Challenges in Building Very Large Teams

Paul Scerri

Carnegie Mellon University
pscerri@cs.cmu.edu

Abstract. When agents coordinate according to the principles of teamwork they can flexibly, robustly and reliably achieve complex goals in complex, dynamic and even hostile environments. An emerging standard for building such teams is via the use of *proxies*, which encapsulate domain independent coordination algorithms in a software module that works closely with a domain specific agent and other proxies to create a team. While succesful, previous generations of proxies and teamwork algorithms were limited to small teams because of their reliance on accurate models of the team and task state. By developing new algorithms that rely on probabilistic models we have been able to build teams several orders of magnitude larger than before. However, key challenges remain before such teams can be deployed in real-world environments, including the need for languages to specify plans for such teams and ways of modeling and predicting team performance in new domains.

1 Introduction

Teamwork allows intelligent, heterogeneous agents (robots, agents and people) to cooperate flexibly, robustly, cohesively and efficiently to achieve complex goals in complex environments. Very large teams are needed for domains such as the military, space operations[10], disaster recovery and commerce, where there are many inter-related, complex activities to be performed by many agents. Typically, the problems for which such teams are most applicable do not require optimal behavior, but require reliably "good" behavior despite an uncertain, complex and potentially hostile environment.

The key technical difference between teamwork and other forms of coordinated behavior is the use of an explicit model of teamwork and an explicit representation of the current team status maintained by each team member. Each team member uses their model of teamwork and the team status to reason about the actions, both actions in the environment and communication acts, that will lead to the best expected outcome for the team. The reasoning an agent performs with its models is derived from early work on the logic of teamwork[5, 11], although practical implementations[30, 14] relax some of the requirements of the logical formulation. The key to building large teams is to find ways for the maintainance of agents' models of team state and reasoning with those models to be scalable.

The key abstraction used in our teamwork models is a *team oriented plan* (TOP). A TOP describes the individual activities, *roles*, that must be performed

to achieve team objectives and any constraints between those individual activities. The team starts with a library of *TOP templates* that are dynamically instantiated when preconditions on the TOP are fulfilled. For example, a TOP template for rescuing an injured civilian in a disaster response domain may have two roles: one for bringing the civilian to safety; and another for administering first aid. A *constraint* would require that the latter role be performed *after* the former. Typically, a large team will be simultaneously executing many TOPs at once. Using the TOP abstraction, the task of coordination via teamwork is to determine what action a team member should take, given the current TOPs and what each team member should communicate about the TOPs. In practice, scalable teamwork requires that each team member does not need to reason and communicate about each TOP, but only some subset.

The "backbone" of our scalable team is an *associates network*, which statically and logically arranges the team into a scale-free network[33]. The team uses the associates network to intelligently share information and dramatically limit the amount of communication required for cohesive activity. The agents exploit the *small-worlds property* of a scale-free network which is that any two agents are separated by a relatively small number of links[33]. Team members can leverage the network to make coordinated decisions by making part of a decision then pushing the decision to its neighbor in the network most likely to be able to continue making the decision. This allows team members to act despite having very limited models of the status of the rest of the team, reducing the need to communicate state information. However, as a consequence the logical guarantees provided by previous teamwork algorithms are replaced by algorithms that have high probability of working correctly. We have developed probabilistic algorithms that leverage the associates network for distributed plan instantiation, role allocation, information sharing and adjustable autonomy with a team.

Our approach to teamwork has been implemented in domain independent software proxies, called Machinetta[26]. Software proxies, which are becoming the standard means for creating heterogeneous teams, are semi-autonomous coordination modules, one for each team member[31]. Each proxy encapsulates the team work algorithms and works closely with an individual team member and with other proxies to implement the teamwork. We have used Machinetta proxies in several domains, including for coordination of large groups of unmanned aerial vehicles.

