Proceedings of the Adaptive and Learning Agents Workshop, 2009

An AAMAS-09 Workshop

Co-chairs:
Matthew E. Taylor, The University of Southern California
(taylorm@usc.edu)

Karl Tuyls, Eindhoven University of Technology
(k.p.tuyls@tue.nl

In conjunction with the Eighth International Conference on Autonomous Agents and Multiagent Systems May 12, 2009
Budapest, Hungary

Preface

This year's edition of ALA is the second after the merger of the ALAMAS and ALAg workshops. In 2008 this joint workshop was organized for the first time under the flag of ALAMAS & ALAg. ALAMAS was a yearly returning European workshop on Adaptive and Learning Agents and Multi-Agent Systems (held eight times). ALAg was the international workshop on Adaptive and Learning agents, usually held at AAMAS. To increase the strength, visibility, and quality of the workshops, both were merged into the ALA workshop, with a steering committee as a backbone for the organization. We are very happy to present you the proceedings of this special edition of the ALA workshop.

As agent-based systems grow larger and more complex, there is a compelling need for agents to learn and adapt to their dynamic environments. Indeed, how to best adaptively control, coordinate, and optimize adaptive multiagent systems is an extremely exciting multi-disciplinary research area. Such systems are often deployed in real-world situations with stochastic environments where agents have limited perception and communication capabilities. Furthermore, in many number of distributed domains without centralized control, different agents will have different behaviors, capabilities, learning strategies, etc. There is a pressing need to better understand and control the behavior of multiple learners and their emergent dynamics. This workshop series intends to explore all agent learning approaches, with particular emphasis on agent settings where the scale and complexity of the environment require novel learning techniques. The goal of this workshop is to bring together not only scientists from different areas of computer science, such as agent architectures, reinforcement learning, and evolutionary algorithms but also from different fields studying similar concepts like game theory, bio-inspired control, and mechanism design.

We thank all authors who responded to our call-for-papers with interesting contributions. We look forward to a lively workshop with informative discussions and constructive exchange of ideas. We are thankful to the members of the program committee for the quality and sincerity of their efforts and service. We would like to thank all the members of the steering committee to make this workshop possible and support it with good advice. We also thank the AAMAS conference for providing us a platform for holding this event.

Matthew E. Taylor and Karl Tuyls
ALA 2009 Co-Chairs

Program Chairs

Matthew E. Taylor
Department of Computer Science
The University of Southern California
USA
taylorm@usc.edu

Karl Tuyls
Department of Industrial Design
Eindhoven University of Technology
The Netherlands
k.p.tuyls@tue.nl

Program Committee

Eduardo Alonso, City University, UK
Bikramjit Banerjee, The University of Southern Mississippi, USA
Ana L.C. Bazzan, UFRGS, Porto Alegre, BR
Marek Grzes, University of York, UK
Zahia Ghuessoum, University of Paris 6, FR
Franziska Klügl, University of Orebro, Sweden
Daniel Kudenko, University of York, UK
Ann Nowé, Vrije Universiteit Brussels, BE
Liviu Panait, Google Inc Santa Monica, USA
Lynne Parker, University of Tennessee, USA
Jeffrey Rosenschein, The Hebrew University of Jerusalem
Michael Rovatsos, Centre for Intelligent Systems and their Applications, UK
Sandip Sen, University of Tulsa, USA
Kagan Tumer, Oregon State University, USA
Katja Verbeeck, KaHo Sint-Lieven, Belgium

Steering Committee

Franziska Klügl
Daniel Kudenko
Ann Nowé
Lynne E. Parker
Sandip Sen
Peter Stone
Kagen Tumer
Karl Tuyls (chair)

Contents

Adaptive Multi-Robot Coordination: A Game-Theoretic Perspective

Gal A. Kaminka, Dan Erusalimchik, and Sarit Kraus
Computer Science Department
Bar Ilan University, Israel

ABSTRACT

Multi-robot systems researchers have been investigating adaptive coordination methods for improving spatial coordination in teams. Such methods adapt the coordination method to 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 success 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 are directly a function of EI estimates. The use of reinforcement learning leads to robots maximizing their EI rewards in equilibrium. This work is a step towards bridging the gap between the theoretical study of interactions, and their use in multi-robot coordination.

1. INTRODUCTION

Multi-robot systems researchers have been investigating coordination methods for improving spatial coordination in teams [9, 18, 17]. Such methods attempt to resolve spatial conflicts between team-members, e.g., by dynamic setting of right-of-way priorities [20, 24], territorial separation [19, 7, 12], or role-based priorities [15]. It is accepted that no one method is always best [8, 6, 17], and that all methods reach a point where adding robots to the group (i.e., increasing the density of the robots in space) reduces overall productivity [19, 18].

There is thus growing interest in adaptive coordination approaches, which adapt the coordination method to the dynamic changes in density. Zuluaga and Vaughan adjust the right-away priorities based on the amount of local effort (or investment) by team-members [24]. Toledo and Jennings [6] propose an algorithm-selection approach, based on reinforcement learning, where fixed coordination methods are switched to accommodate dynamic changes to the environment.

Cite as: Title, Author(s), *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

More recently, Rosenfeld et al. [17] advocated allowing each robot to individually switch coordination methods to reduce its own estimated resource costs. In general, all of these adaptive coordination methods have demonstrated much success in multiple domains of interest.

Unfortunately, while their empirical success is evident, none of these methods have ever been analytically proven to work, nor understood in the context of existing formal work on multi-robot learning and adaptation. As a result, their optimality and the appropriate conditions for their use remain open questions. Put simply, they pose a puzzle: These are methods that work well in practice—both in simulations and with real robots—but the reasons for their success remain elusive.

This paper presents a reinforcement-learning approach to coordination algorithm selection, which is not only shown to work well in experiments, but also explored analytically. The reward function used as the basis for the learning is called *Effectiveness Index* (EI). 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 and analytically evaluate the use of EI. 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 addition, we analytically explore the reasons and assumptions underlying this success. 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 a step towards bridging the gap between the theoretical study of interactions (via game theory), and their use to explain and inform multi-robot coordination.

2. RELATED WORK

Most closely related to our work is earlier work on adaptation based on coordination effort. Rosenfeld et al. [17], 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.

Vaughan et al. [20] presented a method called *aggression* for reducing interference in distributed robot teams. When robots come too close to each other, each of the robots demonstrate its own level of aggression such that the robot with the highest level becomes the winner, while the loser concedes its place. Later, Zuluaga and Vaughan [24] 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. This result is compatible with our findings. However, Effectiveness Index relies solely on task-independent resource measures.

Excelente-Toledo and Jennings [6] 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.

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 [23, 10].

Matarić [14] discusses several techniques for using rewards in multi-robot Q-learning: A local performance-based reward, a global performance-based reward, and a heuristic strategy referred to as shaped reinforcement; it combines rewards based on local rewards, global rewards and coordination interference of the robots. Balch [3] reports on using reinforcement learning in individual robot behavior selection. The rewards for the selection were carefully selected for each domain and application, in contrast to our work. In contrast to these investigations, we explore a domain-independent reward function, based on minimizing resource use, and use them in selecting between coordination methods, rather than task behaviors.

Wolpert et al. [22, 21] 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. Later work by Agogino and Tumer further extended this approach [1]. Similarly to these 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 distinguishes also the time spent coordinating and time spent executing the task.

3. LIMITING RESOURCE SPENDING

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

Multilateral coordination prevents and resolves conflicts among robots in a multi-robot system (MRS). 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 [7, 15, 19, 20]. Not one method is good for all cases and group sizes [17]. 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 $A = \{\ldots, a_i, \ldots\}, 1 \leq i \leq N$ 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 1 illustrates this notation.

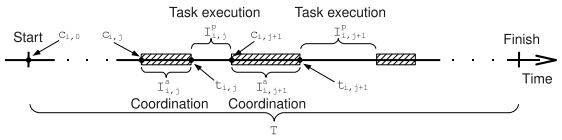


Figure 1: Illustration of task time-line, from the robots' perspective. Task execution is occasionally interrupted by the requirement to spend resources on coordination.

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. Typically, reward functions are given, and indeed most previous work focuses on learning algorithms that use the reward functions as efficiently as possible. Instead, we assume a very basic learning algorithm (a simple Q-Learning variant), and instead focus on defining a reward function (see below).

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).
- Finally, the most general way is in using of a resources manager with capability to monitor resource usage by components of the robot system. The description of such a manager is beyond the scope of this work, though we note in passing that such managers exist already for general operating systems.

