Auton Agent Multi-Agent Sys (2006) 13:97–115
DOI 10.1007/s10458-006-6105-y

Handling communication restrictions and team formation in congestion games

Adrian K. Agogino · Kagan Tumer

Published online: 25 February 2006
Springer Science + Business Media, Inc. 2006

Abstract There are many domains in which a multi-agent system needs to maximize a "system utility" function which rates the performance of the entire system, while subject to communication restrictions among the agents. Such communication restrictions make it difficult for agents that take actions to optimize their own "private" utilities to also help optimize the system utility. In this article we show how previously introduced utilities that promote coordination among agents can be modified to be effective in domains with communication restrictions. The modified utilities provide performance improvements of up to 75% over previously used utilities in congestion games (i.e., games where the system utility depends solely on the number of agents choosing a particular action). In addition, we show that in the presence of severe communication restrictions, team formation for the purpose of information sharing among agents leads to an additional 25% improvement in system utility. Finally, we show that agents' private utilities and team sizes can be manipulated to form the best compromise between how "aligned" an agent's utility is with the system utility and how easily an agent can learn that utility.

Keywords Reinforcement learning · MAS · Teams · Communication

1. Introduction

Many methods exist for coordinating the actions of autonomous agents in a large multi-agent system when those agents can fully communicate with one another [7, 11, 21, 27, 33, 37]. One particular solution to that problem is given within the framework of "collectives" defined as large multi-agent systems where there is a well-defined "system utility" function which rates the possible dynamic histories of the collection, and where each agent is only

A.K. Agogino, (✉)
University of California, Santa Cruz, USA
e-mail: adrian@email.arc.nasa.gov

K. Tumer
NASA Ames Research Center, USA
e-mail: ktumer@mail.arc.nasa.gov

concerned with maximizing its own "private utility" function [37]. However, many problems impose communication restrictions among the agents, rendering the coordination problem more difficult [6]. Examples of these problems, include controlling collections of rovers, constellations of satellites and packet routers, where an agent may only be able to directly communicate with a small number of other agents. In addition, even if there are other indirect ways to share information, they may be costly and an agent may be unwilling to share, if doing so would hurt its private utility. In all of these problems, the system designer faces the following difficult task:

– ensuring that, as far as the provided system utility function is concerned, the agents do not work at cross-purposes (i.e., making sure that the private utilities of the agents and the system utility are "aligned").
– ensuring that, agents are capable of achieving high values of their private utilities, (i.e. making sure an agent's actions have enough impact on its own private utility so that the utility is "learnable" and that it can effectively maximize it).
– ensuring that agents can compute their private utilities when they do not have access to a broad communication network providing them with access to global information.

These issues are at odds with each other and in fact in many cases it will be impossible for agents to achieve high values of a private utility which is "aligned" with the system utility.[1] In addition even if the system utility, computed with global information, can be broadcast to all the agents, agents may not be able to effectively use this information to select actions that will be useful to them and to the overall system. In fact many methods of incorporating local information into the system utility can lead to reduced performance as communication increases (Fig. 1). This example shows the performance of a system (described in detail in Section 3) with respect to the amount of communication available to the agents. Note that increasing the amount of information to which the agents have access can have deleterious effects on the performance of the system. We will discuss the reasons for this apparent paradox and show how problems associated with communication restrictions can be overcome by modifying the agents' utility functions and/or forming teams of agents that pool information.

Furthermore, issues related to communication restrictions can also be addressed by agents aggregating into teams sharing a utility function. Many types of team formation have been shown to be effective in different domains [24, 26]. In our domain utility sharing encourages team members to pool their information together, effectively reducing the impact of the communication restrictions. As the size of a team grows, the amount of information to which an agent has access also grows. However, even if large teams have access to more information, the agents now face the problem of determining the contribution of their actions to the utility.

We will explore these issues of team formation and communication restrictions through the collectives framework. This framework focuses on how to create agent-specific private utilities that are easy for the agents to learn, yet remain aligned with the overall system utility. The collectives framework has been successfully applied to multiple domains including packet routing over a data network [38], congestion games [39], and the coordination of multi-rovers in learning sequences of actions [1, 33]. However, unlike what will be presented in this paper, in all of these other works, agents were not hampered with communication restrictions.

In this paper we show how moderate communication restrictions can be overcome by modifying the agents' utilities. Then we show that team formation can be used when there

[1] By "aligned" we mean that actions that improve the agents private utility will also improve the system utility. We will formalize this concept in Section 2.

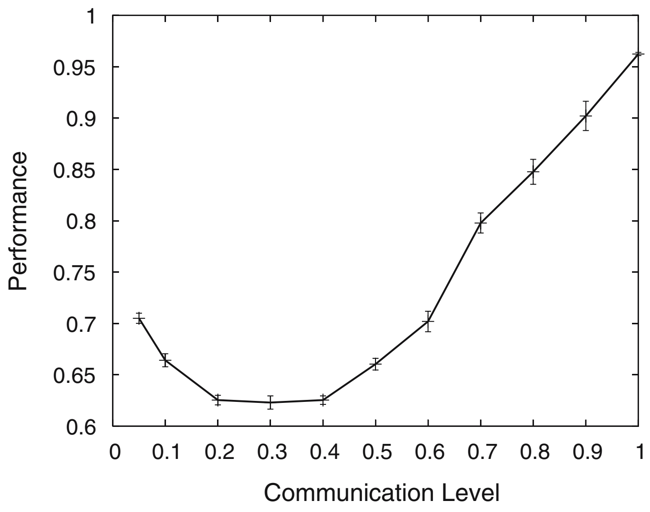

Fig. 1 Performance vs. communication level when system utility is combined with partial information. When the system utility is known, adding additional information about the system can actually hurt performance. Using standard methods more than 60% of system information has to be revealed before improvements can be made on the system utility. With better designed utilities presented later in this paper, even small amounts partial information can be used to increase performance

are severe communication restrictions, and we explore the tradeoff between team size and communication restrictions. In Section 2, we provide background on collectives and present four private utility functions for agents facing communication restrictions. In Section 3, we describe the problem domain and present the collective-based solution to this problem. In Section 4, we present the simulation results. In Section 5 we summarize related work in agent communications, team formation and coordination.

2. Design of agent utilities

One can look at utility design for large systems in a similar way one would look at creating employee incentives in a human company. In a company, the board's objective is to maximize a system utility which represents the "bottom line" of the company. The problem faced by the board is to create incentives for each of the employees such that when the employees maximize their incentives, the company's system utility is maximized. For example, the board might choose to give employees a compensation package that contains incentives tied to the company's stock price (e.g., stock options). The net effect of this action is to align the utility of the employee with the utility of the company, which ensures that what is good for the employee is also good for the company. In addition to being aligned with the system utility of the company it would be beneficial if the relation between an employee's actions and the value of the employees were readily discernable by the employee. If this relationship is easy to learn by the employees, then they would be able to learn to take the correct actions to maximize their compensation.