2 Team Oriented Plans

Team Oriented Plans (TOPs) are the abstraction that define team behavior. The TOPs provide the mapping from team level goals to individual *roles* that are performed by individual team members. Roles are lowest level of abstraction in a team plan and are goals, activities or responsibilities that will most often be performed by a single team member. Often several roles will be required for a particular task. For example, in a disaster response domain, a patient rescue task

may have roles for carrying the front of the stretcher, the back of the stretcher and a role for administering first aid. It is possible (even likely in some domains) that a single team member takes on multiple roles, e.g., one of the stretcher bearers might also administer first aid, but a role is only ever taken on by a single team member.[1]

Importantly, while TOPs describe the breakdown of team activities into individual activities, they do not describe what coordination is required to execute the plan. A TOP does not describe which team member will perform which role or exactly how the role should be performed. This allows the team maximum flexibility to deploy its available resources to execute the current plans. For example, the stretcher bearing roles described above might be filled by robots or humans, depending on their availability and the path taken from the location of the patient to the ambulance will depend on whether robots or humans are carrying the stretcher.

Roles may be performed in whatever way the team member assigned to the role chooses to perform it. However, typically individual roles are not completely decoupled from other roles. The interactions between the roles are represented with *constraints* on the TOP. The team members that are assigned to the roles are informed of the constraints and must coordinate with the team members executing roles with which their roles are constrained to ensure constraints are not violated.

3 The Associates Network: The Backbone of a Large Scale Team

To provide structure to the team, we connect all team members via a static, task independant, logical network. Neighbors in this network are called *associates*. Team members are required to keep their neighbors in the associates network informed of their current activities, including which TOPs they are working on. The associates network can then be used as a backbone for the key coordination algorithms.

Agents performing roles within the TOP must directly communicate with each other. Since agents also keep their neighbors in the associates network informed of their current activities, the group of agents with up-to-date knowledge of progress on a specific TOP consists of the agents actually performing roles in the plan and their neighbors in the associates network. Because the agents performing roles in a particular plan can change over time, the members of the subteam change over time. Moreover, since agents can perform multiple roles and have multiple neighbors, they can be part of several subteams at once. Thus, we can view the structure of the team as a set of dynamic, overlapping subteams. A snapshot of the resulting dynamic, structure is shown in Figure 1. Scalability is possible, because the number of agents needing to know about the plan is a function of the complexity of the TOP, rather than the size of the team.

[1] If a task requires multiple team members doing the same thing, there are multiple roles.

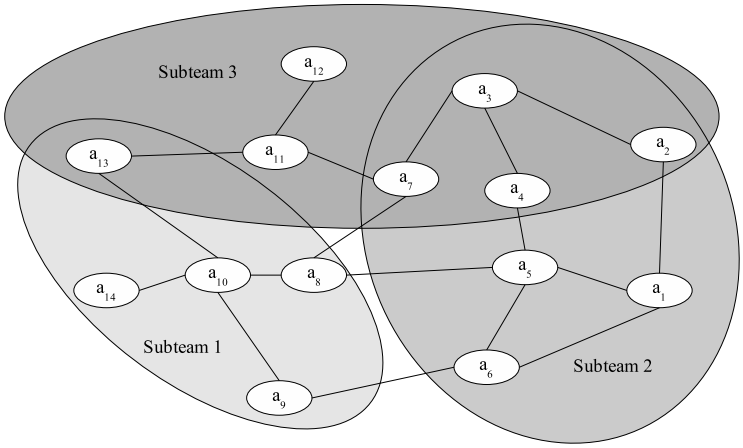


Fig. 1. Relationship between subteams and the associates network

The associates network is a special type of network called a *scale-free network*, which is a type of *small worlds network*[33]. In a small worlds network there is a low *degree of separation* between group members. Having a low degree of separation means that a message can be passed between any two agents via a small number of the point-to-point connections. Such networks are known as *small worlds networks*[33]. Such networks exist among people and are popularized by the notion of "six degrees of separation"[1]. A number of key coordination algorithms, leverage the associates network for scalable teamwork. See [24, 35, 27, 19] for more details.

