Designing Agent Utilities for Coordinated, Scalable and Robust Multi-Agent Systems

Kagan Tumer

NASA Ames Research Center, Mailstop 269-4, Moffett Field, CA 94035
ktumer@mail.arc.nasa.gov

Summary. Coordinating the behavior of a large number of agents to achieve a system level goal poses unique design challenges. In particular, problems of scaling (number of agents in the thousands to tens of thousands), observability (agents have limited sensing capabilities), and robustness (the agents are unreliable) make it impossible to simply apply methods developed for small multi-agent systems composed of reliable agents. To address these problems, we present an approach based on deriving agent goals that are aligned with the overall system goal, and can be computed using information readily available to the agents. Then, each agent uses a simple reinforcement learning algorithm [26] to pursue its own goals. Because of the way in which those goals are derived, there is no need to use difficult to scale external mechanisms to force collaboration or coordination among the agents, or to ensure that agents actively attempt to appropriate the tasks of agents that suffered failures.

To present these results in a concrete setting, we focus on the problem of finding the subset of a set of imperfect devices that results in the best aggregate device [5]. This is a large distributed agent coordination problem where each agent (e.g., device) needs to determine whether to be part of the aggregate device. Our results show that the approach proposed in this work provides improvements of over an order of magnitude over both traditional search methods and traditional multi-agent methods. Furthermore, the results show that even in extreme cases of agent failures (i.e., half the agents failed midway through the simulation) the system's performance degrades gracefully and still outperforms a failure-free and centralized search algorithm. The results also show that the gains increase as the size of the system (e.g., number of agents) increases. This latter result is particularly encouraging and suggests that this method is ideally suited for domains where the number of agents is currently in the thousands and will reach tens or hundreds of thousands in the near future.

1 Introduction

Coordinating a large number of agents to achieve complex tasks collectively presents new challenges to the field of multi-agent systems. The research issues in this area present significant departures from those in traditional multi-agent systems coordination problems where a handful of agents interact with one another. When dealing with a handful of agents, it is reasonable to assume that in many cases agents react to one another, can model one another, and/or enter into contracts with one another [6, 8, 12, 21]. When dealing with thousands of agents on the other hand, such assumptions become more difficult to justify. At best each one can assume that the agents are aware of other agents as part of a background. In such cases, agents have to act within an environment that may be shaped by the actions of other agents, but cannot be interpreted as the the by-product of the actions of any single agent.

This distinction is crucial and makes the coordination problem fundamentally different than that traditionally encountered in many domains, and thus requires new approaches. In this work, we focus on an agent coordination method that aims to handle systems which have the following four characteristics:

1. The agents have limited sensing and decision making capabilities. Therefore, rather than rely on carefully designed agents, the interactions among the agents will be leveraged to achieve the complex task;
2. The agents will not be able to model the other agents in the system. Therefore, they will "react" to the signals they receive from their environment;
3. The agents will not necessarily perform reliably, and a non-negligible percentage of the agents will to fail during the life-cycle of the system. Therefore, the agents will not rely on other agents performing specific tasks at specific performance levels.
4. The number of agents will be in the thousands. Therefore, the agents will need to act with local information and without direct regard for the full system performance.

To study such multi-agent systems within a concrete domain, we focus on the problem of imperfect device subset selection. This problem consists of a set of imperfect devices, and the task is to find the subset of those devices that results in the best aggregate device [5]. It can be viewed as an abstraction of what will likely loom as a major challenge in achieving coordination in large scale multi-agent systems (e.g., systems of nano or micro-scale components) meeting the four criteria listed above. This is a hard optimization problem, and brute force approaches cannot be used for any but its smallest toy instances [5, 10].

We propose addressing this problem by associating each device with an adaptive Reinforcement-Learning (RL) agent [15, 17, 26, 33]) that decides whether or not its device will be a member of the subset. In this problem, there is a well-defined, system-level objective function that needs to be achieved. As such we focus on how the agents' actions further that system-level goal (i.e., global utility). Furthermore, because we intend to scale this system to a large number of agents, the agents need to take their actions without actively soliciting information from other agents in the system. The design problem we face then, is to determine how best to set the private utility functions of the agents in a way that will lead to good values of the global utility, without involving difficult to scale external mechanism that ensure cooperation among the agents. Note that though the agents have simple decisions to make, this is still fundamentally a multi-agent problem: Each agent autonomously makes a decision at each time step based on its estimate of the reward it will receive; and the system is fully distributed as each agent has full autonomy over its actions.

