Conflicts in teamwork: Hybrids to the rescue

M. Tambe, E. Bowring, H. Jung*, G. Kaminka+, R. Maheswaran, J. Marecki, P.J.Modi++
R. Nair**, S. Okamoto++, J.P. Pearce, P. Paruchuri, D. Pynadath, P. Scerri++, N. Schurr,
P. Varakantham
University of Southern California, Los Angeles, CA 90089, USA
* Institute for Human-Machine Cognition, Pensacola, FL 32502, USA
+ Bar Ilan University, Ramat Gan, Israel
** Honeywell Laboratories, Minneapolis, MN 55418, USA
++ School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
http://teamcore.usc.edu

ABSTRACT

Today within the AAMAS community, we see at least four competing approaches to building multiagent systems: belief-desire-intention (BDI), distributed constraint optimization (DCOP), distributed POMDPs, and auctions or game-theoretic approaches. While there is exciting progress within each approach, there is a lack of cross-cutting research. This paper highlights hybrid approaches for multiagent teamwork. In particular, for the past decade, the TEAMCORE research group has focused on building agent teams in complex, dynamic domains. While our early work was inspired by BDI, we will present an overview of recent research that uses DCOPs and distributed POMDPs in building agent teams. While DCOP and distributed POMDP algorithms provide promising results, hybrid approaches help us address problems of scalability and expressiveness. For example, in the BDI-POMDP hybrid approach, BDI team plans are exploited to improve POMDP tractability, and POMDPs improve BDI team plan performance. We present some recent results from applying this approach in a Disaster Rescue simulation domain being developed with help from the Los Angeles Fire Department.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Multi Agent Systems

General Terms
Algorithms

Keywords
POMDP, DCOP, BDI, Game theory

1. INTRODUCTION

The long-term goal of our research is to facilitate building heterogeneous teams where teams may be composed of software agents, robots, people, sensors, etc. Such teams may be composed of thousands of members and operate in domains that are dynamic and real-time. While such teamwork is important in several current applications, e.g. virtual environments for training [30, 26], RoboCup robot soccer [29], and office work [25], it is also crucial in several future applications. For instance, in future disaster rescue applications, robots, people and software agents may team up to save civilians without putting rescue personnel in harm's way [24]. Similarly, future collaborations are envisioned between human astronauts and teams of robots and agents in space domains.

The key research challenge is to understand how to rapidly build such teams and enable the teams to coordinate and adapt in their complex, dynamic environments. Today within the AAMAS community, we see at least four competing approaches to building multiagent teams. First, distributed constraint optimization (DCOP) approaches focus on interaction graphs (limited local interactions) among agents [15, 1, 12], and attaining a local or global optimum given that not all agents interact with all other agents [15, 23]. Second, distributed Partially Observable Markov Decision Problems (POMDPs) focus on team coordination in the presence of uncertainty in actions and observations in real-world domains [25, 17, 2]. Third, game-theoretic and auction approaches focus on coordination among self-interested agents using market-oriented mechanisms [11] which may also be applied in team settings. Fourth, BDI approaches, inspired by logic and psychology, are symbolic approaches which arguably enable better human understanding of the methodology employed.

While there has been excellent progress in each of the four approaches outlined above, there is an unfortunate lack of cross-cutting research. This paper highlights some hybrid approaches for multiagent teamwork. As multiagent teams are built for large-scale complex domains, hybrid systems that combine the key approaches outlined above will become essential — this is true in several of the large-scale multiagent applications we have developed (see Section 2).

More importantly, hybrids enable synergistic interactions among the four approaches, allowing them to overcome each other's weaknesses. For example, current BDI team approaches lack tools for quantitative performance analysis

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

under uncertainty. Distributed POMDPs on the other hand are well-suited for such analysis but the complexity of finding optimal policies in such models is highly intractable. Fortunately, with the BDI-POMDP hybrid approach, BDI team plans are exploited to improve POMDP tractability, and POMDPs improve BDI team plan performance. Similarly, a hybrid DCOP-POMDP approach combines the DCOP strength of reasoning about local interactions among agents (within an agent network/graph) with POMDP ability to reason with uncertainty. We outline several such synergistic interactions in this paper.

2. OVERVIEW OF RESEARCH ON TEAMWORK

2.1 Infrastructure and algorithms

Teamwork enables flexible, robust coordination among multiple heterogeneous entities in order to achieve their shared goals. Our previous work has focused on domain-independent, reusable team coordination algorithms to realize flexible teamwork [27]. Proxies encapsulating team coordination algorithms enable a group of "non-team-ready" agents to work together and facilitate the participation of humans in such teams [22, 24].

Teams of proxies execute Team-Oriented Programs (TOPs) [22, 28], abstract plans that provide high-level descriptions of the activities to be performed. TOPs specify the teams joint plans and the inter-dependencies between those plans but do not contain all of the details of coordination and communication required. Indeed, TOPs are intended to provide a high-level abstraction, where the proxies supply the necessary coordination and communication algorithms to ensure that the team executes TOPs in a coordinated fashion, while optimizing team performance. To that end, the proxies encapsulate two key types of algorithms: (i) task allocation algorithms to determine which team members to recruit given a set of tasks (or how to allocate tasks to existing members); (ii) communication algorithms for reasoning about what, when and how to communicate among team members.

