

Getting Robots, Agents and People to Cooperate: An Initial Report

Paul Scerri, Lewis Johnson, David V. Pynadath, Paul Rosenbloom, Nathan Schurr, Mei Si and Milind Tambe

Information Sciences Institute and Computer Science Department

University of Southern California

4676 Admiralty Way, Marina del Rey, CA 90292

{scerri,johnson,pynadath,rosenbloom,schurr,meisi,tambe}@isi.edu

Abstract

Combining the unique capabilities of robots, agents and people (RAPs) promises to improve the safety, efficiency, reliability and cost at which some goals can be achieved, while allowing the achievement of other goals not previously achievable. Despite their heterogeneity, and indeed, because of their heterogeneity, our key hypothesis is that in order for RAPs to work together effectively, they must work as a team: they should be aware of the overall goals of the team, and coordinate their activities with their fellow team members in order to further the team's goals. This poses a challenge, since different RAP entities may have differing social abilities and hence differing abilities to coordinate with their teammates.

To construct such RAP teams, we make the RAPs "team ready" by providing each of them with teamwork-enabled proxies. While proxy-based architectures have been explored earlier for flexible multiagent teamwork, their application to RAP teams exposes two weaknesses. To address the problems with existing role allocation algorithms for RAP teams operating in dynamic environments, we provide a new approach for highly flexible role allocation and reallocation. Second, we enrich the communication between RAPs and between a RAP and its proxy, to improve teamwork flexibility while limiting the number of messages exchanged. This paper discusses the proxy based architecture and our initial attempts at developing algorithms that address the problems that emerge when the RAP teams are used in complex domains.

Introduction

Groups of Robots, software Agents and People (RAPs) working together promise to revolutionize the way some tasks are performed. In domains such as disaster recovery, military operations, commercial organizations (Rich and Sidner 1997) and industrial plants (Musliner and Krebsbach 1999), collaborative groups of RAPs can make activities safer, more efficient, more reliable and less costly by selectively leveraging the unique capabilities of each type of RAP. In this paper, we present our initial efforts at addressing the challenge of getting such groups to work together effectively.

RAPs have differing levels of social ability, ranging from very socially capable people to completely autistic robots

and agents. This heterogeneity poses difficult challenges for making the RAPs coordinate. Among these challenges are the needs to dynamically form large-scale groups out of these diverse entities, assign tasks to them according to their various capabilities, plan large distributed activities, monitor progress to detect critical changes to the state of the group's execution, and modify and adapt the group in response to such changes so as to ensure robust performance. Our objective is to develop new forms of highly effective, integrated, flexible teams composed of robots, software agents and people.

For these highly heterogeneous RAPs to work effectively towards their shared goals, we are organizing the RAPs as *teams*. Engaging in teamwork imposes constraints on the team members, e.g., informing other team members when there are successes or failures (Cohen and Levesque 1991; Grosz and Kraus 1996; Tambe 1997), which lead to desirable properties of the group behavior, e.g., robustness. To implement teamwork, we are adapting an existing coordination protocol, called STEAM (Tambe 1997). The resulting teams will be able to take advantage of the unique capabilities and strengths of their heterogeneous team members, and avoid having the limitations of individual team members hamper the effectiveness of other team members or of the team as a whole. Also building on previous work, we are developing a framework in which each team member is provided with a common core set of communication and coordination capabilities, in the form of a semi-autonomous *proxy agent* working in close cooperation with them (Tambe *et al.* 2000).

Our current domain of interest is urban disaster recovery, in particular emergency response to an earthquake. We envision large teams of humans, agents and robots fulfilling a wide variety of roles in the response effort. Our experimental platform is partially simulated and partially in the physical world. Figure 1 shows the scenario and the various RAPs that are involved. We simulate the city with the RoboCup Rescue simulation environment (Kitano *et al.* 1999). Agents control fire trucks, driving around the city locating and attempting to put out fires. A human plays the role of fire chief. If a burning building has trapped civilians, a robot and paramedic can be sent to the building to attempt to rescue those civilians. The building search and rescue is performed in a physical environment. The paramedic is



Figure 1: Configuration of experimental platform. Heavy black lines show communication channels between proxies. The dotted line around the bottom left corner shows the division between what is simulated and what is real.

equipped with an iPaq, on which the proxy runs, and can communicate with an iPaq on the robot and with the fire chief. The robot is capable of finding people and leading them to safety. If it finds injured people, it has a microphone and speaker which would allow the paramedic to communicate with the injured person. While proxies and teamwork models deal with some of the key issues in supporting effective heterogeneous teams, scaling up to more complex and more heterogeneous teams exposes new problems. In this paper, we look in detail at two specific problems: role allocation and communication; and how these problems have been addressed.