When the incentive package of an employee is both aligned with the company's utility and has high "learnability", then employees will both have the incentive to help the company and be able to determine how best to do so. Note that in practice, providing stock options is more effective in small companies than in large ones. This makes perfect sense from this perspective since such an incentive package has higher learnability (e.g., is easier for an employee to learn) in small companies than in large companies.

Designing proper private utilities for agents in a multi-agent system parallels the goal of the board in setting the incentive of all the employees: ensure that each agent (employee) has the correct incentive to take actions that will benefit the multi-agent system (company). In this section, we first summarize the formalization of the concepts of factoredness and learnability which are essential in deriving good private utilities for the agents [37]. We then present a class of private utilities satisfying those two properties and discuss four variants (based on different trade-offs of learnability vs. degree of factoredness) applicable to domains with communication restrictions (Section 2.3). Finally, we present how forming teams of agents that pool their information can further improve system performance (Section 2.4).

2.1. Background

Let z characterize the joint move of all agents in the system. In this formulation, z specifies the full system state. The system utility, $G(z)$, is a function of the full system state. The multi-agent system design problem is to find the z (e.g., joint action) that maximizes $G(z)$.

In addition to G, for each agent i, there is a *private utility function* g_i. The agents act to improve their individual private functions, even though, we, as system designers are only concerned with the value of the system utility G. To specify all agents other than i, we use the notation $-i$. Also to specify the part of the system state controlled by agent i and agents $-i$, we use the notation z_i and z_{-i}, respectively. Note that throughout this paper we "zero pad" all of our vectors so that z, z_i and $z{-i}$ have the same length and $z = z_i + z_{-i}$.

There are two properties that are crucial to producing systems in which agents acting to optimize their own private utilities will also optimize the provided system utility. The first of these deals with the concept of "aligning" the private utilities of the agents with the system utility. Formally this first property, the degree of *factoredness* between agent i, utility g_i and the system utility, G, is given by

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

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 same impact on the two utilities (e.g., make both of them go up). For example, when a company offers stock options, the factoredness of an employee's incentives increases because what helps the company, helps the employee. A system is fully factored when $F_{g_i} = 1$. In paradigms where we only care about whether the system is fully factored or not, the property, $F_{g_i} = 1$, can be defined more simply as:

$$g_i(z) \geq g_i(z') \Leftrightarrow G(z) \geq G(z') \quad \forall z, z' \ s.t. \ z_i = z'_i.$$

In a fully factored system for all pairs of states z and z' that differ only for agent i, a change in a i's state that increases its private utility cannot decrease the system utility.

Any system in which all the private utility functions equal G is fully factored [11]. However, such systems often suffer from low signal-to-noise, a problem that get progressively worse as the size of the system grows. This is because for large systems where G sensitively depends on all components of the system, each agent may experience difficulty discerning the effects of its actions on G. As a consequence, each i may have difficulty achieving high g_i.

This signal-to-noise effect, called *learnability* is the second property that is crucial in the designing of the agents' private utility functions. Formally we can quantify the learnability of utility g_i, for agent i at z:

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

where $E[\cdot]$ is the expectation operator, z'_is are alternative actions of agent i at z, and z'_{-i}s are alternative joint actions of all 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 agent i. Intuitively then, higher learnability means it is easier for agent i to achieve a large values of its utility. Note that learnability here is a measure of the relative influence of agents. It is a function of the utility and is independent of the learning algorithm the agents may use.

2.2. Difference utilities

For agents' private utilities consider the *difference* evaluation functions, which are of the form:

$$D_i \equiv G(z) - G(z_{-i} + c_i), \tag{3}$$

where z_{-i} contains all the variable not affected by agent i. All the components of z that are affected by agent i are replaced with the fixed constant c_i. Such difference utilities are fully factored no matter what the choice of c_i, because the second term does not depend on agent i's actions [34, 37]. Furthermore, they usually have better learnability than does a team game, because of the second term of D, which removes a lot of the effect of other agents (i.e., noise) from agent i's evaluation function. In many situations it is possible to use a c_i that is equivalent to taking agent i out of the system. Intuitively this causes the second term of the difference utility function to evaluate the fitness of the system without i and therefore D measures the agent's contribution to the system utility. Note that the effectiveness of the difference utility and of different values of c_i depends on the problem domain.

Figure 2 illustrates the computation of the difference utility in a simple system. As in that example, in many circumstances there is a particular choice for c that is a "null" move for that agent, equivalent to removing that agent from the system. For such a c, DU is closely related to the economics technique of "endogenizing a player's (agent's) externalities" [23].[2]

2.3. Communication restrictions

In general to compute a difference utility there may need to be enough communication to infer the entire system state. For some specific classes of utility such as the DU, this communication demand may be relaxed, since many of the elements of the system state cancel out and may be ignored. However, in many real system problems there is not enough communication between agents to compute even the less demanding utilities. In these cases we must approximate the utility under the constraints of the communication restrictions.

Mathematically, we represent an agent's knowledge of the system state as the union of the states directly observed by the agent and the states that indirectly "observed" through

[2] Indeed, DU has conceptual similarities to Vickrey tolls [35] in economics. However, DU can be applied to arbitrary, time-extended utility functions, and need not be restricted to the "null" clamping operator interpretable in terms of "externality payments". It also appears similar to Groves' mechanism [18] in mechanism design. However, the effect of Groves' mechanism is to create a system utility by subtracting out the benefit an agent already received directly, not computing the counterfactual impact of an agent on the system utility.

$$\begin{array}{cccc} & z & & (z_{-i_2},\vec{0}) \\ \begin{array}{c} i_1\\ i_2\\ i_3\\ i_4\end{array} & \begin{bmatrix}1&0&0\\0&0&1\\1&0&0\\0&1&0\end{bmatrix} & \underset{\substack{i_2 \text{ takes}\\ \text{“null” action}}}{\Longrightarrow} & \begin{bmatrix}1&0&0\\0&0&0\\1&0&0\\0&1&0\end{bmatrix}\end{array}$$

Fig. 2 This example illustrates the computation of the difference utility for agent 2, in a four-agent system. Each agent has three possible actions, and each such action is represented by a three-dimensional unary vector. The first matrix represents the joint state, z, of the system, where agent 1 has selected action 1, agent 2 has selected action 3, agent 3 has selected action 1 and agent 4 has selected action 2. The second matrix displays the virtual state where agent 2s action is the “null” vector (i.e., replacing z_{i_2} with $\vec{0}$). The difference utility of agent 2 is the difference between the system utility of the first state (z) and the system utility of the second state $(z_{-i_2}, \vec{0})$

communicating with other agents. Because this union represents the states whose values are known to that agent, we will refer to this union as “observable” states for agent i, eliminating the distinction between first-hand and second-hand state knowledge. We can decompose the system state z into a component observable by agent i, z^{o_i}, and a component hidden from agent i, z^{h_i} (we will denote the concatenated state z by $z = z^{o_i} + z^{h_i}$). In this paper we will define the communication level for agent i as:

$$B_i = \frac{\int_z I_{z_j}\, dj}{\int_z dj}, \tag{4}$$

where I_{z_j} is an indicator function return 1 when z_j is observable. For a problem with countable state elements, B_i reduces to the number of observable elements in the state divided by the total number of elements in the state. Note that B is always in the range [0.0, 1.0].