For the joint action of agents working in such a system to provide good values of the global utility, we must both ensure that the agents do not work at cross-purposes, and that each one has a learning problem that is relatively easy to solve. Typically these two requirements are in conflict with one another. For example, providing each agent with the system-level goal will ensure that they will not work at cross purposes. However, such a choice will leave the agents with a difficult problem: each of the agents' utilities will depend on the actions of all the other agents, making it all but impossible for the agents to determine the best actions to follow in most systems of interest. At the other extreme, providing each agent with a simple, local utility function will provide a clear signal, but may not necessarily lead the system to high values of global utility.

The challenge is is to find the best trade-off between these two requirements. This design problem is related to work in many other fields, including multi-agent systems (MAS's), computational economics, mechanism design, computational ecologies and game theory [4, 20, 13, 18, 25]. However, because of issues related to the scale of the system, the reliability of the agents and the limited availability of information, they do not provide a full solution to this problem. (See [30] for a detailed discussion of the relationship between these fields, involving hundreds of references.)

This chapter presents an agent utility based multi-agent coordination algorithm that is well-suited for large and noisy multi-agent systems where coordination among simple and coomperative agents is required. In Section 2 we summarize the background material for agent utility derivation and define the desirable properties an agent utility needs to possess for coordination in large multi-agent systems. In Section 3 we present the imperfect device combination problem and derive the specific agent utilities for this domain. In Section 4 we describe the simulations and present results showing the performance of the various utilities, their scaling properties and their robustness to agent failures. Finally, in Section 5 we provide a summary and discuss the implications and general applicability of this work.

2 Background

In this work, we focus on multi-agent systems that aim to maximize a global utility function, $G(z)$, which is a function of the joint move of all agents in the system, z. Instead of maximizing $G(z)$ directly, each agent, i, tries to maximize its private utility function $g_i(z)$. Our goal is to devise private utility functions that will cause the multi-agent system to produce high values of $G(z)$ [2, 28, 34]. Because this method is based on assigning a utility function to each agent, it is better suited for inherently cooperative distributed domains such as multi-rover coordination [1], or the imperfect device combination problem presented here. On the other hand, with some modifications, it is also applicable to more general domains such as data routing [32], job scheduling over heterogeneous servers [29] or multivariate search [35].

In this work, the notation z_i refers to the parts of z that are dependent on the actions of i, and z_{-i} to refer to the components of z that do not depend on the actions of agent i. Instead of concatenating these partial states to obtain the full state vector, we use zero-padding for the missing elements in the partial state vector. This allows us to use addition and subtraction operators when merging components of different states (e.g., $z = z_i + z_{-i}$).

2.1 Properties of Utility Functions

Now, let us formalize the two requirements discussed above that a private utility should satisfy. First, the private utilities have to be aligned with respect to G, quantifying the concept that an action taken by an agent that improves its private utility also improves the global utility. Formally, for systems with discrete states, the degree of factoredness for a given utility function g_i is defined as:

$$\mathcal{F}_{g_i} = \frac{\sum_z \sum_{z'} u[(g_i(z) - g_i(z'))\ (G(z) - G(z'))]}{\sum_z \sum_{z'} 1} \tag{1}$$

for all z' such that $z_{-i} = z'_{-i}$ and where $u[x]$ is the unit step function, equal to 1 if $x > 0$, and zero otherwise. Intuitively, the higher the degree of factoredness between two utilities, the more likely it is that a change of state will have the impact on the two utilities (e.g., make both of them go up). A system is fully factored when $\mathcal{F}_{g_i} = 1$. As a trivial example, a system in which all the private utility functions equal G [7] is fully factored.

Second, the private utilities have to have high **learnability**, intuitively meaning that an agent's utility should be sensitive to its own actions and insensitive to actions of others. Formally we can quantify the learnability of utility g_i, for agent i at z:

$$\lambda_{i,g_i}(z) = \frac{E_{z'_i}[|g_i(z) - g_i(z_{-i} + z'_i)|]}{E_{z'_{-i}}[|g_i(z) - g_i(z'_{-i} + z_i)|]} \tag{2}$$

where $E[\cdot]$ is the expectation operator, z'_i's are alternative actions of agent i at z, and z'_{-i}'s are alternative joint actions of all agents other than i. Intuitively, learnability provides the ratio of the expected value of g_i over variations in agent i's actions to the expected value of g_i over variations in the actions of agents other than i. So at a given state z, the higher the learnability, the more $g_i(z)$ depends on the move of agent i, i.e., the better the associated signal-to-noise ratio for i. Higher learnability means it is easier for i to achieve a large values of its utility. Note that, though a system where all agents' private utilities are set to G is fully factored, such a system will have low learnability since each agent's utility will depend on the actions of all the other agents in the system.