The proxies exhibit hybrid reasoning at two levels. First, the team of proxies as a whole engages in hybrid reasoning. While the execution of TOPs implies a BDI architecture [27], the underlying proxy algorithms may involve executing DCOP or distributed POMDP algorithms, as outlined in Section 3. Second, individual algorithms within a single proxy may also engage in hybrid reasoning. For instance, the communication algorithms are themselves hybrids of BDI theories of teamwork [8, 4] and decision-theoretic reasoning [27, 31].

Tables 1 and 2 categorize the algorithms that we have developed for task allocation and communication. For task allocation, Table 1 classifies the algorithms based on: (i) whether the allocation is among agents or between agents and humans (in which case the problem is identified as one of *adjustable autonomy* [25]), and (ii) whether the allocation reasoning is done on-line or off-line. Typically, off-line reasoners use exact algorithms that guarantee optimality, whereas on-line reasoners employ approximate algorithms. For communication, Table 2 categorizes the algorithms based on: (i) whether communication is among agents or between agents and humans and (ii) whether the communication is explicit or implicit. The key point to note is that the algorithms involve DCOP, BDI, game theory and POMDPs, demonstrating the hybrid nature of the proxies. The proxies may be equipped with algorithms from multiple categories e.g. an algorithm for off-line multiagent role allocation complemented with an on-line role reallocation algorithm for dynamic domains. Where there are multiple algorithms within one category, choosing the right one to apply is a matter of tradeoffs that are not yet well-understood — this is an issue for future work.

	Multiagent	Agent-human
Offline optimal	DCOP: ADOPT complete [15], [1] BDI + POMDP hybrid: [16]	POMDPs: Transfer of control policy [32], [25]
Online	DCOP + Games k-coordination incomplete [10] DCOP: LA incomplete [23]	BDI Plans: [26]

Table 1: Task Scheduling/allocation algorithms

	Multiagent	Agent-human
Explicit	BDI + Decision-theoretic filter [22] Distributed POMDPs [25]	
Implicit	BDI: Plan Recognition [6], Distributed POMDPs [17]	Probabilistic plan recognition : Overhearing [5]

Table 2: Communication Algorithms

2.2 Teamwork applications

The reusable infrastructure and algorithms introduced in the previous section have been applied to a wide range of domains. Our early work focused on teams of synthetic pilots flying simulated helicopters for mission rehersal simulations [27] and player teams for RoboCup Soccer simulations [29]. While this early work focused on small-scale homogeneous agent teams in simulated environments, our recent work addresses larger-scale heterogeneous teams. We describe here three recent application domains and our continuing work in these domains.

Personal assistant agents: Individual software agents embedded within an organization can represent each human user in the organization and act on their behalf. These personal assistant agents work together in teams toward service of cooperative tasks. Such agentified organizations could potentially revolutionize the way a variety of tasks are carried out. Agentified organizations may be highly beneficial in domains like disaster rescue, where teams composed of agent-assisted response vehicles, robots and people may enable more rapid crisis response. To this end, we have constructed a software prototype system that leverages the effectiveness of human and agent teams confronted with a disaster.

Personal assistant teams are also be useful in office environments. In an earlier research project called "Electric Elves,", an agent system was deployed at USC and ran continuously for nine months [25]. The resulting small-scale team of 15-20 agents aided in daily tasks such as reschedul-

ing meetings, selecting presenters for research meetings, tracking people and ordering meals. The agents adjusted their decision making autonomy dynamically. Figure 2 shows a human user with a PDA connected to a GPS device that agents used to track the user's location and that allowed mobile communication between the personal assistant and the user. Section 3.1 describes the hybrid approach adopted in these proxies. Partly building on this experience, work has begun on a more comprehensive joint project with SRI International called CALO.

Distributed sensor nets: This domain consists of multiple stationary sensors, each controlled by an independent agent, and targets moving through their sensing range (see Figure 3) [7] [14]. Each sensor is equipped with a doppler radar with multiple sectors. An agent may activate one sector at a time or switch the sensor off. All of the sensor agents must act as a team to cooperatively track the targets. In order for a target to be tracked accurately, multiple agents must concurrently turn on overlapping sectors. There may not be enough sensors to track all possible targets so agents have to sacrifice tracking some lower priority targets in order to ensure that they globally optimize tracking performance. Additionally, sensor readings may be noisy, and the situation may be dynamic with targets moving through the sensing range. Our early work utilized a standard DCOP approach to address the resource allocation problem — allocating sensors to targets — that arises in this domain. While DCOP addresses the locality of agent interactions in this domain, it is unable to address the sensor uncertainty by itself. A hybrid approach that combines DCOPs with POMDPs called ND-POMDPs, promises to address this shortcoming.