To illustrate the intuition, consider Milgram's famous experiment with the United States population as a small world's network. In that experiment, a letter needed to be passed from acquaintance to acquaintance to eventually arive at some incompletely described recipient. Consider a letter to be received by "a Pittsburgh botanist who plays the flute" that starts in Los Angeles. If the original recipient knows someone in Pittsburgh (botanist or not), the letter will be passed to them. If the Pittsburgher knows neither a botanist nor a flutist, they might pass the letter to an acquiantance at a university computer science department, believing that their friend may have acquiantances in the music or biology departments that might be able to move the letter to its destination. The process of pushing the letter closer to its destination continues until a recipient is found. According to Milgram's experiment, this taskes on average six steps in a network the size of the United States population.[2] The key property of this example is that no person needed to know enough to get the letter all the way from Los Angeles to the botanist in Pittsburgh, only enough to get it closer to the destination. The analogy to scaling teamwork is that each team member does

[2] More recent experiments have cleaned up some poor experiment design in the original experiment, but the basic conclusion still holds.

not need to know everything that is going on in the team in order for effective coordination to occur. It is the ability to act while having an imcomplete picture of the rest of the team that allows our algorithms to effectively scale.

3.1 Plan Instantiation with the Associates Network

The most common way to introduce TOPs into the team is to create *team plan templates*, with logical conditions that define when the plan template should be instantiated. The TOP templates typically have open parameters which are instantiated with specific domain level information at run time. For example, we can write one template to represent a generic plan of "Fight a fire at building x", rather than writing hundreds of plans of the form "Fight a fire at building 1", "Fight a fire at building 2", "Fight a fire at building 3", etc. When the preconditions of a plan template match the proxy's current state of beliefs, a new plan belief is instantiated with the specific details of the particular precondition match. The TOP includes each plan's termination conditions, under which a team plan is achieved, irrelevant or unachievable. Such explicit specification ensures common knowledge of such conditions, so that the team can terminate the goal coherently.

Notice that this approach to plan instantiation is completely decentralized; there is no imposed hierarchy or centralization. Once TOPs are instantiated, subteams develop around the TOP resulting in a type of structure, however this structure emerges from the interaction with the team and the environment. This *dynamic, partial centralization* is often a feature of large scale coordination infrastructure,[20] likely because it balances the need for flexibility against the structure required for effective coordination though it is not exactly clear why this structure is a consistent feature of large scale teams. By not imposing any particular structure, we allow the team to allocate resources most flexibly to the current situation. Although some structure might lead to some efficiency gains, it is our research hypothesis that eventually no imposed structure will yield the best results.

While the distributed plan instantiation process allows the team to instantiate plans efficiently and robustly, two possible problems can occur. First, the team could instantiate different plans for the same goal, based on different preconditions detected by different members of the team. For example, two different plans could be instantiated by different team members for fighting the same fire depending on what particular team members know or sense. Second, the team may initiate multiple copies of the same plan. In a large team, it is infeasible to require all agents to agree on the plan before execution of the plan can commence, since this may take a long time. Instead, we use a two pronged approach, leveraging the associates network of minimizing the number of conflicting plans created, detecting and then removing the ones that are. More details can be found in [19].

4 Machinetta

A number of algorithms work together to achieve the teamwork, given the framework described above. There are algorithms for allocation roles[9], instantiating plans[19], sharing information[35], human interaction[27] and resource allocation. To avoid requiring a reimplementation of the algorithms for each new domain, the coordination algorithms are encapsulated in a *proxy*[13, 31, 21, 26]. Proxies are becoming a standard mechanism for building heterogeneous teams. Each team member works closely with a single proxy that coordinates with the other proxies to implement the teamwork. The basic architecture is shown in Figure 2. The proxy communicates via a high level, domain specific protocol with the robot, agent or person it is representing in the team. Most of the proxy code is domain independent and can be readily used in a variety of domains requiring distributed control. Our current proxy code, known as Machinetta, is a substantially extended and updated version of the TEAMCORE proxy code[31]. Machinetta proxies are in the public domain and can be downloaded from http://teamcore.usc.edu/doc/Machinetta.