2.2 Private Utility Functions

Now, let us present two utilities that are fully factored and have high learnability. The **Estimated Difference Utility** is given by:

$$EDU_i \equiv G(z) - E_{z_i}[G(z)|z_{-i}] \tag{3}$$

where $E_{z_i}[G(z)|z_{-i}]$ gives the expected value of G over the possible actions of agent i. Such a private utility for the agents is fully factored with G because the second term does not depend on agent i's state [34] (these utilities are referred to as AU in [34]). Furthermore, because it removes noise from an agent's private utility, EDU yields far better learnability than does G [34]. This noise reduction is due to the subtraction which (to a first approximation) eliminates the impact of states that are not affected by the actions of agent i.

The second utility we consider is the **Wonderful Life Utility** [34], given by:

$$WLU_i \equiv G(z) - G(z_{-i}). \tag{4}$$

The major difference between EDU and WLU is in how they handle z_{-i}. EDU provides an estimate of agent i's impact by sampling all possible actions of agent i whereas WLU simply removes agent i from the system WLU is also factored with G, because the second term does not depend on the actions of agent i [34]. In general, WLU also has better learnability than G, and in the next section we discuss this in more detail for this problem domain.

3 Combination of Imperfect Devices

We now explore the use of these private utility functions for the problem of combining imperfect devices [5]. A typical example of this problem arises when many simple and noisy observational devices (e.g., nano or micro devices, low power sensing devices) attempt to accurately determine some value pertinent to the phenomenon they're observing. Each device will provide a single number that is slightly off, similar to sampling a Gaussian centered on the value of the real number. The problem is to choose the subset of a fixed collection of such devices so that the average (over the members of the subset) distortion is as close to zero as possible.

3.1 Problem Definition

Formally, the problem is to minimize

$$\varepsilon \equiv \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k}, \tag{5}$$

where $n_j \in \{0, 1\}$ is whether device j is or is not selected, and there are N devices in the collection, having associated distortions $\{a_j\}$. This is a hard optimization problem that is similar to known NP-complete problems such as subset sum or partitioning [5, 10], but has two twists: the presence of the denominator and that $a_j \in R\ \forall j$. In this work we set the system-level utility function to $G = -\varepsilon$ (we do this so that the goal is to "maximize" G, which is more consistent with the concept of "utility" design).

The system is composed of N agents, each responsible for setting one of the n_j. Each of those agent has its own private utility function, though the overall objective is to maximize system level performance. The aim is to give those agents private utilities so that, as they learn to maximize their private utilities, they also maximize G.

3.2 Expected Difference Utility

For this application, the EDU discussed in the previous section becomes:

$$\begin{aligned} EDU_i(z) = &- \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \\ &+ \left(p(n_i = 1) \frac{|\sum_{j \neq i}^{N} n_j a_j + a_i|}{\sum_{k \neq i}^{N} n_k + 1} + p(n_i = 0) \frac{|\sum_{j \neq i}^{N} n_j a_j|}{\sum_{k \neq i}^{N} n_k} \right) \end{aligned} \qquad (6)$$

where $p(n_i = 1)$ and $p(n_i = 0)$ give the probabilities that agent i set its n_i to 1 or 0 respectively. In what follows, we will assume that those two actions are equally likely (i.e., for all agents i, $p(n_i = 1) = p(n_i = 0) = 0.5$).

Depending on which action agent i chose (0 or 1), EDU can be reduced to:

$$EDU_i(z) = 0.5 \frac{|\sum_{j=1}^{N} n_j a_j - a_i|}{\sum_{k=1}^{N} n_k - 1} - 0.5 \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \quad \text{if } n_i = 1 \, , \qquad (7)$$

or:

$$EDU_i(z) = 0.5 \frac{|\sum_{j=1}^{N} n_j a_j + a_i|}{\sum_{k=1}^{N} n_k + 1} - 0.5 \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \quad \text{if } n_i = 0 \, . \qquad (8)$$

Note that in this formulation, EDU provides a very clear signal. If EDU is positive, the action taken by agent i was beneficial to G, and if EDU is negative, the action was detrimental to G. Thus an agent trying to maximize EDU will efficiently maximize G, without explicitly trying to do so. Furthermore, note that the computation of EDU requires very little information. Any system capable of broadcasting G can be minimally modified to accommodate EDU. For each agent to compute its EDU, the system needs to broadcast the two numbers needed to compute G: the number of devices that were turned on (i.e., the denominator in Equation 5) and the associated subset distortion as a real number (i.e., the numerator in Equation 5 before the absolute value operation is performed. Based on those two numbers, the agent can compute its EDU.

