Team Behavior in Interactive Dynamic Influence Diagrams with Applications to Ad Hoc Teams

Muthu Chandrasekaran
Dept. of Computer Science
University of Georgia
Athens, GA 30602, USA
mku@uga.edu

Prashant Doshi
Dept. of Computer Science
University of Georgia
Athens, GA 30602, USA
pdoshi@cs.uga.edu

Yifeng Zeng
School of Computing
Teesside University
Tees Valley, TS1 3BA, UK
y.zeng@tees.ac.uk

ABSTRACT

Planning for Ad Hoc teamwork among autonomous agents has been recognized as a challenging problem as it involves agents trying to collaborate without any prior coordination or communication protocol. Although this problem pertains to teamwork, the real challenge lies in building a single agent with such capabilities, not an entire team, which is why we investigate this in the context of individual decision making frameworks. However, individual decision making in multiagent settings faces the task of having to reason about other agents' actions who themselves could be reasoning about others. An approximation that enables the application of this approach is to bound the infinite nesting from below by introducing level 0 models. A consequence of the finitely nested modeling is that we may not obtain optimal team solutions in cooperative settings. We address this limitation by including models at level 0 whose solution involves learning. We demonstrate that the integrated learning with planning facilitates optimal team behavior and thus facilitating ad hoc teamwork. We investigate our novel approach within the framework of interactive dynamic influence diagrams and evaluate its performance.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents, Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

multiagent systems, ad hoc teamwork, sequential decision making and planning, reinforcement learning

1. INTRODUCTION

Autonomous agents often find themselves in situations where they must collaborate with one another to collectively achieve certain tasks in partially observable environments. Sometimes collaboration is expected without prior coordination with teammates. Such scenarios are identified as *ad hoc settings*. Planning in ad hoc settings is recognized to be a challenging problem in multiagent systems research [30, 31].

Appears in: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.*

Several attempts have been made to address ad hoc coordination in multiagent systems [2, 6, 32]. Since some knowledge about the teammates is hidden from a centralized planner, the approaches of *Decentralized Partially Observable Markov Decision Processes* (Dec-POMDPs) [7, 28] –which generally assume common initial beliefs for all agents –may not be appropriate. Most of other existing approaches rely on substantial pre-coordination and/or make several assumptions therein to simplify the problem domain in order to achieve ad hoc teamwork. For example, the *Online Planning for Ad Hoc Agent Teams* (OPAT) algorithm – a state-of-the-art online planning algorithm for ad hoc agent teams – assumes that all agents can fully observe both the environment states and the joint actions at each step. Additionally, their agents don't plan for their teammates to observe and interpret their own actions. Arbrecht et al. on the other hand, provide a framework – the *Harsanyi-Bellman Ad Hoc Coordination* (HBA) algorithm – where they make assumptions about the behavior types of the opponents thereby simplifying the problem [3]. In this paper however, we make no such assumptions and present a generalized framework for achieving teamwork.

Interactive Partially Observable Markov Decision Processes (I-POMDPs) [14, 17] and their graphical counterparts, Interactive Dynamic Influence Diagrams (I-DIDs) [15, 34], have been recognized by researchers to be complex but more general frameworks for ad hoc settings [3]. The approach used in these frameworks allows for agents to plan individually at their own level in the context of other agents acting and observing in a partially observable environment. Such frameworks particularly provide a solution to a fundamental challenge in ad hoc teamwork –building a single autonomous agent (as opposed to planning for the entire team) with the ability to coordinate in ad hoc settings [33].

However, individual decision making is confronted with having to reason about how the other agents behave, who in turn could be reasoning about others' behaviors, and so on *ad infinitum*. A common initial belief about the state circumvents this problem, but its conceptualization is itself problematic [16] and neither does it solve the problem nor does it apply to ad hoc settings. Motivated by approaches in game theory and interactive epistemology [1, 4, 20], we may bound this heirarchy from below as in finitely-nested I-POMDPs by assuming the presence of level 0 models that do not involve reasoning about others. This leads to an approximation of the optimal decision making motivated by considerations of computability.

A direct consequence of the finitely nested modeling is that we may not obtain optimal team solutions in cooperative settings in general. In this paper, we address the limitation and induce optimal team behavior by enhancing the reasoning ability of lower level agents thereby making I-POMDPs, as well as I-DIDs, theoretically perfectly suited for ad hoc settings. Notice that I-DIDs demonstrate

a computational advantage over I-POMDPs as they allow us to exploit the embedded structure in the problem [15, 34]. In this paper, we investigate ad hoc teamwork within the framework of I-DIDs.

We augment the I-DID framework by additionally attributing a new type of level 0 model. This type distinguishes itself by utilizing reinforcement learning (RL) either online or in simulation to discover an optimal policy. As the setting is partially observable, few RL approaches apply. A promising one is Perkins' Monte-Carlo Exploring Starts for POMDPs (MCESP) [25], which is an action-value based RL algorithm for POMDPs. This type of models mainly differs from the previous type in its frame, which in addition to the augmented capabilities and preferences, contains a fixed seed policy for agent j, the learning rate, and a candidate i's policy. Together with the belief, the frame is sufficient to learn a – possibly locally optimal – policy online or offline in a simulated environment.

The contributions of this research are two-fold; First, we show the plausibility of true team behavior to emerge when the reasoning ability of lower level agents is enhanced via learning. Consequently, we are able to achieve globally optimal teammate solutions when our agents are modeled in finitely-nested *augmented* I-DIDs while *traditional* I-DIDs fail. Second, we demonstrate the applicability of augmented I-DIDs to ad hoc settings; show its robustness, and effectiveness for different types of teammates. In this regard, we experiment with multiple well-known cooperative problem domains and generate ad hoc team behavior for all of them. We also perform a baseline comparison of augmented I-DIDs with our implementation of a generalized version of OPAT – to account for the partial observability – in simulation.