If the DU for agent i depends on any component of z^{h_i} then agent i cannot compute it directly. Instead we introduce different approximations to the DU that vary in their balance between learnability and factoredness. In the four utilities discussed below, the first two letters of the utility represent how the two terms of the difference utility get their information. “B” stands for “broadcast” meaning that the system utility is broadcast to the system, “T” stands for “truncated” meaning that the hidden values are ignored, and “E” stands for “estimated” meaning that the hidden variable is estimated from the observed variables.

2.3.1. Broadcast/Truncated Utility (BTU)

The first private utility we present for systems with communication restrictions is a variant of DU, where the subtraction in the second term not only removes agent is contribution, but also the contribution of all agents that i cannot observe z^{h_i}:

$$\mathrm{BTU}_i(z) = G(z) - G(z - z^{h_i} - z_i). \tag{5}$$

Note that BTU (as well as BEU discussed below) assume that the true system utility can be broadcast despite the communication restriction (agents have access to $G(z)$, but not to z). In many applications, this is a reasonable assumption since the system utility can often be computed once and broadcast throughout the environment [16]. More complex forms of broadcasting are often used for distributed multi-agent systems [9], but in this paper we will assume a very simple global broadcast of a single number. In many domains it is also reasonable to assume system utility can even be obtained directly from the environment without broadcasting [28].

Note that despite this “broader” subtraction, BTU is still fully factored. This is because BTU is in the form of a difference utility (Equation **??**) which only requires that constant c_i cannot depend on the state of agent i. Here c_i is $-z^{h_i}$, which is independent of the state of agent i since it can always observe itself. Since an agent cannot influence the second term of the BTU, the only way it can affect the value of BTU is through the first term, which is the system utility. However, while being fully factored, BTU may have much more noise than a pure DU since much more is subtracted out in the second term. Intuitively, only part of the noise, the part that was observable, has been removed from is utility.

As an example, consider a situation where agent i is an employee in a large company. Proper DU would remove the impact of the other employees from employee i’s private utility, since their general effect would be present both in the first term G and the second term $G(z - z^{h_i} - z_i)$. But if employee i can only communicate with a fraction of the employees, all the employees with whom it cannot communicate will also be clamped in the second term. Then the subtraction will not remove their effect from employee is utility. The influence those employees have on i’s utility will be noise, and employee i will have a harder time seeing the effect of its actions on its utility.

2.3.2. Truncated/Truncated Utility (TTU)

The second private utility is conceptually similar to BTU except that both terms are computed under the communication restrictions:

$$\text{TTU}_i(z) = G(z - z^{h_i}) - G(z - z^{h_i} - z_i). \tag{6}$$

This utility is no longer fully factored with respect to the system utility, because the first term in the difference utility is $G(z - z^{h_i})$ instead of $G(z)$. While not being fully factored with system utility, TTU can have better learnability than BTU. This is because both terms are computed using the same truncated state, and thus the systematic error may be removed in the subtraction for certain types of system utility functions [36].

Continuing with the company example, in this case the contribution of employees that are hidden from i will not appear in either term of TTU, since both terms are computed with the communication restriction. Therefore, this utility will have good learnability, since the noise from the hidden employees will not clutter employee is utility. As long as $G(z - z^{h_i})$ is sufficiently close to $G(z)$, this utility will have a high degree of factoredness and gains due to reduced noise will outweigh the loss in factoredness. However, if the assumption that $G(z - z^{h_i})$ is close to $G(z)$ does not hold (e.g., some hidden employees are crucial to the company’s profit) then TTU will not produce good system performance. For example agents using TTU are likely to fail in a congestion game, since the truncation will make the system seem less congested, inducing agents to make different choices than they would in a congested system.

2.3.3. Broadcast/Estimated Utility (BEU)

This utility is similar to BTU, except that instead of subtracting out all the components of z^{h_i}, their values are estimated given the values of z^{o_i}:

$$\text{BEU}_i(z) = G(z) - G(z^{o_i} + E[z^{h_i}|z^{o_i}] - z_i), \tag{7}$$

where $E[\cdot]$ is the expectation operator.[3] As long as this estimate is not influenced by the actions of i beyond z_i, this utility is still fully factored, since the first term of the difference

[3] While the expectation operator is used in this paper, any function of the observable components could be used instead.

equation is still $G(i)$. While both BTU and BEU are fully factored, BEU may have less noise, depending on how good the estimate of z^{h_i} is.

As in the previous example, suppose that there are a large number of employees that are hidden from employee i, but that employee i can approximate their contribution to the company based on the employees that it can observe. In this case the first term of BEU will contain all of the employees' contribution to $G(z)$, but the second term will subtract out the hidden employees' inferred contribution. Even if effects of the hidden elements cannot be perfectly estimated, a lot of noise can still potentially be eliminated from the system. Note however that if the estimate is particularly poor, noise can also be introduced into the system.

2.3.4. Estimated/Estimated Utility (EEU)

This utility is similar to TTU, except that in both terms, the value of z^{h_i} is estimated:

$$\mathrm{EEU}_i(z) = G(z^{o_i} + E[z^{h_i}|z^{o_i}]) - G(z^{o_i} + E[z^{h_i}|z^{o_i}] - z_i). \tag{8}$$

As was the case with TTU this utility is not fully factored with respect to the system utility G. However, with a good estimate of z^{h_i}, the value $G(z^{o_i} + E[z^{h_i}|z^{o_i}])$ will be much closer to $G(z)$ than $G(z - z^{h_i})$, so this utility can be much higher degree of factoredness with respect to $G(z)$ than can TTU.

Following the example, EEU provides an advantage over TTU in that even if there are hidden employees whose actions strongly impact the company's profits, if the actions of those employees can be predicted, then EEU will be close to being fully factored. This utility retains the benefits of TTU (both terms computed the same way, leading to good learnability) while in general having higher factoredness than TTU. Note that unlike with BEU, if the estimate of the hidden components is not particularly good, in general, noise will not be added to the system because both terms of the utility use the same estimate. Instead, the quality of the estimate only affects how close this utility is to being fully factored with respect to $G(z)$. If there are enough observable elements to make a good estimate we expect agents using EEU to do well. However, if there are so few observable elements that a reasonable estimate is impossible, then agents maximizing EEU may be lead to lower performance than having agents simply maximize G directly.