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 [17]), and is limited to only considering individual intrinsic resources (unlike [24]).

Let us use a cost function $cost_i(\alpha, t)$ to represent the costs due to using coordination method $\alpha \in M$ at any time t during the lifetime of the robot. 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 $C_{i,j}^C$ of a particular event of robot i at time $c_{i,j}$:

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

$C_{i,j}^C$ 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_i^a is better. Note that this is true regardless of the use of other resources (which are measured by C_i^C). 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:

$$ACC_{i,j}(\alpha) \equiv I_{i,j}^a(\alpha) + C_{i,j}^C(\alpha) \quad (2)$$

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 $I_{i,j}^p$ 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:

$$EI_{i,j}(\alpha) \equiv \frac{ACC_{i,j}(\alpha)}{I_{i,j}^a(\alpha) + I_{i,j}^p(\alpha)} = \frac{I_{i,j}^a(\alpha) + C_{i,j}^C(\alpha)}{I_{i,j}^a(\alpha) + I_{i,j}^p(\alpha)} \quad (3)$$

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.

In this paper we use the simple single-state Q-learning algorithm to estimate the EI values from the robot's individual perspective. The learning algorithm we use is stateless:

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

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

4. EXPERIMENTS IN MULTIPLE DOMAINS

We now turn to briefly survey a subset of experiment results, in multiple domains, supporting the use of EI in multi-robot team tasks. Due to lack of space, we only provide representative results in each domain.

Foraging in TeamBots Simulation. Foraging is a canonical task in multi-robot systems research. Here, robots locate target items (pucks) within the work area, and deliver them to a goal region. As was the case in Rosenfeld et al.'s work [17], we used the TeamBots simulator [2] 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.

We compare the EI method with random coordination algorithm selection (RND), and to the method of Rosenfeld et al. (ACIM) (which uses offline learning [17]). Each of these selection methods selectssss between three types of coordination methods (α), described also in [17]: Noise (which essentially allows the robots to collide, but increases their motion uncertainty to try to escape

collisions), Aggression [20] (where one robot backs away, while the other moves forward), and Repel, in which robots move away (variable distance) to avoid an impending collision.

Figures 2(a)–2(c) show a subset of results. In all, the X axis marks the group size, and the Y axis marks the number of pucks collected. Figure 2(a) 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. Figure 2(b) shows the advantage of EI over ACIM when resource costs apply. Here, when ACIM takes fuel costs into account, it performs well. But when it does not, its performance is very low. On the other hand, EI with fuel costs and without perform well. Finally, Figure 2(c) shows how ACIM and EI respond to unknown costs. 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, $p = 0.0027$).

Foraging in AIBO Robots. We have also utilized EI-based adaptation in foraging experiments with Sony AIBO robots, shown in Figure 3. Three robots were placed within a boxed arena, measuring 2m by 2m, and containing four pucks The robots were allowed up to 10 minutes to collect the pucks. We implemented two basic coordination methods: Noise and Repel (described above). We ran 8 trials of Noise, and 9 of Repel.

Figure 3: 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.

We contrasted the performance of the three groups (Noise, Repel, and EI). Figure 4(a) shows the pucks collected per minute by each of the three methods (median). 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. This is to be expected, because the EI algorithm utilized constant exploration rate (up 19% of the conflicts of each robot). Thus even under the best of conditions, the EI runs are expected to worse. We see the same trend in Figure 4(b), which shows the average number of conflicts in the different groups. We again 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(c) shows the average EI values that were achieved at the end of each run. 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).

EI in Virtual Environments. Finally, we evaluated the use of EI with robots in virtual environments. Here, we utilized robots that operate in VR-Forces[13], 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.

Within this environment, we conducted experiments with four virtual robots, where the coordination was implicit, rather than explicit. All of the four robots 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 ($path2$). 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 (Figure 5) 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.

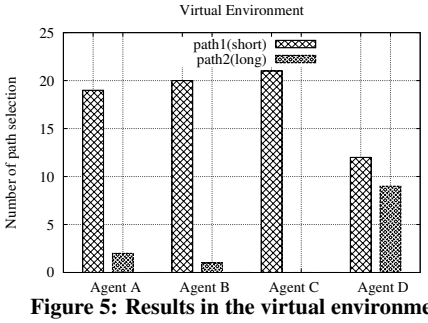
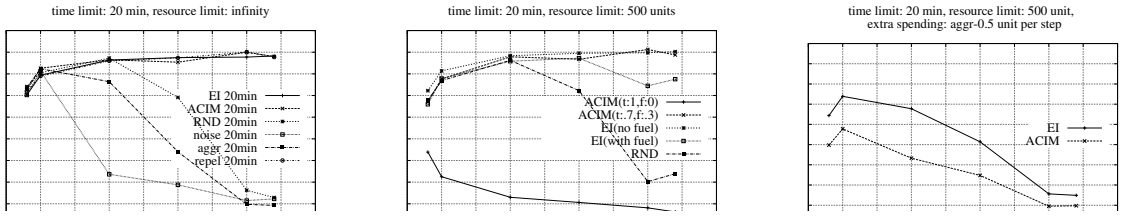


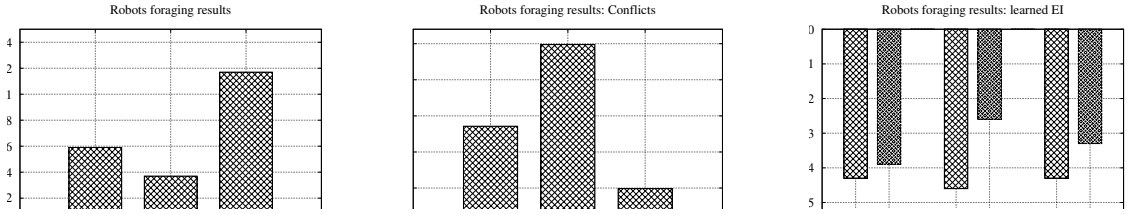
Figure 5: Results in the virtual environment domain.

5. WHY DOES EI WORK?



(a) $T = 20$, no resource limits. (b) $T = 20$, severe fuel limits. (c) $T = 20$, resource cost unknown.

Figure 2: Results from the TeamBots foraging domain.



(a) Pucks collected per minute (median). (b) Mean number of conflicts (lower result is better). (c) Negative EI values for Noise and Repel in the three robots (higher is better).

Figure 4: Results from the AIBO foraging domain.

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) the task is bound in time. 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 turn to game theory to represent 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. Assume for simplicity of the discussion that we limit ourselves to two robots, and that whenever they are in conflict, they are both aware of it, and they both enter the conflict at the same time. 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.

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

[1]This is true in all communication-less coordination methods, which are used in most previous work [20, 17]. When used with communication-based coordination method, this restriction may be removed. It might also be possible to relax this restriction if robots

appears in Figure 6.

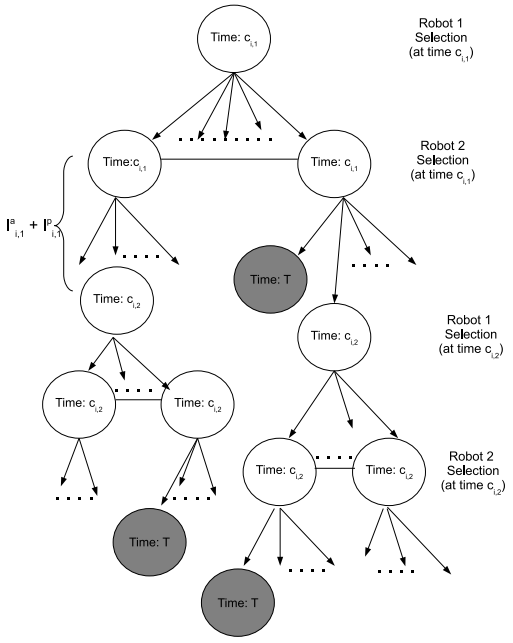


Figure 6: An illustration of the extensive-form game tree for an LCT task. Conflict times are denoted in the nodes. Terminal nodes (total time=T) are dark. Note that the second conflict $c_{i,2}$ may occur at different absolute times depending on the choices of the robots at time $c_{i,1}$.