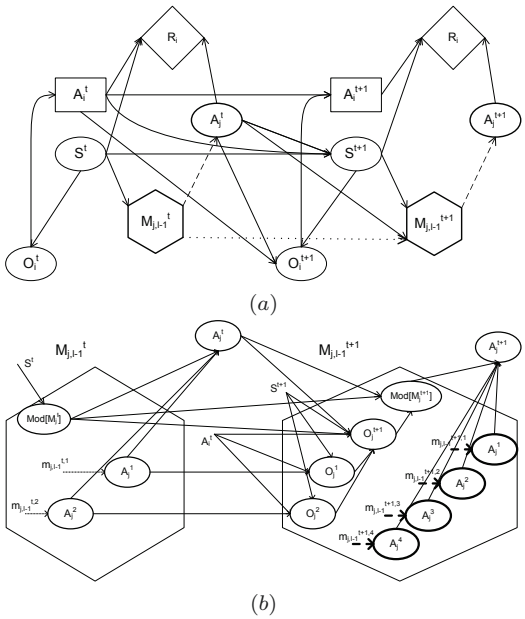


Figure 1: (a) **A generic two time-slice level l I-DID for agent i. The dotted model update link represents the update of j's models and the distribution over the models over time;** (b) **Implementation of the model update link using standard dependency links and chance nodes; e.g., two models, $m_{j,l-1}^{t,1}$ and $m_{j,l-1}^{t,2}$, are updated into four models (shown in bold) at time $t+1$.**

2. BACKGROUND: INTERACTIVE DIDS

We sketch I-DIDs below and refer readers to [34] for more details.

2.1 Representation

A traditional DID models sequential decision making for a single agent by linking a set of chance, decision and utility nodes over multiple time steps. To consider multiagent interactions, I-DIDs introduce a new type of node called the *model node* (hexagonal node, $M_{j,l-1}$, in Fig. 1(a)) that represents how another agent j acts as subject agent i reasons about its own decisions at level l. The model node contains a set of j's candidate models at level l 1 ascribed by i. A link from the chance node, S, to the model node, $M_{j,l-1}$, represents agent i's beliefs over j's models. Specifically, it is a probability distribution in the conditional probability table (CPT) of the chance node, $Mod[M_j]$ (in Fig. 1(b)). An individual model of j, $m_{j,l-1} = \rangle b_{j,l-1}, \hat{\theta}_j|$, where $b_{j,l-1}$ is the level l 1 belief, and $\hat{\theta}_j$ is the agent's *frame* encompassing the decision, observation and utility nodes. Each model, $m_{j,l-1}$, could be either a level l 1 I-DID or a DID at level 0. Solutions to the model are the predicted behavior of j and are encoded into the chance node, A_j, through a dashed link, called a *policy link*. Connecting A_j with other nodes in the I-DID structures how agent j's actions influence i's decision-making process.

Expansion of an I-DID involves the update of the model node over time as indicated by the *model update link* - a dotted link from $M_{j,l-1}^t$ to $M_{j,l-1}^{t+1}$ in Fig. 1(a). As agent j acts and receives observations over time, its models should be updated. For each model $m_{j,l-1}^t$ at time t, its optimal solutions may include all actions and agent j may receive any of the possible observations. Consequently, the set of the updated models at $t+1$ contains up to { $_{j,l-1}^{t}$ A_j Ω_j models. Here, { $_{j,l-1}^{t}$ is the number of models at time t, and A_j and Ω_j the largest spaces of actions and observations respectively among all the models. The CPT of $Mod[M_{j,l-1}^{t+1}]$ specifies the function, $\tau(b_{j,l-1}^t, a_j^t, o_j^{t+1}, b_{j,l-1}^{t+1})$ which is 1 if the belief $b_{j,l-1}^t$ in the model $m_{j,l-1}^t$ using the action a_j^t and observation o_j^{t+1} updates to $b_{j,l-1}^{t+1}$ in a model $m_{j,l-1}^{t+1}$; otherwise, it is 0. We may implement the model update link using standard dependency links and chance nodes, as shown in Fig. 1(b), and transform an I-DID into a traditional DID.

2.2 Solution

A level l I-DID of agent i expanded over T time steps is solved in a bottom-up manner. To solve agent i's level l I-DID, all lower level l 1 models of agent j must be solved. Solution to a level l 1 model, $m_{j,l-1}$, is j's policy that is a mapping from j's observations in O_j to the optimal decision in A_j given its belief, $b_{j,l-1}$. Subsequently, we may enter j's optimal decisions into the chance node, A_j, at each time step and expand j's models in $Mod[M_j]$ corresponding to each pair of j's optimal action and observation. We perform this process for each of level l 1 models of j at each time step, and obtain the fully expanded level l model. We outline the algorithm for exactly solving I-DIDs in Fig. 2.

The computational complexity of solving I-DIDs is mainly due to the exponential growth of lower l-1 j's models over time. Although the space of possible models is very large, not all models need to be considered in the model node. Models that are behaviorally equivalent (BE) [26] – whose behavioral predictions for the other agent are identical – could be pruned and a single representative model considered. This is because the solution of the subject agent's I-DID is affected by the behavior of the other agent only; thus we need not distinguish between BE models. Let **PruneBehavioralEq** ({ $_{j,l-1}$) be the procedure that prunes BE models from { $_{j,l-1}$ returning the representative models (line 6).

Note that lines 4-5 (in Fig. 2) solve level l-1 I-DIDs or DIDs and

then supply the policies to level l I-DID. Due to the bounded rationality of level l-1 agents, the solutions lead to a suboptimal policy of agent j, which certainly compromises agent i's performance in the interactions particularly in a team setting. We will show in the coming sections that solving I-DIDs using RL to get the policy of j at level l-1 may generate the expected team behavior among agents i and j.

I-DID EXACT(level $l \geq 1$ I-DID or level 0 DID, horizon T)
Expansion Phase
1. **For** t **from** 0 **to** $T-1$ **do**
2. **If** $l \geq 1$ **then**
Populate $M^{t+1}_{j,l-1}$
3. **For each** m^t_j **in** $\mathcal{M}^t_{j,l-1}$ **do**
4. Recursively call algorithm with the $l-1$ I-DID (or DID) that represents m^t_j and horizon, $T-t$
5. Map the decision node of the solved I-DID (or DID), $OPT(m^t_j)$, to the corresponding chance node A_j
6. $\mathcal{M}^t_{j,l-1} \leftarrow$ **PruneBehavioralEq**($\mathcal{M}^t_{j,l-1}$)
7. **For each** m^t_j **in** $\mathcal{M}^t_{j,l-1}$ **do**
8. **For each** a_j **in** $OPT(m^t_j)$ **do**
9. **For each** o_j **in** O_j (part of m^t_j) **do**
10. Update j's belief, $b^{t+1}_j \leftarrow SE(b^t_j, a_j, o_j)$
11. $m^{t+1}_j \leftarrow$ New I-DID (or DID) with b^{t+1}_j
12. $\mathcal{M}^{t+1}_{j,l-1} \overset{\cup}{\leftarrow} \{m^{t+1}_j\}$
13. Add the model node, $M^{t+1}_{j,l-1}$, and the model update link
14. Add the chance, decision, and utility nodes for $t+1$ time slice and the dependency links between them
15. Establish the CPTs for each chance node and utility node
Solution Phase
16. **If** $l \geq 1$ **then**
17. Represent the model nodes, policy links and the model update links as in Fig. 1 to obtain the DID
18. Apply the standard look-ahead and backup method to solve the expanded DID

Figure 2: Algorithm for exactly solving a level $l \geq 1$ I-DID or level 0 DID expanded over T time steps.