The first issue we address is how to allocate and reallocate roles to team members. In earlier versions of our proxies (Tambe *et al.* 2000), RAPs would volunteer for roles, with the first volunteer getting the role. Later, to improve the allocation of roles to RAPs we added a mechanism for the team to auction off roles. Neither of these mechanisms scale to larger teams, with many roles in many plans, that we are now considering. Giving the role to the first volunteer can lead to a poor assignment of RAPs to roles and auctions are not feasible for large teams in distributed environments, since participating in a large number of auctions poses too high a burden on proxies and communication channels. Other coordination protocols typically do not have mechanisms for dynamic role allocation. Even if a coordination mechanism allows role reallocations they are typically considered only when a RAP has completely failed to perform the role. This precludes the team from taking advantage of additional or freed up team members or changing allocations to suit emerging circumstances. We are now developing a layered

role allocation algorithm that addresses the above problems. The key idea is to provide multiple mechanisms for role allocation that the proxies and RAPs choose between depending on the circumstances. Each role is initially assigned to a RAP but can be exchanged or reallocated in a distributed manner. If distributed role reallocation does not lead to a good allocation of RAPs to roles, then a group of roles can be assigned by a single member of the team. These layers of role allocation allow the team to robustly and quickly find and maintain an effective assignment of RAPs to roles.

The second issue addressed in this paper is when, how and what to communicate. There are two types of communication which we consider separately: communication between RAP/proxy pairs and communication between a RAP and its proxy. Reasoning about communication between a proxy and its RAP builds on our previous work on adjustable autonomy (Scerri *et al.* 2002). In general, the problem of reasoning about when and what to communicate between RAPs is highly intractable (Pynadath and Tambe 2002) and approximate solutions are required. While STEAM communication reasoning has been shown to perform better than communication reasoning in other coordination mechanisms (Pynadath and Tambe 2002), iohs twe 7TEAM communicatl cabou-

(m)-o(m)-t(c)h(m)-e(p)-1nt 6AP2-36co(m)-6(m415(pl)5eca

even more interesting. Different types of RAPs have different abilities to use information and different abilities to produce information in a form useful for other RAPs. For example, an agent that plans routes has likely no need for information about the context in which that route was requested. People can use far richer context information about the status of the team to adapt their own behavior to help the team function more effectively, though care must be taken not to overwhelm people with information. Conversely, the amount of information each RAP can provide and the cost of providing that information varies. For example, robots are likely to have much less accurate information about their own context than are people, but people may not be able to communicate information in a way that a robot can easily understand. Thus, the heterogeneity of the RAPs leads to a need for careful reasoning about which information should be requested from a RAP and where that information is sent. The second key limitation of STEAM is that it relies on explicit, static specifications of the cost of miscoordination and the probability of this miscoordination occurring. In complex environments, with potentially many interacting plans, specification of these costs and probabilities is not feasible. To address the issues with previous communication reasoning, we are casting the problem of what to communicate as a decision theoretic planning problem which we solve with COM-MDPs (Pynadath and Tambe 2002).

RAP Team Architecture

The foundation for creating robust, highly heterogeneous teams is an architecture that uses semi-autonomous *proxy agents* to create an homogeneous coordination layer “above” the highly heterogeneous RAPs (Tambe *et al.* 1999). By providing each RAP with a proxy that has teamwork knowledge we raise the coordination ability of the RAP-proxy pair to the level at which the rest of the team is operating. Via the use of adjustable autonomy (Mulsiner and Pell 1999), the amount of coordination ability that each proxy actually provides (and the amount which the RAP provides) is dynamically adjusted to the particular context of the RAP at that point in time. The proxies manage the coordination, performing the routine operations that are required for co-operation; e.g., informing others when plans are completed, and assisting in the handling of exceptional situations; e.g., finding RAPs to fulfill roles due to failures or overloading. The proxies can assist in making adjustments to the plan the group is following, when required. The proxies also perform more routine tasks, such as information sharing, that ensure the continued smooth operation of the team, while freeing the RAP from engaging in these activities. The proxies facilitate the coordination among RAPs, while adjustable autonomy ensures that RAPs can have input into coordination decisions when required.