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, 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:

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

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:

$$\begin{aligned} U_{i,j} \equiv \; & [gain(I^p_i(\alpha_i)) + gain(I^p_j(\alpha_j))] \\ & - [C^C_i(\alpha_i) + C^C_j(\alpha_j)] \end{aligned} \tag{4}$$

where we use (for robot i) the notation $gain(I^p_i(\alpha_i))$ to denote the gains achieved by robot i during the task execution time $I^p_i(\alpha_i)$. 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 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^2_1	α^2_2
α^1_1	$K_{1,1}U_{1,1}$	$K_{1,2}U_{1,2}$
α^1_2	$K_{2,1}U_{2,1}$	$K_{2,2}U_{2,2}$

Table 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

$$u_k(\alpha_k) \equiv gain(I^p_k(\alpha_k)) - C^C_k(\alpha_k).$$

This results in the revised matrix game appearing in Table 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 selecting method k can be rewritten as $\frac{T}{I^a_l(\alpha_k)+I^p_l(\alpha_k)}u_l$.

could infer each others' choices post-factum.

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

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

$$\frac{T[g(I_l^p(\alpha)) - c(I_l^a(\alpha))]}{I_l^a(\alpha) + I_l^p(\alpha)} \propto \frac{g(I_l^p(\alpha)) - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \quad (5)$$

$$\propto \frac{I_l^p(\alpha) - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \quad (6)$$

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:

$$\frac{I_l^p(\alpha) - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} = \frac{I_l^p(\alpha) + [I_l^a(\alpha) - I_l^a(\alpha)] - c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \quad (7)$$

$$= \frac{[I_l^p(\alpha) + I_l^a(\alpha)] - [I_l^a(\alpha) + c(I_l^a(\alpha))]}{I_l^a(\alpha) + I_l^p(\alpha)} \quad (8)$$

$$= \frac{I_l^p(\alpha) + I_l^a(\alpha)}{I_l^p(\alpha) + I_l^a(\alpha)} - \frac{I_l^a(\alpha) + c(I_l^a(\alpha))}{I_l^a(\alpha) + I_l^p(\alpha)} \quad (9)$$

$$= 1 - EI_l(\alpha) \quad (10)$$

$$\propto -EI_l(\alpha) \quad (11)$$

Thus the game matrix is in fact equivalent to the following matrix (Table 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	$-EI_1(\alpha_2^1), -EI_2(\alpha_1^2)$	$-EI_2(\alpha_2^2), -EI(\alpha_2^2)$

Table 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 [4, 23, 10]. 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 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. 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).

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 [11]. 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.

There remains the question of why do agents converge to the maximal payoff Nash equilibrium. 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 [5, 16]. However, these works typically assume public knowledge of some kind, which is absent in our domain. Thus we leave this as a conjecture.

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 our 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 chal-

lenged 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.

The use of the 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.

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 was that not one single equilibrium point existed, as one combination of paths works always (i.e., a mixed Nash equilibrium). Indeed, the algorithm converged quickly to selecting between two almost equally-valued alternatives, reflecting the two top choices.

6. SUMMARY

This paper 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.

Acknowledgements. We thank Dov Miron and Shai Shlomai for their assistance with the AIBO experiments.

7. REFERENCES

[1] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *JAAMAS*, 17(2):320–338, 2008.

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

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

[4] 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.

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

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

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

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

[9] 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.

[10] 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.

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

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

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

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

[15] E. Ostergaard, G. Sukhatme, and M. Matarić. Emergent bucket brigading. In *Agents-01*, pages 29–30, 2001.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] D. H. Wolpert and K. Tumer. Collective intelligence, data routing and braess' paradox. *JAIR*, 16:359–387, 2002.

[22] D. H. Wolpert, K. R. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In *Agents-99*, pages 77–83. ACM Press, 1999.

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

[24] M. Zuluaga and R. Vaughan. Reducing spatial interference in robot teams by local-investment aggression. In *IROS*, Edmonton, Alberta, August 2005.

Decentralized Learning in Wireless Sensor Networks

Mihail Mihaylov
Vrije Universiteit Brussel
Brussels, Belgium
mike@como.vub.ac.be

Karl Tuyls
Technische Universiteit
Eindhoven
Eindhoven, The Netherlands
k.p.tuyls@tue.nl

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium
ann.nowe@como.vub.ac.be

ABSTRACT

In this paper we use a reinforcement learning algorithm with the aim to increase the autonomous lifetime of a Wireless Sensor Network (WSN) and decrease latency in a decentralized manner. WSNs are collections of sensor nodes that gather environmental data, where the main challenges are the limited power supply of nodes and the need for decentralized control. To overcome these challenges, we make each sensor node adopt an algorithm to optimize the efficiency of a small group of surrounding nodes, so that in the end the performance of the whole system is improved. We compare our approach to conventional ad-hoc networks of different sizes and show that nodes in WSNs are able to develop an energy saving behaviour on their own and significantly reduce network latency, when using our reinforcement learning algorithm.

Keywords

Energy Efficiency, Latency, Reinforcement Learning, Wireless Sensor Network

1. INTRODUCTION

An increasingly popular approach for environmental and habitat monitoring is the use of Wireless Sensor Networks (WSNs) [2, 6]. The nodes in such a WSN are limited in power, processing and communication capabilities, which requires that they optimize their activities, in order to extend the autonomous lifetime of the network and minimize latency. A complicating factor is communication, because some nodes can fall outside the transmission range of the base station, or can belong to different stakeholders, serving various purposes, thus rendering the common centralized approach inapplicable for large networks.

This paper extends the work done in [5] to a random network topology, reduces the communication overhead and significantly improves the results. In this work we use a reinforcement learning algorithm to optimize the energy efficiency of a WSN and reduce its latency in a decentralized manner. We achieve that by making nodes (hereby regarded as agents) develop energy-saving schemes by themselves without a central mediator. The idea behind this approach is that agents learn to reduce the negative effect of their actions on other agents in the system, based on a certain reward function. We investigate the performance of our algorithm in two networks of different sizes. We show that when agents learn to optimize their behaviour, they can increase the energy efficiency of the system and significantly decrease its latency with minimal communication overhead.

The outline of the paper is as follows: Section 2 presents the background of our approach by describing the basics of a wireless sensor network and the MAC communication protocol. Section 3 describes the idea behind our algorithm and its application to the energy efficiency optimization of nodes. In Section 4 we explain the experiments and discuss our findings. Lastly, Section 5 presents our conclusions from this research and suggests some areas for improvement in the future.

Cite as: Decentralized Learning in Wireless Sensor Networks, Mihail Mihaylov, Karl Tuyls and Ann Nowé, *Proc. of the Adaptive and Learning Agents Workshop (ALA 2009)*, Taylor and Tuyls (eds.), May, 12, 2009, Budapest, Hungary.

2. BACKGROUND

In this section we describe the basics of a Wireless Sensor Network and the MAC communication protocol. Subsection 2.1 elaborates on WSNs and Subsections 2.2 and 2.3 explain the working of the MAC protocol and the way nodes communicate.

2.1 Wireless Sensor Networks

A Wireless Sensor Network is a collection of densely deployed autonomous devices, called sensor nodes, that gather environmental data with the help of sensors. The untethered nodes use radio communication to transmit sensor measurements to a terminal node, called the sink. The sink is the access point of the observer, who is able to process the distributed measurements and obtain useful information about the monitored environment. Sensor nodes communicate over a wireless medium, by using a multi-hop communication protocol that allows data packets to be forwarded by neighbouring nodes to the sink. This concept is illustrated in Figure 2.1. The environmental or habitat monitoring is usually done over a long period of time, taking into account the latency requirements of the observer.

The WSN can vary in size and topology, according to the purpose it serves. The sensor network is assumed to be homogeneous where nodes share a common communication medium (e.g. air, water, etc.). We further assume that the communication range is equal in size and strength for all nodes. They have a single omnidirectional antenna that can only *broadcast* a message, delivering it to all nodes in range. In our network, sensor nodes can neither vary their transmission power, nor are they able to estimate their distance

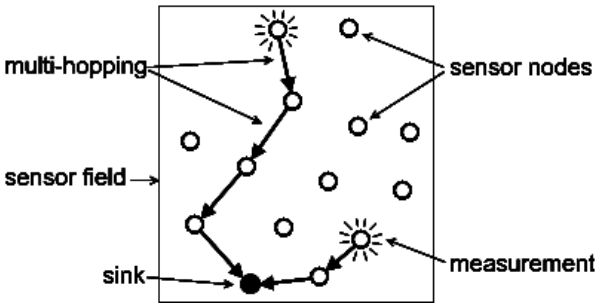


Figure 1: Wireless Sensor Network

from the transmitting node by measuring the signal strength – such features are not generally available in sensor nodes and therefore are not considered here. The motivation to use such simple devices is to reduce the overall cost of nodes and to keep our solution applicable to the most general sensor network.