2.4. Team formation

As discussed above, communication restrictions can have serious negative effects on the utility functions of the agents. One way to remedy this situation is to let agents form "teams" which "share" their knowledge of the system state. More precisely the observable states for any member of a team will be the union of all the observable states of the individual team members. We use the notation z^{o_T} for the observable system state for team, T. Note that team information sharing can be viewed as another form of communication and we can define the effective communication level of an agent in a team, T, as:

$$B_{eff_T} = \frac{\int_z I_{z_j,T}\, dj}{\int_z dj}, \tag{9}$$

where $I_{z_i,T}$ is an indicator function return 1 when z_i is observable from team T. However, we will always refer to it as information sharing to differentiate from communication that happens between agents independent of the formation of teams. In real systems, team information sharing may have very different properties from general communication. It may have different constraints, different costs and may be imposed at different times in the creation of a system. There are also many different ways to form teams, but in this paper we use a simple model that contains similar properties to many other team models [8, 20, 27]. In this paper, a team is defined as an aggregation of agents where each agent:

1. belongs to one and only one team;
2. receives the utility of the team; and
3. shares knowledge of the system state with its team members.

This definition of a team models many domains, where agents tend to be clustered geographically in such a way where it is realistic for an agent to be part of one fully communicating team, but not be part of several teams. Also since each agent is part of a single team and all the members of the team share a utility, anything an agent does to help another agent in the team will also help itself. Therefore, it is natural to assume that if possible the agents will share information with each other, whenever it is needed.

3. Congestion games

This paper tests the effectiveness of the proposed utilities and teams formations on variants of Arthur's bar problem [4]. This problem is chosen since it can be analyzed theoretically yet relates to many important congestion problems, including network routing and traffic management. Loosely speaking, in this problem at each time step each player i decides whether to attend a bar by predicting, based on its previous experience, whether the bar will be too crowded to be "rewarding" at that time, as quantified by a utility function G. The selfish nature of the players frustrates the system goal of maximizing G. This is because if most players think the attendance will be low (and therefore choose to attend), the attendance will actually be high, and vice-versa.

3.1. Non-binary congestion games

Here, we focus on the following more general variant of the bar problem investigated in Wolpert et al. [39]: There are N players, each picking one out of m bars every week. Each week, every player chooses a single bar. Then the associated private utilities for each player are communicated to that player, and the process is repeated. More formally, the global system utility in any particular week is:

$$G(z) \equiv \sum_{k=1}^{m} x_k(z) \exp\left(\frac{-x_k(z)}{c}\right) \tag{10}$$

where $x_k(z)$ is the total attendance at bar k; z_i is is move in that week; and c is a real-valued parameter. In this problem when either too few or too many players attend some bar in some week, the system utility G is low.

Since we wish to concentrate on the effects of the utilities rather than on the RL algorithms that use them, we use (very) simple RL algorithms. We would expect that even marginally more sophisticated RL algorithms would give better performance. In our algorithm each player i has a m-dimensional vector giving its estimates of the utility it would receive for choosing each possible bar. The decisions are made using the vector, with an ϵ-greedy learner with ϵ set to 0.05. All of the vectors are initially set to zero and there is a learning rate decay is 0.99.

3.2. Multi-time-step congestion games

To test the effects of communication restrictions and teams on a more difficult domain we use a variant of the previous congestion game called the Time Extended Bar Problem (TEBP)(Fig. 3). The TEBP is similar to the Bar Problem, except that each week a player can only choose to attend the same bar, or the bars next to the one he attended the previous week. To make it an episodic task, every four weeks the agents are reset to a random position, at which point they are given a reward based on their last choice of bar to attend. This task forces the agents to come up with a sequence of four actions that will maximize their final utility at the end of four weeks.

An agent learns in the TEBP using a Sarsa-learner. In every episode its 3 first rewards are zero and the last reward depends on the final bar attendance as computed in the single time step variant. The learner is an ϵ-greedy learner with ϵ set to 0.05 and $\gamma = 0.9$. The learning rate is set to $0.99^{v(s,a)}$ where $v(s, a)$ is a count of the number of times an agent took action a in state s.

3.3. Communication restrictions and team formation

We model communication restrictions in the bar problem by controlling how many other agents one agent can "talk" to. Without this communication the agent cannot know what the other agents have done. Here the communication level B will represent the fraction of

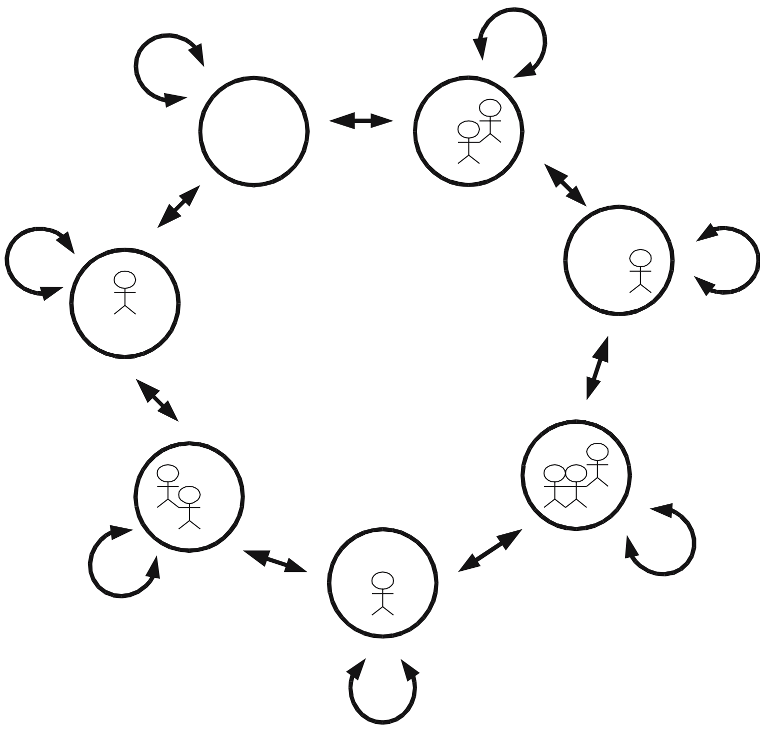

Fig. 3 Time extended bar problem. Circles represent bars, figures represent patrons attending a bar and arrows represent the transitions that patrons can take from week to week. Each week a patron can choose to attend the same bar or the bars next to the one he attended the previous week

all the agents to which an agent can talk. When $B = 1.0$ an agent can talk to the all other agents, whereas when $B = 0.0$ an agent has no communication, and thus is only aware of its own action. In the Bar Problem, communication restrictions are reduced to how $x_k(z)$ is computed. For truncated versions of the DU, (BTU and TTU), we use $x_k(z^{o_i})$ which returns how many of the observable patrons are going on bar k (note since in BTU the first term is broadcast, the agent does not need to compute it). For utilities using an estimate of the state (BEU and EEU), $x_k(z^{o_i})$ is scaled, and $\frac{1}{B}x_k(z^{o_i})$ represents the estimate of how many patrons actually went on bar k. For example when $B = 0.25$, we assume that $x_k(z^{o_i})$ is really only accounting for one quarter of the patrons, so we scale it by $\frac{1}{0.25} = 4$. Note this is an extremely simple estimation procedure and does not take any information an agent collects to modify how it forms this estimate.