Teams of RAP-proxy pairs implement *team-oriented programs* (TOPs). TOPs are high-level descriptions of the activities to be performed by the team, expressed as abstract team plans. With team-oriented programming, we specify the plans that a team may need to perform jointly, and the inter-dependencies between those plans; but are spared the effort of specifying all of the coordination and communi-

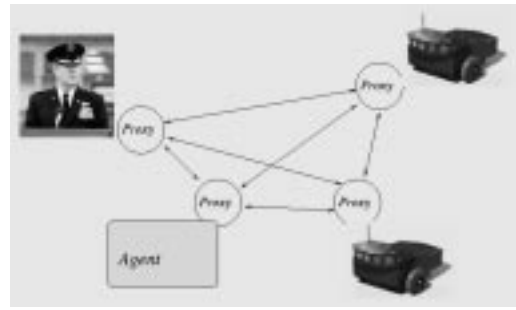


Figure 2: An example of a proxy based RAP team.

cation. Such programs identify the domain capabilities required to perform each task, which helps in determining how best to allocate tasks to team members. There may be a small number of different plans (and sub-plans) to be followed if a goal is to be achieved under different circumstances. The proxies execute a plan by allocating relevant plan steps (i.e., tasks) to RAPs that will perform the plan steps. The proxies can make small changes to the plan and dynamically change assignments of tasks to RAPs to better achieve the goals. Because the team plan is abstract, the individual RAPs are free to carry out their assigned tasks as they see fit, based upon their own domain knowledge and capabilities.

When operating in complex, dynamic and unpredictable environments the coordination among RAPs must be robust and ductile. As a mechanism, teamwork provides these characteristics, in part by performing much of the coordination reasoning online, in reaction to the specific state of the team. For example, RAPs (or their proxies) decide which plan to follow depending on the situation and decide what to communicate depending on what they believe other team members know. The flexibility inherent in teamwork has an interesting, useful impact on the reasoning required of each RAP. Team members are not rigidly coordinated with their teammates and actively work at staying coordinated. This reduces the need to perform exactly as a team mate expected a priori, giving a team member more freedom to perform their own role as the situation demands. This is especially important when people are members of the team, since people are very good (relative to agents and robots) at assessing situations and actively flexibly and are also unlikely to be satisfied performing very rigidly defined roles. To allow this level of flexibility in the team, we have sometimes used multiple algorithms for the same coordination problem. For example, allocation of RAPs to roles can be performed statically at design time, dynamically in a distributed manner or dynamically in a centralized way. The team chooses between role allocation techniques depending on the circumstances and may even use more than one algorithm for a single role allocation problem.

Proxies for RAPs

Our proxies are lightweight, domain independent pieces of software, capable of performing the activities required to work cooperatively within a larger team on some abstract

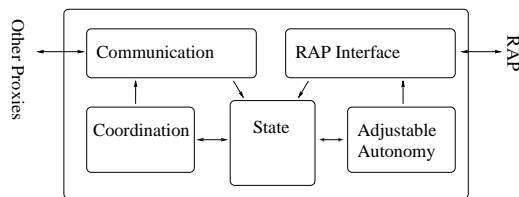


Figure 3: Proxy software architecture.

team plan(s). Each proxy works closely with a single RAP and in cooperation with other proxies. A proxy can be thought of as a kind of a “personal assistant” for the RAP, acting on behalf of the RAP in the group activities. The proxy is implemented in Java and is designed to run on a number of platforms including laptops and hand-held devices.

The proxy software is made up of five components: communication, coordination, state, adjustable autonomy and RAP interface. Each component in the architecture abstracts away details allowing other components to work without considering those details. For example, the RAP interface component deals with the details of the type of RAP and the media with which to interact with the RAP, while the coordination component deals with RAPs as abstract entities having heterogeneous capabilities. The abstraction the component provides for other components is independent of both the details of its internal reasoning and of the peculiarities of the components it interacts with. For example, the communication component will be tailored to the medium being used to communicate, e.g., wireless or wired, but the coordination component will only be told available bandwidth and cost of communication. Likewise, the adjustable autonomy component uses the RAP interface component as an abstract tool for sending and receiving messages from the RAP. It will know abstract characteristics, like how costly a communication is and the probability the RAP will respond, but not know whether the RAP interface is inferring responses from an analysis of a person’s emotional reaction or directly altering the memory of a robot.

An emerging, unanswered research question is how much knowledge of both the RAP and the current state of the domain the proxy needs in order to perform its job effectively. Our current implementation is attempting to minimize the detailed knowledge used by the proxy, with future testing aimed at showing whether more knowledge is required.