3. TEAMWORK IN INTERACTIVE DIDS

Ad hoc teamwork involves multiple agents working collaboratively in order to optimize the team reward. Each ad hoc agent in the team behaves according to a policy, which maps the agent's observation history or beliefs to the action(s) it should perform. We begin by showing that the finitely-nested hierarchy in I-DID does not facilitate ad hoc teamwork. However, augmenting the traditional model space with models whose solution is obtained via RL provides a way for team behavior to emerge.

3.1 Implausibility of Teamwork

Fig. 3 shows an ad hoc team setting of the two-agent *grid meeting* problem [8]. Each ad hoc agent, i or j, moves in the grid and collects rewards as the number indicated in the occupied cell. If they move to different cells, the agents get their own individual reward. However, if they move to the same cell allowing them to hold a meeting, they will be rewarded with the twice of the sum of their individual rewards. This is the type of ad hoc teamwork expected in practice. Initial positions of the two agents are shown colored and we focus on their immediate actions.

If each agent deliberates at its own level, agent i modeled at level 0 will choose to move left while a level 0 agent j chooses to move down. Each agent would obtain a reward of 15 with the whole team getting 30. Agent i modeled at level 1 and modeling j at level 0 thinks that j will move down, and its own best response to predicted j's behavior is to move left. Analogously, a level 1 agent j would choose to move down. A level 2 agent i will predict that a level 1 j moves down as mentioned previously, due to which it decides to move left. Analogously, a level 2 agent j continues to decide to move down. We may apply this reasoning inductively to conclude that level l agents i and j would move left and down, respectively, thereby earning a joint reward of 30. However, the optimal team behavior in this setting is for i to move right and j to move up thereby obtaining a team reward of 40.

Clearly, these finite hierarchical systems preclude the agents' optimal teamwork due to the bounded reasoning of the lowest level (level 0) agents. Observation 1 states this more formally:

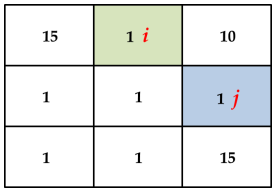


Figure 3: Agents i and j in the grid meeting problem with the numbers being their individual rewards.

Observation *There exist cooperative multiagent settings in which intentional agents each of which is modeled using the finitely-nested I-DID may not choose the jointly optimal behavior of working together as a team.*

PROOF. Observe that an offline specification of level 0 models in cooperative settings is necessarily incomplete. This is because the true benefit of cooperative actions often hinges on others performing supporting actions, which by themselves may not be highly rewarding to the agent. Therefore, despite solving the level 0 models optimally, the agent may not engage in optimal teammate behavior. One such cooperative setting was shown in Fig. 3. □

In general, this observation holds for cooperative settings where the self-maximizing level 0 models result in predictions that are not consistent with optimal team behavior for the modeled agent. Of course, settings may also exist where the level 0 model's solutions coincide ad hoc with the policy of a teammate thereby leading to joint teamwork. Nevertheless, the significance of this observation is that we may not rely on finitely-nested I-DIDs to generate optimal teammate policies.

We observe that team behavior is challenging in the context we study above because of the bounded rationality imposed by assuming a level 0. The boundedness precludes modeling others at the same level as one's own – as an equal teammate. However, at the same time, this imposition is, (a) motivated by reasons of computability, which allow us to operationalize such a paradigm; and (b) allows us to avoid some self-contradicting, and therefore impossible beliefs, which exist when infinite belief hierarchies are considered [9, 11]. Consequently, answering this question is of significance because it may provide us with a way of generating optimal team behavior in finitely-nested frameworks, which so far have been utilized for noncooperative settings, and provides a principled way to solving ad hoc teamwork problem.

3.2 Augmented Level 0 Models that Learn

We present a principled way to induce tam behavior by enhancing the reasoning ability of lower level agents. While it is difficult to *a priori* discern the benefit of moving up for agent j in Fig. 3, it could be *experienced* by the agent. Specifically, it may explore moving in different directions including moving up and learn about its benefit from the ensuing, possibly indirect, team reward.

If we place agents in a cooperative problem domain or its simulation, an agent may come to *experience* the benefit of performing

team actions. Subsequently, we may expect an agent to learn policies that are consistent with optimal teammate behavior because the corresponding actions provide large reinforcements. For example, given that agent i moves right in Fig. 3, j may choose to move up in its explorations, and thereby receive a large reinforcing reward. This observation motivates formulating level 0 models that utilize RL in order to generate the predicted policy for the modeled agent.

Because the level 0 models generate policies for the modeled agent only, we focus on the modeled agent's learning problem. However, the rewards in the setting usually depend on actions of all agents due to which the other agent must be simulated as well. The other agent's actions are a part of the environment and its presence hidden, thereby making the problem one of single-agent learning.

We augment the level 0 model space, denoting it as $M'_{j,0}$, by additionally attributing a new type of level 0 model to the other agent j: $m'_{j,0} = \langle b_{j,0}, \hat{\theta}'_j |$, where $b_{j,0}$ is j's belief and $\hat{\theta}'_{j,0}$ is the frame of the learning model. The frame, $\hat{\theta}'_{j,0}$, consists of the learning rate, α, a seed policy, π'_j, of planning horizon, T, which includes a fair amount of exploration, and the chance and utility nodes of the DID along with a candidate policy of agent i, which could be an arbitrary policy from i's policy space, Π_i, as agent i's actual behavior is not known. This permits a proper simulation of the environment.

This type of model, $m'_{j,0}$, differs from a traditional DID based level 0 model in the aspect that $m'_{j,0}$ does not describe the offline planning process of how agent j optimizes its decisions, but allows j to learn an optimal policy, π_j, with the learning rate, either online or in a simulated setting. Different models of agent j differ not only in their learning rates and seed policies, but also in the i's candidate policy that is used. Therefore, in principle, while the learning rate and seed policies may be held fixed, j's model space could be as large as i's policy space. Consequently, our augmented model space becomes larger.