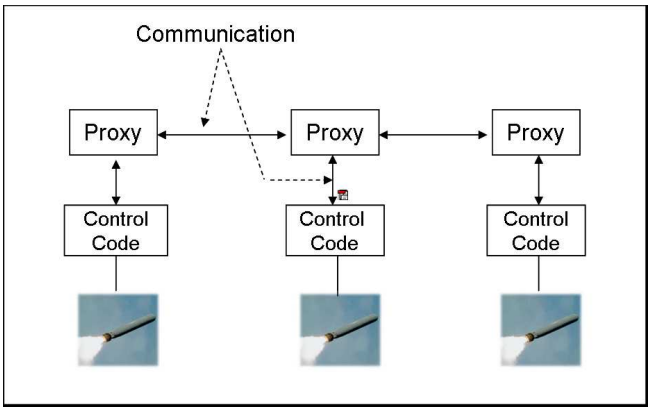


Fig. 2. The basic system architecture showing proxies, control code and Unmanned Aerial Vehicles (UAVs) being controlled.

Coordination Agents In a dynamic, distributed system, protocols for performing coordination need to be extremely robust. When we scale the size of a team to hundreds of agents, this becomes more of an issue than simply writing bug-free code. Instead we need abstractions and designs that promote robustness. Towards this end, we are encapsulating "chunks" of coordination in *coordination agents*. Each coordination agent manages one specific piece of the overall coordination. When control over that piece of coordination moves from one proxy to

another proxy, the coordination agent moves from proxy to proxy, taking with it any relevant state information. We have coordination agents for each plan or subplan (PlanAgents), each role (RoleAgents) and each piece of information that needs to be shared (InformationAgents). For example, a RoleAgent looks after everything to do with a specific role. This encapsulation makes it far easier to build robust coordination.

Coordination agents manage the coordination in the network of proxies. Thus, the proxy can be viewed simply as a mobile agent platform that facilitates the functioning of the coordination agents. However, the proxies play the additional important role of providing and storing local information. We divide the information stored by the proxies into two categories, domain specific knowledge, K, and the coordination knowledge of the proxy, CK. K is the information this proxy knows about the state of the environment. For example, the proxy for a UAV knows its own location and fuel level as well as the the location of some targets. This information comes both from local sensors, reported via the domain agent, and from coordination agents (specifically InformationAgents, see below) that arrive at the proxy. CK is what the proxy knows about the state of the team and the coordination the team is involved in. For example, CK includes the known team plans, some knowledge about which team member is performing which role and the TOP templates. At the most abstract level, the activities of the coordination agents involve moving around the proxy network adding and changing information in C and CK for each agent. The content of K as it pertains to the local proxy, e.g., roles for the local proxy, govern the behavior of that team member. The details of how a role is executed by the control agent, i.e., the UAV, are domain (and even team member) dependant.

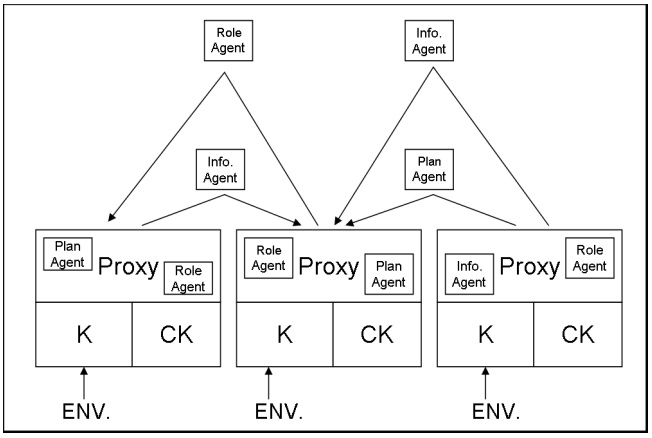
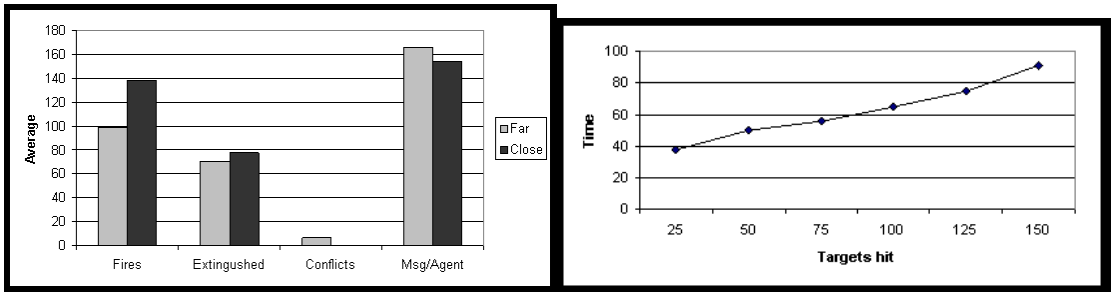


Fig. 3. High level view of the implementation, with coordination agents moving around a network of proxies.