In this paper we show that the selfish and computationally bounded agents can optimize their own performance, in a decentralized manner, in order to reduce both their own energy consumption and the latency of the network. We assume that communication between the agents is limited and that central control is not possible. We further require that the communication protocol considers not only energy efficiency, but also scalability and fault tolerance, so that our approach is able to adapt to a dynamic topology, where nodes may move, fail or new nodes may be added to the system. The communication protocol, therefore, constitutes an important part of the WSN design.

2.2 The MAC Protocol

The Medium Access Control (MAC) protocol is a data communication protocol, concerned with sharing the wireless transmission medium among the network nodes. Typical MAC protocols, used by ad-hoc networks, cannot be applied to WSNs, due to a number of differences between the two types of networks. Some differences include the large number and density of sensor nodes in a WSN, compared to the nodes in ad-hoc networks; the frequently changing topology of sensor nodes and their power constraints, etc.

We use a simple asynchronous MAC protocol that divides the time into small discrete units, called frames. Each node independently determines its sleep duration (or schedule), i.e. the amount of time in a frame that the node's antenna will be turned off. During that time the agent is not able to communicate with other nodes and therefore saves energy. Nevertheless, the agent continues its sensing and processing tasks. Our protocol allows nodes to synchronize their schedules prior to communication and thus avoid collisions and overhearing – typical sources of energy waste.

Since communication is the most energy expensive action [7], it is clear that in order to save more energy, a node should sleep more. However, when sleeping, the node is not able to send or receive any messages, therefore it increases the latency of the network, i.e., the time it takes for messages to reach the sink. On the other hand, a node does not need to listen to the channel when no messages are being sent, since it loses energy in vain. As a result, nodes should learn on their own the number of time slots they should spend sleeping within a frame. For example, nodes far away from the sink may learn to sleep more, since they will have fewer messages to forward, while nodes close to the sink should learn to listen more, because the workload near the sink is usually heavier. Learning to optimize nodes' own schedules will ensure good energy efficiency of the network, while minimizing the latency. The MAC protocol should therefore support the exchange of additional information, necessary for the algorithm for optimization. It is clear that the amount of this information within message packets should be kept as little as possible, in order to minimize the energy waste by control packet overhead. A brief description of the communication protocol is presented next.

2.3 Communication and Routing

When the WSN is deployed, nodes first need to determine their hop distance to the sink, i.e. the minimum number of nodes that will have to forward their packets. This is achieved by broadcasting SYNchronization (SYN) packets in the following way: the sink broadcasts a SYN packet, containing a counter, initially set to 0; all receivers set their hop equal to the counter, increment it and broadcast the new SYN packet further on, with a small random delay to avoid collisions. For example, a node right next to the sink will receive a SYN packet with `hop=0` and will broadcast a new one with `hop=1`.

When a node has a message to send[1], it broadcasts a Request To Send (RTS) packet to all nodes within range, which we call neighbours (or neighbouring nodes). All neighbours at an equal or higher hop simply go to sleep, since they do not need to forward the sender's message. All lower-hop neighbours wait a small random amount of time before replying with a Clear To Send (CTS) packet. Once one node broadcasts a CTS packet, all its neighbours go to sleep, except the sender of the RTS, who in turn broadcasts the actual data. In other words, all immediate neighbours of the two communication partners are sleeping during the broadcast of the data, in order to avoid collisions and overhearing. Once the receiver obtains the data packet, it replies with an ACKnowledgment (ACK) and thus the communication is over.

3. LEARNING ALGORITHM

Besides on its hardware, the energy consumption of a node is also dependent on its position in the WSN. Nodes, closer to the sink have to forward more messages and therefore need to listen more, while those far away from the sink could spend more time sleeping. For this reason, the behaviour of agents cannot be the same for all (e.g. all listen and sleep the same amount of time in a frame). Each node needs to *learn* what behaviour is energy efficient in the network. To achieve that, we make nodes adopt an algorithm for optimization in order to improve the performance of the whole system.

Each agent in the WSN uses a reinforcement learning (RL) algorithm to learn an optimal schedule (i.e. sleep duration in a frame) that will maximize the energy efficiency and minimize the latency of the system in a distributed manner. The main challenge in such a decentralized approach is to define a suitable reward function for the individual agents that will lead to an effective emergent behaviour as a group. Another challenge is that agents in a WSN can obtain only local information from surrounding nodes, due to their small

[1]We assume that all messages are forwarded toward the sink.

transmission range. To tackle these challenges, we proceed with the definition of the basic components of the reinforcement learning algorithm.

3.1 Actions

The actions of each agent are restricted to selecting a sleep duration for a frame. The action space consists of a discrete number of sleep durations at equal increments within one frame length. Defining the size of the increment constitutes a tradeoff, since a rather large value will result in only few actions for the agent to choose. On the other hand, a small increment will result in a large action set, which makes it difficult for the algorithm to converge [4]. Agents choose their actions according to a probability distribution and use that action for a certain number of frames, which we call a frame window. The reason for using an action for more than one frame is that the agent will thus have enough time to experience the effect of that action on the system. The size of the frame window and the discretization increment will be discussed in Section 4.1.

3.2 Rewards

Before proceeding with the formulation of the reward signal, we first need to define what Energy Efficiency (EE) of a single agent is.

3.2.1 Energy Efficiency

We consider an agent to be energy efficient when it minimizes most of the major sources of energy waste in WSN communication – idle listening, overhearing and unsuccessful transmissions, while quickly forwarding any packets in its queue to ensure low network latency. Formally, the energy efficiency for agent i in frame f is:

$$\begin{aligned} EE_{i,f} = \quad & \alpha(1 - IL_{i,f}) + \beta(1 - OH_{i,f}) + \\ & + \gamma(1 - UT_{i,f}) + \delta(1 - DQ_{i,f}) + \epsilon BL_i \end{aligned}$$

where:

- $IL_{i,f}$ is the duration of idle listening of agent i within frame f;
- $OH_{i,f}$ is the duration of overhearing of agent i within frame f;
- $UT_{i,f}$ is the amount of unsuccessful transmissions of agent i within frame f;
- $DQ_{i,f}$ is the sum of the durations that each packet spent in the queue of agent i within frame f;
- BL_i is the remaining battery life of agent i;
- the constants α, β, γ, δ and ϵ weight the different terms accordingly.

All values are in the unit interval.

It is easy to show that if agents try to increase simply their own energy efficiency, they will prefer to sleep until they obtain a measurement (thus minimizing energy waste) and then wake up only to broadcast it (to ensure low latency). That will not lead to high global efficiency, due to the high number of collisions and unsuccessful transmissions that nodes will experience. Therefore, individual agents should also consider other agents in the system when optimizing their own behaviour. A similar approach was undertaken by Wolpert and Tumer in [8], where they apply their Collective Intelligence framework to align the selfish agents' goals with the system goal.

3.2.2 Effect Set

Our belief is that if each agent "cares about others" that will improve the performance of the whole system. To achieve that, we introduce the concept of an Effect Set (ES) of a node, which is the subset of that node's neighbourhood, with which it communicates within a frame window. In other words, the ES of agent i is the set N_i of nodes, whose messages agent i (over)hears within a frame window. Thus, the energy efficiency of agent i is directly dependent on the actions of all agents in N_i and vice versa.

3.2.3 Effect Set Energy Efficiency

As a result of the influence of agents on each other's performance, we form our hypothesis. We believe that if each agent seeks to increase not only its own efficiency, but also the efficiency of its ES, this will lead to higher energy efficiency of the whole system. For this reason, we set the reward signal of each agent to be equal to its mean Effect Set Energy Efficiency (ESEE) over a frame window of size $|F|$. We define the ESEE of agent i in the frame window F as

$$\mathrm{ESEE}_{i,F} = \frac{1}{|F|} \cdot \sum_{f}^{F} \frac{\mathrm{EE}_{i,f} + \sum_j \mathrm{EE}_{j,f}}{|N_i| + 1} \qquad \forall j \in N_i$$

where $EE_{i,f}$ is the energy efficiency of agent i in frame f and $|N_i|$ is the number of agents in the effect set of agent i. In other words, the reward signal that each agent receives at the end of each frame window is the mean energy efficiency of its effect set and of itself, averaged over the size of the frame window. Thus, agents will try to increase the value of their ESEE by optimizing their own behaviour.