3. CASE STUDIES IN HYBRID MODELS

This section provides an overview of five specific projects which employed hybrid approaches and the benefits gained via such hybrids.

3.1 Human-agent task allocation: BDI-POMDP hybrid

Adjustable autonomy refers to entities dynamically varying their own autonomy, transferring decision making control to human users in appropriate situations. The main question in adjustable autonomy is whether and when agents should make autonomous decisions and when they should transfer decision making control to other entities. Previous adjustable autonomy research framed the problem in terms of two choices: either transfer control or take autonomous action. However, in a team context, miscoordination costs may be incurred while an agent waits for input from a user. Thus, with only these two options an agent is forced to either take a risky decision or risk incurring the cost of miscoordination. To reduce this risk, we introduced the notion of a *transfer-of-control strategy* which is a pre-planned sequence of transfer-of-control and deadline delaying actions. For example, a simple strategy might allow the agent to first transfer control to a human but then re-take control if the human fails to respond within a fixed time interval.

Thus, the key adjustable autonomy problem in agent-team settings is to select the right strategy, i.e. the one that provides the benefit of high-quality decisions without risking significant costs in interrupting the user and miscoordination with the team. Furthermore, an agent must select the right strategy despite significant uncertainty about whether the user will respond to a request for input and whether the agent itself can make a correct decision. MDPs are a natural choice for implementing such reasoning because they explicitly represent costs, benefits and uncertainty as well as doing lookahead to examine the potential consequences of sequences of actions [25]. Specifically our MDPs encoded the possible transfer-of-control actions, the decisions an agent could make autonomously and any actions the agent could take to "buy" more time for user input (typically at some cost.) The optimal policy of the MDP corresponded to both the optimal transfer-of-control strategy and the appropriate decision for an agent to make should it decide to act autonomously.

The usefulness of the hybrids is seen in that we are not using MDPs to solve the whole team coordination problem. The team is actually executing a TOP, where communications among team members is controlled by BDI coordination [27] [25]. It is while executing a single task or a role in service of executing this TOP, that MDPs get employed to find optimal transfer-of-control strategies. Thus, instead of a complex MDP that reasons about all team coordination, MDPs are restricted to specific tasks, reducing their state-space and enabling easy computation of the optimal policy.

Though MDPs provide for sequential decision making in the presence of transitional uncertainty, they are hampered in not being able to handle observational uncertainty. A natural step to address this issue is to use partially observable MDPs (or POMDPs) to model the adjustable autonomy problem. However, existing techniques for solving POMDPs either provide loose quality guarantees on the solutions (approximate algorithms) or are computationally very expen-

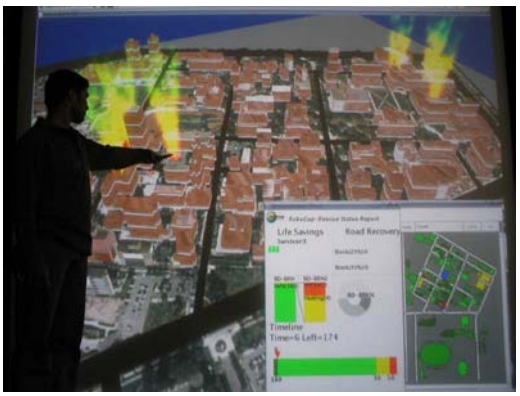

Figure 1: Disaster rescue simulations.

Figure 2: Electric elves domain.

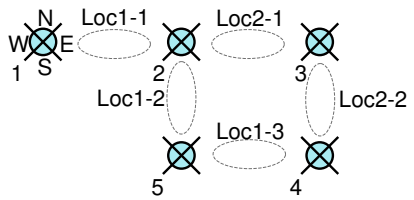

Figure 3: Sensor nets

	Individually Observable	Collectively Observable	Collectively Partially Observable
No Comm.	P-complete	NEXP-complete	NEXP-complete
Gen. Comm.	P-complete	NEXP-complete	NEXP-complete
Free Comm.	P-complete	P-complete	PSPACE-complete

Table 3: Time complexity of COM-MTDPs.

sive (exact algorithms). We provide efficient exact algorithms for POMDPs, deployed in service of adjustable autonomy, by exploiting the notions of progress in the environment. The key insight is that given an initial (possibly uncertain) set of starting states, the agent needs to be prepared to act only in a limited range of belief states; most other belief states are simply unreachable given the dynamics of the monitored process so no action needs to be generated for such belief states. These bounds on the belief probabilities are obtained using Lagrangian techniques [32].

We tested this enhanced algorithm against two of the fastest exact algorithms: GIP (Generalized Incremental Pruning) and RBIP (Region Based Incremental Pruning). We obtained orders of magnitude speedups over RBIP and GIP in problems taken from the personal assistant domain, namely meeting scheduling and task management.