Teams in the Bar Problem are modeled by creating disjoint groups of agents of approximately equal size. Every member of the team receives the same utility. In addition, we allow the members of a team to pool together all the information known by the team members. This means that each team member can get information about any agent that any of the team members can talk to. Therefore for the attendance for bar k that an agent i receives as a member of team k is: $x_k(z^{o_T})$ where z^{o_T} contains all the patrons that can be observed by at least one member of team T.

4. Results

We tested the performance of the four utilities, BTU, TTU, BEU and EEU with varying levels of communication, with and without teams. The test were conducted using the Bar Problem and Time Extended Bar Problem with 100 agents and with $c = 5$. All of the trials were conducted for 1000 episodes, and were run 25 times.

4.1. Communication restrictions without teams

The first set of experiments were conducted without teams (team size = 1). Figure 4 shows the performance of the four utilities with different levels of communication. When the communication level is high, the utilities converge to DU so the resulting performance converges. When communication is very low, the BTU and BEU have the best performance because their first term G is not affected by the communication restriction. They essentially are reduced to using the system utility as their individual utility, and give moderately good performance. Note that the performance of BTU is worse at 50% communication than at 5%. This counterintuitive result is explained by how the utility is computed in the bar problem. With less communication, the total number of agents that can be seen is small, and the contribution of the second term is small. With 50% communication on the other hand, the second term will be large enough to have an impact on the utility. However, because both at 5% and 50% communication levels $x_k(z^{o_i})$ is significantly different than $x_k(z)$, neither provide a usable second term. In fact, rather than subtracting out noise, the second term adds noise.

For most levels of communication restriction, the EEU performs the best and performs up to 75% closer to optimal than utilities which use the same information. Recall that EEU and TTU are not fully factored, whereas BTU and BEU are. What helps EEU in this case is that though it is not fully factored, as long as the estimate for G in the first term is sufficiently close to G, it is close to being fully factored. Furthermore, because both the first and second terms use the same estimate for the state, the subtraction does remove noise, as intended. The utility TTU performs worst of all since, even though there may not be much noise in

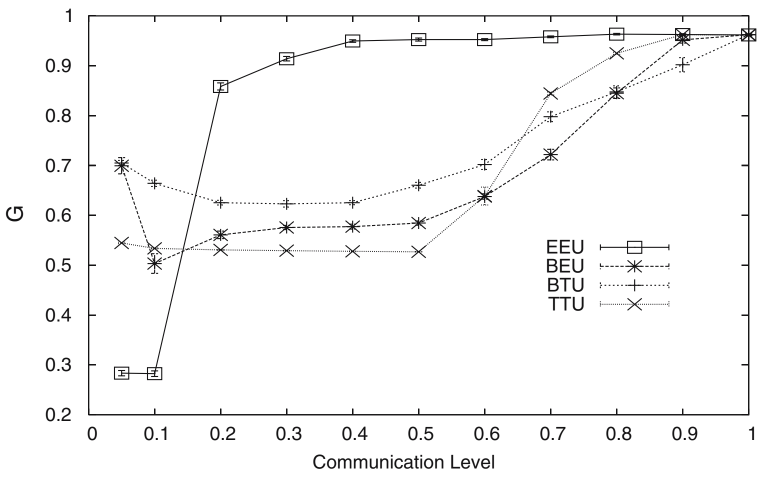

Fig. 4 Performance of four utility functions without teams for a range of communication levels (including error bars). For moderate communication levels EEU performs best. For very low communication BTU performs best since, it uses information from system utility

the utility, not only is it not fully factored with respect to the system utility, but due to the truncation, it may have a very low degree of factoredness.

Figure 5 gives a clearer view of the performances at a fixed level of communication restriction (40% and 70%). EEU is clearly superior at 40% communication. At 70% communication TTU displays the problem with utilities that have low factoredness: the more the agents learn the worse the system performance becomes. Because this system is not fully factored (or in this case, not close to being fully factored) the agents optimizing their private utilities do not optimize the system utility. Ironically, because TTU has good learnability (i.e., the slope of TTU shows no sign of flattening out at $t = 1000$) the agents learn to do the wrong thing successfully. BTU and BEU on the other hand, are fully factored so G does not decrease. However, because of learnability issues, after an initial period of improvement, the agents encounter a difficult signal to noise problem and the system performance stops improving.

4.2. Communication restrictions with teams

Even using the best utility, EEU, a high level of performance cannot be achieved if the communication level is too low. However, if agents can form small teams where information sharing is allowed between team members, good performance is possible even when communication between teams is low. Figure 6 shows the tradeoffs between choices of team size at different levels of communication. At most communication levels, there is an optimal team size that lies between the extremes of not having teams (team size = 1), and only having a single team (team size = 100). The best team size is typically around 5 or 10 agents. This optimum represents the best balance between having small team sizes which produce a more learnable utility and large team sizes which allows for more information sharing.

With the non-fully factored utilities EEU and TTU this balance comes from the tradeoff between factoredness and learnability. Even though as team sizes get smaller, the utilities become more learnable, they also become less factored since as information sharing goes down, the first term in the difference equation diverges from G. For the fully factored utilities BEU and BTU there is a tradeoff between two different ways noise comes into the system.

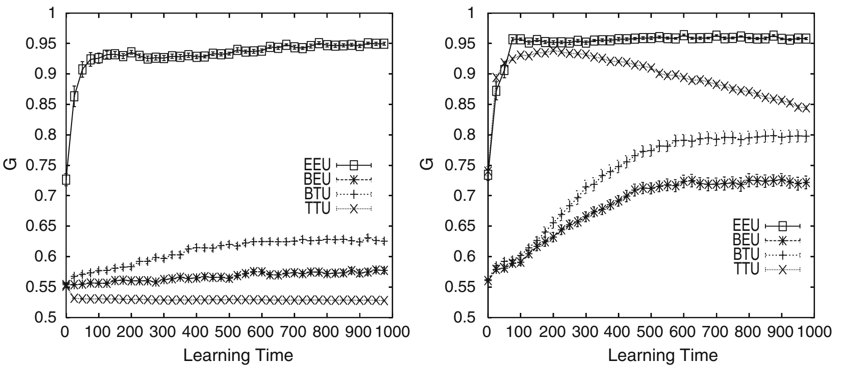

Fig. 5 Learning rates of four utility functions at 40% communication (left). EEU learns far quicker, since it produces a much less noisy signal. Note that even though TTU is highly learnable, it has a low degree of factoredness with respect to G, so it has a flat learning curve. At 70% communication (right) TTU is closer to being fully factored and can learn quickly, but still not fully factored, causing performance to eventually go down