3.2.4 Challenge

One challenge in our reward signal is that nodes cannot compute their ESEE directly, because to do so, they would have to obtain the efficiency of each agent in N_i. To achieve that, nodes simply include the value of their own EE in the three control packets – RTS, CTS and ACK, so that neighbouring agents can (over)hear these values and compute their ESEE. This is the only information that nodes need to exchange for our algorithm to work. Although including additional information in control packets is expensive, we will show that the network performs still better than one without learning. We will now show how each agent can learn to optimize its ESEE.

3.3 Update Rule

At the end of each frame window, agents compute the average ESEE from the past frames and use this value to learn the best sleep duration that will maximize efficiency and minimize latency. Agents use the update rules of a classical learning automata to update their action probabilities. More specifically, after executing action x in every frame of F, its probability $p_i(x)$ is updated in the following way

$$p_i(x) \leftarrow p_i(x) + \lambda \cdot ESEE_{i,F} \cdot (1.0 - p_i(x))$$

where λ is a user-defined learning rate. The probability $p_i(y)$ for all other actions $y \neq x$ in the action set of agent i then

becomes

$$p_i(y) \leftarrow p_i(y) - \lambda \cdot ESEE_{i,F} \cdot p_i(y) \qquad \forall y \neq x$$

At the beginning of each frame, agents select their actions according to the updated probabilities and execute them in that frame window. As a result, the learning process is done on-line – the algorithm adapts to the topology of the network and the traffic pattern, which typically cannot be known in advance in order to train nodes off-line.

4. RESULTS

4.1 Experimental Setup

We applied our algorithm on two networks of random topology and different sizes – one small network with 10 nodes and a large one with 50 nodes. The density of both networks was the same, i.e. on average each node had 4 neighbours, because we found out empirically that it influences the speed of learning. In this work we focus on how well learning scales in terms of the number of nodes, rather than in terms of the density. The reason for the slower learning in more dense networks is the higher degree of interdependence of the actions of neighbouring agents. In other words, agents in dense networks have to consider more neighbours in optimizing the performance of their ESEE and thus converge to an optimal action slower than agents in less dense networks. An in-depth study of the optimal density of sensor networks is presented in [3].

We considered networks of random topology, rather than organized in a grid structure (as in [5]), so that the WSN can be deployed more freely (e.g. nodes can be scattered from a moving vehicle). The synchronization phase of the network was set to 20 seconds – this duration was enough for all nodes to find their hop distance to the sink in both networks. During this phase, agents do not learn to optimize their behaviour, since the resulting traffic pattern is independent of that from the actual data. We set the duration of a frame to 0.5 seconds and the message rate – to 1 sensor measurement in a frame on average. We chose this high message rate to make the effect of agents' actions more apparent and to give agents enough information in order to learn a good policy. A sufficient frame window size was found to be 4, i.e. agents repeat their selected action for 4 times, before obtaining a reward signal. The discretization coefficient (Subsection 3.1) was selected such that it results in 11 different actions (or sleep durations). The 5 weighting coefficients in the computation of the EE (Subsection 3.2) were experimentally chosen in the following way: $\alpha = 0.2$, $\beta = 0.3$, $\gamma = 0.1$, $\delta = 0.3$ and $\epsilon = 0.1$. The best learning rate λ was found to be 0.280 for the small network and 0.299 for the large one, where in both cases the initial action probability was uniform. Finally, the networks were allowed to run for 500 seconds, i.e. 1000 frames, before the simulation was terminated.

4.2 Experiments

As stated above, we evaluated our algorithm on two random topology networks of the same density, but of different sizes. We compared the performance of each setting to a network of the same size where agents do not optimize their behaviour, but rather all sleep the same pre-defined amount of time. In each experiment we measured six performance criteria:

1. Average **remaining battery** at the end of the simulation (i.e. after 1000 frames). This value shows what the battery levels of nodes will be after 500 seconds of runtime with the selected settings.
2. **Standard deviation of the average remaining battery** – indicates the difference between the most and the least efficient nodes. Here a small deviation is desirable, since it signifies a rather equal dissipation of energy over time.
3. **Average latency** of the network over all packets delivered to the sink. This criterion measures the average time a message takes from the moment it was generated to the time it reaches the sink.
4. **Standard deviation of the average latency** of the network. Again, a small deviation is preferable, because it signifies consistent traffic latency.
5. **Maximum latency** of the network, i.e. the latency of the packet that took the most time to be delivered to the sink. This value indicates the worst case scenario for the latency that the user of the WSN can experience for a packet.
6. Number of **received packets** by the sink within 500 seconds. This is an inverse indication of latency and it shows how many messages actually reached the sink during the simulation runtime.

Small Network (10 nodes)

performance criteria	obj.	not learning	learning	improvement
End battery - mean (%)	max	23.283	25.706	10.4% (increased)
End battery - std. dev. (%)	min	4.514	2.220	50.8% (decreased)
Latency - mean (sec.)	min	11.413	3.937	65.5% (decreased)
Latency - std. dev. (sec.)	min	8.455	3.348	60.4% (decreased)
Latency - max (sec.)	min	62.359	18.975	69.6% (decreased)
Packets arrived at Sink	max	2007	2167	8.0% (increased)
Sleeping time - mean (sec.)	n/a	0.120	0.094	n/a
Sleeping time - std. dev. (sec.)	n/a	0.000	0.136	n/a

arge Network (0 nodes)

performance criteria	obj.	not learning	learning	improvement
End battery - mean (%)	max	22.375	22.789	1.9% (increased)
End battery - std. dev. (%)	min	4.362	5.251	20.4% (increased)
Latency - mean (sec.)	min	20.552	5.823	71.7% (decreased)
Latency - std. dev. (sec.)	min	14.768	5.850	60.4% (decreased)
Latency - max (sec.)	min	88.669	50.892	42.6% (decreased)
Packets arrived at Sink	max	544	2296	322.1% (increased)
Sleeping time - mean (sec.)	n/a	0.220	0.166	n/a
Sleeping time - std. dev. (sec.)	n/a	0.000	0.176	n/a

Figure 2: Comparison between no learning and learning in the small and large networks

The sleep duration of the two networks without learning was selected such that it maximizes the above six performance criteria. The same technique was used to select the best learning rate of the networks with optimization. In other words we compared the optimal "non-learning" system to the optimal one *with* learning. This comparison is displayed in Figure 2. The first column shows the above six performance criteria, where the last two rows indicate the average sleeping time of the agents and the standard deviation. The second column indicates the objective (*obj.*) of

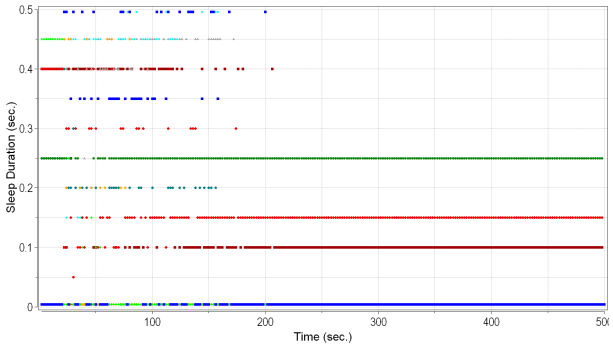


Figure 3: Sleep Duration over Time when learning, Small Network (10 nodes)

the corresponding performance criterion – whether it should be maximized (*max*) or minimized (*min*). The third and forth column display the results from our experiments when agents are not learning and when they are learning, respectively. The column labeled *improvement* displays the percentage increase of the six performance measures when agents adopt our learning algorithm.[2]

As it can be seen from Figure 2, in both cases our learning agents sleep on average less than those in the non-learning network. One would expect that less sleeping results in lower battery level, due to idle listening and overhearing, and higher latency, due to collisions. However, our learning algorithm aims to reduce precisely those sources of energy waste, by making nodes optimize their behaviour, based on the actions of neighbouring nodes. Thus, agents learn to avoid "harming" other agents by adapting to the traffic pattern and therefore learning the optimal sleep duration in their neighbourhood. In other words, agents learn to sleep when their neighbours communicate (so as to avoid overhearing); stay awake enough to forward messages quickly (and thus decrease latency); and yet sleep enough (to ensure longer network lifetime). Figure 3 shows agents' actions (sleep durations) over time. Each coloured dot represents that agent's selected action at the corresponding time in the simulation. The graph indicates that in the small network agents learn, as the time progresses, to sleep less and listen more, so that they reduce the latency of the network, while increasing its lifetime.[3] The figure also shows that in the beginning of the simulation agents explore their action set and after approximately 200 seconds, the policy of all agents converges to an optimal action. In other words, after 400 frames, each agent finds the sleep duration that maximizes its ESEE and then sticks to it. The effect of adapting to the traffic pattern is even more apparent in the large network, where agents are able to decrease the average latency with over 70%, resulting in three times more packets delivered to the sink (cf. Figure 2).