3.3 Learning Algorithm:Generalized MCESP

Learning has been applied to solve decision-making problems in both single- and multi-agent settings. Both model based [13, 23] and model free [19, 21] learning approaches exist for solving POMDPs. Banerjee *et al.* [5] utilized a distributed RL approach solving finite horizon Dec-POMDPs. Recently, Ng *et al.* [22] incorporated model learning in the context of I-POMDPs where adversarial agents learn the transition and observation probabilities by augmenting the interactive states with the learning parameters.

Because the setting in which the learning takes place is partially observable, RL approaches that compute a table of values for state-action pairs do not apply. We adapt Perkins' Monte Carlo Exploring Starts for POMDPs (MCESP) [25], which has been shown to learn good policies in less iterations. MCESP maintains a Q table indexed by observation, o_j, and action, a_j, that gives the value of following a policy, π_j, except when observation, o_j, is obtained at which point action, a_j, is performed. An agent's policy in MCESP maps a single observation to actions over T time horizons. We generalize MCESP so that observation histories of length up to T, denoted as $\vec{o}$, are mapped to actions. A table entry, $Q^{\pi_j}_{\vec{o},a}$, is updated over every simulated trajectory of agent j, $\tau = \} \pm a^0_j, r^0_j, o^1_j, a^1_j, r^1_j, \times\times, o^{T-1}_j, a^{T-1}_j, r^{T-1}_j, o^T_j \langle$, where r_j is the team reward received. Specifically, the $Q^{\pi_j}_{\vec{o},a}$ value is updated as:

$$Q^{\pi_j}_{\vec{o},a} \in (1 \quad \alpha) Q^{\pi_j}_{\vec{o},a} + \alpha R_{post-\vec{o}}(\tau) \tag{1}$$

where α is the learning rate and $R_{post-\vec{o}}(\tau)$ is the sum of rewards of a portion of the observation-action sequence, τ, following the first occurrence of $\vec{o}$ in τ, say at t': $R_{post-\vec{o}}(\tau)$=$\sum_{t=t'}^{T-1} \gamma^t r_t$, where γ $\mathcal{M}[0, 1)$ is the discount factor. Alternate policies are considered by perturbing the action for randomly selected observation histories.

Level 0 agent j learns its policy while agent i's actions are hidden in the environment. In other words, agent j needs to reason with unknown behavior of i while it learns level 0 policy using the generalized MCESP algorithm. Agent j considers the entire policy space of agent i, Π_i, and a fixed policy of i, $\pi_i(\mathcal{M}[\Pi_i)$, results in one learned j's policy, π_j.

We show the algorithm for solving level 0 models using the generalized MCESP in Fig. 5. The algorithm takes as input agent j's model whose solution is needed and the policy of i, which becomes a part of the environment. We repeatedly obtain a trajectory, τ, of length T either by running the agent online or simulating the environment by sampling the states, actions and observations from the appropriate CPTs (lines 5-10). The trajectory is used in evaluating the value of the current policy, π_j, of agent j (line 11). Initially, we utilize the seed policy contained in agent j's model. If another action, a', for the observation sequence, $\vec{o}$, is optimal, we update π_j to conditionally use this action, otherwise the policy remains unchanged (lines 12-13). This is followed by generating a perturbed policy in the neighborhood of the previous one (line 14), and the evaluation cycle is repeated. If the perturbation is discarded several times, we may terminate the iterations and return the current policy.

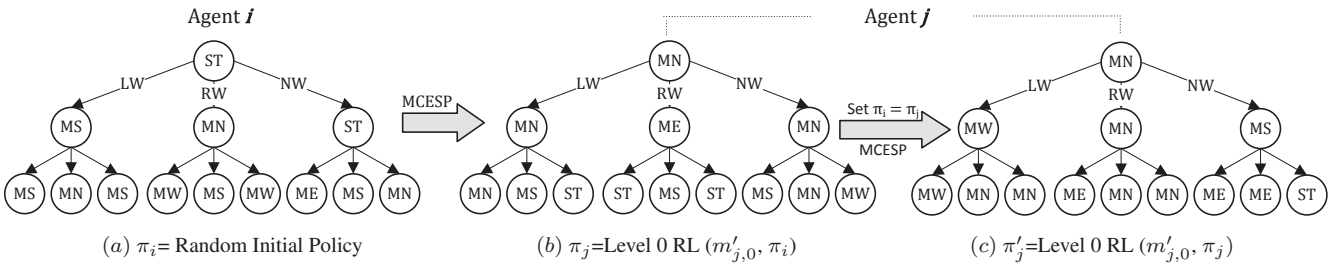
As the space of i's policy becomes very large particularly for a large planning horizon, it is intractable for j to learn a policy for all i's policies. In addition, considering that few of i's policies are actually collaborative, we formulated a principled way that allows us to reduce the full space to those policies of i, denoted as $\hat{\Pi}_i$, that could be collaborative. We elaborate this method with the help of the 3x3 Grid domain as shown in Fig. 4. For this, we pick a random initial policy of i. We use the corresponding i's policy in the frame of a new model of j, and create a new model for j. When we perform this step from second time onwards, we set the initial policy that MCESP uses as the previous π_j. MCESP then checks for neighbors of π_j, which would improve on the joint utility of (π_j,π_j)[1]. If successful, the improved neighboring policy, say π'_j, is returned. This ensures that (π_j,π'_j) is greater than (π_j,π_j). If π_j cannot be improved upon by MCESP – because π_j is already the best response to π_j (as i's policy) – we will do a random restart while picking the initial policy of i. One reason for this is because the pair represents the global optimality. This way, we exploit MCESP's approach as well.

3.4 Augmented I-DID Solutions

Solving augmented I-DIDs is similar to solving the traditional I-DIDs except for the fact that the candidate models of the agent at level 0 may be learning models. We show the revised algorithm in Fig. 6. When the level 0 model is a learning model, the algorithm invokes the method LEVEL 0 RL shown in Fig. 5. Otherwise, it follows the same procedure as shown in Fig. 2 to recursively solve the lower level models.