3.3 Wonderful Life Utility

For this application, the WLU discussed in the previous section becomes:

$$WLU_i(z) = - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} + \frac{|\sum_{j \neq i}^{N} n_j a_j|}{\sum_{k \neq i}^{N} n_k} \qquad (9)$$

Note however, that unlike with EDU, the action chosen by agent i has a large impact on the WLU. If agent i chooses action 0, the two terms in Equation 9 are identical, resulting in a WLU of zero. Depending on which action agent i chose (0 or 1), WLU can be reduced to:

$$WLU_i(z) = \frac{|\sum_{j=1}^{N} n_j a_j - a_i|}{\sum_{k=1}^{N} n_k - 1} - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \qquad \text{if } n_i = 1\,, \qquad (10)$$

or:

$$WLU_i(z) = 0 \qquad \text{if } n_i = 0\,. \qquad (11)$$

In this formulation, unlike EDU, WLU provides a clear signal only if agent i had chosen action 1. In that case, a positive WLU means that the action was beneficial to G, and a negative WLU means that the action was detrimental for G. However, if agent i had chosen action 0, it receives a reward of 0 regardless of whether that action was good or bad for G. This means that on average half the actions an agent takes will be random as far as G is concerned. Considering learnability implications, this means that on average WLU will have half the learnability of EDU for this problem.

4 Experimental Results

In this work we purposefully used computationally unsophisticated and easy to build agents for the following reasons:

1. To ensure that we remained consistent with our purpose of showing that a large scale system of potentially failure-prone agents can be coordinated to achieve a system level goal. Indeed, building thousands of sophisticated agents may be prohibitively difficult; therefore though systems that will scale up to thousands may use sophisticated agents, they cannot rely on such sophistication.
2. To focus on the design of the utility functions. Having sophisticated agents can obscure the differences in performance due to the agent utility functions and the algorithms they ran. By having each agent run a very simple algorithm we kept the emphasis on the effectiveness of the utility functions.

Each agent had a data set and a simple reinforcement learning algorithm. Each agents' data set contained time, action, utility value triplets that the agent stored throughout the simulation. At each time step each agent chose what action to take, which provided a joint action which in turn set the system state. Based on that state the system level utility, and the private utility of all the agents are computed. The new time, action take and utility value for agent i then gets added to the data set maintained by agent i. This is done for all agents and then the process repeats.

To choose its actions, an agent uses its data set to estimate the values of the utility it would receive for taking each of its two possible move. Each agent i picks its action at a time step based on the utility estimates at that time. Instead of simply picking the largest estimate, to promote exploration it probabilistically selects an action, with a higher likelihood of selecting the actions with higher utility estimates (e.g., it uses a Boltzmann distribution across the utility values). Because the experiments were run for short periods of time, the temperature in the Boltzmann distribution did not decay in time. However to reflect the fact that the environment in which an agent is operating changes with time (as the other agents change their moves), and therefore the optimal action changes in time, the two utility estimates are formed using exponentially aged data: for any time step t, the utility estimate i uses for setting either of the two actions n_i is a weighted average of all the utility values it has received at previous times t' that it chose that action, with the weights in the average given by an exponential of the values $t - t'$. Finally, to form the agents' initial data sets, there is an initialization period in which all actions by all agents are chosen uniformly randomly, with no learning used. It is after this initialization period ends that the agents choose their actions according to the associated Boltzmann distributions.

For all learning algorithms, the first 20 time steps constitute the data set initialization period (note that all learning algorithms must "perform" the same during that period, since none are actually in use then). Starting at $t = 20$, with each consecutive time step a fixed fraction of the agents switch to using their learner algorithms instead, while others continue to take random actions. Because the behavior of the agents starting to use their learning algorithm changes, having all agents start learning simultaneously provides a sudden "spike" into the system which significantly slows down the learning process. This gradual introduction of the learning algorithms is intended to soften the "discontinuity" in each agent's environment. In these experiments, for $N = 50$ and $N = 100$, three agents turned on their learning algorithms at each time step, and for $N = 1000$, sixty agents turned on their learning algorithms at each time step.

4.1 Agent Utility Performance

Figures 1-3 show the convergence properties of different agent utilities and a search algorithm in systems with 50, 100 and 1000 agents respectively. The results reported are based on 20 different $\{a_i\}$ configurations, where each $\{a_i\}$ is selected from a Gaussian distribution with zero mean and unit variance. For each configuration, the experiments were run 50 times (i.e., each point on the figures is the average of $20 \times 50 = 1000$ runs). The graphs labeled G, EDU and WLU show the performance of agents using reinforcement learners with those reinforcement signals provided by G (team game), EDU and WLU respectively. S shows the performance of local search where new n_i's are generated at each step by perturbing the current state and selected if the solution is better than the current best solution (in the experiments reported here, 25% of the actions were randomly changed at each time step, though somewhat surprisingly, the results are not particularly sensitive to this parameter). Because the runs are only 200 time steps long, algorithms such as simulated annealing do not outperform local search: there is simply no time for an annealing schedule. This local search algorithm provides the performance of an algorithm with centralized control.