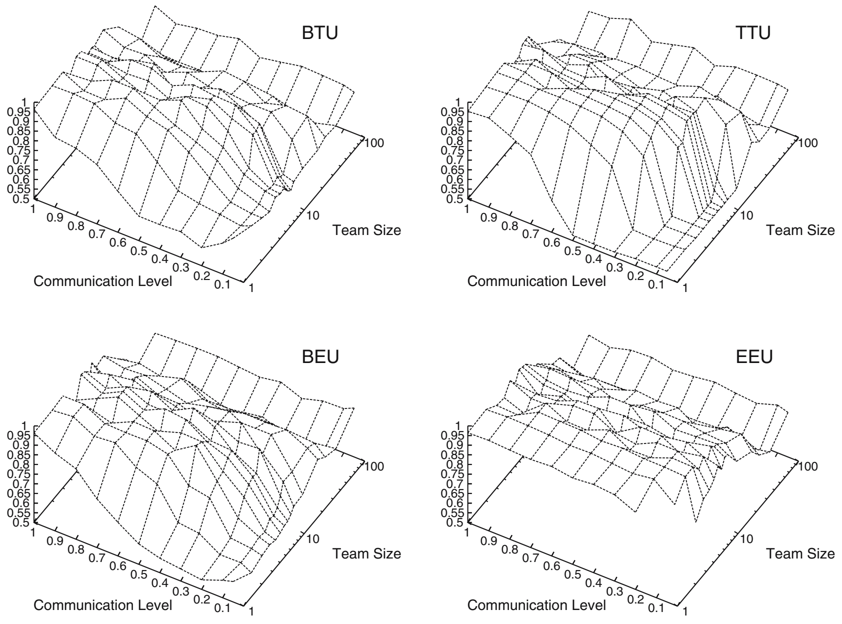

Fig. 6 Performance with different team sizes and communication levels. Each graph is for a different utility. From top-left, clockwise the utilities used are: BTU, TTU, EEU, BEU. The two utilities, BEU and EEU, that estimate hidden values rather than ignoring them, perform much better than their conterparts, BTU and TTU

When teams are large, more components have to be clamped in the second term of the difference equation allowing more noise from the first term to remain. When teams are small, the lack of information sharing has a similar effect, in that many of the components in the second term are clamped because their values are unknown.

Figure 7 shows that even when having teams is possible, the choice of utility is still critical. As in the case without teams, the EEU tends to perform best under most team sizes.

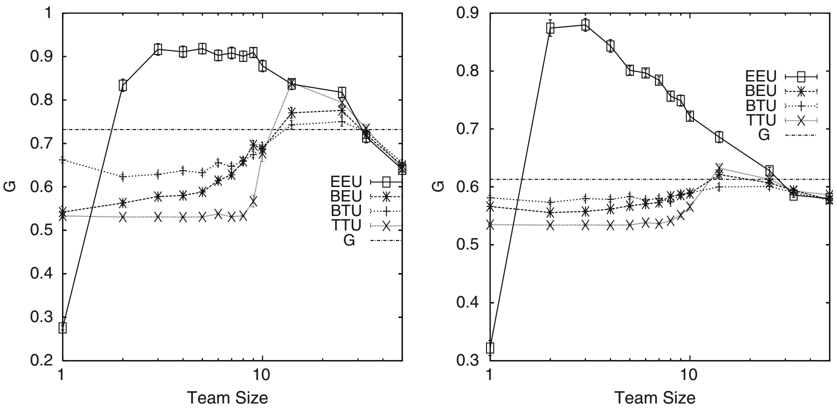

Fig. 7 Performance of four utility functions at 10% communication. EEU performs best for most team sizes under normal learning time (left).The signal-to-noise advantages of EEU become more apparent when learning time is reduced to 1/8 of original time (right)

Even though it is not fully factored, it has up to 25% higher performance than a $g_i = G$ system. Only with very small team sizes do the fully factored utilities perform better. When team sizes are very large, there are no hidden agents, so all the utilities converge to the same values. Due to the high learnability of EEU, its superiority is even more pronounced when the agents do not have much time to learn as shown in figure 7 (right).

4.3. Time extended results

To test the effectiveness of our methods on a more difficult problem, we performed the same experiments on Time Extended Bar Problem. This problem is harder since it is a Markov Decision Process, unlike the conventional Bar Problem which is a single step problem. In this problem agents have to find the best sequence of four actions instead of just a single action. To maximize comparability between the two problems, the time expended problems were tested identically to the non-time extended problems, except that they were conducted over 4000 learning steps instead of 1000.[4]

Figure 8 shows that the time extended problem is significantly harder. In trials with large team sizes, the agents were unable to learn at all on this problem. This happens because when the teams were large, the signal-to-noise ratio of the agents' utilities went down since the utilities contain the noise from all the other agents on a team. The signal-to-noise problem is a bigger issue with the Time Extended Bar Problem than with the original Bar Problem, since the noise is compounded in every time step. Even if an agent were able to take the correct action in the last three time steps, it may perform poorly if a noisy utility caused it not to take the correct action in the first time step. Also agents using utilities BTU and BEU suffered at low communication levels because the amount of noise in these utilities goes up as the communication level goes down, since less of the noise from other agents gets subtracted out. Figure 9 shows that at 5% communication, the only effective utility is EEU. All the other utilities have very low performance, resulting in actions that are not much better than random for most team sizes. This situations contrasts to the easier non-time-extended problem, where many of the utilities would result in a reasonable performance, especially with large team sizes.

[4] Both experiments were conducted for 1000 episodes, but the Time Extended problem has four steps per episode.

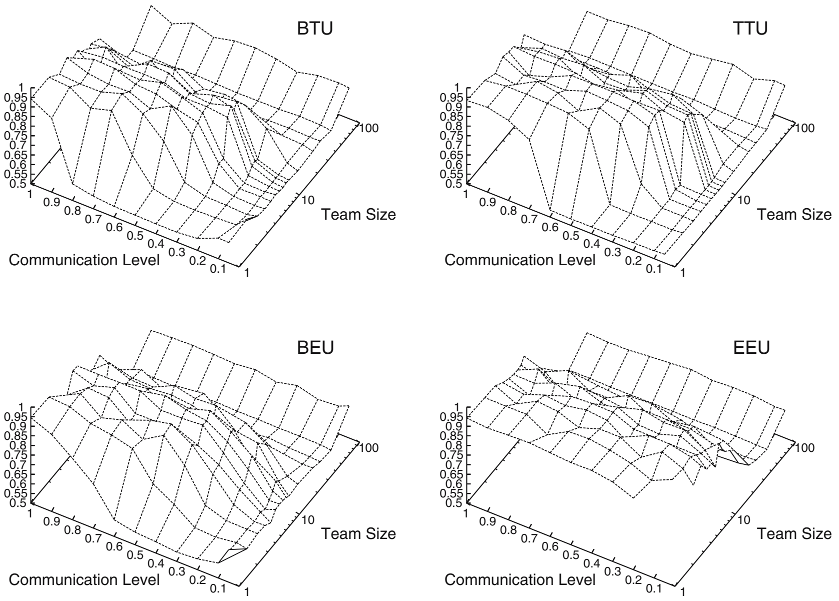

Fig. 8 Performance with different team sizes and communication levels for Time Extended Bar Problem. Each graph is for a different utility. From top-left, clockwise the utilities used are: BTU, TTU, EEU, BEU. The two utilities, BEU and EEU, that estimate hidden values rather than ignoring them, perform much better than their conterparts, BTU and TTU. Note that with large team sizes, agents can not effectively learn in the time extended problem