3.2 Multiagent Communication: BDI and Distributed POMDPs

We next shift our focus from single agent POMDPs to distributed POMDPs. In particular, we introduce the multiagent team decision problem (MTDP) [25], which is defined as a tuple: $\langle S, A, P, \Omega, O, R\rangle$. S is a finite set of world states $\{s_1, \ldots, s_m\}$. $A = \times_{1\leq i\leq n} A_i$, where $A_1, \ldots, A_n$, are the sets of action for agents 1 to n. A joint action is represented as $\langle a_1, \ldots, a_n\rangle$. $P(s_i, \langle a_1, \ldots, a_n\rangle, s_f)$, the transition function, represents the probability that the current state is s_f, if the previous state was s_i and the previous joint action was $\langle a_1, \ldots, a_n\rangle$. $\Omega = \times_{1\leq i\leq n}\Omega_i$ is the set of joint observations where Ω_i is the set of observations for agent i. $O(s, \langle a_1, \ldots, a_n\rangle, \omega)$, the observation function, represents the probability of joint observation $\omega \in \Omega$, if the current state is s and the previous joint action is $\langle a_1, \ldots, a_n\rangle$. We assume that observations of each agent are independent, i.e. $O(s, \langle a_1, \ldots, a_n\rangle, \omega) = O_1(s, \langle a_1, \ldots, a_n\rangle, \omega_1) \cdot \ldots \cdot O_n(s, \langle a_1, \ldots, a_n\rangle, \omega_n)$. The agents receive a single, immediate joint reward $R(s, \langle a_1, \ldots, a_n\rangle)$. Each agent i chooses its actions based on its local policy, Π_i, which is a mapping of its observation history to actions. Thus, at time t, agent i will perform action $\Pi_i(\vec{\omega}_i^t)$ where $\vec{\omega}_i^t = \omega_i^1, \ldots.., \omega_i^t$. $\Pi = \langle \Pi_1, \ldots.., \Pi_n\rangle$ refers to the joint policy of the team of agents.

MTDP may be extended to examine the tradeoffs between complexity and optimality in multiagent systems. COM-MTDP extends MTDP in service of analyzing communication in multiagent teams. We have used the COM-MTDP framework to characterize the difficulty of problem domains in agent teamwork along the dimensions of communication cost and observability. Table 3 summarizes our results, which show that the greatest challenges lie in domains with either *collective observability* or *partial observability* and with nonzero communication cost. Under *collective observability* and *partial observability* with *free communication*, the complexity becomes on par with that of single-agent planning problems. While free communication may be rare, these results show that investment in communication in teamwork can pay off with a significant simplification of optimal teamwork. On the other hand, when the world is *individually observable*, communication makes little difference in performance.

Table 3 shows that providing optimal domain-level and coordination policies for teams is a challenge. We could give up optimality and generate locally optimal policies, as done in our joint equilibrium-based search for policies (JESP) and Communicating JESP (Comm-JESP) algorithms [17]. Unfortunately, while the results are encouarging, they are not yet at a stage where large teams of agents can be easily constructed using policies generated with these algorithms.

Rather than applying COM-MTDP directly to generating policies, we may use it to analyze existing BDI systems. The goal of this hybrid approach is to improve the performance of the BDI team by optimizing its communications.

In many BDI systems, planners or domain experts provide the domain-level team plans [27]. The task for the agents is generating the appropriate team coordination to ensure that they properly execute the domain-level plans. For instance, the joint intentions and SharedPlans theories [4, 8] provide a basis for many existing approaches to team coordination. We have used COM-MTDP to analyze different instantiations of these theories, e.g. the STEAM algorithm, which applied joint intentions to several real-world domains [27]. Although the STEAM policy is based on decision-theoretic communication selectivity, it remains unanswered whether it is *optimally* selective, and researchers continue to struggle with the question of when agents should communicate [33]. The few reports of suboptimal communication in STEAM were characterized as an exceptional circumstance, but it is possible that STEAM's optimal performance is the exception. We used the COM-MTDP model to derive an analytical characterization of optimal communication in STEAM, and also investigated its performance empirically. Our experiments illustrated that the observed suboptimality was not an isolated phenomenon, but, in fact, that STEAM has a propensity towards extraneous communication in situations involving low observability and high communication costs. This result matches the situation where the "aberration" occurred in the more realistic domain.

3.3 Multiagent task allocation: BDI and Distributed POMDPs

Next we describe a more complex hybrid BDI-POMDP approach [16], where BDI team plans are exploited to improve POMDP tractability and POMDP analysis improves BDI team plan performance through improved role allocation, i.e. which agents to assign to the different roles in the team.

This hybrid approach (see Figure 4) combines the strengths of BDI plans and RMTDP (role-based multiagent team decision problem), an extension of MTDP that enables quantitative evaluation of role allocations. This synergistic interaction enables RMTDPs to improve the performance of BDI-based teams. We have also identified four ways in which BDI team plans make it easier to build RMTDPs and to efficiently search RMTDP policies. First, we use the preconditions and post-conditions in the BDI plans to mathematically define the domain via an RMTDP. Second, the BDI plans provide incomplete policies to RMTDPs, restricting the policy search. Next, the BDI plan hierarchy helps decompose the RMTDP policy search, thus improving its efficiency. In particular, we use the plan hierarchy to come up with an admissible heuristic called MAXEXP that allows us to do a branch-and-bound search in the role allocation policy space. Finally, belief representation in BDI team plans is exploited to enable faster RMTDP policy evaluation.