While we may reduce the space of i's policies to a subset, $\hat{\Pi}_i$ in a principled fashion, and therefore j's learning models, we may further reduce agent j's policy space by keeping top-K policies of j, K>0, in terms of their expected utilities (line 11). Given the same belief, the team behavior(s) is guaranteed to generate the largest utility in a cooperative problem. Subsequently, the resulting top-K policies shall include j's optimal collaborative policies. Hence this filtering of j's policy space may not compromise the quality of I-DID solutions at level 1. Ideally, top-K policies shall contain all K optimal team policies of j. As the number of optimal poli-

[1]This denotes the joint policy where agent i's policy, π_i, is set to π_j and agent j's policy, π_j, is the only candidate model in agent i's I-DID. We can then compute the joint expected utility given the agents' policy trees.



(a) π_i= Random Initial Policy (b) π_j=Level 0 RL ($m'_{j,0}$, π_i) (c) π'_j=Level 0 RL ($m'_{j,0}$, π_j)

Figure 4: We illustrate the principled way to generate collaborative policies for lower level agent j in the 3x3 Grid domain. We start with (a) a random initial policy of agent i in the frame of agent j's model; (b) execute MCESP to generate a collaborative policy for agent j; and (c) check whether the neighboring policy of j, π'_j, has a better joint utility.

LEVEL 0 RL (agent j's model, $m'_{j,0}$, agent i's policy, π_i, T)

1. Sample the initial state s from $b_{j,0}$ in j's model
2. Set current policy of j denoted as π_j using the seed policy in j's model
3. Set $\tau \leftarrow \{*\}$ – empty observation
4. **Repeat**
5. **For** t = 0 to $T-1$ **do**
6. Obtain i's action from π_i and j's action, a_j^t, from current policy of j using observation history
7. Obtain the next state, s', either by performing the actions or sampling
8. Obtain team reward, r_j^t, using state and joint actions
9. Obtain j's observation, o_j^{t+1}, using next state and joint actions
10. Generate trajectory, $\tau \leftarrow \tau \cup \{a_j^t, r_j^t, o_j^{t+1}\}$
11. Update $Q^{\pi_j}_{\vec{o},a}$ according to Eq. 1
12. **If** $\underset{a'}{max}\, Q^{\pi_j}_{\vec{o},a'} > Q^{\pi_j}_{\vec{o},a}$
13. $\pi_j(\vec{o}) \leftarrow a'$
14. Perturb an action, a, in π_j for some $\vec{o}$
15. **Until** termination condition is met
16. **Return** π_j and $Q^{\pi_j}_{\vec{o},a}$

Figure 5: Algorithm for learning agent j's policies when modeled at level 0.

AUGMENTED I-DID (level $l \geq 1$ I-DID or level 0 DID, T)

Expansion Phase

1. **For** t **from** 0 **to** $T-1$ **do**
2. **If** $l \geq 1$ **then**
Populate $M^{t+1}_{j,l-1}$
3. **For each** m_j^t **in** $\mathcal{M}^t_{j,l-1}$ **do**
4. **If** $l > 1$ **then**
5. Recursively call algorithm with the $l-1$ I-DID that represents m_j^t and the horizon, $T-t$
6. **else**
7. **If** the level 0 model is a learning model **then**
8. Solve using LEVEL 0 RL in Fig. 5 with horizon, $T-t$
9. **else**
10. Recursively call algorithm with level 0 DID and the horizon, $T-t$
11. Select top-K j's policies based on expected utility values given the same belief
12. The remaining steps of the expansion phase are the same as in Fig. 2.

Solution Phase

13. This is similar to the solution phase in Fig. 2.

Figure 6: Algorithm for solving a level $l \geq 1$ I-DID or level 0 DID expanded over T time steps with $M'_{j,0}$ containing level 0 models that learn.

cies is unknown, we normally use a sufficiently large K value. The proposition below formalizes this observation.

Proposition *Top-K policies of level 0 models of agent j given same initial beliefs, K>0, guarantee inclusion of j's optimal team policy resulting in the optimal team behavior of agent i at level 1.*

Agent j's policy space will be additionally reduced because behaviorally equivalent models – models whether they are learning or not with identical policies – will be clustered in line 6 of Fig. 2. In summary, we take several steps to reduce the increase in j's model space. Using a subset of i's policies preempts growing j's policy space at level 0 while the top-K technique removes possibly non-collaborative policies.

4. EXPERIMENTAL RESULTS

From our experiments, we show that I-DIDs augmented with level 0 models that learn, facilitate optimal team behavior which was previously implausible. Consequently, we also show the applicability of augmented I-DIDs to ad hoc teamwork. We empirically evaluate the performance in *three* well-known cooperative problem domains involving two agents, i and j: $3 * 3$ grid meeting (Grid) [8], box-pushing (BP) [27], and multiple access broadcast channel (MABC) [18] problems.

Table 1: Domain Dimension and Experimental Settings

Domain	T	$\lvert\mathcal{M}_j^0\rvert$	$\lvert\hat{\Pi}_i\rvert$	Dimension
Grid	3	100	32	$\lvert S_j\rvert$=9, $\lvert S_i\rvert$=81, $\lvert\Omega\rvert$=3, $\lvert A\rvert$=5
	4	200	64	
BP	3	100	32	$\lvert S\rvert$=50, $\lvert\Omega\rvert$=5, $\lvert A\rvert$=4
MABC	3	100	32	$\lvert S_j\rvert$=2, $\lvert S_i\rvert$=4, $\lvert\Omega\rvert$=2, $\lvert A\rvert$=2
	4	100	64	
	5	200	64	

We summarize the domain statistics and parameter settings of the AUGMENTED I-DID in Table 1. Note that $\{\ ^0_j$ is the number of initial models of agent j at level 0 and $\hat{\Pi}_i$ is the subset of i's policies that were generated using a principled approach which allowed us to reduce the full space of agent i's policies to those that are possibly collaborative.

4.1 Plausibility of Teamwork

Experimental Settings. We implemented the algorithm AUG. I-DID as shown in Fig. 6 including an implementation of the generalized MCESP for performing level 0 RL. We demonstrate the performance of the augmented framework for generating team behavior in light of the consequences of bounded rationality in I-DIDs.