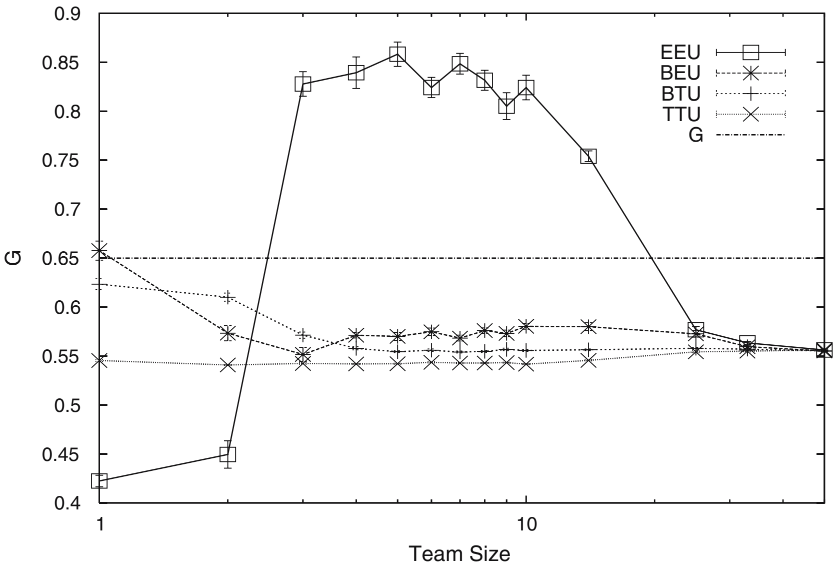

Fig. 9 Performance of four utility functions at 5% communication. Superiority of EEU is even more clear in the time extended problem

5. Related work

Issues related to agent communication and team formation have been studied separately for many years from a variety of viewpoints. Literature closer to the focus of this paper, where teams are used to overcome limited communication is less common and tends to come from sensor-fusion research. In addition there is a large body of related work on multi-agent systems and how to coordinate multiple agents.

5.1. Communication among agents

The study of communication among agents has taken on many forms. Much work has been done low level communication issues such as agent communication languages and physical implementation of communications [12, 13, 29]. Pynadath and Tambe [25] have formalized many aspects of agent communications, including observability and explicit communication. For multi-agent Markov decision processes, Xuan et al. [40] dealt with the problem of partially hidden states of other agents. In their system communication of an agent state had a cost and they presented a number of algorithms that traded off cost of communication vs. the expected gain from the knowledge obtained from the communication. A number of researches have noted that often little communication is needed to coordinate agents [5] and that in many cases local communication is sufficient [15, 28], though such conclusions necessarily depend on the chosen domain.

5.2. Teams

There has been extensive research on rule-based agent team formations. Tambe has shown that coordination rules can be used successfully in many fields including military engagement [31]. A common mechanism to coordinate team agents is for teams to have "joint intentions" [10] where team agents need to work for a common goal. Groz coins the term "SharedPlan" [17] to refer to this concept. In this paper we borrow from this concept by having team members share a common utility. Even more related to this paper is work done in the field of sensor fusion. Fox has shown that when the amount of information that a robot receives is restricted, teams of robots, with different sensors, can work together to solve the robot localization problem [14]. In addition it has been shown that teams can share sensor information to estimate unobservable parts of the system in robotic soccer domains [30].

Significant work has also been done on coalition formation in distributed systems for the purpose of increasing efficiency, where coalitions are formed dynamically as the system runs. In these domains one significant problem is how to assign a value to an individual agent's contribution to a coalition. Ketchpel [20] addresses problem through a local auction mechanism. Another issue with coalitions is that, while larger coalitions may provide more benefit, they may entail substantial costs. Sandholm and Lesser [27] show properties of coalitions when computational cost of being part of a coalition goes up as the coalition gets bigger. While in the previous examples agents are trying to maximize coalition utility or system utility, in Brooks and Durfee [8] agents are self centered, but join congregations to try to increase their own utilities. In all these works, as in our work, it is assumed that an agent only belongs to one coalition/team.

5.3. Agent coordination

Learning agents in multi-agent systems must ensure that agents learn to cooperate to optimize the overall system goal. Leveraging game theory and reinforcement learning, Hu and

Wellman [19] accomplish this through an algorithm where through Q-learning a pair of agents could reach a Nash equilibrium in general sum games, even when the agents do not know the reward function or state transition probabilities. However, this algorithm was not designed to scale to a large number of agents. In contrast, Mataric [22] has shown that large groups of foraging robots can be made to cooperate by constructing a set of utilities appropriate for the domain.

6. Discussion

In this work we focus on the problem of designing a collective with a large number of autonomous agents in the presence of severe communication restrictions. This problem is particularly challenging in that the following issues are all present at once:

- There are a large number of agents.
- Agents must learn solution.
- Greedy solutions are highly suboptimal.
- Agents can only observe small fraction of other agents.

In this problem we must promote agent coordination, even when agents may not be able to communicate with one another. In such cases, private utilities which rely on agents having access to a fully connected communication network will break down. To address this issue, we presented four different utility functions that each make different tradeoffs among what information is available to an agent and how that information should be used. We showed that one utility in particular, EEU, does far better than all the others in almost all experiments. Agents using this utility learn much faster and achieve better results on the traditional and time extended variants of the Bar Problem. Furthermore, agents using this utility can perform well with far more restricted communication.

In addition, we showed that team formation improves the performance of the system by as much as 25% on top of the 75% performance increase achieved by using better utilities. This was due to the increased information available to the members of a team, which alleviated the communication restriction imposed on the agents. While increasing the size of the team increased the information available, it also increased the complexity of each team's problem. The improved performance illustrated that a good balance point between these tradeoffs could be found.

These results were all obtained on a (possibly time dependent) congestion problem with 100 agents. This problem is part of an important class of problems, that cover a wide variety of domains including network routing, traffic control, and multi-robot coordination. Similar use of difference utilities have shown to work well on an even wider variety of domains [1, 2, 32, 33, 38]. In addition while scaling results were not explicitly shown, the methods presented here performed well with 100 agents, and scaled well with different number of teams. In other works we have shown similar methods to scale well, from 10 to 400 agents [3, 33].

This performance boost was achieved using a simple team model, where team members which had a common utility always chose to share information. Also for simplicity, there were no costs to share information. In certain domains sharing costs may be significant, but in many cases this cost will not effect the applicability of our utility functions. For example if the sharing cost is solely a function of team size, it will cancel out in the difference utilities. In these domains the collective designer should include the cost of sharing in the system utility and adjust the group size so as to maximize performance in the specific domain.

Furthermore, in many problems agents can choose whether to share information or not, and consequently incur a cost or not. Preliminary results show that in such cases, agents have a difficult time in learning to maximize their utility functions. This is due to the constant change in how an agent perceives the system, which now depends on the sharing choices of many other agents. This in effect creates a more noisy learning environment. Our current research focuses on these issues and on prodding the agents to share information in the absence of teams.

Acknowledgements The authors would like to thank David Wolpert for helpful discussions.

References

1. Agogino, A., & Tumer, K. (2004). Efficient evaluation functions for multi-rover systems. In *The genetic and evolutionary computation conference* (pp. 1–12). Seatle, WA.
2. Agogino, A., & Tumer, K. (2005a). Multi agent reward analysis for learning in noisy domains. In *Proceedings of the fourth international joint conference on autonomous agents and multi-agent systems*. Utrecht, Netherlands.
3. Agogino, A., & Tumer, K. (2005b). Reinforcement learning in large multi-agent systems. In *AAMAS-05 workshop on coordination of large scale multi-agent systems*. Utrecht, Netherlands.
4. Arthur, W. B. (1994). Complexity in economic theory: Inductive reasoning and bounded rationality. *The American Economic Review, 84*(2), 406–411.
5. Balch, T., & Arkin, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots, 1*(1), 27–52.
6. Blumrosen, L., & Nisan, N. (2002). Auctions with severely bounded communication. In *The 43rd annual IEEE symposium on foundations of computer science*. Vancouver, Canada.
7. Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the sixth conference on theoretical aspects of rationality and knowledge,* Holland.
8. Brooks, C. H., & Durfee, E. H. (2003). Congregation formation in multiagent systems. *Autonomous Agents and Multi-Agent Systems Journal*, 145–170.
9. Busetta, P., Dona, A., & Nori, M. (2002). Channeled multicast for group communications. In *Proceedings of the first international joint conference on autonomous agents and multi-agent systems* (pp. 1280–1287). Bologna, Italy.
10. Cohen, P., & Levesque, H. N. (1991). Teamwork. *Special Issue on Cognitive Science and AI, 25*(4), 487–512.
11. Crites, R. H., & Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo, (Eds.), *Advances in neural information processing systems* Vol. 8 (pp. 1017–1023).
12. Dastani, M., van der Ham, J., & Dignum, F. (2002). Communication for goal directed agents. In *Proceedings of the agent communication languages and conversation policies*. Bologna, Italy.
13. Dignum, F., Dunin-Keplicz, B., & Verbrugge, R. (2000). Agent theory for team formation by dialogue. In *Proceedings of the agents, theories, architectures and languages (ATAL 2000)* (pp. 150–166). Boston, MA.
14. Fox, D., Burgard, W. Kruppa, H., & Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*.
15. Fredslund, J., & Mataric, M. J. (2002). Robots in formation using local information. In *Proceedings, 7th international conference on intelligent autonomous systems (IAS-7)* (pp. 100–107). Marina del Rey, CA.
16. Gage, D. (1993). How to communicate with zillions of robots. In *Proceedings of SPIE mobile robots VIII* (pp. 250–257). Boston, MA.
17. Grosz, B., & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence, 2*(86), 269–357.
18. Groves, T. (1973). Incentives in teams. *Econometrica, 41*, 617–631.
19. Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the fifteenth international conference on machine learning* (pp. 242–250).
20. Ketchpel, S. P. (1994). Forming coalitions in the face of uncertain rewards. In *National conference on artificial intelligence* (pp. 414–419).
21. Kraus, S. (1997). Negotiation and cooperation in multi-agent environments. *Artificial Intelligence, 94*, 79–97.

22. Mataric, M. J. (1994). Reward functions for accelerated learning. In *Machine learning: Proceedings of the eleventh international conference* (pp. 181–189). San Francisco, CA.
23. Nicholson, W. (1998), *Microeconomic theory*, 7th ed. The Dryden Press.
24. Petersen, S. A., & Divitini, M. (2002). Using agents to support the selection of virtual enterprise teams. In *Proceedings of fourth international bi-conference workshop on agent-oriented information systems (AOIS-2002) (at AAMAS 2002)*. Bologna, Italy.
25. Pynadath, D., & Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research, 16*, 389–423.
26. Pynadath, D., Tambe, M., Chauvat, N., & Cavedon, L. (1999). Toward team-oriented programming. In *Proceedings of the agents, theories, architectures and languages (ATAL'99)* (pp. 77–91). Orlando, Florida.
27. Sandholm, T., & Lesser, V. R. (1997). Coalitions among computationally bounded agents. *Artificial Intelligence, 94*, 99–137.
28. Sen, S., Sekaran, M., & Hale, J. (1994). Learning to coordinate without sharing information. In *Proceedings of the twelfth national conference on artificial intelligence* (pp. 426–431). Seattle, WA.
29. Smith, I. A., & Cohen, P. R. (1995). Toward a semantics for a speech act based agent communications language. In T. Finin, & J. Mayfield, (Eds.), *Proceedings of the CIKM '95 workshop on intelligent information agents*. Baltimore, Maryland.
30. Stroupe, A., Martin, M. C., & Balch, T. (2001). Distributed sensor fusion for object position estimation by multi-robot systems. In *IEEE international conference on robotics and automation, May, 2001*.
31. Talukdar, S., Baerentzen, L., Gove, A., & de Souza, P. (1998). Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, *4*, 295–321.
32. Tumer, K., & Agogino, A. (2005). Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *The genetic and evolutionary computation conference*. Washington, DC.
33. Tumer, K., Agogino, A., & Wolpert, D. (2002). Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the first international joint conference on autonomous agents and multi-agent systems* (pp. 378–385). Bologna, Italy.
34. Tumer, K., & Wolpert, D. (Eds.) 2004. *Collectives and the design of complex systems*. New York: Springer.
35. Vickrey, W. (1961). Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance, 16*, 8–37.
36. Wolpert, D., & Lawson, J. (2002). Designing agent collectives for systems with Markovian dynamics. In *Proceedings of the first international joint conference on autonomous agents and multi-agent systems*. Bologna, Italy.
37. Wolpert, D. H., & Tumer, K. (2001). Optimal payoff functions for members of collectives. *Advances in Complex Systems, 4*(2/3), 265–279.
38. Wolpert, D. H., Tumer, K., & Frank, J. (1999). Using collective intelligence to route internet traffic. In *Advances in neural information processing systems*. Vol. 11 (pp. 952–958).
39. Wolpert, D. H., Wheeler, K., & Tumer, K. (2000). Collective intelligence for control of distributed dynamical systems. *Europhysics Letters, 49*(6). 708–714.
40. Xuan, P., Lesser, V., & Zilberstein, S. (2001). Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the fifth international conference on autonomous agents* (pp. 616–623). Montreal.