Local Learning to Improve Organizational Performance in Networked Multiagent Team Formation

Blazej Bulka and **Matthew Gaston** and **Marie desJardins**
Department of Computer Science
University of Maryland Baltimore County
{bulka1, mgasto1, mariedj}@cs.umbc.edu

Abstract

Networked multiagent systems are comprised of many autonomous yet interdependent agents situated in a virtual social network. Two examples of such systems are supply chain networks and sensor networks. A common challenge in many networked multiagent systems is decentralized team formation among the spatially and logically extended agents. Even in cooperative multiagent systems, efficient team formation is made difficult by the limited local information available to the individual agents. We present a model of distributed multiagent team formation in networked multi-agent systems, describe a policy learning framework for joining teams based on local information, and give empirical results on improving team formation performance. In particular, we show that local policy learning from limited information leads to a significant increase in organizational team formation performance compared to a naive heuristic.

Introduction

Distributed team formation is a common challenge in many multiagent systems. In sensor networks, collections of sensors, sector managers, and other agents must dynamically decide to work together to track targets or adapt to environmental changes (Culler, Estrin, & Srivastava 2004; Horling, Benyo, & Lesser 2001). In supply networks, manufacturers, suppliers, distributors, wholesalers, and consumers form informal teams in order to efficiently distribute goods and to adapt to supply and demand (Thadakamalla *et al.* 2004; Swaminathan, Smith, & Sadeh 1998). The process of distributed team formation involves agents autonomously deciding to cooperate with other agents in order to accomplish joint tasks.

Several factors compound the challenge of designing agents for efficient distributed team formation. In large agent organizations, the agent-to-agent interactions may be limited by a social network. The social network may result from a variety of factors, including physical proximity, communication limitations, limited knowledge of other agents and their capabilities, trust relationships, and organizational structures. Dynamic, real-time task environments, where the agents have limited information about the structure of future tasks, make the design of effective team joining strategies difficult. Finally, limited local information as a result of the virtual social network and the lack of a centralized controller makes it difficult for agents to determine the behaviors of other agents and to decide which teams to join.

Most of the previous work on agents learning to form teams in multiagent systems focused on environments with access to global information and no restrictions arising from the social network. Kraus, Shehory, & Taase described and evaluated several heuristics for coalition formation (Kraus, Shehory, & Taase 2003). These heuristics involved an agent evaluating all possible coalitions, assuming that the agent could observe all possible tasks and all other agents. Nair, Tambe, & Marsella formulated the problem of allocating roles among existing teams as a communicating decentralized POMDP (Nair, Tambe, & Marsella 2002). Although it is possible to solve the problem when formulated in this way, it is shown to be NEXP-complete (Bernstein *et al.* 2002), making solutions for systems with more than a few agents intractable. This observation supports our emphasis on the distributed, on-line learning of heuristics for dynamic team formation. Other work focused on learning to select appropriate coalition partners assuming no restrictions imposed by the social network (Dutta & Sen 2003), or learning to select an optimal, predefined coordination method (Excelente-Toledo & Jennings 2003).

This paper focuses on the problem of dynamic, distributed team formation in a networked multiagent system with a real-time task environment. In particular, we develop and empirically test a learning framework within which agents attempt to learn effective team initiating and team joining policies online. The goal is to test whether agents can learn local policies that improve the aggregate performance of the agent organization, measured as the proportion of successfully completed tasks, given only access to local information. We find that it is possible for agents to learn effective policies from experience starting from a default random policy. The learned policies are adapted to an agent's position in the social network, which restricts its interactions with other agents. Ultimately, the locally learned policies lead to significant improvement of the overall organizational performance in a variety of network structures.

A Model of Multiagent Team Formation

To explore simultaneous learning of team joining policies in a dynamic environment, we have selected a simple multi-agent system model, motivated by previous work on agent team formation (Abdallah & Lesser 2004; Kraus, Shehory, & Taase 2003; Nair, Tambe, & Marsella 2002). The model provides a dynamic team formation environment in which agent teams form spontaneously in a completely decentralized manner and the agents' decision making is based solely on local information. In addition, the model allows for potentially very large agent organizations where the agents are embedded in a social network. The model is only concerned with the dynamic formation of teams and not with post-formation teamwork mechanisms or protocols, for which there is a large body of previous work (Hunsberger & Grosz 2000; Nair, Tambe, & Marsella 2003; Pynadath & Tambe 2002; Tambe 1997).