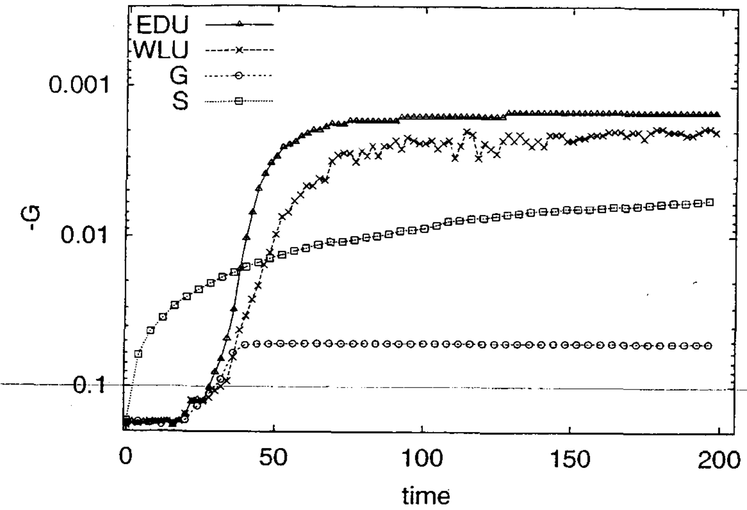

Fig. 1. Combination of Imperfect Devices Problem, N=50.

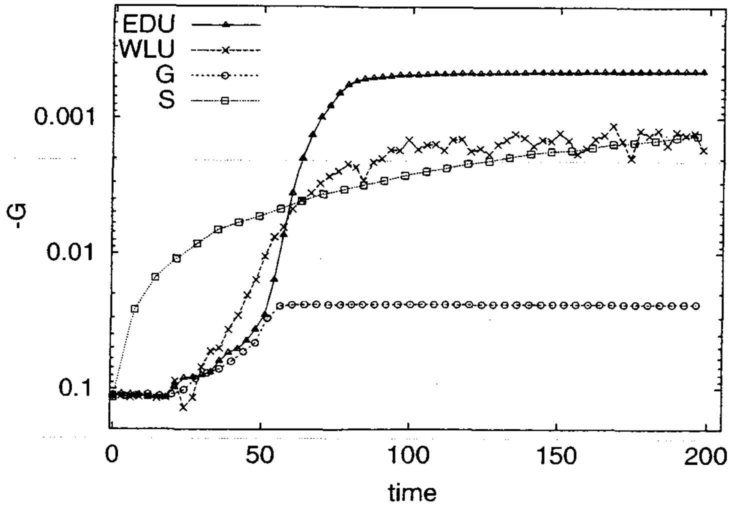

Fig. 2. Combination of Imperfect Devices Problem, N=100.

In all cases in which agents use the *G* utility, they have a difficult time learning. Even for 50 agents, the noise in the system is too large for such agents to learn how to select their actions. For 50 agents (Figure 1) both WLU and EDU outperform the centralized search algorithm. In this case, both utility functions sufficiently "clean-up" the signal for the agents to perform well. For 100 agents (Figure 2), WLU starts to suffer. Because agents only receive useful feedback when they take one of the two

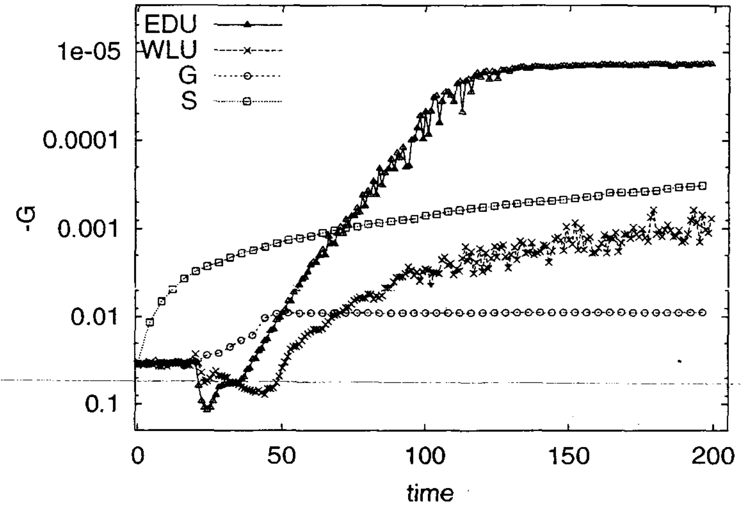

Fig. 3. Combination of Imperfect Devices Problem, N=1000.

actions, the noise in the system is increasing. This "noise" becomes too much for systems with 1000 agents (Figure 3), where WLU is outperformed by the centralized algorithm. EDU, on the other hand, continues to provide a clean signal for all systems up to the largest we tested (1000 agents).

Note that because agents turning on their learning algorithm changes the environment, the performance of the system as whole degrades immediately after learning starts (i.e., after 20 steps) in some cases. Once agents adjust to the new environment, the system settles down and starts to converge.