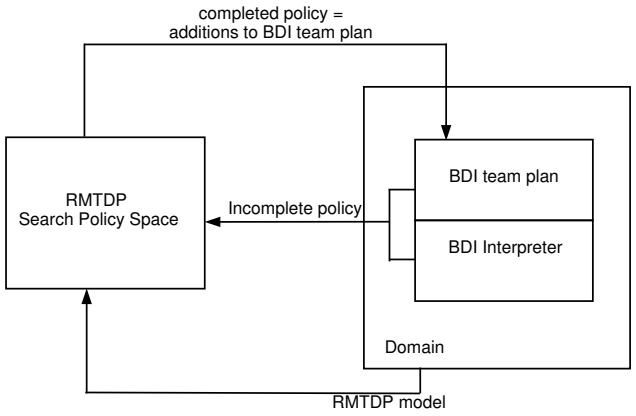

Figure 4: Integration of BDI and POMDP

We demonstrate the advantages of this hybrid approach via a scenario (see Figure 5) from the RoboCupRescue disaster simulation environment [29]. Here, five fire engines at three different fire stations (two each at stations 1 & three and the last at station 2) and five ambulances stationed at the ambulance center must collaborate to put out two fires (in top left and bottom right corners of the map) and to save the surviving civilians. The first goal is to determine which fire engines to assign to each fire. Once the fire engines have gathered information about the number of civilians at each fire, this is transmitted to the ambulances. The next goal is to allocate the ambulances to fires to rescue the civilians trapped there.

Figure 6 demonstrates the improvements in performance from using the MAXEXP heuristic over a brute force search (NOPRUNE). While both search methods find the best allocation, there is a significant savings for MAXEXP over NOPRUNE in terms of both nodes in the role allocation space as well as run time.

Next, we compare the performance of the allocations found

Figure 5: RoboCupRescue Scenario: C1 and C2 denote the two fire locations, F1, F2 and F3 denote fire stations 1, 2 and 3 respectively and A denotes the ambulance center.

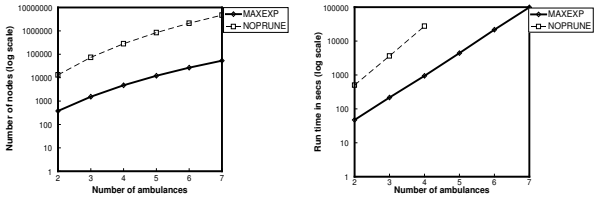

Figure 6: Performance of role allocation space search in RoboCupRescue, a: (left) Number of nodes evaluated on a log scale, and b: (right) Runtime in seconds on a log scale.

via the role allocation policy search against the performance of human subjects (human1, human2, human3) and RescueISI (the third place team in RoboCupRescue 2001). This comparison was done via multiple runs in the RoboCupRescue simulation environment. We used two different settings for the distribution from which civilian locations were drawn: uniform and skewed. The metric for comparison was the number of civilian casualties and the amount of building damage. The three human subjects were familiar with the RoboCupRescue domain and were given time to study the setup and to provide their allocations. As can be seen in Figure 7(a), the RMTDP allocation did better than the other five allocations in terms of a lower number of civilians dead. Using the skewed distribution, the difference between the allocations was greater (see Figure 7(b)). The RMTDP allocation does much better than the humans in terms of the number of civilians dead.

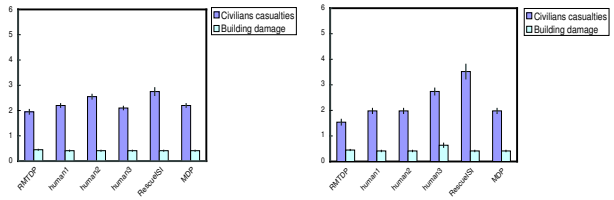

Figure 7: Comparison of performance in RoboCupRescue, a: (left) uniform, and b: (right) skewed.

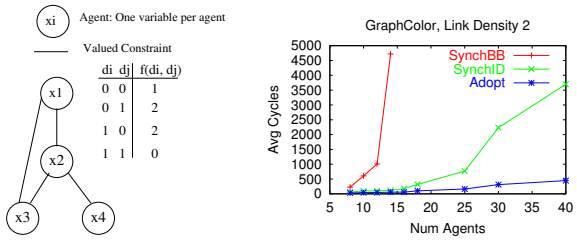

Figure 8: (a) An example DCOP with four agents. Agents must choose values for their respective variables to minimize the sum cost over all valued constraints. (b) The Adopt algorithm for DCOP significantly outperforms synchronous methods on benchmark problems.