Figure 4 compares the overhearing duration of nodes over time in the small network when all agents sleep the same amount of time (4(a)) and when they learn their optimal sleep duration (4(b)). Each coloured dot represents that agent's overhearing duration within a frame at the corresponding time in the simulation. It is evident that when learning, agents reduce this source of energy waste, resulting in higher end battery level.[4] In other words, as the time progresses, agents learn to sleep when their neighbours are communicating, in order to reduce the amount of packets they overhear. This is evident from the fewer dots in Figure 4(b). As a consequence of the convergence to an optimal policy (explained above), one can see a large reduction in overhearing duration after approximately 200 seconds of network runtime. However, we did not measure significant decrease in the overhearing duration of the large network, as it can be predicted from Figure 2. The end battery level of the large network increased with only 2%. This was a result of the large number of nodes and consequently the time they need to find an optimal action. Nevertheless, our learning agents had higher overall energy efficiency, due to the lower amount of unsuccessful transmission and the shorter stay of packets in the queues of the nodes.

The improved ESEE of agents in the large network can be seen in Figure 5(b), as compared to their non-learning counterparts (5(a)). Each coloured dot represents that agent's ESEE within a frame window at the corresponding time in the simulation. In other words, the graph shows the relative energy efficiency of each node's neighbourhood over time. Although the efficiency of the worst performing nodes is comparable, the average ESEE of the learning agents is higher, than that of the non-learning nodes. This means that when using our algorithm for optimization, on average agents are more energy efficient than when they are not learning. The mean ESEE of both graphs, however, is constantly decreasing, since the remaining battery level of nodes is included in this reward signal (cf. Subsection 3.2). In other words, since battery level is inevitably decreasing, so is the ESEE of both networks.

5. CONCLUSION

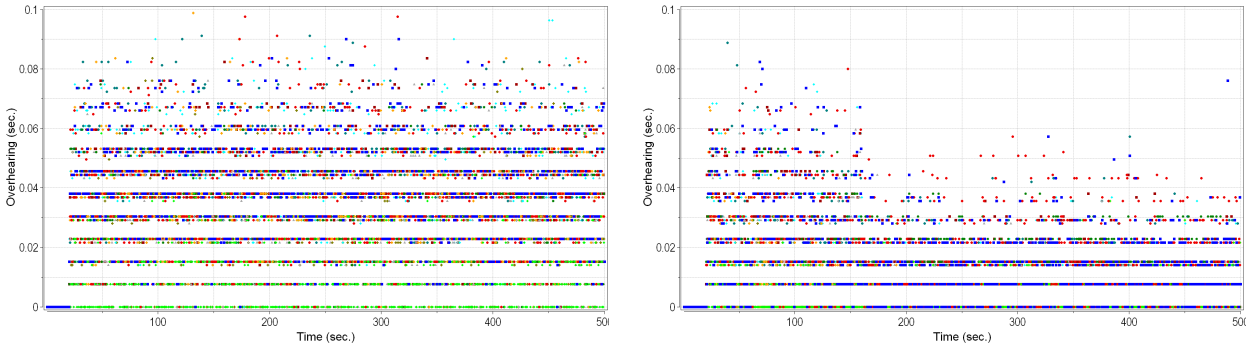
In this paper we used a reinforcement learning algorithm to improve the performance of Wireless Sensor Networks (WSN) in a decentralized manner, in order to prolong the autonomous lifetime of the network and reduce its latency. We were able to show that when agents in a WSN use an algorithm for optimization, they can learn to reduce the negative effect of their actions on other agents in the system, without a central mediator. Our results indicate that both in a small and large network, agents can learn to optimize their behaviour in order to increase the energy efficiency of the system and significantly decrease its latency with minimal communication overhead. Our results outperformed a conventional ad-hoc network, where all agents equally listen and sleep for a pre-defined amount of time. Thus, based on our experiments we can conclude that it is more beneficial for the sensor network when nodes *learn* what actions to take, rather than follow a pre-defined schedule. In our algorithm each node seeks to improve not only its own efficiency, but also the efficiency of its neighbourhood, which ensures that the agents' goal is aligned with the system goal of higher energy efficiency and lower latency.

We are currently focusing on comparing the performance of our algorithm to the X-MAC protocol [1], which aims to

[2]The concept of "improvement" is not applicable to the last two rows.

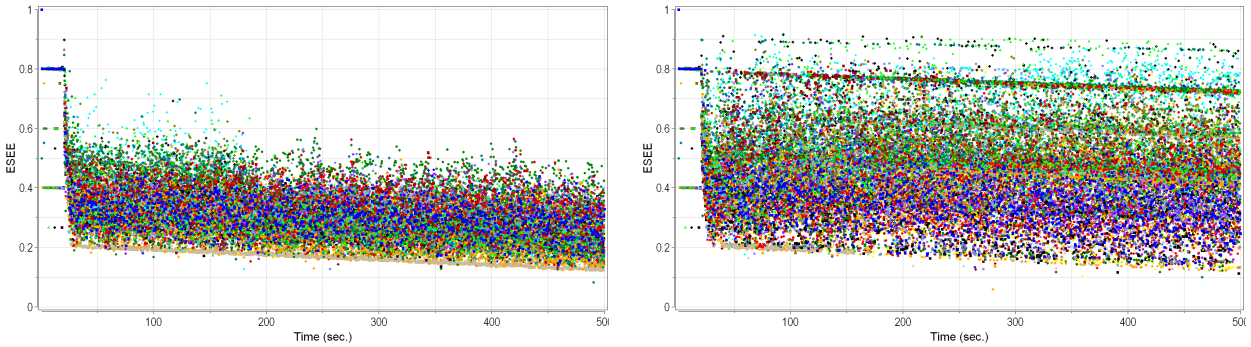
[3]Due to the discrete values in this graph, some colours overlap and thus not all of them can be displayed at the same time.

[4]The discrete steps in the graph are a result of the fixed control and data packet lengths that nodes overhear.



(a) not learning (b) learning

Figure 4: Overhearing duration over Time, Small Network (10 nodes)



(a) not learning (b) learning

Figure 5: Effect Set Energy Efficiency over Time, Large Network (50 nodes)

increase energy efficiency in a decentralized way without any communication overhead. Additionally, we aim to extend our approach, presented in this paper, to make it suitable for a larger set of WSN applications, where the network will adapt to the latency requirement of the user directly.

Future work involves computing the energy requirements of the algorithm itself and experimenting with different network topologies and reward functions to obtain a yet bigger improvement in energy efficiency and latency.

6. ACKNOWLEDGEMENTS

The authors would like to thank anonimous referees for their useful comments and suggestions.

7. REFERENCES

[1] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. Technical Report CU-CS-1008-06, University of Colorado at Boulder, May 2006.

[2] J. Carle and D. Simplot-Ryl. Energy-efficient area monitoring for sensor networks. *IEEE Computer Society*, 47(2):40–46, 2004.

[3] M. Esseghir and N. Bouabdallah. Node density control for maximizing wireless sensor network lifetime. *Int. J. Netw. Manag.*, 18(2):159–170, 2008.

[4] J. Leng. *Reinforcement learning and convergence analysis with applications to agent-based systems*. PhD thesis, University of South Australia, 2008.

[5] M. Mihaylov, A. Nowé, and K. Tuyls. Collective intelligent wireless sensor networks. In *Proceedings of the 20th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, Enschede, The Netherlands, October 2008.

[6] A. Rogers, R. K. Dash, N. R. Jennings, S. Reece, and S. Roberts. Computational mechanism design for information fusion within sensor networks. In *Ninth International Conference on Information Fusion*, 2006.

[7] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings Of The First International Conference On Embedded Networked Sensor Systems*, pages 171 – 180, Los Angeles, California, USA, 2003.

[8] D. H. Wolpert and K. Tumer. An introduction to collective intelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 2008.

The Evolution of Agent Strategies and Sociability in a Commons Dilemma

Enda Howley
Department of Information Technology
National University of Ireland
Galway
enda.howley@nuigalway.ie

Jim Duggan
Department of Information Technology
National University of Ireland
Galway
jim.duggan@nuigalway.ie

ABSTRACT

This paper explores the evolution of strategies in a n-player dilemma game. These n-player dilemmas provide a formal representation of many real world social dilemmas. Those social dilemmas include littering, voting and sharing common resources such as sharing computer processing time. This paper explores the evolution of altruism using an n-player dilemma. Our results show the importance of sociability in these games. For the first time we will use a tag-mediated interaction model to examine the n-player dilemma and demonstrate the significance of sociability in these games.