An Initial Approach to Role Allocation Using Layers

The role allocation algorithm is responsible for assigning RAPs to roles within a team plan and reallocating RAPs to roles when the situation or plans change (Nair *et al.* 2002). Given the complex, dynamic environments in which the RAP teams will act, reallocations of roles in response to failures, difficulties or opportunities will be the norm, rather than the exception. Our preliminary approach to role allocation and reallocation uses three “layers” of algorithms

in order to maximize flexibility and robustness. First, the TOP provides an initial allocation of RAPs to roles. This static allocation does a simple, reasonable allocation. The initial allocation may be an allocation of a the role to a particular RAP or it may be a simple “rule” for finding a RAP, e.g., “*The RAP detecting the fulfillment of the plan pre-condition*” or “*The physically closest, capable RAP*”. For example, when a role to help an injured patient is triggered by a (human) paramedic finding the patient, a reasonable initial allocation of the role will be to allocate it to the paramedic. In some domains, this simple static allocation of roles may all that is required much of the time.

Second, the RAP that is currently allocated to a role can attempt to find another RAP to perform the role. This distributed role reallocation allows simple switching of roles to improve performance or recover from failure. For example, if a mobile robot found an injured patient that it could not help autonomously, it might offer that role to an available (human) paramedic, perhaps then taking over that paramedic’s current role of searching another part of the building. Such role reallocations allow small sub-groups of RAPs to leverage each others capabilities in an efficient manner.

Finally, the responsibility for reallocating a role can itself be transferred. This mechanism allows groups of roles that need to be reallocated to be “centralized” for a more globally efficient allocation. If the proxies detect that there are several roles, which the same entities are simultaneously being asked to perform, they can transfer responsibility for allocating all roles to some central point. For example, if several injured patients were found in a short period of time, the allocation of paramedics to patients could be performed by a central command center, perhaps minimizing the distance each paramedic needed to travel. After this centralized role reallocation has been performed, further local role reallocations (i.e., the second type, described above) could be performed to address small problems with the centralized allocation. For example, particular paramedics allocated to particular patients as a part of a larger allocation might find themselves dealing with a patient for which they do not have appropriate equipment. By performing a local role reallocation with a nearby paramedic, the injured patients could be better cared for. The combination of “centralized” role allocation and subsequent “distributed” reallocation reduces both the need for the central role allocation to know all details and make an optimal allocation and the need to for extensive local negotiations. Thus, each role allocation algorithm complements the other algorithms and provides an additional layer of flexibility to handle a type of role allocation problem efficiently.

Preliminary Thought on Communication Algorithms

When a large group of RAPs and proxies attempts to act in a coordinated manner in a distributed, unpredictable and dynamic environment there is a large volume of information which there is some use to communicate. However, there are two important reasons why all this information cannot

and should not be communicated. Most simply, there is too much information to communicate everything, especially when there are a large number of RAPs and some are using relatively low bandwidth communication media. The second, perhaps more interesting reason, is that it is important not to overwhelm RAPs with information or with requests for information. Below we describe in more detail the algorithms for inter-RAP communication and RAP-proxy communication and how they balance the desire for information with the need not to overwhelm communication media or RAPs.

Inter-RAP Communication

The coordination module makes all of the decisions regarding inter-proxy communication in an attempt to ensure proper team behavior. It seeks to balance the need to communicate relevant information in a timely manner, against the need to avoid overburdening the team members with an overload of messages to process. In general, the problem of choosing the best messages to send and the best times to send them is NEXP-complete (Pynadath and Tambe 2002). Given this prohibitive complexity, we instead focus on making approximately optimal decisions in a manner which is computationally feasible within our target domains, but which still attains reasonable team performance.

Our first step in achieving practical, approximately optimal coordination is to reduce the communication decisions to binary choices on new beliefs. Thus, when a proxy receives a new piece of information (e.g., from an observation made by its RAP), it can potentially communicate that information to its teammates; otherwise, it never communicates the information at all. We thus ignore the possibility of communicating the information at some later time, which prevents realization of optimal communication in certain domains (e.g., the example domain of (Pynadath and Tambe 2002)). However, many existing coordination algorithms have found success within this restricted space of immediate communication (Tambe 1997; Jennings 1995). Furthermore, we gain significant computational savings by so restricting our space (Pynadath and Tambe 2002). The end result is that our proxy architecture implements a communication *policy* that maps a new belief into a decision on whether to communicate that belief or not.