3.4 Multiagent task allocation: Graphical games and DCOPs

Distributed constraint optimization (DCOP) is a promising framework for modeling team optimization problems. A DCOP consists of a set of variables assigned to agents who control their values. The agents must coordinate their local choice of variable values so that a global objective function is optimized. The global objective function is modeled as a set of distributed valued constraints. Figure 8(a) shows an example DCOP with four agents. Each constraint is defined by the table shown, where $f(d_i, d_j)$ denotes the cost of assigning $x_i = d_i$ and $x_j = d_j$. The objective is to find an assignment $\mathcal{A}^*$ such that the total cost, denoted F, is minimized:

$$F(\mathcal{A}) = \sum_{x_i, x_j \in V} f_{ij}(d_i, d_j) \quad , where\ x_i \leftarrow d_i, \\ x_j \leftarrow d_j\ in\ \mathcal{A}$$

Distributed constraints are a natural model for representing team optimization problems and are powerful enough to represent complex tasks, roles and capabilities. Previous research in AI and other areas [13] has shown that constraints can be used to represent and solve many interesting computational problems. A key feature of DCOP is that it allows agents to reason about optimization rather than a more limited model of only good/nogood solutions.

We have recently developed a new algorithm, named ADOPT (asynchronous distributed optimization), for solving DCOP [15]. The ADOPT algorithm has a number of novel features. In particular, ADOPT is able to provide strong theoretical guarantees on the global quality of solutions obtained while allowing agents to execute concurrently and communicate asynchronously. It is the first algorithm for DCOP that is both asynchronous and complete. As shown in Figure 8(b), initial results indicate that because ADOPT allows agents to execute concurrently, it is significantly more efficient that existing methods in terms of time required to find the globally optimal solution.

While ADOPT is a *complete* DCOP algorithm that arrives at a global optimal or a fixed distance from the global optimal, *incomplete* DCOP algorithms may compute a locally optimal solution. A more precise classification of incomplete algorithms is useful as a first step toward understanding the tradeoff between runtime and solution quality or the likelihood of finding an optimal solution. Capturing incomplete DCOP algorithms in a graphical-game model provides a theoretical foundation to address this problem. We provide a hybrid solution concept, called k-optimality [21, 10], that draws from both graphical games and constraint reasoning to categorize incomplete DCOP algorithms and the local optima they reach. A k-optimal DCOP solution is an assignment of values to agents such that no subset S of cardinality k of these agents can improve its local utility, defined as the sum of the rewards on all constraints incident on any agent in S. A k-optimal algorithm is an algorithm guaranteed to converge to a k-optimal solution. Under some assumptions, incomplete algorithms with higher k-optimality provide higher expected solution quality, and may require fewer restarts to reach a global optimum.

In experiments, while algorithms with lower k converged to a stable solution more rapidly, higher k algorithms achieved a higher solution quality on average. Figure 9 shows the performance of DSA, an existing 1-coordinated algorithm, against SCA-2, a new 2-coordinated algorithm based on DSA [10]. The comparison was done over many randomly generated examples, both in a three-coloring domain and in a domain in which all costs on constraints are chosen from a uniform random distribution.

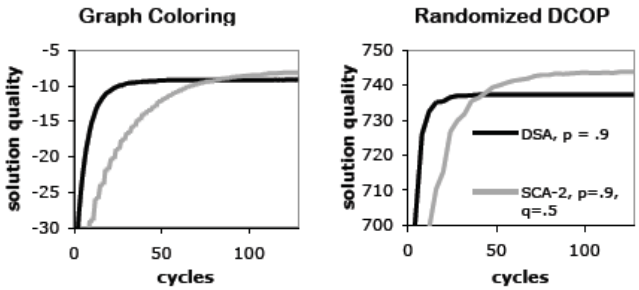

Figure 9: SCA-2 vs. DSA in two DCOP domains

K-optimality also provides a novel tool to enumerate sets of multiple solutions with desirable properties. A set of k-optimal solutions is guaranteed to have a certain level of diversity (any two solutions must be separated by a Hamming distance of at least $k + 1$) as well as relative quality (any solution X is of higher quality than any solution $\tilde{X}$ within a Hamming distance of k). Upper bounds on the number of possible k-optimal solutions to a DCOP can be obtained by leveraging results from coding theory on bounds on codewords in a word space [20]. In many domains, agent teams must generate multiple possible joint actions, either to execute in series or to provide a choice to a human operator. Each joint action generated may consume a resource, such as fuel (for vehicles), supplies (for troops) or time (for a human who must choose among the generated options). These bounds allow a human operator to choose an appropriate k in order to guarantee a particular level of diversity in the solution set, as well as to ensure that resources are not exhausted before all k-optimal solutions are found.