In the model, tasks are generated periodically and are globally advertised to the organization. Agents attempt to form teams to accomplish these tasks. The agents in the organization are embedded in a virtual social network that restricts the set of possible agent teams: specifically, a legal team must be a connected subgraph of the social network. Since we are only concerned with the formation process, tasks are generic in that they only require that a team of agents with the necessary skills form to accomplish the specific task.

In this model of team formation, the organization consists of N agents, $A = \{a_1, a_2, \ldots, a_N\}$, where each agent can be considered as a unique node in a social network. The social network is modeled as an adjacency matrix E, where an element of the adjacency matrix $e_{ij} = 1$ if there is an edge between agent a_i and a_j and $e_{ij} = 0$ otherwise. The social relationships among the agents are undirected, or symmetric, so $e_{ij} = e_{ji}$. In the agent organization, each agent is also assigned a single fixed skill, $\sigma_i \in [1, \sigma]$, where σ is the number of different types of skills that are present in the organization. In the experiments presented in this paper, skills are assigned to each agent uniformly at random from the set of potential skills.

During the team formation process, each agent can be in one of three states: UNCOMMITTED, COMMITTED, or ACTIVE. An agent in the UNCOMMITTED state is available and not assigned to any task. An agent in the COMMITTED state has selected a task, but the full team to work on the task has not yet formed. Finally, an agent in the ACTIVE state is a member of a team that has fulfilled all of the skill requirements for a task and is actively working on that task. Only uncommitted agents can commit to a new or partially filled task. Committed agents cannot decommit from a given task. Upon task completion, agents in the ACTIVE state return to the UNCOMMITTED state. An agent's state is denoted s_i.

Tasks are introduced at fixed task introduction intervals, where the length of the interval between tasks is given by the model parameter, μ. Tasks are globally advertised (i.e., announced to all agents). Each task T_k has an associated size requirement, $|T_k|$, and a $|T_k|$-dimensional vector of required skills, R_{T_k}. The skills required for a given task T_k are chosen uniformly with replacement from $[1, \sigma]$. Each task is advertised for a finite number of time steps γ, ensuring that the resources (i.e., agents) committed to the tasks are freed if the full requirements of the task cannot be met. That is, if a task is advertised for more than γ time steps without a full team forming, the agents committed to the task return to the UNCOMMITTED state. Similarly, teams that successfully form to fill the requirements of a given task are only active for a finite number of time steps α.

Agent Social Networks Since we are primarily interested in large agent organizations where the control of the agents does not necessarily fall under a single authority, we explicitly model the agent social network. Real-world constraints such as cognitive limitations, geographical considerations, and limited communication capabilities make it impossible for an individual agent to know about or communicate with *all* other agents. In our team formation model, the agent social network explicitly restricts the sets of agents that can form teams.

Definition 1 *A* **valid team** *is a set of agents $M = \{a_i\}$ that induce a connected subgraph of the agent social network and whose skill set $\{\sigma_i\}$ fulfills the skill requirements for task T_k.*

The requirement of a team to induce a connected subgraph of the agent social network means that for each agent in the team, $a_i \in M$, there must exist at least one other agent, $a_j \in M, i \neq j$, such that $e_{ij} = 1$. This implies that an agent satisfying skill requirements is only eligible to commit to a task in two situations: 1) when no other agents are committed to the task, or 2) when at least one neighbor of the agent is already committed to the task. In essence, this requires that an agent on a team must "know" at least one other agent on the team.

Previous work on networked multi-agent team formation has demonstrated that the ability of an organization of agents to effectively form teams depends heavily on the structure of the agent social network (Gaston & desJardins 2003). In particular, networks with short average path length and hub-like structures support a diverse set of possible teams and, therefore, a large number of task configurations. We have also explored, given the dependence of team formation performance on network structure, policies for adapting the network structure as a means of improving performance. In this paper, however, we embed the agents in fixed network structures and focus on the ability of individual agents to learn effective team joining policies.

Team Formation Policies During each iteration of the model, the agents receive advertisements about the current tasks in the system. Each agent with $s_i =$ UNCOMMITTED considers all currently advertised tasks for which it is eligible and selects at most one of them. If the selected task has no other agents committed to it, an agent initiates a new team; otherwise, it joins the existing team. We refer to an agent's strategy for selecting teams to initiate as the agent's *team initiation policy*. The strategy for joining existing teams is termed the agent's *team joining policy*.