Unfortunately, existing communication policies produced by multiagent research will have only limited success in RAP teams. Many coordination algorithms (e.g., STEAM) though successfully applied to real-world domains often display inflexibility that can lead to non-negligible degradation in performance in our intended domains. Furthermore, existing coordination algorithms typically rely on heuristic rules that may not always apply to RAP domains or on user-specified parameters that may overburden RAP system designers. For example, STEAM requires that the TOP contain team plans for each important piece of information to communicate, as well as the costs and likelihood of miscoordination of these team plans. For our RAP proxy architecture, we would instead like a policy that can make more automated decisions on communication of arbitrary beliefs.

Our previous work has identified an algorithm that auto-

matically computes the locally optimal communication policy, and this algorithm was feasible within our original example problem (Pynadath and Tambe 2002). However, this example domain focused on only one belief to be communicated or not communicated. The locally optimal policy's exponential computational requirements will not scale over the very many beliefs relevant in RAP domains. We instead need a new communication policy that is more flexible and automated than existing real-world coordination algorithms, but whose computational complexity remains acceptable as the number of beliefs increases in scale.

We have designed our proxy architecture to allow for easy substitutability of different communication policies, thus providing a testbed for exploring the space of communication policies within our architecture and examining their performance within RAP teams. In addition to this empirical evaluation of policies, we will use our COM-MTDP model to perform *analytical* evaluations of RAP communication policies, similar to work analyzing existing coordination policies from the literature (Pynadath and Tambe 2002). The goal of these evaluations is to provide detailed profiles (perhaps even guarantees) of performance of these policies within RAP domains.

Furthermore, we plan to exploit knowledge of the team's TOP in designing our candidate communication policies. In particular, the team's plan hierarchy can provide scoping information that can localize the necessary decision-making (e.g., certain beliefs may be relevant to only a subset of possible plans). By modeling TOP execution within our COM-MTDP framework, we expect to gain unique insight into the communication requirements of such teams. Given the specialized structure of our TOP, it may even be possible to design a practical communication policy that is optimal within the architectural assumptions of our TOP semantics.

One of the biggest keys to a successful deployment of RAP teams will be dealing with the special case of *people*. People are different from agents and robots in the way they process information, they can potentially utilize context information better than agents and robots and can often do better problem solving. However, people also have special limitations, such as limited memory and less aptitude at things like mathematical calculations. Also, people can get tired and miss information or process it less well. Things they can do well when they are fresh they may do poorly when they are tired or stressed. The workings of the communication algorithm and, especially, the design of the computer interface to the person must be carefully designed to deal with the peculiarities of people. Two key principles are guiding the development of the interface between the proxy and person. First, information should be presented graphically when good spatial metaphors exist for organizing the information, both to present information compactly and to exploit the human visual system's ability to detect patterns. For example, we superimpose information about fires and fire trucks onto a plan view display of the the urban area, to aid the people in reasoning about how to deploy fire trucks to fight the fires. Secondly, the interface needs to intelligently abstract away details, presenting enough information to help people make decisions, but not too much to over-

whelm them.

Adjustable Autonomy Issues

Adjustable autonomy (AA) reasoning decides whether a proxy will make a coordination decision autonomously or whether the decision will be referred to the RAP (Dorais *et al.* 1998; Ferguson *et al.* 1996). For example, AA reasoning would decide whether the proxy can autonomously accept (or decline) responsibility for a role on behalf of the RAP or whether the RAP itself should make that decision.

AA performs two intertwined functions related to providing flexibility in the team. First, AA provides the ability for the proxy to make coordination decisions on behalf of the RAP, thus ensuring all RAPs have an ability to make coordination decisions. Because each RAP has an ability to perform sophisticated coordination reasoning (perhaps only by relying on its proxy) the team can engage in sophisticated coordination. In the case of a person, AA could assign responsibility to the person for coordination decisions, to ensure personal preferences, etc. were adequately taken into account. Notice, that AA does not simply pass all decisions to the RAP if the RAP is capable of making them, but flexibly chooses between autonomous action and transferring autonomy, depending on the situation and the decision.