5 Results



(a) **(b)**

Fig. 4. Coordinating 200 agents in (a) disaster response simulation (average on y-axis, fires, extinguished, conflicts and messages per agent on x-axis); and (b) simulated UAVs in a battlespace (time on y-axis, targets hit on x-axis).

Due to limited space, we present only some high level experiments. For more deailed experiments see [28, 25]. In Figures 4(a) and (b), we show the results of an experiment using 200 Machinetta proxies. These experiments represent high fidelity tests of the coordination algorithms and illustrate the overall effectiveness of the approach. In the first experiment, the proxies control fire trucks responding to an urban disaster. The trucks must travel around an environment, locate fires (which spread if they are not extinguished) and extinguish them. The top level goal of the team, G, was to put out all the fires. A single plan requires that an individual fire be put out. In this experiment, the plan had only one role which was to put out the fire. We varied the sensing range of the fire trucks ("Far" and "Close") and measured some key parameters. The most critical thing to note is that the approach was successful in coordinating a very large team. The first column compares the number of fires started. The "Close" sensing team required more searching to find fires, and as a result, unsurprisingly, the fires spread more. However, they were able extinguish them slightly faster than the "Far" sensing team, partly because the "Far" sensing team wasted resources in situations where there were two plans for the same fire (see Column 3, "Conflicts"). Although these conflicts were resolved it took an nontrivial amount of time and slightly lowered the team's ability to fight fires. Resolving conflicts also increased the number of messages required (see Column 4), though most of the differences in the number of messages can be attributed to more fire fighters sensing fires and spreading that information. The experiment showed that the overall number of messages required to effectively coordinate the team was extremely low, partially due to the fact that no low level coordination between agents were required (since one fire truck per plan). Figure 4(b) shows high level results from a second domain using exactly the same proxy code. The graph shows the rate at which 200 simulated unmanned aerial vehicles, coordinated with Machinetta proxies,

searched a battle space and destroyed targets. Taken together, the experiments in the two domains show not only that our approach is effective at coordinating very large teams but also suggests that it is reasonably general.

6 Open Problems

While significant progress has been made in developing large scale teams, significant challenges remain. In this section we briefly discuss a selection of key challenges.

Over a long period of time good algorithms have been developed to solve the problems of task allocation, resource allocation and scheduling for a variety of domains. However, most of the algorithms treat these three problems independently which can be problematic. For example, an appropriate task allocation may depend on the scheduling constraints entailed by that allocation. In the most general case, integrating scheduling, task allocation and resource allocation essentially solves the mutliagent planning problem, hence it is unlikely good general solutions can be found. However, algorithms for each of the problems that can make the other problems simpler will improve the performance of teams.

As teams get bigger, specifying the required behavior becomes more difficult and time consuming. Emergent behavior and unforeseen circumstances also become more common in larger teams. We believe that fundametally new abstractions are required to effectively specify the required behavior of large teams. As well as things that all team programming languages must encode, a language for large teams must be able to encode things like preferences for tradeoffs, what types of emergent behavior are undesirable and whether fast or good coordination is more important in a particular situation.

Simulations of large scale coordination in a particular domain can provide some confidence that behavior is appropriate, however it is not feasible to do enough simulations to ensure that there are not particular domain circumstances that will not induce very poor behavior. Thus, we require analytic techniques for predicting and understanding coordinated behavior. Unfortunately, no mathematical techniques are currently available for this task.

When there are errors in coordination code, it is often very difficult to even detect that something is going wrong, in part because it is difficult to know what to expect. Once a problem is found, it can be extremely difficult to track the problem down, especially when interactions between agents caused the problem. Before large teams requiring high reliability can be developed, techniques will need to be develop to allow developers to effectively find and fix bugs.