Table 2: Performance comparison between the trad. I-DID, aug. I-DID, and DP-JESP in terms of the expected utility

	Aug. I-DID			Trad I-DID
Domain	**K**	**Uniform**	**Diverse**	**Uniform**
Grid (T=3)	32	41.875	41.93	25.70
	16	40.95	41.93	
	8	40.95	41.93	
	Dec-POMDP(DP-JESP): 43.10			
Grid (T=4)	64	28.76	32.61	21.55
	32	28.76	32.61	
	16	27.85	32.61	
	Dec-POMDP(DP-JESP): *			
BP (T=3)	32	73.45	76.51	4.75
	16	73.45	76.51	
	8	71.36	76.51	
	Dec-POMDP(DP-JESP): 85.18			
MABC (T=3)	32	2.12	2.30	1.79
	16	2.12	2.30	
	8	2.12	2.30	
	Dec-POMDP(DP-JESP): 2.99			
MABC (T=4)	64	3.13	3.17	2.80
	32	3.13	3.17	
	16	3.13	3.17	
	Dec-POMDP(DP-JESP): 3.89			
MABC (T=5)	64	4.08	4.16	3.29
	32	3.99	4.16	
	16	3.99	4.16	
	Dec-POMDP(DP-JESP): 4.79			

We compare the expected utility values of agent i's policies with those obtained from a Dec-POMDP formulation of the same problem domains. Note that the Dec-POMDP framework is designed for exclusively solving the class of problems that constitute a cooperative multiagent setting. For different beliefs over the models, the augmented I-DID solution approaches optimal team behavior in all problem domains while outperforming traditional I-DID solutions.

Comparison with DP-JESP. We used an exact algorithm (**DP-JESP**) – a recognized technique for solving Dec-POMDP – to compute the optimal team policy of Dec-POMDP formulations of the problem domains [29] while solving the traditional I-DID using the exact DMU method [34]. For both the traditional I-DID (**Trad. I-DID**) and augmented I-DID (**Aug. I-DID**) frameworks, we utilized $\{\ ^{0}_{j}$ models of level 0 agent j that differ either in initial beliefs or in the frame itself. In the running of the I-DID, all level 0 j's traditional models, $m_{j,0}$, are evenly weighted by level 1 agent i and solved as DIDs. To run the augmented I-DID, we maintained the top K of $\{\ ^{0}_{j}$ - K (in Table. 2) traditional models, $m_{j,0}$, and as many learning models, $m'_{j,0}$, as i's policies in $\hat{\Pi}_i$ each with the same belief over the state space as i's. We then employed two ways to weight both the non-learned policies (from $m_{j,0}$) and learned policies (from $m'_{j,0}$) in agent i's level 1 belief: (a) **Uniform**: all policies are evenly weighted; (b) **Diverse**: policies with larger expected utility are assigned proportionally larger weights.

Performance Evaluation. In Table 2, we evaluate the performance in terms of the expected utility values of level 1 i's policies obtained from solving three cooperative problem domains. We observe that the augmented I-DID framework significantly outperforms the traditional I-DID where level 0 agent j does not learn. The augmented I-DID solutions approach **DP-JESP** that generates the globally optimal team behavior. In cooperative games, the globally optimal solution is the pareto optimal nash equilibrium. We observe that larger the weights on the learned policies, the better the quality of i's policies. This restates the importance of the augmented level 0 j's models that learn. The small gap from the optimal value is mainly due to the beliefs over different models of j. We further looked into the results (see Fig. 8), and found that the augmented I-DID generates the optimal team behavior if i's belief converges to the true model of j (as is done in Dec-POMDP). This is consistent with our observations of the effect of weighting policies. Varying K does not make a visible impact on the performance as the K values are probably large enough to cover a large fraction of collaborative policies of agent j including the optimal team one.

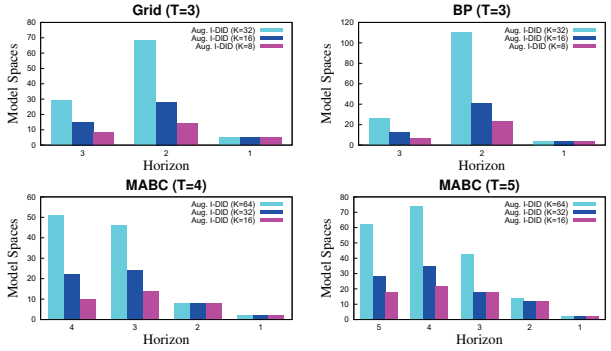


Figure 7: Top-K method reduces the added solution complexity of the augmented I-DID. The complexity is dominated by the number of models in each time step.

In Fig. 7, we show the reduction of model space that occurs due to smaller values of K, which facilitates solutions of the augmented I-DID. Fig. 7 shows the augmented level 0 models increases the model space compared to the traditional I-DID (red bar). However, the growth in the number of models has been effectively prevented using the previous model clustering method (DMU) and the proposed top-K technique.

4.2 Applications to Ad Hoc Teams

Experimental Settings. We also tested the performance of our online implementation of the algorithm AUGMENTED I-DID in ad hoc applications involving different types of teammates. We also performed a baseline comparison with a well-known ad hoc planner - OPAT. The goal of this experiment is to test if augmented I-DIDs can adapt well when its teammates' policies are not particularly effective in advancing the joint goal. We evaluate their ability to adapt effectively against three types of teammates; (a) **Random** - when the teammate plays according to a randomly generated action sequence for the entire length of the run. Some predefined random seeds are used to guarantee that each test had the same action sequences. (b) **Predefined** - when the teammate plays according to some predefined patterns which are sequences of random actions with some fixed repeated lengths that are randomly chosen at the beginning of each run. For example, if the action pattern is"1324" and the repetition value is 2, the resulting action sequence will be "11332244". (c) **Optimal** - when the teammate plays rationally and adaptively. In the case of OPAT, an optimal MMDP policy of teammate is computed offline by value iteration.

In order to speed up the generation of RL models at level 0, we implemented an approximate version of our generalized MCESP called the Sample Average Approximation (MCESP-SAA) that estimates action values by taking a fixed number of samples and then comparing the sample averages [25]. For this set of experiments, we used n=25 sample trajectories to compute the approximate value of the policy that generated them for MCESP-SAA. We set α=0.9, and terminate the RL (line 15 in Fig. 5) if no policy changes are recommended after taking n samples of the value of each observation sequence-action pair [25]. We also tested with some domain-specific seed policies to investigate speedup in the

Table 3: Baseline Comparison with OPAT with different types of teammates. Each datapoint is the average of 10 runs.