The policies for initiating and joining teams are an important factor in the performance of multiagent team formation. There are many possibilities for these policies. One simple policy involves an agent always trying to commit to a task by randomly selecting from all tasks for which the agent is eligible. Such a policy, which combines team initiation and team joining, may create unnecessarily high demand on, or inefficient use of, the resources in the system (e.g., agents may commit to tasks for which it is impossible for a team to form). Another simple policy, which may place less burden on the agents, is a policy in which agents commit to tasks with a given probability that depends in a fixed way on local information, such as the relative percentage of positions filled. We are interested in designing agents that autonomously learn effective team formation policies to improve organizational performance. Our framework for learning team joining and team initiating policies is discussed below (*Learning to Initiate and Join Teams*).

In our experiments, we use as a lower baseline the *random policy*, which selects uniformly from all of the tasks it is eligible for, including a virtual *wait task*. Let E denote the number of tasks for which an agent is eligible. An agent using the random policy selects from the eligible tasks (including the wait task) with probability

$$P(J_i) = \frac{1}{E+1}, \tag{1}$$

where J_i is the act of committing to the ith eligible task. The purpose of the wait task is to prevent agents from always committing to a team at the first opportunity, since doing so may not necessarily be the best option. Although the random policy is quite simple, we have found it to outperform other more sophisticated hand-designed strategies for over-constrained networked team formation environments.

Organizational Performance We measure the team formation performance of the agent organization as the ratio of number of teams successfully formed to the total number of tasks introduced to the system:

$$\text{org. performance} = \frac{\text{\#of teams successfully formed}}{\text{\#of tasks introduced}} \tag{2}$$

This measure of performance provides a global measure of the effectiveness of the agent organization in forming teams to execute the advertised tasks. The value of the measure is comparable only for environments with the same task load. An increase of the number of tasks above the theoretical maximum (imposed by the number of agents and network shape) causes the value to be low, even when the agents' actions are optimal. In our experiments on learning team joining and team initiating policies, we are only concerned with cooperative agents that share the collective goal of improving organizational performance. In the following section, we describe a framework for individual agents to learn effective team formation policies and discuss the challenges associated with distributed simultaneous learning.

Learning to Initiate and Join Teams

Rather than designing specific team formation policies for agents, mechanisms for the individual learning of effective team joining and team initiating policies allow the agents and the organization to deal with variability in the team formation environment, including task structure and network structure (e.g., node failure). Moreover, in many network structures, the agents have diverse patterns of connectivity with the rest of the organization. It is difficult to specify team joining policies for all possible positions in a social network—for example, a hub agent with many connections may need a different policy than a "fringe" agent with only one or two connections—so learning to effectively join teams given any position in the network is extremely useful.

In order to capture a realistic team formation environment, agents are forced to make decisions based solely on local information. Any access to global data would require either extensive communication among agents or a centralized point in the system. Moreover, relying only on local information, reduces the computational burden on the agents.

Some of the key challenges involved in multiagent learning in a distributed team formation environment include:

- **Decisions based solely on local information:** Each agent must make decisions about joining teams, initiating teams, or waiting given only partial information about the state of the organization. The information is limited by the position of the agent in the social network and by the dynamic nature of the environment (i.e., an agent cannot know what tasks will be introduced to the system in the near future).
- **Local policy interference:** The policies of two nearby agents (e.g., neighbors or neighbors of neighbors) can interfere with each other when the agents attempt to select teams to initiate and join. For example, one agent may learn to join teams only when a specific neighbor is UNCOMMITTED (hoping that the UNCOMMITTED neighbor will subsequently join the team), while the neighbor may learn to avoid teams of which the first agent is a member. Moreover, interference can propagate through the agent organization. We observed this problem to be common in simultaneous interdependent adaptation.
- **Learning to wait:** In the dynamic team formation environment, it may be in an agent's best interest to delay, rather than committing to any task for which the agent is eligible. However, the reward for waiting is necessarily delayed, making it difficult to explicitly learn to wait. Additionally, the value of deciding to wait is dependent upon the actions of an agent's neighbors (and, indirectly, on the rest of the agent network).
- **Blocking:** If agents are strictly greedy and attempt to join as many successful teams as possible, blocking may increase. Blocking occurs when an agent is in the COMMITTED or ACTIVE state, preventing teams from forming in that part of the network structure. This phenomenon is similar to a bottleneck in a flow network. It is desirable for agents to learn to prevent or reduce blocking.
- **Local performance $\neq$ global performance:** If the agents in the organization attempt to naively optimize local performance (i.e., join as many successful teams as possible), global performance is not necessarily optimized. One

"greedy" agent can reduce the performance of all of its neighbors, by making poor task choices or "stealing" skill slots on a specific task, resulting in lower collective performance.