Finally, evaluation of multiagent systems and teamwork remains an ad hoc and somewhat weak process. While evaluation by proof of concept, i.e., success in a particular domain, shows that an approach works, it does not allow effective objective comparison of competing techniques. Without the ability to objectively compare approaches to coordination, the rate of progress will be limited.

7 Related Work

Coordination of distributed entities is an extensively studied problem[5, 4, 15, 18, 29]. A key design decision is how the control is distributed among the group members. Solutions range from completely centralized[8], to hierarchical[7, 12] to completely decentralized[34]. While there is not yet definitive, empirical evidence of the strengths and weaknesses of each type of architecture, it is generally considered that centralized coordination can lead to behavior that is closer to optimal, but more distributed coordination is more robust to failures of communications and individual nodes[2]. Creating distributed groups of cooperative autonomous agents and robots that must cooperate in dynamic and hostile environments is a huge challenge that has attracted much attention from the research community[16, 17]. Using a wide range of ideas, researchers have had moderate success in building and understanding flexible and robust teams that can effectively act towards their joint goals[3, 6, 13, 23].

Tidhar [32] used the term "team-oriented programming" to describe a conceptual framework for specifying team behaviors based on mutual beliefs and joint plans, coupled with organizational structures. His framework also addressed the issue of team selection [32] — team selection matches the "skills" required for executing a team plan against agents that have those skills. Jennings's GRATE* [13] uses a teamwork module, implementing a model of cooperation based on the joint intentions framework. Each agent has its own *cooperation level* module that negotiates involvement in a joint task and maintains information about its own and other agents' involvement in joint goals. The Electric Elves project was the first human-agent collaboration architecture to include both proxies and humans in a complex environment[3]. COLLAGEN [22] uses a proxy architecture for collaboration between a single agent and user. While these teams have been successful, they have consisted of at most 20 team members and will not easily scale to larger teams.

8 Conclusions

In this paper, we have presented an approach to developing very large teams that overcomes the limitations of previous team work algorithms via the use of an associates network. The algorithms we have developed for scalable teamwork are encapsulated in domain independant proxies that have been used to demonstrate effective teamwork with 200 team members in two distinct domains. While progress has been made, significant challenges must be overcome before large teams can be reliably deployed. If these challenges can be overcome, large scale teamwork will revolutionize the way some very complex tasks will be completed, improving efficiency, safety and robustness.

Acknowledgments

This research was supported by AFSOR grant F49620-01-1-0542 and AFRL/MNK grant F08630-03-1-0005. Many people have contributed to this work including

Milind Tambe, David Pynadath, Nathan Schurr, Steven Okamoto, Alessandro Farinelli, Katia Sycara, Mike Lewis, Yang Xu, Elizabeth Liao and Bin Yu.

References

1. Albert-Laszla Barabasi and Eric Bonabeau. Scale free networks. *Scientific American*, pages 60–69, May 2003.
2. Johanna Bryson. Hierarchy and sequence vs. full parallelism in action selection. In *Intelligent Virtual Agents 2*, pages 113–125, 1999.
3. Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, and Milind Tambe. Electric Elves: Agent technology for supporting human organizations. *AI Magazine*, 23(2):11–24, 2002.
4. D. Cockburn and N. Jennings. *Foundations of Distributed Artificial Intelligence*, chapter ARCHON: A Distributed Artificial Intelligence System For Industrial Applications, pages 319–344. Wiley, 1996.
5. Philip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.
6. K. Decker and J. Li. Coordinated hospital patient scheduling. In *Proceedings of the 1998 International Conference on Multi-Agent Systems (ICMAS'98)*, pages 104–111, Paris, July 1998.
7. Vincent Decugis and Jacques Ferber. Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In *Proceedings of the Second International Conference on Autonomous Agents*, 1998.
8. T. Estlin, T. Mann, A. Gray, G. Rapideau, R. Castano, S. Chein, and E. Mjolsness. An integrated system for multi-rover scientific exploration. In *Proceedings of AAAI'99*, 1999.
9. Alessandro Farinelli, Paul Scerri, and Milind Tambe. Building large-scale robot systems: Distributed role assignment in dynamic, uncertain domains. In *Proceedings of Workshop on Representations and Approaches for Time-Critical Decentralized Resource, Role and Task Allocation*, 2003.
10. Dani Goldberg, Vincent Cicirello, M Bernardine Dias, Reid Simmons, Stephen Smith, and Anthony (Tony) Stentz. Market-based multi-robot planning in a distributed layered architecture. In *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, volume 2, pages 27–38. Kluwer Academic Publishers, 2003.
11. Barbara Grosz and Sarit Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996".
12. Bryan Horling, Roger Mailler, Mark Sims, and Victor Lesser. Using and maintaining organization in a large-scale distributed sensor network. In *In Proceedings of the Workshop on Autonomy, Delegation, and Control (AAMAS03)*, 2003.
13. N. Jennings. The archon systems and its applications. Project Report, 1995.
14. N. R. Jennings. Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *Intl. Journal of Intelligent and Cooperative Information Systems*, 2(3):289–318, 1993.
15. David Kinny. The distributed multi-agent reasoning system architecture and language specification. Technical report, Australian Artificial intelligence institute, Melbourne, Australia, 1993.
16. Hiraoki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, , and Hitoshi Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