Keywords

Evolution, Learning, Cooperation, Agent Interactions, Tragedy of the Commons, Tag-Mediated Interaction Models

1. INTRODUCTION

When a common resource is being shared among a number of individuals, each individual benefits most by using as much of the resource as possible. While this is the individually rational choice, it results in collective irrationality and a non Pareto-optimal result for all participants. These n-player dilemmas are common throughout many real world scenarios. For example, the computing community is particularly concerned with how finite resources can be used most efficiently where conflicting and potentially selfish demands on those resources are common. Those resources may range from access to processor time or bandwidth.

One example commonly used throughout existing research is the *Tragedy of the Commons* [5]. This outlines a scenario thereby villagers are allowed to graze their cows on the village green. This common resource will be over grazed and lost to everyone if the villagers allow all their cows to graze, yet if everyone limits their use of the village green, it will continue to be useful to all villagers. Another example is the *Diners Dilemma* where a group of people in a restaurant agree to equally split their bill. Each has the choice to exploit the situation and order the most expensive items on the menu. If all members of the group apply this strategy, then all participants will end up paying more [2].

Cite as: The Evolution of Agent Strategies and Sociability in a Commons Dilemma, Enda Howley and Jim Duggan, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

These games are all classified as n-player dilemmas, as they involve multiple participants interacting as a group. These games involve only two players interacting through pairwise interactions. N-player dilemmas have been shown to result in widespread defection unless agent interactions are structured. This is most commonly achieved through using spatial constraints which limit agent interactions through specified neighbourhoods on a spatial grid. Limiting group size has been shown to benefit cooperation in these n-player dilemmas [14].

In this paper we will examine an n-player dilemma, and study the evolution of strategies when individuals can bias their interactions through a tag mediated environment. Furthermore, we will show how certain strategies evolve with respect to their sociability towards their peers. The simulations presented in this paper use the n-player Prisoner's Dilemma (NPD). The purpose of this paper is to examine the evolution of cooperation and sociability throughout the agent population in the NPD. The research presented in this paper will deal with a number of specific research questions:

1. Can a tag-mediated interaction model be used to determine group interactions in a game such as the NPD?
2. If agents have an evolvable trait which determines their sociability, then will this trait prove significant to the emergence of cooperation in the agent society?

The following section of his paper will provide an introduction to the NPD and a number of well known agent interaction models. In the Experimental Setup Section we will discuss our simulator design and our experimental parameters. Our Results Section will provide a series of game theoretic simulations. Finally we will outline our conclusions and future work.

2. BACKGROUND RESEARCH

In this section we will introduce the NPD game while also discussing some existing background research relevant to this paper.

2.1 The N-Player Prisoner's Dilemma

The n-player Prisoner's Dilemma is also known as the Tragedy of the Commons [5] and the payoff structure of this game is shown in Figure 1.

On the horizontal axis is the fraction of cooperators in the group of n players in a particular game. On the vertical axis is the payoff for an individual participating in a

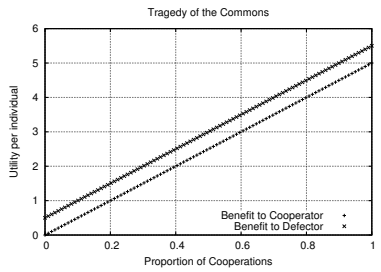


Figure 1: The N-Player Prisoner's Dilemma

game. There is a linear relationship between the fraction of cooperators and the utility received by a game participant. Importantly, the payoff received for a defection is higher than for a cooperation. The utility for defection dominates the payoff for cooperation in all cases. Therefore, an individual that defects will always receive a higher payoff than if they had chosen to cooperate. The result of this payoff structure should result in an advantage to defectors in the agent population. Despite this, a cooperator in a group of cooperators will do much better than a defector in a group of defectors.

This game is considered a valid dilemma due to the fact that individual rationality favors defection despite this resulting in state which is less beneficial to all participants. In our case where all individuals defect they all receive 0.5. This state is a non-pareto, sub-optimal, and collectively irrational outcome for the agent population. For all values of x this can be expressed as follows: $U_d(x) > U_c(x)$. x is the fraction of cooperators while U_d and U_c are utility functions based on the fraction of cooperators in the group.

2.2 Agent Interaction Models

A number of alternative agent interaction models have been proposed and examined, such as spatial constraints [11, 10] and tag mediated interactions, [13]. The importance of group size has been demonstrated explicitly through tags in the PD by [7]. Similarly in the NIPD [14]. Yao and Darwin demonstrated the effects of limiting group size, which was shown to benefit cooperation. Increasingly complex aspects of agent interactions have been examined by a number of authors, these include the effects of community structure on the evolution of cooperation [12, 1]. These have shown that neighbourhood structures benefit cooperation.

In this paper we are most concerned with tag-mediated interactions. Tags are visual markings or social cues which can help bias social interactions [6]. They are a commonly used agent interaction model and can be considered akin to football supporters identifying each other through wearing their preferred team colours. Similarly individuals can identify each other in conversations through a common language, dialect, or regional accent. Tag-mediated interaction models are often considered as more abstract interaction models, and thereby useful to represent agent interactions more abstractly without the specific characteristics of a specific topology or implementation. The research presented by Riolo demonstrated how tags can lead to the emergence of cooperation in the Prisoner's Dilemma [13]. Riolo investigated both a fixed and an evolved tag bias. More recently tags have been successfully applied to multi-agent problems [3, 4]. Tags have been shown to promote mimicking and thereby have major limitiations where complimentary actions are required by agents. Cooperation that can be achieved through identical actions is quite easily achieved using tags, yet behaviours that require divergent actions are problematic [9, 8].

In this paper we will augment existing research to show the effects of using a tag-mediated interaction model to determine group interactions in the NPD. The following section will provide a detailed specification of our simulator and the overall design of our experiments.

3. EXPERIMENTAL SETUP

In this section we will outline our agent structure, our agent interaction model and our evolutionary algorithm.

3.1 Agent Genome

In our model each agent is represented through an agent genome. This genome holds a number of genes which represents how that particular agent behaves.

$$Genome = G_C, G_T, G_S, \tag{1}$$

The G_C gene represents the probability of an agent cooperating in a particular move. Each agent has G_C gene which never changes throughout their lifetime. The G_T gene represents the agent tag. This is represented in the range $[0 \ldots 1]$ and is used to determine which games each agent participates. Finally, the G_S gene represents the sociability of each agent. This gene is also a number in the range $[0 \ldots 1]$ which acts as a degree of sociability for that individual agent. Initially these agent genes are generated using a uniform distribution for the first generation. Over subsequent generations new agent genomes are generated using our genetic algorithm.

3.1.1 Tag Mediated Interactions

In our simulations each agent interacts through a simple tag mediated interaction model. We adopt a similar tag implementation as that outlined by Riolo [13]. In our model each agent has a G_T gene which is used as their tag value. Each agent A is given the opportunity to make game offers to all other agents in the population. The intention is that this agent A will host a game and the probability other agents will participate is determined as follows.

$$d_{A,C} = 1 - |A_{GT} - C_{GT}| \tag{2}$$

This equation is based on the absolute value between the tag values of two agents A and C. This value is used to generate two roulette wheels R_a and R_c for A and C. These two roulette wheels will then be used to determine agent A's attitude to C and agent C's attitude to A. An agent C will only participate in the game when both roulette wheels have indicated acceptance. The distribution of these roulette wheels are also influenced by each agents sociability gene. This gene acts like a scalar value which is used to reflect that some agents are more sociable than others and will therefore be more willing to play with their peers. This is shown in the following equation, where R_a represents the roulette wheel probability of entering a game.

$$R_a = d_{A,C} \times A_{GS} \tag{3}$$

Each agent in the population makes a game offer to all other agents, and the set of agreed players then participate in the NPD game.

3.1.2 Genetic Algorithm

In our simulator we have implemented a simple genetic algorithm. In each generation individuals participate in varying numbers of games. Therefore, fitness is determined by summing all their payoffs received and getting an average payoff per game. In each generation, the top 10% of agents are copied directly into the following generation. The other 90% of the agent population in generation $G+1$ are generated through evolving new strategies based on agent fitness in G. Individuals are selected through roulette wheel selection based on their fitness from generation G. Parent pairs are selected and then these are used to generate a single new agent offspring for generation $G+1$. Crossover occurs through averaging the genes between the two parent strategy genomes $G_C, G_T, G_S,$. These averaged strategy genes are then used for the new agent. A 5% chance of mutation on each of these strategy genes is also used, and once this occurs a gaussian distribution is used to determine the degree of change.