3.5 Multiagent task allocation: DCOPs and Distributed POMDPs

In many real-world multiagent applications, e.g. distributed sensor nets, a network of agents is formed based on each agent's interactions with a small number of neighbors. While distributed POMDPs capture the real-world uncertainty in multiagent domains, they fail to exploit such locality of in-

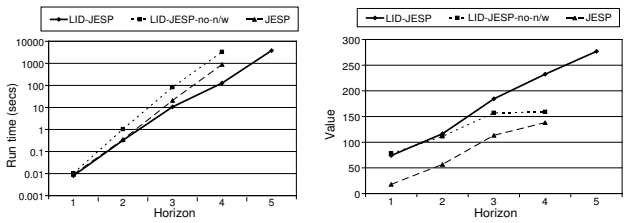

Figure 10: (a) Run time (secs), (b) Value

teraction. DCOP naturally captures the locality of interaction but fails to allow for planning under uncertainty. Hence, we introduce the networked distributed POMDP (ND-POMDP) model [18], a hybrid of distributed POMDP and DCOP, that can handle uncertainty in the domain and take advantage of locality of interaction.

The ND-POMDP model is a transition-independent, observation-independent distributed POMDP where the reward function can be expressed as the sum of rewards for groups of agents that interact. In the case of sensor nets, the reward is obtained by summing the rewards for interacting sensor agents. ND-POMDP can be thought of as an *n-ary* DCOP where the variable at each node is an individual agent's policy. The constraint graph for this DCOP is called an *interaction hypergraph* and is derived from the reward function of the ND-POMDP.

We developed a locally optimal policy generation algorithm called LID-JESP (locally interacting distributed joint equilibrium search for policies) is based on the DCOP DBA algorithm [34] and JESP [17]. We present some initial results for a sensor net scenario (Figure 3(a). In this domain five sensors need to track two moving targets. Each sensor can scan in four directions (N, S, E, and W), and two neighboring sensors are needed to successfully track a target in the sector. Further, there can be false positives and false negatives in the sensors' observations. We compared LID-JESP with Nair *et al.*'s JESP algorithm [17], which uses a centralized processor to find a locally optimal joint policy and does not consider the *interaction graph*. We also compare with LID-JESP-no-nw, which is LID-JESP with a fully connected *interaction graph*. Figure 10(a) shows the run time in seconds on a log scale on the y-axis for increasing finite horizon T on the x-axis, while Figure 10(b) shows the value of policy found on the y-axis and increasing finite horizon T on the x-axis. The values obtained for LID-JESP, JESP and LID-JESP-no-nw are quite similar, although LID-JESP and LID-JESP-no-nw often converged on a higher local optimum than JESP. In comparing the run times, LID-JESP outperforms LID-JESP-no-nw and JESP which highlights the advantage of exploiting network structure to reduce the complexity of distributed POMDPs.

4. TEAMWORK: TOWARDS THE FUTURE

This paper emphasized the role of hybrid representations for scalability, expressiveness and robustness in our current research on agent teams. It illustrated the synergistic interactions between distributed POMDPs, DCOPs, BDI systems and game theoretic representations. While this research focused on teams where team members are fully dedicated to their common goal (i.e. team members do not have additional explicitly represented constraints), our recent research has begun focusing on such additional constraints. These constraints arise as we push teamwork into domains where there may be individual resources that must not be shared with team members, or where there may be privacy considerations that prevent individual team members from revealing information about resource constraints or available options with their teammates. We describe here three issues in current research addressing these problems:

- Formalization of resource-constrained teamwork using distributed MDPs: While the COM-MTDP framework focused on agents with a joint reward function, we have also introduced the EMTDP framework to model agent teams where agents have additional individual resource constraints. The challenge that arises as a result of these bounds on expected resource consumption is that each agent gets a randomized policy and hence miscoordination arises in team settings. We have developed an algorithm for EMTDP transformation to allow resulting policies, even if randomized, to avoid such miscoordination [19].
- Multi-criteria optimization: While previous work in DCOP attempts to optimize a single global criterion, in multicriteria DCOP the goal is to also satisfy agents' individual constraints. We have developed a unified algorithm that tailors its performance to the structure of the network and whether the constraint is to be kept private. The key techniques involved are problem transformation and mutually constraining search [3].
- Privacy in DCOP: While a key motivation for using DCOPs has been privacy, the effectiveness of DCOP algorithms in achieving this goal has not been investigated quantitatively across multiple metrics. This has been due to the lack of a formal framework that can capture the various notions of privacy in a common syntax. We have developed a framework [9] which has allowed us to identify several key properties that lay hidden under the assumption that distribution automatically provides privacy. We have shown that centralization can outperform distribution in critical multi-agent scheduling problems. The key seems to be generating a constraint network without yielding knowledge of the total graph structure to the agents. We are developing tools and methodologies for inference (which is the determining factor for privacy loss) that will enable us to further identify and decipher areas where privacy is lost and where this loss can be prevented.

5. ACKNOWLEDGEMENTS

We thank Sven Koenig, Madhuri Kottamraju, Xiaoming Zheng, S. M. Raza Ali, Shriniwas Kulkarni and Don Dini for their teamwork!

