a high-fidelity simulation, the actual movement velocity of the robots is not always the same, and varies slightly from one run to the next. The result is that it is not immediately obvious how robots should divide up the paths between them. Using EI to select between the paths is not a selection of a coordination method, but is instead a selection of a task, such that coordination is implicit.

We conducted 21 runs, where the EI values were saved from one run to the next. The results of these runs are shown in Figure 6.7. The X axis lists the four different robots (agents), A–D. The Y axis measures the number of runs (out of the 21) that each agent selected a particular path. The two bars mark, for each agent, the path selections: $path1$ or $path2$.

The results show convergence of the first three robots to selecting $path1$, while the fourth and last robot jumps back and forth between $path1$ and $path2$. When we examine the results in detail, we discover that indeed the decision of the fourth robot is difficult: On one hand, four robots on $path1$ often interfere with each other. On the other hand, the use of $path2$ does add to the overall task time of the robot. Thus the EI values are very close to each other, and the robot in fact converges to arbitrary selection between the two paths.

An interesting lesson can be learned from the experiments in the virtual environment. Here, EI was applied to a task that it was not meant for, involving implicit, rather than explicit, coordination. The nature of this task is that there is more than one single equilibrium point, as two combination of paths are possible. Thus our intuition as to the convergence properties of the EI algorithm should not hold. However, the algorithm converged quickly to selecting between two almost equally-valued alternatives, reflecting the two top choices.

Chapter 7

Conclusions and Future Work

This thesis examined in depth a novel reward function for cooperative settings, called Effectiveness Index (EI). EI estimates the resource spending velocity of a robot, due to its efforts spent on coordination. By minimizing EI, robots dedicate more time to the task, and are thus capable of improving their team utility. We used EI as a reward function for selecting between coordination methods, by reinforcement-learning. This technique was shown to work well in three different domains: Simulation-based multi-robot foraging, real AIBO multi-robot foraging, and high-fidelity commercial virtual environment. The experiments explore the scope of the technique, its successes and limitations. In addition, we have formally explored multi-robot tasks for which EI is intended. We have shown that under some assumptions, EI emerges analytically from a game-theoretic look at the coordination in these tasks. We believe that this work represents a step towards bridging the gap between theoretical investigations of interactions, and their use to inform real-world multi-robot system design. Improved results can be achieved by extending both the theory underlying the use of EI, and the learning algorithms in which it is used.

Bibliography

[1] A. K. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, pages 980–987, 2004.

[2] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

[3] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-06)*, 2006.

[4] T. Balch. Reward and diversity in multirobot foraging. IJCAI-99 Workshop on Agents Learning, July 1999.

[5] T. Balch. www.teambots.org, 2000.

[6] T. Balch and R. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, December 1998.

[7] T. R. Balch. Integrating learning with motor schema-based control for a robot soccer team. In *RoboCup*, pages 483–491, 1997.

[8] M. Bowling and M. Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical Report CMU-CS-00-165, Computer Science Department, Carnegie Mellon University, 2000.

[9] G. Ellison. Cooperation in the prisoner's dilemma with anonymous random matching. *The Review of Economic Studies*, 61(3):567–588, July 1994.

[10] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-07)*, 2007.

[11] C. B. Excelente-Toledo and N. R. Jennings. The dynamic selection of coordination mechanisms. *Autonomous Agents and Multi-Agent Systems*, 9:55–85, 2004.

[12] M. Fontan and M. Matarić. Territorial multi-robot task division. *IEEE Transactions of Robotics and Automation*, 14(5):815–822, 1998.

[13] J. R. Galbraith. *Designing Complex Organizations*. Addison-Wesley Longman Publishing Co., Inc., 1973.

[14] D. Goldberg and M. Matarić. Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In *Robot Teams: From Diversity to Polymorphism*, pages 315–344, 2001.

[15] D. Goldberg and M. J. Mataric. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 637–642, Providence, RI, 1997. AAAI Press.

[16] N. Hazon and G. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 2008. To Appear.