4.2 Scaling Characteristics of Utilities

Figure 4 shows scaling results (the $t = 200$ average performance over 1000 runs) along with the associated error bars (differences in the mean). As N grows two competing factors come into play. On the one hand, there are more degrees of freedom to use to minimize G. On the other hand, the problem becomes more difficult: the search space gets larger for S, and there is more noise in the system for the learning algorithms. To account for these effects and calibrate the performance values as N varies, we also provide the baseline performance of the "algorithm" that randomly selects its action ("Ran"). Note that the difference between the performances of all algorithms and EDU increases when the system size increases, reaching a factor of twenty for S and over 600 for G for $N = 1000$.

Also note that all algorithms but EDU have slopes similar to that of "Ran", showing that they cannot use the additional degrees of freedom provided by the larger N. Only EDU effectively uses the new degrees of freedom, providing gains that are proportionally higher than the other algorithms (i.e., the rate at which EDU's per-

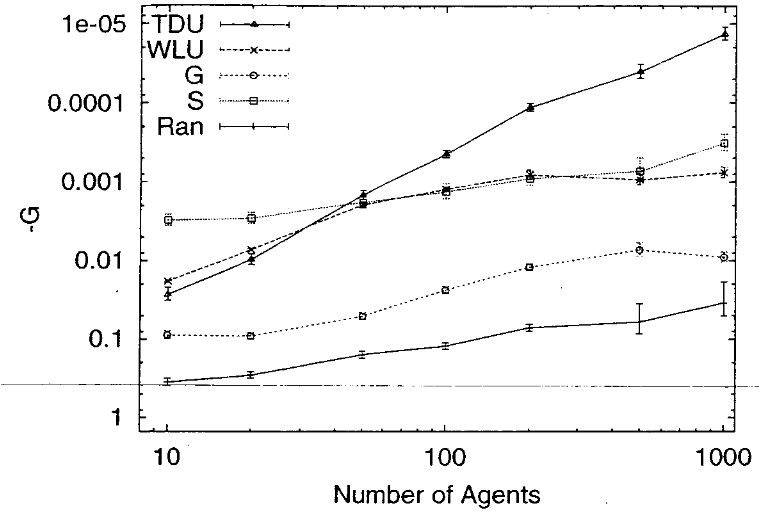

Fig. 4. Scaling in the Combination of Imperfect Devices Problem.

formance improves outpaces what is "expected" based on the random algorithm's performance).

4.3 Robustness

In order to evaluate the robustness of the proposed utility functions for multiagent coordination, we tested the performance of the system when a subset of the agents failed during the simulation. At a given time ($t = 100$ in these experiments), a certain percentage of agents failed (e.g., were turned off) simulating hazardous condition in which the functioning of the agents cannot be ascertained. The relevance of this experiment is in determining whether the proposed utility functions require all or a large portion of the agents to perform well to be effective, or whether they can handle sudden changes to their environment.

Figure 5 shows the performance of EDU, WLU, and G for 50 agents when 10% of the agents fail at time step $t = 100$. Similarly Figure 6 shows the performance of 100 agents where 20% of them fail. The results of the centralized search algorithm with no failures ("S" from Section 4.1), is also included for comparison.

In these experiments, none of the agent learning algorithms were adjusted to account for the change in the environment. In agents that continued to function, the learning proceeded as though nothing had happened. As a consequence, not only did the agents need to overcome the sudden change in their task but they had to do so with parameters tuned to the previous environment. Despite these limitations, EDU and WLU recover rapidly for the 50 agent case, whereas G does not. For the case with 100 agents and 20% agent failure, only EDU outperforms the centralized search algorithm. Note this is a powerful results: a distributed algorithm with only

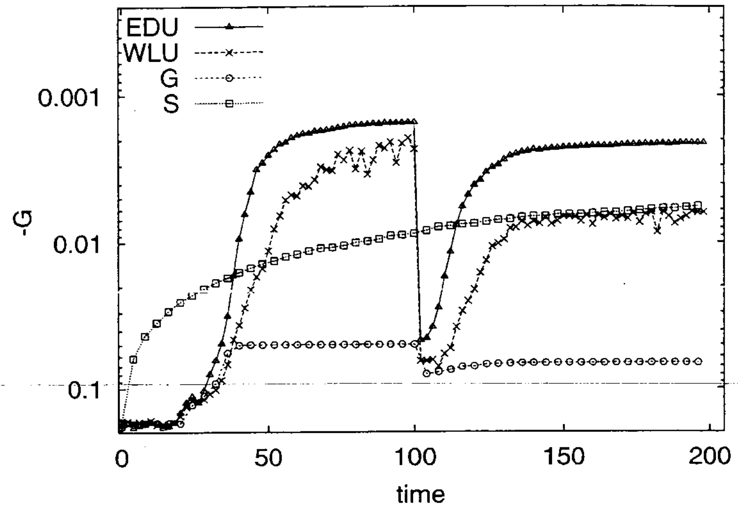

Fig. 5. System performance for 50 agents, 10% of which fail at time t=100.