6. REFERENCES

[1] S. Ali, S.Koenig, and M.Tambe. Preprocessing techniques for accelerating the dcop algorithm adopt. In *AAMAS*, 2005.

[2] D. Bernstein, S.Zilberstein, and N.Immerman. The complexity of decentralized control of markov decision processes. In *UAI*, 2000.

[3] E. Bowring, M. Tambe, and M. Yokoo. Distributed multi-criteria coordination in multi-agent systems. In *Workshop on DALT*, 2005.

[4] B. Grosz and S.Kraus. Collaborative plans for complex group action. *AIJ*, 86:269–357, 1996.

[5] G. Kaminka, D.V.Pynadath, and M.Tambe. Monitoring teams by overhearing: A multi-agent plan recognition approach. *JAIR*, 17, 2002.

[6] G. Kaminka and M.Tambe. Robust multi-agent teams via socially-attentive monitoring. *JAIR*, 12:105–147, 2000.

[7] V. Lesser, C.Ortiz, and M.Tambe. *Distributed sensor nets: A multiagent perspective*. Kluwer academic publishers, 2003.

[8] H. Levesque, P.R.Cohen, and J.H.T.Nunes. On acting together. In *AAAI*, 1990.

[9] R. Maheswaran, J. Pearce, P. Varakantham, E. Bowring, and M. Tambe. Valuation of possible states: A unifying quantitative framework for evaluating privacy in collaboration. In *AAMAS*, 2005.

[10] R. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *PDCS*, 2004.

[11] R. Maheswaran and T.Basar. Coalition formation in proportionality fair divisible auctions. In *AAMAS*, 2003.

[12] R. Mailler. Comparing two approaches to dynamic, distributed constraint satisfaction. In *AAMAS*, 2005.

[13] S. Minton, M.D.Johnston, A.B.Philips, and P.Laird. Minimizing conflicts: A heuristic method for constraint-satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[14] P. Modi, H.Jung, M.Tambe, W.Shen, and S.Kulkarni. A dynamic distributed constraint satisfaction approach to resource allocation. In *CP*, 2001.

[15] P. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 161:149–180, 2005.

[16] R. Nair and M.Tambe. Hybrid bdi-pomdp framework for multiagent teaming. *JAIR*, 23:367–413, 2005.

[17] R. Nair, M.Tambe, M.Yokoo, D.Pynadath, and S.Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

[18] R. Nair, P.Varakantham, M.Yokoo, and M.Tambe. Networked distributed pomdps: A synergy of distributed constraint optimization and pomdps. In *IJCAI*, 2005.

[19] P. Paruchuri, M.Tambe, F.Ordonez, and S.Kraus. Towards a formalization of teamwork with resource constraints. In *AAMAS*, 2004.

[20] J. Pearce, R. T. Maheswaran, and M. Tambe. Dcop games for multi-agent coordination. In *Workshop on DCR*, 2005.

[21] J. Pearce, R.T.Maheswaran, and M.Tambe. How local is that optimum? k-optimality for dcop. In *AAMAS*, 2005.

[22] D. Pynadath and M.Tambe. Automated teamwork among heterogeneous software agents and humans. *JAAMAS*, 7:71–100, 2003.

[23] P. Scerri, A.Farinelli, S.Okamoto, and M.Tambe. Allocating tasks in extreme teams. In *AAMAS*, 2005.

[24] P. Scerri, L.Johnson, D.Pynadath, P.Rosenbloom, M.Si, N.Schurr, and M.Tambe. A prototype infrastructure for distributed robot, agent, person teams. In *AAMAS*, 2003.

[25] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real-world. *JAIR*, 17:171–228, 2002.

[26] N. Schurr, J. Marecki, P. Scerri, J. Lewis, and M. Tambe. The defacto system: Training tool for incident commanders. In *IAAI*, 2005.

[27] M. Tambe. Towards flexible teamwork. *JAIR*, 7:83–124, 1997.

[28] M. Tambe, D.Pynadath, and N.Chauvat. Building dynamic agent organizations in cyberspace. *IEEE Internet Computing*, 4, 2000.

[29] M. Tambe, G.Kaminka, S.Marsella, I.Muslea, and T.Raines. Two fielded teams and two experts: A robocup response challenge from the trenches. In *IJCAI*, 1999.

[30] M. Tambe, W. Johnson, R. Jones, F. Koss, J. Laird, P. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, page 16(1), 1995.

[31] M. Tambe and W.Zhang. Towards flexible teamwork in persistent teams. *JAAMAS*, 3:159–183, 1998.

[32] P. Varakantham, R. Maheswaran, and M. Tambe. Exploiting belief bounds: Practical pomdps for personal assistant agents. In *AAMAS*, 2005.

[33] J. Yen, J.Yin, T.R.Ioerger, M.S.Miller, D.Xu, and R.A.Volz. Cast: Collaborative agents for simulating teamwork. In *IJCAI*, 2001.

[34] M. Yokoo and K.Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *ICMAS*, 1996.