Ad Hoc Teammate	OPAT	Aug. IDID
Grid *T=20, look-ahead=3*		
Random	12.25 ◯1.26	14.2 ◯0.84
Predefined	11.7 ◯1.63	16.85 ◯1.35
Optimal	28.35 ◯2.4	27.96 ◯1.92
BP *T=20, look-ahead=3*		
Random	29.26 ◯2.17	36.15 ◯1.95
Predefined	41.1 ◯1.55	54.43 ◯3.38
Optimal	52.11 ◯0.48	59.2 ◯1.55
MABC *T=20, look-ahead=3*		
Random	9.68 ◯1.37	12.13 ◯1.08
Predefined	12.8 ◯0.65	13.22 ◯0.21
Optimal	16.64 ◯0.28	15.97 ◯1.31

convergence of MCESP. We ran the simulations and reported the average (of 10 trials) cumulative value at the end of 20 steps with the discount factor = 0.9. We show that the augmented I-DID solution *significantly* outperforms OPAT solutions in all problem domains for *Random* and *predefined* teammates while performing comparably for *optimal* ones.

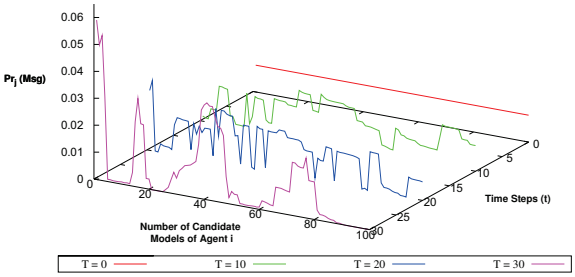


Figure 8: MABC simulation belief updates over 30 steps showing the distribution approaching the agent j's true type if it is present in agent i's candidate model space

Comparison with OPAT. In ad hoc teams, it may be noted that the agents may not be aware that they take part in a team or they may not be capable of working efficiently with other agents. Therefore, we also implemented a generalized version of the OPAT algorithm – that accounts for partial observability – to simulate ad hoc teamwork in partially observable settings and used this as the baseline for comparison with our online implementation of Augmented I-DIDs. The fundamental idea behind OPAT is that it solves a series of stage games, one for each step, and then using biased adaptive play to choose actions. Note that OPAT in its original form assumes full observability of the environmental state and joint actions, and the availability of a black box generative model for drawing samples of team actions. We relax the observability assumption in our generalized implementation by considering partial observability. We maintain similar simulation settings as in augmented I-DIDs such as the random seeds for generating *Random* and *Predefined* teammates. The OPAT simulations with several sample sizes were run also for 20 steps and the average cumulative reward over 10 trials was reported.

Performance Evaluation. In Table 3, we show how augmented I-DIDs compare with OPAT for three types of ad hoc teammates in online simulations spanning 20 steps with a look-ahead of 3 for the three ad hoc domains. At each step, we maintain top 100 candidate models in agent i's I-DID and generate a 3 step look-ahead policy for computing the action at each step. Each data point represents the average cumulative reward of 10 trials over some fixed time T, and the standard deviation. We computed the *Student's (Unpaired) T-Test* for each pair of data points in order to measure the significance of the results. In these empirical results, we notice that I-DIDs *significantly* outperform OPAT especially when the other agents are *random* or *predefined* teammates in all three problem domains except in the case of the MABC domain with a *predefined* teammate where the improvement over OPAT was *not quite* significant (2-tailed P-value = 0.0676). The reason for augmented I-DID's good performance may be attributed to the fact that if the true model of the other agent is present in agent i's candidate model space, over time, the augmented I-DID's belief update facilitates the belief distribution over the models to increase over the agent j's true model (See Fig. 8). However, although Aug. I-DIDs performed significantly better than OPAT when faced with *optimal* teammates only for the BP domain, the results from the other domains were comparable to OPAT. As expected, we noticed that both OPAT and Aug. I-DIDs each played the game better against *optimal* teammates as opposed to *random* or *predefined* ones.

Timing Results. In the Fig. 9, we illustrate the simulation run times for the augmented I-DID and generalized OPAT approaches for solving the three problem domains online. Expectedly, OPAT performs much better in terms of timing because it approximates the problem by solving a series of stage games. In the case of augmented I-DIDs, we observe that generating the learning models took the bulk of the time. We show the learning overhead for Grid, BP, and the MABC domains in red. To reduce this overhead and speedup augmented I-DIDs, we may try other faster RL methods to substitute MCESP.

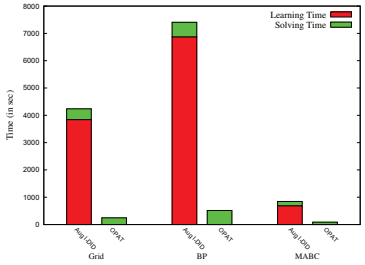


Figure 9: Timing results for augmented I-DID simulations and OPAT with an OPTIMAL teammate.

Scalability Results. While we recognize that MCESP is the bottleneck to scaling augmented I-DIDs for larger problems, in a preliminary study of scalability, we computed the augmented I-DID solutions in a $4 * 4$ grid domain and a box-pushing problem of larger planning horizon. For example, in the larger grid domain for T=3, the optimal team value of 29.6 is achieved by the augmented I-DID comparable to 19.82 obtained by solving the traditional I-DID; while the optimal team policies found by the augmented I-DID are the same as those computed by **DP-JESP** in the T=4 box-pushing problem. Using the augmented I-DID, we are also able to solve MABC for T=5 and get the same optimal team behavior as that obtained by **DP-JESP**. Hence we are optimistic about the scalability of the augmented I-DID.

5. RELATED WORK

Previously, there have been several attempts to address ad hoc coordination in multiagent systems. However, the assumptions that were made by the solutions therein entail that they only address few aspects of the larger problem. For example, some assume that all agents follow pre-specified plans that include roles with synchronized action sequences for each role [10, 12] while some assume that the other agents' behaviors are fixed and known and that all agents' frames are common [2, 6, 31, 32]. However, a good ad hoc agent would model its teammates in an attempt to converge to their true types. These types of agents could differ from one an-

other in their frames in the way their payoffs are structured and/or in the way their mobility/sensing capability is compromised. The framework used in this paper assumes nothing about the behaviors or plans of the other agents and seeks to converge to the agents' respective true models while permitting their frames to differ. Wu et al. on the other hand, simplify the problem by assuming that the state and joint actions are fully observable [33]. In this paper however, we relax this assumption to account for partial observability as experienced in most real-world scenarios. I-POMDPs and I-DIDs provide a formal model general enough to accomodate a wide spectrum of problems including those of ad hoc coordination as we note that most existing work in ad hoc planning are procedural in nature and pertain to only specific (types of) problems. A related solution is *Harsanyi-Bellman Ad Hoc Coordination (HBA)* by Albrecht et al. [3] where they provide a generalized framework that is capable of maintaining a distribution over a set of user-defined behavior types of the opponents, and utilises them in the planning to obtain its optimal actions. However, there is no guarantee that the space of user-defined types considered will include the true type space of the opponents. Broadly, HBA relates pre-defined types to the true type whereas I-DIDs can recognize the true type from all possible types of unknown opponents. Although additionally, Albrecht et al. propose a opponent modelling method by combining case-based reasoning and fictitious play called *conceptual types*, they do not hypothesise behaviors directly, instead they simplify it by hypothesising a world conceptualization underlying an agent's behavior.