Second, the fact that the RAPs can make coordination decisions gives them flexibility to go outside the default coordination of the team for arbitrary and not necessarily predefined reasons. This is important for any RAP that might have reasons for wanting coordination done in a particular way. In particular, it allows RAPs to make decisions using information or reasoning capabilities outside the information and capabilities of the proxies. For example, a fire fighter might have some particular limitation that is not modeled by the proxy that makes her incapable of fighting a particular fire. This capability is especially important for people who will have far superior reasoning abilities and access to information than their proxies. In fact, we hypothesize that people will not be comfortable working in RAP teams unless they have a large degree of freedom.

Status

To evaluate our RAP teams we are using a mixture of real-world and simulation environments, as described earlier. We are currently experimenting with scenarios in which a single human fire chief interacts with up to ten fire trucks (controlled by agents). The fire chief is responsible for assisting with role allocations and assigning priorities to fires. Currently, the fire fighting plan is relatively simple (see Figure 4), though reasonably effective. In the near term, we are working on integrating robots and mobile people (with proxies running on hand-held devices) into the scenario.

Conclusion

In this paper, we have presented initial efforts to develop teams of RAPs that can successfully cooperate on shared tasks in complex dynamic environments. When we adapted an approach that used teamwork and proxies two key issues

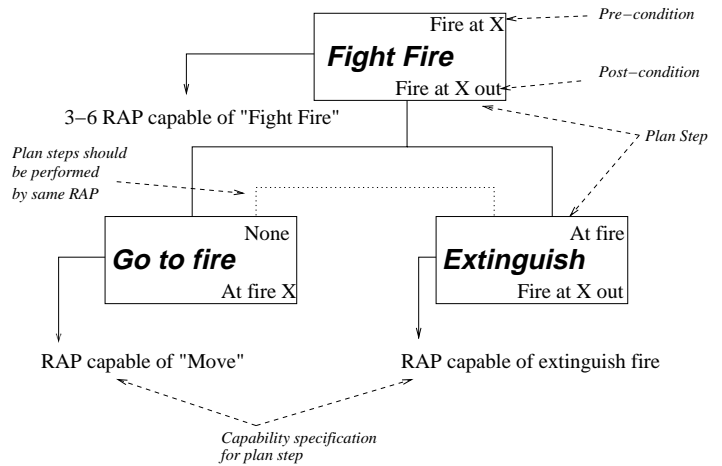


Figure 4: A sample team plan for a fighting fire.

emerged. Firstly, a new role allocation and reallocation algorithm was needed to more flexibly transfer responsibilities for roles within a team plan. Our preliminary efforts employ a layered approach to this role allocation that allows both static, distributed and centralized allocation mechanisms to be used, depending on the situation. Second, communication between a large number of highly heterogeneous entities raises the challenging problem of ensuring adequate information is communicated to where it is required. We are intending to apply sophisticated decision theoretic reasoning to make situation specific decisions about when and what to communicate.

Acknowledgments

This research was supported by DARPA award no.F30602-01-2-0583. We would like to thank Dylan Schmorrow at DARPA and Allen Sears at CNRI. We also thank our colleagues, especially, Guarav Sukhatme, Sameera Poduri and Dennis Wolf.

References

Philip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

G. Dorais, R. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars. In *Proceedings of the First International Conference of the Mars Society*, pages 397–420, August 1998.

G. Ferguson, J. Allen, and B. Miller. TRAINS-95 : Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems*, pages 70–77, May 1996.

Barbara Grosz and Sarit Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.

Nick Jennings. Controlling cooperative problem solving in

industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75:195–240, 1995.

Hiroaki Kitano, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsushi Shinjoh, and Susumu Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Proc. 1999 IEEE Intl. Conf. on Systems, Man and Cybernetics*, volume VI, pages 739–743, Tokyo, October 1999.

Dave Mulsiner and Barney Pell. Call for papers: AAAI spring symposium on adjustable autonomy. www.aaai.org, 1999.

D. Mulsiner and K. Krebsbach. Adjustable autonomy in procedural control for refineries. In *AAAI Spring Symposium on Agents with Adjustable Autonomy*, pages 81–87, Stanford, California, 1999.

R. Nair, T. Ito, M. Tambe, and S. Marsella. Task allocation in robocup rescue simulation domain. In *Proceedings of the International Symposium on RoboCup*, 2002.

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.

C. Rich and C. Sidner. COLLAGEN: When agents collaborate with people. In *Proceedings of the International Conference on Autonomous Agents (Agents'97)*, 1997.

P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 2002.

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.

M. Tambe, D. Pynadath, C. Chauvat, A. Das, and G. Kaminka. Adaptive agent architectures for heterogeneous team members. In *Proceedings of ICMAS'2000*, pages 301–308, 2000.

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