17. John Laird, Randolph Jones, and Paul Nielsen. Coordinated behavior of computer generated forces in TacAir-Soar. In *Proceedings of the fourth conference on computer generated forces and behavioral representation*, pages 325–332, Orlando, Florida, 1994.
18. V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S. Zhang. The UMASS intelligent home project. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 291–298, Seattle, USA, 1999.
19. E. Liao, P. Scerri, and K. Sycara. A framework for very large teams. In *AAMAS'04 Workshop on Coalitions and Teams*, 2004.
20. Regis Vincent Paul Scerri and Roger Mailler, editors. *Proceedings of AAMAS'04 Workshop on Challenges in the Coordination of Large Scale MultiAgent Systems*, 2004.
21. David Pynadath and Milind Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, 2002.
22. C. Rich and C. Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the International Conference on Autonomous Agents (Agents'97)"*, 1997.
23. P. Rybski, S. Stoeter, M. Erickson, M. Gini, D. Hougen, and N. Papanikolopoulos. A team of robotic agents for surveillance. In *Proceedings of the fourth international conference on autonomous agents*, pages 9–16, 2000.
24. P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating roles in extreme teams. In *Proceedings of AAMAS'04, Poster Presentation*, 2004.
25. P. Scerri, E. Liao, Yang. Xu, M. Lewis, G. Lai, and K. Sycara. *Theory and Algorithms for Cooperative Systems*, chapter Coordinating very large groups of wide area search munitions. World Scientific Publishing, 2004.
26. P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.
27. P. Scerri, K. Sycara, and M Tambe. Adjustable autonomy in the context of coordination. In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004. Invited Paper.
28. P. Scerri, Yang. Xu, E. Liao, J. Lai, and K. Sycara. Scaling teamwork to very large teams. In *Proceedings of AAMAS'04*, 2004.
29. Munindar Singh. Developing formal specifications to coordinate hetrogeneous agents. In *Proceedings of third international conference on multiagent systems*, pages 261–268, 1998.
30. Milind Tambe. Agent architectures for flexible, practical teamwork. *National Conference on AI (AAAI97)*, pages 22–28, 1997.
31. Milind Tambe, Wei-Min Shen, Maja Mataric, David Pynadath, Dani Goldberg, Pragnesh Jay Modi, Zhun Qiu, and Behnam Salemi. Teamwork in cyberspace: using TEAMCORE to make agents team-ready. In *AAAI Spring Symposium on agents in cyberspace*, 1999.
32. G. Tidhar, A.S. Rao, and E.A. Sonenberg. Guided team selection. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.
33. Duncan Watts and Steven Strogatz. Collective dynamics of small world networks. *Nature*, 393:440–442, 1998.

34. Tony White and Bernard Pagurek. Towards multi swarm problem solving in networks. In *Proceedings of the International conference on multi-agent systems*, pages 333–340, Paris, July 1998.
35. Y. Xu, M. Lewis, K. Sycara, and P. Scerri. Information sharing in very large teams. In *In AAMAS'04 Workshop on Challenges in Coordination of Large Scale MultiAgent Systems*, 2004.