4. EXPERIMENTAL RESULTS

In this section we will present a series of simulations showing the results of our experiments. Firstly, we will examine a set of graphs depicting the results from a single run over 1000 Generations. The aim of this single run is to show the inherent links between certain agent gene values and the overall cooperation throughout the agent population. Later in this section we will present simulations showing results from a number of experimental runs. These will demonstrate the overall stability of our results over multiple runs. All our simulations were conducted using an agent population of 100 agents.

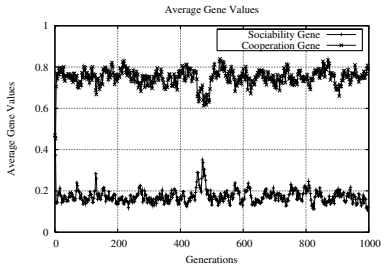


Figure 2: Average Gene Value (1 Run)

Figure 2 shows the rapid emergence of cooperation throughout the agent population. This graph depicts the average G_C and G_S genes throughout the agent population in each generation. The results show the emergence of cooperation as the average G_S gene falls throughout the population. These results show a rapid drop in the average G_S gene which reflects the tendency of the agent population to interact with fewer peers. The increased levels of cooperation throughout the population are closely linked with the tendency of individuals to act less sociably. It is clear from the results that the heightened cooperative gene is linked directly with the lower sociability gene.

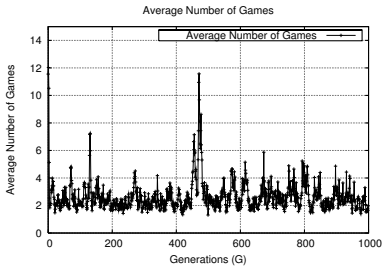


Figure 3: Average Number of Games (1 Run)

The results in Figure 3, depict the average number of games each agent participates in throughout successive generations. These results show the underlying dynamics that resulted in the heightened average cooperation shown in Figure 2. Once agents begin to participate in multiple n-player dilemmas they are exposed to exploitation and they are then heavily penalised. It is clear that cooperation is achieved through agents participating in as few games as possible. This serves to limit their exposure to potential exploitative peers.

The simulations shown are from a single run over 1000 generations. These simulations show the close relationship between the various agent gene values, and the collective behaviour of the agent population. For example around Generation 440 we can identify a period of increased sociability and a corresponding drop in cooperativeness throughout the population. This feature is clearly identifiable through examining the average gene values in Figure 2 and also the average game participation results in Figure 3.

These results are confirmed when examined over multiple experimental runs. The following graphs are averaged over 25 experimental runs. The purpose of these experiments is to confirm that the overall trends identified previously are repeated over many runs.

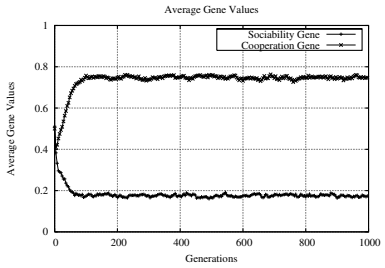


Figure 4: Average Gene Value (25 Runs)

The data shown in Figure 4 show the average strategy genes averaged over many experiments. The results show that the agent population consistently converges on cooperation throughout multiple experiments. We also notice the low G_S genes recorded throughout the simulations. Through limiting game participation to a tiny number of games, each agent minimises the opportunity of less cooperative individuals to exploit them. Once cooperative strategies benefit heavily by limiting their interactions they receive heightened payoffs and then this feature is propagated throughout new agents in the population.

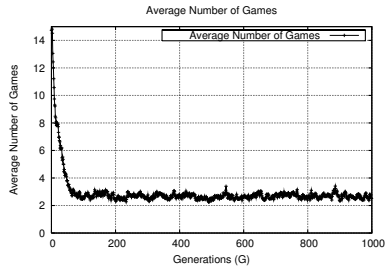


Figure 5: Average Number of Games (25 Runs)

The tendency to interact in a small number of games is confirmed in Figure 5, which depicts the average number of games each agent participates in. Our results indicate a clear benefit to individuals who are less sociable and thereby choose to be far more discerning regarding game participation. This facilitates the emergence of cooperation and helps to maintain cooperation it over successive generations.

5. CONCLUSIONS

This paper has examined the NPD game with respect to group participation. For the first time this game has been investigated using a tag-mediated interaction model. Our results demonstrate that despite there being a clear incentive to defect, cooperation can still emerge. This stems from the ability of individuals in our agent population to determine their degree of sociability towards their peers. This reinforces much of the existing literature involving the traditional Prisoner's Dilemma [7] and also the NIPD [14]. Our models reinforces these observations through an alternative approach. In our case we have not explicitly determined the sociability of our agent population. Instead we have allowed the agent population to evolve with respect to their cooperative and sociability genes. Our results have demonstrated the significance of sociability in games such as the NPD. Furthermore, we have also demonstrated the advantage to cooperative individuals who act less sociably towards their peers. Limiting game participation provides a very effective defence against exploiters. Earlier in our introduction we posed two specific research questions.

1. Our results show that tags can successfully bias interactions in the the NPD. We believe this is the first time a tag model has been applied to the NPD. Our results show the resulting levels of cooperation that emerged.

2. The significance of the sociability gene in our simulations is clear from the obvious link between cooperation and sociability in our simulations.

This paper has presented an evolutionary model capable of modeling sociability within the agent strategy genome. We have also shown how tags can be used to determine n-player games. Finally, our results have shown through an evolutionary model that there is a clear benefit to agent strategies who are cooperative in tandem with being less sociable through limiting their exposure to exploitation.

In summary this paper has shown that tags can be successfully adapted to bias agent interactions in a n-player game such as the NPD. Furthermore, we have demonstrated how an agent population can engender and maintain cooperation through an evolvable sociability trait. In future work we hope to examine how cooperation can be engendered without limiting game participation so dramatically.

6. ACKNOWLEDGMENTS

The authors would like to acknowledge the continued support of Science Foundation Ireland (SFI).

7. REFERENCES

[1] R. Chiong, S. Dhakal, and L. Jankovic. Effects of neighbourhood structure on evolution of cooperation in n-player iterated prisoner's dilemma. In *IDEAL*, pages 950–959, 2007.

[2] N. S. Glance and B. A. Huberman. The dynamics of social dilemmas. *Scientific American*, 270(3):76–81, 1994.

[3] D. Hales and B. Edmonds. Evolving social rationality for mas using "tags". In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 497–503, New York, NY, USA, 2003. ACM.

[4] D. Hales and B. Edmonds. Applying a socially inspired technique (tags) to improve cooperation in p2p networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):385–395, 2005.

[5] G. Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, December 1968.

[6] J. Holland. The effects of labels (tags) on social interactions. *Working Paper Santa Fe Institute 93-10-064*, 1993.

[7] E. Howley and C. O'Riordan. The emergence of cooperation among agents using simple fixed bias tagging. In *Proceedings of the 2005 Congress on Evolutionary Computation (IEEE CEC'05)*, volume 2, pages 1011–1016. IEEE Press, 2005.

[8] M. Matlock and S. Sen. Effective tag mechanisms for evolving coordination. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[9] A. McDonald and S. Sen. The success and failure of tag-mediated evolution of cooperation. In *LAMAS*, pages 155–164, 2005.

[10] M. Nowak and R. May. The spatial dilemmas of evolution. *Int Journal of Bifurcation and Chaos*, 3:35–78, 1993.

[11] M. Oliphant. Evolving cooperation in the non-iterated prisoner's dilemma: the importance of spatial organisation. In *Proceedings of Artificial Life IV*, 1994.

[12] C. O'Riordan and H. Sorensen. *Stable Cooperation in the N-Player PrisonerŠs Dilemma: The Importance of Community Structure*, volume 4865 of *Lecture Notes in Computer Science (lncs) 4865*, pages 157–168. Springer-Verlag Berlin, 2008.

[13] R. Riolo. The effects and evolution of tag-mediated selection of partners in populations playing the iterated prisoner's dilemma. In *ICGA*, pages 378–385, 1997.

[14] X. Yao and P. J. Darwen. An experimental study of n-person iterated prisoner's dilemma games. *Informatica*, 18:435–450, 1994.