Many of these challenges are interdependent, manifesting themselves simultaneously in the dynamic team formation environment.

The Learning Framework In our team formation model, the agents receive advertisements of tasks for which they are eligible during each iteration. The agents decide whether to initiate or join a team based on the currently advertised tasks and on the locally perceived state of the agent organization. The agents also try to learn effective team joining and team initiating policies in real time based on their previous experiences. Whenever an agent decides to join a team, it records the current state of its environment, including the state of the team and the states of its neighbors. Later, when it is known whether the team formed successfully or failed to form within the time limit, this stored state is used to create a training instance. The instance is labeled as *JOIN* or *IGNORE*, based on the success or failure of the team, respectively. Subsequently, the learning model is updated using these training instances to influence future team formation decisions.

A training instance includes the following attributes of an agent's local environment: the states of each of the neighbors; the skills of each of the neighbors; the number of unfilled slots in the task; the skills required for the task; and the amount of time that the task has been advertised. The information about the skills required for a task includes whether each of the σ skills is *not required for this task*, *required but already filled*, or *required but not filled*.

In our learning framework, the agents use two sets of classifiers: one for team initiation and another for joining existing teams. The decision to use two classifiers was based on the fact that initiating a team and joining a partially filled team are very different situations. For example, joining a team that already has most of its skill slots filled should have a greater probability of success than joining a team with mostly empty skill slots. However, if all agents only join nearly full teams, then overall organizational performance would suffer, since agents would never initiate new teams.

The knowledge summarized in a classifier may become outdated over time, because the neighboring agents also learn and change their behavior. This creates a need to discard old experience that does not reflect the current situation. We solved this problem using a simple incremental learning scheme. We maintain two classifiers for team initiation and two classifiers for team joining. One of the two classifiers – the *decision classifier* – is considered to be sufficiently trained and is used to make current decisions. While it is being used, the decision classifier is also updated in real time as training instances are created. The second classifier – the *classifier-in-training* – is not used to make decisions until it becomes well trained, which occurs after a certain number of time steps. At that point, the existing decision classifier is discarded and replaced with the classifier-in-training, and the agent creates a new classifier-in-training. This framework for swapping classifiers ensures that agents are making decisions based on their most recent experiences.

We used the WEKA library (Witten & Frank 2000) to provide the actual implementation of the classifiers. Initially, we experimented with a range of classifiers but finally we selected Naïve Bayes (Domingos & Pazzani 1997) as our classifier of choice because of its ability to be updated with new learning instances in real time, its simplicity, and its performance.

Agents decide which tasks to join by consulting their decision classifiers. For each available task, an agent creates a testing instance related to the task. Using Naïve Bayes, these testing instances (i.e., all eligible tasks) are assigned a probability of success, f_i, by the appropriate decision classifier. The agent randomly selects a task to commit to based on the probabilities of success. The probability $P(J_i)$ of selecting a task J_i is proportional to its estimated probability of success.

$$P(J_i) = \frac{f_i}{\sum_i f_i} \tag{3}$$

Additionally, if the success probabilities of all eligible tasks sum to less than one, the agent adds a virtual *wait task* with probability equal to $1 - \sum_i f_i$. The wait task allows the agent to delay taking action if current conditions are unfavorable, meaning that no current task is likely to be completed.

Initially, the agents have insufficient experience to make sensible decisions. As a result, the agents usually estimate task success probabilities as very low. Because of this, no teams would be joined, no experience gathered, and a vicious cycle of inexperience would ensue. To avoid this form of "learned helplessness," agents always switch to using the random policy after periods of inactivity. We denote the length of this period of inactivity as the *idleness limit*. If an agent exceeds its idleness limit (i.e., it does not attempt to join a team for a specified amount of time), it reverts to selecting a task to join at random.

Filtering Negative Instances Since we focus on heavily loaded task environments, most of the agents see far more team failures than successes. If this is not taken into account, the perceived task success probability may drop. As a result, the agents will quickly stop joining any teams (unless forced by the idleness limit) and the performance will degrade below that of the random policy. In order to avoid this difficulty, we introduced filtering, or sampling, of negative training instances. In our learning framework, negative training instances are filtered if the ratio of positive training instances to negative training instances is too low. Specifically, a negative training instance will be kept with probability p_{acc}, where

$$p_{acc} = \min\left(1.0, 2 \times r \times \frac{\text{\#of positive instances}}{\text{\#of all instances}}\right). \tag{4}$$

The ratio r determines the acceptance rate of negative instances.