6. DISCUSSION AND CONCLUSION

Self-interested individual decision makers face hierarchical (or nested) belief systems in their multiagent planning problems. In this paper, we explicate one negative consequence of bounded rationality: the agent may not behave as an optimal teammate. In the I-DID framework that models individual decision makers modeling other agents, we show that reinforcement learning integrated with the planning allows the models to produce sophisticated policies. For the first time, we see the principled and comprehensive emergence of team behavior in I-DID solutions as demonstrated by our results.

This result facilitated I-DIDs to be applied to ad hoc team settings for which they are just naturally well-suited for. We show that integrating learning in the context of I-DIDs helped us provide a solution to a fundamental challenge in ad hoc teamwork; building a single autonomous agent that can function well in an ad hoc team setting. Not only did augmented I-DIDs compare well with a standard baseline algorithm (OPAT), it adapted well with different kinds of ad hoc teammates.

While individual decision-making frameworks such as interactive POMDPs and I-DIDs are thought to be well suited to non-cooperative domains, we show a way in which they may be applied to cooperative domains as well. Integrating learning while planning provides a middle ground between multiagent planning frameworks such as the Dec-POMDP and joint learning for cooperative domains [24]. Additionally, the augmented I-DID differentiates itself from other centralized cooperative frameworks by focusing on the behavior of an individual agent in a multiagent setting. Immediate lines of future work involve improving the scalability of the framework, and particularly its learning component, by utilizing larger problems.

7. REFERENCES

[1] B. Adam and E. Dekel. Hierarchies of beliefs and common knowledge. *International Journal of Game Theory*, 59(1):189–198, 1993.

[2] N. Agmon and P. Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *AAMAS*, 2012.

[3] S. Albrecht and S. Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. Technical report, School of Informatics, The University of Edinburgh, United Kingdom, 2013.

[4] R. J. Aumann. Interactive epistemology II: Probability. *International Journal of Game Theory*, 28:301–314, 1999.

[5] B. Banerjee, J. Lyle, L. Kraemer, and R. Yellamraju. Solving finite horizon decentralized pomdps by distributed reinforcement learning. In *AAMAS Workshop on MSDM*, pages 9–16, 2012.

[6] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS*, 2011.

[7] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[8] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized pomdps. In *IJCAI*, pages 1287–1292, 2005.

[9] K. Binmore. *Essays on Foundations of Game Theory*. Pittman, 1982.

[10] M. H. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, pages 53–58, 2005.

[11] A. Brandenburger. The power of paradox: Some recent developments in interactive epistemology. *International Journal of Game Theory*, 35:465–492, 2007.

[12] D. H. Browning, M. B. Dias, T. K. Harris, B. Browning, E. G. Jones, B. Argall, M. Veloso, A. Stentz, and A. Rudnicky. Dynamically formed human-robot teams performing coordinated tasks. In *AAAI Spring Symposium "To Boldly Go Where No Human-Robot Team Has Gone Before"*, 2006.

[13] L. Chrisman. Reinforcement learning with perceptual aliasing: the perceptual distinctions approach. In *AAAI*, pages 183–188, 1992.

[14] P. Doshi. Decision making in complex mulitiagent contexts: A tale of two frameworks. *AI Magazine*, 4(33):82–95, 2012.

[15] P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive pomdps: Representations and solutions. *JAAMAS*, 18(3):376–416, 2009.

[16] R. Fagin, J. Geanakoplos, J. Halpern, and M. Vardi. The hierarchical approach to modeling knowledge and common knowledge. *International Journal of Game Theory*, 28, 1999.

[17] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *JAIR*, 24:49–79, 2005.

[18] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715, 2004.

[19] A. K. Mccallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, University of Rochester, 1996.

[20] J. Mertens and S. Zamir. Formulation of bayesian analysis for games with incomplete information. *International Journal of Game Theory*, 14:1–29, 1985.

[21] N. Meuleau, L. Peshkin, K. eung Kim, and L. P. Kaelbling. Learning finite-state controllers for partially observable environments. In *UAI*, pages 427–436, 1999.

[22] B. Ng, K. Boakye, C. Meyers, and A. Wang. Bayes-adaptive interactive pomdps. In *AAAI*, 2012.

[23] D. Nikovski. *State-Aggregation Algorithms for Learning Probabilistic Models for Robot Control*. PhD thesis, Carnegie Mellon University, 2002.

[24] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *JAAMAS*, 11(3):387–434, 2005.

[25] T. J. Perkins. Reinforcement learning for pomdps based on action values and stochastic optimization. In *AAAI*, pages 199–204, 2002.

[26] D. Pynadath and S. Marsella. Minimal mental models. In *AAAI*, pages 1038–1044, 2007.

[27] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized pomdps. In *UAI*, pages 344–351, 2007.

[28] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *JAAMAS*,

17(2):190–250, 2008.

[29] M. Spaan and F. Oliehoek. The multiagent decision process toolbox: Software for decision-theoretic planning in multiagent systems. In *AAMAS Workshop on MSDM*, pages 107–121, 2008.

[30] P. Stone, G. A. Kaminka, S. Kraus, J. R. Rosenschein, and N. Agmon. Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. *Artificial Intelligence*, 2013.

[31] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010.

[32] P. Stone and S. Kraus. To teach or not to teach? decision making under uncertainty in ad hoc teams. In *AAMAS*, 2010.

[33] F. Wu, S. Zilberstein, and X. Chen. Online planning for ad hoc autonomous agent teams. In *IJCAI*, pages 439–445, 2011.

[34] Y. Zeng and P. Doshi. Exploiting model equivalences for solving interactive dynamic influence diagrams. *JAIR*, 43:211–255, 2012.