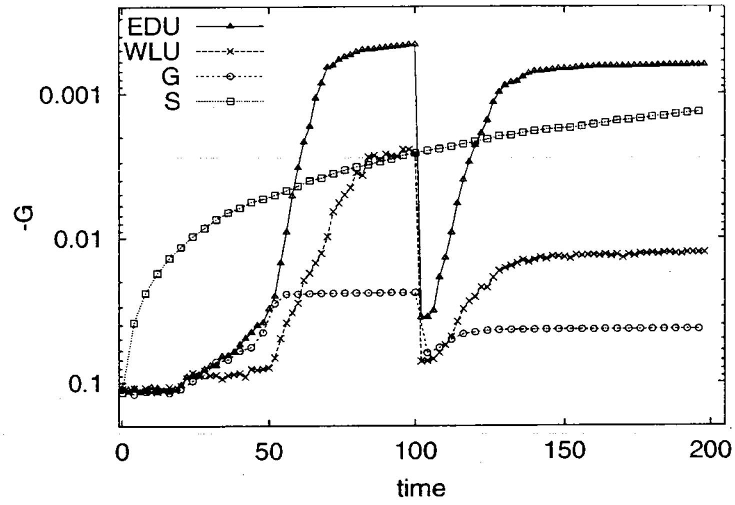

Fig. 6. System performance for 100 agents, 20% of which fail at time t=100.

80% functioning agents, each tuned to a different environment outperforms a 100% functioning centralized algorithm.

Figures 7 and 8 show the performance of EDU when the percentage of agent failures increases from 10 to 50% for 50 and 100 agents respectively. For comparison purposes, the search results (From Section 4.1) are also included. After the initial drop in performance when the agents stop responding, EDU trained algorithms re-

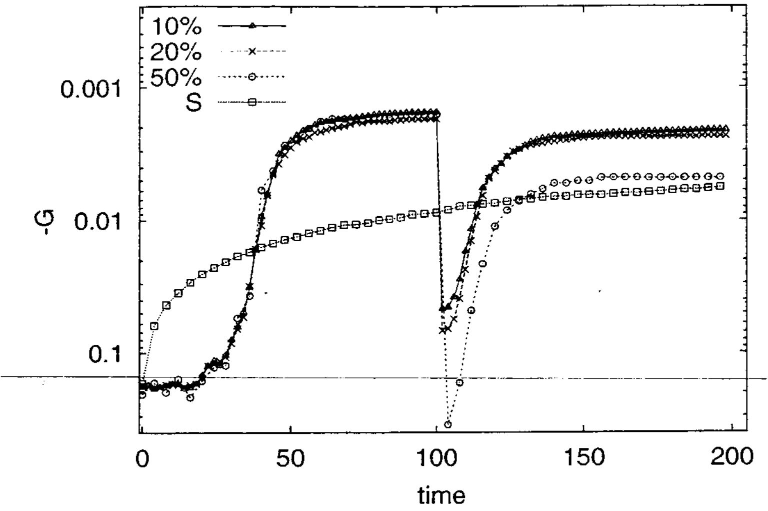

Fig. 7. Effect of agent failures on EDU for 50 agents (S has no agent failures).

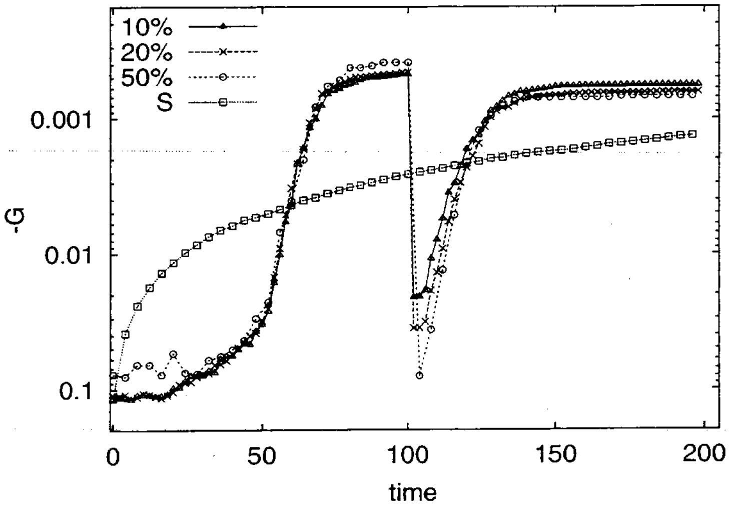

Fig. 8. Effect of agent failures on EDU for 100 agents (S has no agent failures).

cover rapidly and even with half the agents outperform the fully functioning and centralized search algorithm. These results demonstrate both the adaptability of the EDU and its robustness to failures of individual agents, even in extreme cases.