Results

The goal of our experiments was to determine if it is possible for agents to learn effective team joining and team initiating policies. We also wished to evaluate the ability of the agents to learn, *independently of the agent social network*. That is, the goal was to develop and test a learning framework for an agent embedded in an arbitrary network structure. In that light, we tested our approach in a variety of commonly used and realistic network structures: random graphs (Erdos & Renyi 1959), scale-free graphs (Albert & Barabási 2002), regular graphs (lattices), and small-world networks (Watts & Strogatz 1998).

Experimental Design In order to evaluate our learning framework, we ran simulations of the team formation model with the agents embedded in various social network structures. All of our experiments were in agent societies that consisted of 100 agents connected in a social network with approximately 400 edges. We generated 10 such network structures from each of the four network classes: lattice, small-world, random, and scale-free.

We ran a series of simulations on each of the network structures with the agents using the random policy to initiate and join teams, and a second series with the agents learning team formation policies in real time. Each simulation was run for 20,000 time steps. In all simulations, $\sigma = 10$ and $|T_k| = 10$. A new task was introduced to the system every time step ($\mu = 1$) and a task was advertised for up to 10 time steps ($\gamma = 10$). Once a task began execution, it executed for 10 time steps ($\alpha = 10$). Given this environment, an agent could receive at most ten simultaneous task advertisements (a task rich environment; because of the restrictions imposed by the shape of network, agents were not capable of executing all tasks introduced).

The idleness limit was set to 30; i.e., learning agents were forced to join some team after 30 time steps without attempting to join any teams. The classifiers were trained for 2000 iterations before they were used as decision classifiers (except for the classifiers at the beginning of the simulation), and the negative instances were filtered with ratio $r = 0.1$ (see Equation 4). These parameters were determined experimentally. A lower idleness limit led to situations where agents rarely joined teams. A higher limit led to a situation where agents perceived that their neighbors were not willing to join teams, and therefore they also declined to join (especially at the beginning of the simulation, when the classifiers were yet untrained). Similarly, a shorter training duration for the classifiers caused poor decisions (too few positive training instances), and a longer training time made some of the agent's experience outdated.

Comparison of Strategies We found that agents were able to learn effective team formation policies (compared to the random policy), regardless of the underlying social network structure. In all of the networks, the learned policies significantly outperformed the random policy. The average results over all networks are presented in Figure 1. The error bars show the 95% confidence interval for each measurement. The agents consistently improved their performance over the first 5000 time steps. This gain in performance can be explained by the agents' use of accumulated experience. Moreover, there is a visible increase between 4000 and 5000 time steps; this is because at the 4000th iteration, the agents started to use a new decision classifiers that did not incorporate "experience" gained during the initial period (0–2000 time steps). It is beneficial to discard the experience from the initial period because during that period, agents' decisions are more random and inefficient.

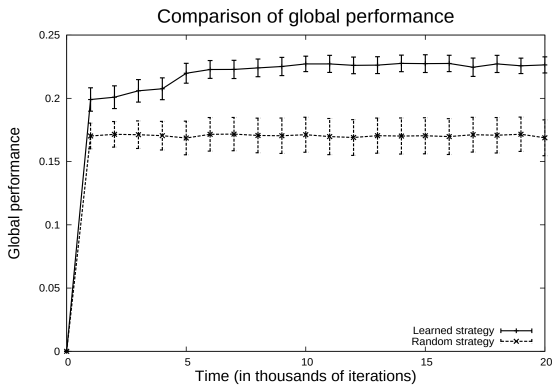
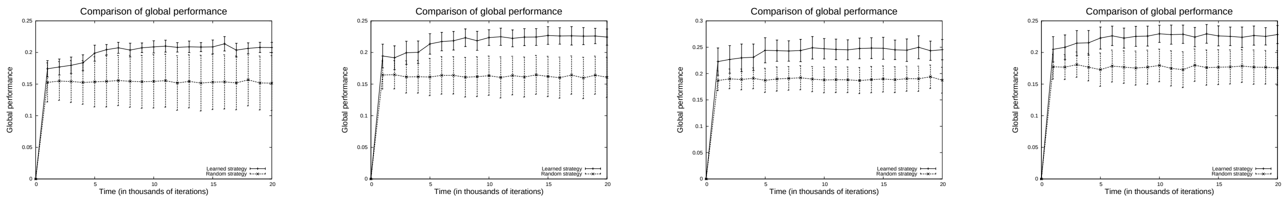


Figure 1: Average performance.