[17] P. J. Hoen, K. Tuyls, L. Panait, S. Luke, and J. A. L. Poutré. An overview of cooperative and competitive multiagent learning. In K. Tuyls, P. J. Hoen, K. Verbeeck, and S. Sen, editors, *First International Workshop on Learning and Adaption in Multi-Agent Systems*, volume 3898 of *Lecture Notes in Computer Science*, pages 1–46. Springer, 2006.

[18] L. Hogg and N. Jennings. Socially intelligent reasoning for autonomous agents. *IEEE Transactions on Systems, Man and Cybernetics - Part A*, 31(5):381–399, 2001.

[19] E. Hopkins. Learning, matching, and aggregation. *Games and Economic Behavior*, 26:79–110, 1999.

[20] I. Ikar. Area coverage by a multi-robot system. Master's thesis, Bar Ilan University, 2007.

[21] M. Jager and B. Nebel. Dynamic decentralized area partitioning for cooperating cleaning robots. In *ICRA 2002*, pages 3577–3582, 2002.

[22] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pages 326–331, 2002.

[23] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, pages 1258–1259, 2004.

[24] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[25] J. Lee and R. Arkin. Adaptive multi-robot behavior via learning momentum. In *IEEE Conference on Intelligent Robot Systems (IROS-03)*, 2003.

[26] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *National Conference on Artificial Intelligence*, pages 796–802, 1990.

[27] MÄK Technologies. VR-Forces. http://www.mak.com/vrforces.htm, 2006.

[28] M. J. Matarić. Reinforcement learning in the multi-robot domain. *Auton. Robots*, 4(1):73–83, 1997.

[29] E. Ostergaard, G. Sukhatme, and M. Matarić. Emergent bucket brigading. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents-01)*, pages 29–30, 2001.

[30] A. Ram, R. C. Arkin, and R. J. Clark. Learning momentum: On-line performance enhancement for reactive systems. Technical report, 1991.

[31] A. J. Robsona and F. Vega-Redondob. Efficient equilibrium selection in evolutionary games with random matching. *Journal of Economic Theory*, 70(1):65–92, July 1996.

[32] A. Rosenfeld, G. A. Kaminka, and S. Kraus. Adaptive robot coordination using interference metrics. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-2004)*, pages 910–916, 2004.

[33] A. Rosenfeld, G. A. Kaminka, and S. Kraus. Adaptive robotic communication using coordination costs. In *Distributed Autonomous Robotic Systems 7*. Springer-Verlag, 2006.

[34] A. Rosenfeld, G. A. Kaminka, S. Kraus, and O. Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6–7):633–655, 2008.

[35] P. Rybski, A. Larson, M. Lindahl, and M. Gini. Performance evaluation of multiple robots in a search and retrieval task. In *Proc. of the Workshop on Artificial Intelligence and Manufacturing*, pages 153–160, Albuquerque, NM, August 1998.

[36] M. Schneider-Fontan and M. Matarić. A study of territoriality: The role of critical mass in adaptive task division. In P. Maes, M. Matarić, J.-A. Meyer, J. Pollack, and S. Wilson, editors, *From Animals to Animats IV*, pages 553–561. MIT Press, 1996.

[37] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[38] K. Tumer. Designing agent utilities for coordinated, scalable, and robust multi-agent systems. In P. Scerri, R. Vincent, and R. Mailler, editors, *Challenges in the Coordination of Large-Scale Multiagent Systmes*, pages 173–188. Springer, 2005.

[39] R. Vaughan, K. Støy, G. Sukhatme, and M. Matarić. Go ahead, make my day: robot conflict resolution by aggressive competition. In *Proceedings of the 6th int. conf. on the Simulation of Adaptive Behavior*, Paris, France, 2000.

[40] D. H. Wolpert and K. Tumer. Collective intelligence, data routing and braess' paradox. *Journal of Artificial Intelligence Research*, 16:359–387, 2002.

[41] D. H. Wolpert, K. R. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, pages 77–83. ACM Press, 1999.

[42] E. Yang and D. Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Technical Report CSM-404, University of Essex, 2004.

[43] M. Zuluaga and R. Vaughan. Reducing spatial interference in robot teams by local-investment aggression. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, August 2005.