5 Discussion

The combination of imperfect devices is a simple abstraction of a problem that will loom large in the near future: How to coordinate a very large numbers of agents – many of which may have limited access to information and perform unreliably – to achieve a prespecified system-level objective. This problem is fundamentally different from traditional multi-agent problems in at least four ways: (i) the agents have limited sensing and decision making capabilities; (ii) the agent do not model the actions of other agents; (iii) the agents are unreliable and failure-prone; and (iv) the number of agents is in the thousands.

The work summarized in this chapter is based on ensuring coordination while eliminating external mechanisms such as contracts and incentives to allow the systems to scale to large system. In the experimental domain of selecting a subset of imperfect devices, the results shows the promise of this method by providing improvements of up to twenty times better than a centralized algorithm and of nearly three orders of magnitude over a multi-agent system using a team game approach. Furthermore, when as many as half the agents failed during simulations, the proposed method still outperformed a fully functioning centralized search algorithm.

This approach is well-suited for addressing coordination in large scale cooperative multi-agent systems where the agents do not have pre-set and possibly conflicting goals, or when the agents do not need to hide their objectives. The focus is on ensuring that the agents do not inadvertently frustrating one another in achieving their goals. The results show that in such large scale, failure-prone systems, this method performs well precisely because it does not rely on the agents building an accurate model of their surroundings, modeling the actions of other agents or requiring all agents in the system to reach a minimum performance level.

Acknowledgements: The author would like to thank David Wolpert for invaluable discussions and for bringing the faulty devices problem to his attention, Adrian Agogino for his many comments, as well as the participants in the *Coordination of Large Scale Multi-Agent Systems* workshop at AAMAS 2004 for their helpful suggestions.

References

1. A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, Seatle, WA, June 2004.
2. A. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, July 2004.
3. S. Arai, K. Sycara, , and T. Payne. Multi-agent reinforcement learning for planning and scheduling multiple goals. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 359–360, July 2000.
4. C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, Holland, 1996.

5. D. Challet and N. F. Johnson. Optimal combinations of imperfect objects. *Physical Review Letters*, 89:028701, 2002.
6. B. Clement and E. Durfee. Theory for coordinating concurrent hierarchical planning agents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 495–502, 1999.
7. R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.
8. K. Decker and V. Lesser. Designing a family of coordination mechanisms. In *Proceedings of the International Conference on Multi-Agent Systems*, pages 73–80, June 1995.
9. J. Fredslund and M. J. Mataric. Robots in formation using local information. In *Proceedings, 7th International Conference on Intelligent Autonomous Systems (IAS-7)*, pages 100–107, Marina del Rey, CA, March 2002.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Fransisco, 1979.
11. T. Hogg and B. A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7:R1–R14, 1998.
12. J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.
13. B. A. Huberman and T. Hogg. The behavior of computational ecologies. In *The Ecology of Computation*, pages 77–115. North-Holland, 1988.
14. N. F. Johnson, S. Jarvis, R. Jonson, P. Cheung, Y. R. Kwong, and P. M. Hui. Volatility and agent adaptability in a self-organizing market. preprint cond-mat/9802177, February 1998.
15. M. Kearns and D. Koller. Efficient reinforcement learning in factored MDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 740–747, 1999.
16. S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, pages 79–97, 1997.
17. M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.
18. D. C. Parkes. *Iterative Combinatorial Auctions: Theory and Practice*. PhD thesis, University of Pennsylvania, 2001.
19. D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.
20. T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.
21. T. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94:99–137, 1997.
22. P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara. Scaling teamwork to very large teams. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, July 2004.
23. Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, 1994.
24. P. Stone. *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer*. MIT Press, Cambridge, MA, 2000.

25. P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3), 2000.
26. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.
27. M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.
28. K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 378–385, Bologna, Italy, July 2002.
29. K. Tumer and J. Lawson. Collectives for multiple resource job scheduling across heterogeneous servers (poster). In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, July 2003.
30. K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems.* Springer, New York, 2004.
31. K. Tumer and D. Wolpert. A survey of collectives. In *Collectives and the Design of Complex Systems*, pages 1,42. Springer, 2004.
32. K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.
33. C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.
34. D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.
35. D. H. Wolpert, K. Tumer, and E. Bandari. Improving search algorithms by using intelligent coordinates. *Physical Review E*, 69:017701, 2004.
36. P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 616–623, Montreal, January 2001. ACM Press.
37. Y. C. Zhang. Modeling market mechanism with evolutionary games. *Europhysics Letters*, March/April 1998.