We also found that the improved performance of the learned strategy did not increase the average number of team joining attempts over that of the random policy. This result suggests that the learned strategies were also able to effectively decide when to wait, and were able to choose tasks that were more likely to succeed when they did join teams.

Discussion The results of our simulation show that the agent community ultimately learned how to coordinate their team formation decisions. We observed that the agents sometimes effectively "partition" the network into areas, depending on the current conditions, so that agents from one area do not interfere with forming teams from another area. The agents also developed different joining policies that were sensitive to their varying positions in the network. Moreover, the fact that the number of joining attempts did not increase despite the improved performance indicates that agents learned to wait more efficiently. That is, agents do not block other agents with a higher probability of success from joining if they perceive that the conditions are unfavorable.

Although the learned strategy always achieved higher performance than the random strategy, the performance for both strategies also varied depending on the underlying network structure. The values for organizational performance qualitatively matched previously reported results (Gaston & desJardins 2003). Figure 2 shows the average performance of the learned strategies compared to the random policy for each of the network structures. For all network structures, the agents that learned were able to increase organizational performance, demonstrating that it is possible to learn, independent of the social network topology.



(a) lattice (b) small-world (c) random (d) scale-free

Figure 2: Average organizational performance for each of the four network structures.

Conclusion and Future Work

We have demonstrated that agents can learn efficient and effective policies for joining and initiating teams in a real-time, heavily loaded task environment. Additionally, our learning framework allows the agents to significantly improve organizational performance in a wide variety of network structures.

In the future, we plan to further investigate the use of various machine learning techniques in our learning framework. In particular, we are exploring the use of more sophisticated windowing and ensemble techniques for online learning, as well as mechanisms for learning from imbalanced data sets. It would be useful for the designers of agents for large-scale multiagent systems to understand the trade-offs among different learning techniques for team formation and the impact of these techniques on organizational performance and individual agent policies. We also plan to conduct a more in-depth study of the policies that agents learn, with a particular focus on comparing the learned strategies for various social network structures and for different positions within these structures.

References

Abdallah, S., and Lesser, V. 2004. Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology (IAT)*.

Albert, R., and Barabási, A. 2002. Statistical mechanics of complex networks. *Review of Modern Physics* 99(3):7314–7316.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research* 27(4):819–840.

Culler, D.; Estrin, D.; and Srivastava, M. 2004. Overview of sensor networks. *IEEE Computer* 37(8):41–19.

Domingos, P., and Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29:103–130.

Dutta, S., and Sen, S. 2003. Forming stable partnerships. *Cognitive Systems Research* 4(3):211–221.

Erdos, P., and Renyi, A. 1959. On random graphs. *Publicationes Mathematicae* (6):290–297.

Excelente-Toledo, C. B., and Jennings, N. R. 2003. Learning when and how to coordinate. *Web Intelligence and Agent System* 1(3 - 4):203–218.

Gaston, M., and desJardins, M. 2003. Team formation in complex networks. In *Proceedings of the 1st NAACSOS Conference*.

Horling, B.; Benyo, B.; and Lesser, V. 2001. Using self diagnosis to adapt organizational structure. In *Proceedings of the 5th International Conference on Autonomous Agents*.

Hunsberger, L., and Grosz, B. 2000. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*.

Kraus, S.; Shehory, O.; and Taase, G. 2003. Coalition formation with uncertain heterogeneous information. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03)*.

Nair, R.; Tambe, M.; and Marsella, S. 2002. Team formation for reformation. In *Proceedings of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems*.

Nair, R.; Tambe, M.; and Marsella, S. 2003. Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *Proceedings of the Second International Joint Conference on Agents and Multiagent Systems (AAMAS)*.

Pynadath, D., and Tambe, M. 2002. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS '02)*.

Swaminathan, J. M.; Smith, S. F.; and Sadeh, N. M. 1998. Modeling supply chain dynamics: A multiagent approach. *Decision Sciences* 29(3):607–632.

Tambe, M. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research* 7:83–124.

Thadakamalla, H. P.; Raghaven, U. N.; Kumera, S.; and Albert, R. 2004. Survivability of multiagent-based supply networks: A topological perspective. *IEEE Intelligent Systems* 19(5):24–31.

Watts, D., and Strogatz, S. 1998. Collective dynamics of 'small-world' networks. *Nature* 393:440–442.

Witten, I. H., and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann.