Generalized and Bounded Policy Iteration for Finitely-Nested Interactive POMDPs: Scaling Up

Ekhlas Sonu
THINC Lab, Dept. of Computer Science
University of Georgia
Athens, GA. 30602
esonu@uga.edu

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

ABSTRACT

Policy iteration algorithms for partially observable Markov decision processes (POMDP) offer the benefits of quick convergence and the ability to operate directly on the solution, which usually takes the form of a finite state controller. However, the controller tends to grow quickly in size across iterations due to which its evaluation and improvement become costly. Bounded policy iteration provides a way of keeping the controller size fixed while improving it monotonically until convergence, although it is susceptible to getting trapped in local optima. Despite these limitations, policy iteration algorithms are viable alternatives to value iteration.

In this paper, we generalize the bounded policy iteration technique to problems involving multiple agents. Specifically, we show how we may perform policy iteration in settings formalized by the interactive POMDP framework. Although policy iteration has been extended to decentralized POMDPs, the context there is strictly cooperative. Its generalization here makes it useful in non-cooperative settings as well. As interactive POMDPs involve modeling others, we ascribe nested controllers to predict others' actions, with the benefit that the controllers compactly represent the model space. We evaluate our approach on multiple problem domains, and demonstrate its properties and scalability.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.8 [**Problem Solving, Control Methods, and Search**]: Dynamic Programming

General Terms

Theory, Performance

Keywords

policy iteration, decision making, multiagent settings

1. INTRODUCTION

Decision making in sequential and partially observable, single-agent settings is typically formalized by partially observable Markov decision processes (POMDP) [11, 19]. In the multiagent context, the decision making is exponentially harder, and depending on the type and perspective of the interaction, is formalized by one of multiple frameworks. In cooperative settings requiring solutions for all agents, decentralized POMDPs [2] sufficiently model the joint decision-making problem. Additionally, interactive POMDPs [7] formalize the decision-making problem of an individual agent in a multiagent setting, which need not be cooperative. Both these frameworks generalize POMDPs in different ways and have relied on extending approximation techniques for POM DPs to their own formalizations for tractability.

One such technique involves searching the solution space directly. Initially proposed in the context of POMDPs [9], the technique represents the solution, called the policy, as a *finite state controller* and iteratively improves it until convergence. The benefit is that the controller typically converges before its value converges across all states and it is useful for an infinite horizon. However, nodes in the controller grow quickly making it computationally difficult to evaluate the controller and continually improve it. Bounded policy iteration (BPI) avoids this growth by keeping the size of the controller fixed as it seeks to monotonically improve the controller's value by replacing a node and its edges with another one [15]. Expectedly, this scales POMDP solutions to larger problems, but the controllers often converge to a local optima. Nevertheless, the benefits of this approach are substantial enough that it has been extended to decentralized POMDPs [3] leading to improved scalability.

In this paper, we introduce a generalization of BPI to the context of finitely-nested interactive POMDPs (I-POMDP) thereby improving on previous approximation techniques on two important fronts: we may solve larger problem domains and generate solutions of much better quality. In contrast to decentralized POMDPs, I-POMDPs do not assume common knowledge of initial beliefs of agents or common rewards, due to which others' beliefs, capabilities and preferences are modeled. They allow for others modeling other agents, and terminate the nesting at some finite level. Recent applications of I-POMDPs testify to its significance and growing appeal. They are being used to explore strategies for countering money laundering by terrorists [12, 13] and enhanced to include trust levels for facilitating defense simulations [17, 18]. They have been used to produce winning strategies for playing the lemonade stand game [21] and even modified to include empirical models for simulating human behavioral data pertaining to strategic thought and action [6].

Being a generalization of POMDPs, solutions of I-POMDPs are also affected by the curses of dimensionality and history

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.
Copyright © 2012, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

that affect POMDPs [14]. The dimensionality hurdle is further aggravated because an agent maintains belief not only over the physical state but also over the models of the other agents, which grow over time as the agents act and observe. Previous approximations for finitely-nested I-POMDPs include interactive particle filtering [4] and interactive point-based value iteration [5]. Particle filtering seeks to mitigate the adverse effect of the curse of dimensionality by forming a sampled, recursive representation of the agent's nested belief, which is then propagated over time. However, its efficiency is still impacted by the number of models because this increases the need for more samples, and it is better suited for solving I-POMDPs with a given prior belief. Interactive point-based value iteration generalizes point-based value iteration [14] to multiagent settings, and reduces the effect of the curse of history. While this approach significantly scales I-POMDPs to longer horizons, we must include all reachable models of the other agent in the state space, which grows exponentially over time, thereby making it susceptible to the dimensionality hurdle. We may also group together models that are behaviorally equivalent resulting in a partition of the model space of the other agent into a finite number of equivalence classes [16]. This approach, analogously to using finite state controllers, allows a compact representation of the model space but computing the exact equivalence requires solving the models.

In generalizing BPI, the interactive state space of the subject agent is reformulated to include the physical states and the set of nodes in a controller without loss of generality. For multiple other agents with differing capabilities and preferences, we include multiple controllers, one for each other agent. Each iteration involves evaluating and possibly improving the nested controller of the other agent followed by improvement of the subject agent's controller. In order to account for the dynamically changing state space, we interleave the evaluation and improvement of controllers at different levels. This approach differs from BPI's implementation in decentralized POMDPs where controllers for each agent are improved independently, but a correlation device is introduced for coordination among them. Such a shared source of randomness may not be feasible in non-cooperative settings. Importantly, we observe that convergence of the subject agent's controller is often dependent on the lower-level controllers converging first.

As we mentioned previously, the benefit is that the space of possible models may be compactly represented using the set of nodes in a controller. On the other hand, the presence of controller(s) embedded in the state space makes evaluation and improvement for the subject agent much more expensive than in the context of POMDPs or decentralized POMDPs. We call our approach *interactive BPI* and experimentally evaluate its properties using benchmark problems. In particular, we show that the converged controller for the subject agent generates solutions of good quality in proportionately less time compared to results reported by the previous best I-POMDP approximation. Ultimately, this allows the application of I-POMDPs to scale to more realistic domains with reduced trade off, as we demonstrate by applying the technique to larger problem domains.

2. BACKGROUND

We briefly review the framework of finitely-nested I-POMDPs and outline previous policy iteration in the context of POMDPs.

2.1 Interactive POMDP

A finitely-nested I-POMDP [7] for an agent i with strategy level, l, interacting with another agent j is defined using the tuple:

$$\text{I-POMDP}_{i,l} = \langle IS_{i,l}, A, T_i, \Omega_i, O_i, R_i, OC_i \rangle$$

where:

- $IS_{i,l}$ denotes the set of *interactive states* defined as, $IS_{i,l} = S \times M_{j,l-1}$, where $M_{j,l-1} = \{\Theta_{j,l-1} \cup SM_j\}$, for $l \geq 1$, and $IS_{i,0} = S$, where S is the set of physical states. $\Theta_{j,l-1}$ is the set of computable, intentional models ascribed to agent j: $\theta_{j,l-1} = \langle b_{j,l-1}, \hat{\theta}_{j,l-1} \rangle$, where $b_{j,l-1}$ is agent j's level $l-1$ belief, $b_{j,l-1} \in \triangle(IS_{j,l-1})$, and $\hat{\theta}_{j,l-1} = \langle A, T_j, \Omega_j, O_j, R_j, OC_j \rangle$, is j's frame. Here, j is assumed to be Bayes-rational. For simplicity, we assume that the frame of agent j is known and remains fixed; it need not be the same as that of agent i. SM_j is the set of subintentional models of j. For the sake of simplicity, in this paper we focus on ascribing intentional models only.
- $A = A_i \times A_j$ is the set of joint actions of all agents.

  The remaining parameters – transition function, T_i, observations, Ω_i, observation function, O_i, preference function, R_i, and the optimality criterion, OC_i – have their usual meaning as in POMDPs [11]. Note that the optimality criterion here is the discounted infinite horizon sum.

An agent's belief over its interactive states is a sufficient statistic, fully summarizing the agent's observation history. Beliefs are updated after the agent's action and observation using Bayes rule. Two differences complicate the belief update in multiagent settings. First, since the state of the physical environment depends on the actions performed by both agents the prediction of how it changes has to be made based on the probabilities of various actions of the other agent. Probabilities of other's actions are obtained by solving its models. Second, changes in the models of other agents have to be included in the update. The changes reflect the other's observations and, if they are modeled intentionally, the update of other agent's beliefs. In this case, the agent has to update its beliefs about the other agent based on what it anticipates the other agent observes and how it updates.

Given the extended belief update, solution to an I-POMDP is a *policy*, analogous to that in a POMDP. Using the Bellman equation, each belief state in an I-POMDP has a value which is the maximum payoff the agent can expect starting from that belief state and over the future. Gmytrasiewicz and Doshi [7] provide additional details on I-POMDPs and how they compare with other multiagent frameworks.

2.2 Policy Iteration

Policy iteration provides an alternative to iterating over the value function, by searching directly over the policy space. While the traditional representation of a policy is as a function from beliefs to actions and iterating over these functions is indeed possible [20], a more convenient representation for the purpose of policy iteration is as a *finite state controller*. At any point in the iteration, the controller represents the infinite horizon policy of the agent.

We may define a simple controller for an agent i, as:

$$\pi_i = \langle \mathcal{N}_i, \mathcal{E}_i, \mathcal{L}_i, \mathcal{T}_i \rangle$$

where $\mathcal{N}_i$ is the set of nodes in the controller, $\mathcal{E}_i$ is the set of edge labels which are observations, Ω_i, in a POMDP, $\mathcal{L}_i$ is the mapping from each node to an action, $\mathcal{L}_i : \mathcal{N}_i \rightarrow A_i$, and $\mathcal{T}_i$ is the edge transition (successor) function, $\mathcal{T}_i : \mathcal{N}_i \times A_i \times \Omega_i \rightarrow \mathcal{N}_i$. For convenience, we group together $\mathcal{E}_i$, $\mathcal{L}_i$ and $\mathcal{T}_i$ in $\hat{f}_i$.

Policy iteration algorithms improve the value of the controller by interleaving steps of evaluating the policy with improving it by backing up the linear vectors that make up the value function. We may view each node in the controller as representing the action and value associated with a vector. As the value function is improved, new vectors may be introduced causing additional nodes in the controller, while some nodes may be dropped if their corresponding vectors are dominated at all states by some other vector [9].

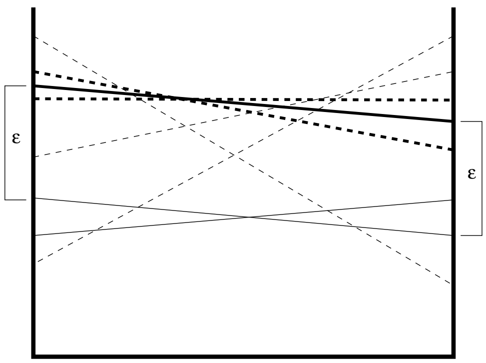

Figure 1: Value vector (solid line in bold), representing a node in the improved controller, is a convex combination of the two dashed backed-up vectors in bold and point-wise dominates a vector that constitutes the value function of the previous controller by ϵ. The dashed vectors constitute the optimal, backed up value function.

Controllers often grow exponentially in size during improvement making evaluation and further improvement intractable. Poupart and Boutilier [15] show that the controller size may be minimized and, in fact kept bounded, in two ways: First, we may replace a node whose corresponding vector is dominated by a convex combination of updated vectors. A convex-combination vector passing through the point of intersection of the combined vectors and parallel to the dominated vector is selected, as we illustrate in Fig. 1. This replaces multiple vectors with a single one and allows us to prune nodes, which previously would not have been removed. This leads to a controller whose transitions due to observations, $\mathcal{T}_i$, may be stochastic. Second, note that if the controller hasn't converged, a backup is guaranteed to improve it. Thus, we may replace some node with another that represents a convex combination of backed up vectors, and whose value is better. This causes the action mapping, $\mathcal{L}_i$, to be stochastic as well. Of course, the technique is susceptible to local optima.

3. GENERALIZED POLICY ITERATION

We generalize the bounded policy iteration technique to the context of I-POMDPs nested to a finite level, l. Notice that a finite state controller partitions the intentional model space, $\Theta_{j,l-1}$, among its nodes. *This is because for a belief in any model in $\Theta_{j,l-1}$, a node exists in the controller that will provide the (boundedly) optimal action(s).* Therefore, the interactive state space, $IS_{i,l} = S \times \Theta_{j,l-1}$, becomes:

$$IS_{i,l} = S \times \mathcal{F}_{j,l-1}$$

where $f_{j,l-1} \in \mathcal{F}_{j,l-1}$ is, $f_{j,l-1} = \langle n_{j,l-1}, \hat{f}_{j,l-1}, \hat{\theta}_{j,l-1} \rangle$. Here, $n_{j,l-1}$ is a node in the set of nodes in the controller, $n_{j,l-1} \in \mathcal{N}_{j,l-1}$; $\hat{f}_{j,l-1}$ is as defined previously in Section 2.2 for a controller; and $\hat{\theta}_{j,l-1}$ is j's frame and is fixed. Notice that the controller represents an initial solution for the entire intentional model space. If there are K other agents with differing capabilities and preferences, the interactive state space becomes, $IS_{i,l} = S \times_{k=1}^{K} \mathcal{F}_{k,l-1}$, where $\mathcal{F}_{k,l-1}$ represents a different controller for each k. This is because the agents differ in their frames and consequently, how their controllers evolve.

Because the set of nodes in $\mathcal{F}_{j,l-1}$ is finite, an important benefit of the above representation is that the infinite model space is represented using a finite node space, thereby making the interactive state space finite as well (assuming that the physical state space is finite). The large model space is often a hurdle for previous approximation techniques that operate on it, such as the interactive point-based value iteration [5]. This motivated arbitrary limitations on the models and on how they evolve, which are no longer necessary. Other, parameterized representations of the model space are also under investigation [8].

Let $\mathcal{F}_{i,l}$ be an initial, level l controller for the subject agent i. Next, we move to evaluating and improving agent i's controller iteratively. Because the controller of the other agent is embedded in i's state space, these steps utilize updated controllers at the lower levels as well thereby generalizing the iterations to multiagent settings.

3.1 Policy Evaluation

As we mentioned previously, each node, $n_{i,l}$, in the controller is associated with a vector of values, $V(\cdot, n_{i,l})$, that gives the expected (converged) value, at each interactive state, of following the controller beginning from that node. In the context of I-POMDPs, this is a $|S \times \mathcal{N}_{j,l-1}|$-dimensional vector for each node. A step of policy evaluation involves computing this vector for each node in the controller. We may do this by solving the following system of linear equations:

$$\begin{aligned} V(s, n_{j,l-1}, n_{i,l}) = &\sum_{a_i \in A_i} Pr(a_i|n_{i,l}) \sum_{a_j \in A_j} Pr(a_j|n_{j,l-1}) \\ \times \Bigg\{ & R_i(s, a_i, a_j) + \gamma \sum_{o_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s')\ O_i(s', a_i, a_j, \\ o_i) \times & \sum_{o_j} O_j(s', a_i, a_j, o_j)\ Pr(n'_{j,l-1}|n_{j,l-1}, a_j, o_j) \\ \times & \sum_{n'_{i,l}} Pr(n'_{i.l}|n_{i,l}, a_i, o_i)\ V(s', n'_{j,l-1}, n'_{i,l}) \Bigg\} \\ \forall s, & n_{j,l-1}, n_{i,l} \end{aligned} \quad (1)$$

In Eq. 1, we compute the expectation over i's actions because multiple actions are possible from a single node of the stochastic controller. Given the multiagent setting, actions of both agents appear in the transition, observation and reward functions in the equation. The terms $Pr(a_i|n_{i,l})$, $Pr(n'_{i.l}|n_{i,l}, a_i, o_i)$ and $Pr(a_j|n_{j,l-1})$, $Pr(n'_{j.l-1}|n_{j,l-1}, a_j, o_j)$ are obtained from $\hat{f}_{i,l}$ and $\hat{f}_{j,l-1}$, respectively, and O_j is obtained from j's frame; all of which are present in $f_{j,l-1}$. Equation 1 is defined for each physical state, s, j's controller node, $n_{j,l-1}$, and i's controller node, $n_{i,l}$. Notice that the update of the other agent's belief is represented using a transition from one node to another by the term, $Pr(n'_{j.l-1}|n_{j,l-1}, a_j, o_j)$.

Solution of the system results in a vector of values for each node in agent i's controller. In the next step, we improve the controller by introducing new nodes with updated value vectors that may uniformly dominate, possibly in combination, those of an existing node and prune the dominated node.

3.2 Policy Improvement

The controller is improved by evaluating whether a node, $n_{i,l}$, in i's controller may be replaced with another whose value, possibly a convex combination of the updated vectors, is better at all interactive states. Instead of first updating the value vectors using the backup operation and then checking for pointwise dominance, Poupart and Boutilier [15] proposed to integrate the two in a single linear program. Our linear program differs from that for a POMDP in involving additional terms related to j's controller. We show this linear program below:

$$
\begin{aligned}
& max \;\; \epsilon \\
& \;\; s.t. \; V(s, n_{j,l-1}, n_{i,l}) + \epsilon \leq \sum_{a_i} c_{a_i} \sum_{a_j} Pr(a_j|n_{j,l-1}) \\
& \times \Bigg\{ R_i(s, a_i, a_j) + \gamma \sum_{o_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s') \\
& \times O_i(s', a_i, a_j, o_i) \sum_{o_j} O_j(s', a_i, a_j, o_j) \\
& \times Pr(n'_{j,l-1}|n_{j,l-1}, a_j, o_j) \sum_{n^{o_i}_{i,l}} c_{a_i, n^{o_i}_{i,l}} V(s', n'_{j,l-1}, n^{o_i}_{i,l}) \Bigg\} \\
& \forall \;\; s, n_{j,l-1}; \\
& \sum_{a_i} c_{a_i} = 1; \;\; \sum_{n^{o_i}_{i,l}} c_{a_i, n^{o_i}_{i,l}} = c_{a_i} \;\; \forall a_i, o_i, n^{o_i}_{i,l}; \\
& \;\; c_{a_i, n^{o_i}_{i,l}} \geq 0 \;\; \forall a_i, o_i, n^{o_i}_{i,l}; \; c_{a_i} \geq 0 \;\; \forall a_i
\end{aligned}
\qquad (2)
$$

The value function terms in Eq. 2 are obtained from the previous policy evaluation step. We run this linear program for each of i's nodes until a positive ϵ is obtained for a node. $\epsilon > 0$ signals that node, $n_{i,l}$, may be pruned because a convex combination of the backed up value vectors dominate it at least by ϵ at all physical states and nodes of j's controller. Because a single ϵ value is sought for all s, $n_{j,l-1}$, the dominating value vector will be parallel to the pruned one. The solution of the program allows us to construct a new node (say, $n'_{i,l}$) with stochastic actions of agent i as, $Pr(a_i|n'_{i,l}) = c_{a_i}$, and the transition probability to a node $(n^{o_i}_{i,l})$ on performing action a_i and receiving observation o_i as, $Pr(n^{o_i}_{i,l}|n'_{i,l}, a_i, o_i) = c_{a_i, n^{o_i}_{i,l}}$.

We iterate over the evaluation and improvement steps until a positive ϵ is not obtained for any node in i's current controller and the value vectors from Eq. 1 have fixated for every node. Because of the strategy of obtaining a value vector that is parallel to the pruned one, the iterations may converge on a peculiar local optima in which all the value vectors are tangential to the intersections of the exact value function at that step, due to which no further improvement using Eq. 2 is possible. Poupart and Boutilier [15] mention a simple approach of potentially dislodging from the local optima, which is applicable in the context of I-POMDPs as well. Specifically, we pick a belief reachable from the tangential belief and add a node to the controller that corresponds to the value vector associated with the reachable belief. This allows the value of the node representing the tangential vector to improve.

3.3 Nested Controllers

Given that the other agent's controller is embedded in agent i's interactive state space, a naive but efficient approach would be to iteratively improve i's controller while holding j's controller in the state space fixed. However, the corresponding solution will likely be poor as better quality controllers may be available to predict the other agent's actions. This is particularly relevant because I-POMDPs model the other agent as being rational. Subsequently, a more sophisticated approach is to interleave improvements of the other agent's controller with improvements of agent i's controller. However, not only is this approach computationally more intensive, but agent i's interactive state space may change dynamically at every iteration.

Fortunately, the previously detailed approach of BPI keeps the number of nodes fixed as it seeks to improve the controller. Consequently, the size of agent i's interactive state space – and that of j if the level of nesting is greater than 1 leading to embedded controllers in j's interactive state space – remains fixed. Although nodes may be added to j's controller initially or to escape local optima, we perform these iterations before beginning the improvement of the higher-level, agent i's, controller. After this point, the subject agent's interactive state space remains fixed in size, although the individual states may change across iterations due to updates in the stochastic distributions, $\hat{f}_{j,l-1}$.

Finally, at level 0 the I-POMDP collapses into a POMDP. Consequently, we may utilize the traditional BPI [15] for POMDPs in order to evaluate and improve the level 0 agent's controller.

4. ALGORITHM

Algorithm 1 outlines the procedure, labeled Interactive BPI (I-BPI), for performing BPI in the context of finitely nested I-POMDPs. It begins by creating a trivial controller having a single node with a randomly selected action, at each level for agent i or j as appropriate. The interactive state space is then reformulated to include the nodes from the other agent's controller (lines 1-2) as we mentioned previously. In order to apply BPI on a controller of reasonable size, we perform a single full backup at each level to obtain controllers of size $|A_i|$ or $|A_j|$, as appropriate (line 3). If there are more agents with differing frames, then a controller is initialized for each other agent. Subsequent steps are performed for each of these controllers.

Algorithm 2, Evaluate&Improve, then recursively performs

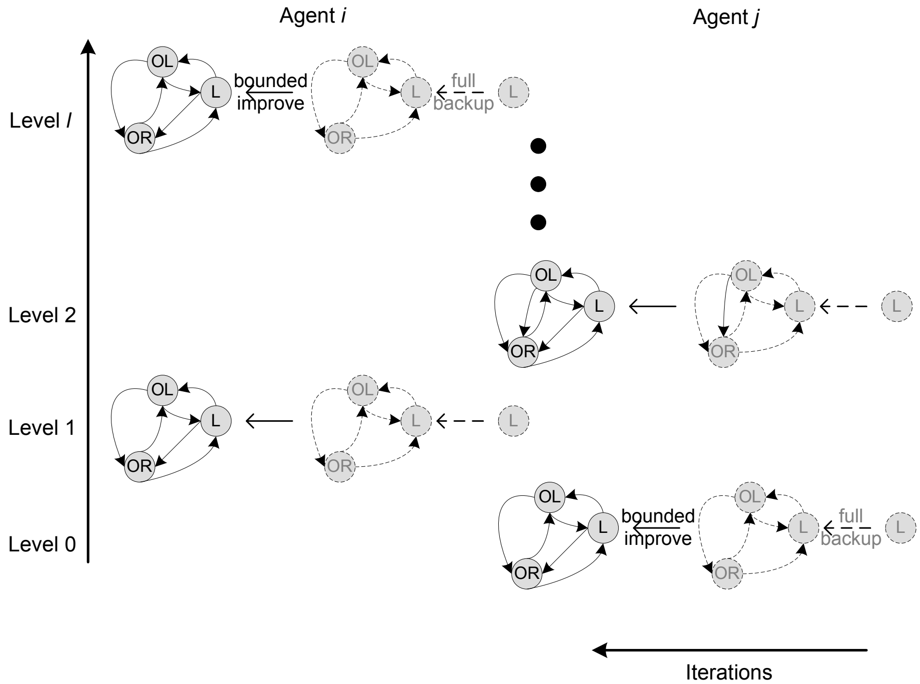

Figure 2: Recursive invocations lead to evaluation and improvement beginning at the bottom and up the nesting levels. Controllers were initialized with a single node and a full backup takes place in the previous iteration (shown dashed and greyed out). Current iteration involves evaluation and bounded improvement of the nested controller bottom up as the recursion unwinds. We demonstrate this process in the context of an example: the well-known multiagent tiger problem. The node labels represent actions of the agents: listen (L), open the left door (OL), and open the right door (OR).

Algorithm 1 Interactive BPI for I-POMDPs

Interactive BPI (I-POMDP: $\theta_{i,l}$) **returns** solution, $\pi^*_{i,l}$

1: Recursively initialize controllers, $\pi_{i(j),l}$, for both agents, such that $|\mathcal{N}_{i(j),l}| = 1$, down to level 0
2: Reformulate, $IS_{i(j),l} = S \times \mathcal{F}_{j(i),l}$ at each l in $\theta_{i,l}$
3: Beginning with level, $l = 0$, perform a single-step full backup at each level, l, resulting in $|\mathcal{N}_{i(j),l}| \leq |A_{i(j)}|$ nodes in a controller, $\pi_{i(j),l}$
4: **repeat**
5:    **repeat**
6:       $\pi_{i,l} \leftarrow$ Evaluate&Improve ($\pi_{i,l}$)
7:    **until** no more improvement is possible
8:    Push controllers at each level from local optima
9: **until** no more escapes are possible
10: **return** converged (nested) controller, $\pi^*_{i,l}$

Algorithm 2 Evaluation and bounded improvement of the nested controllers

Evaluate&Improve (nested controller: $\pi_{i(j),l}$) **returns** controller, $\pi'_{i(j),l}$

1: **if** $l \geq 1$ **then**
2:    $\pi_{j(i),l-1} \leftarrow$ Evaluate&Improve ($\pi_{j(i),l-1}$)
3: **if** l=0 **then**
4:    Evaluate controller, $\pi_{i(j),0} = \langle \mathcal{N}_{i(j)}, \mathcal{E}_{i(j)}, \mathcal{L}_{i(j)}, \mathcal{T}_{i(j)} \rangle$
5:    Improve controller, if possible, analogously to a POMDP [15]
6: **else**
7:    Evaluate controller, $\pi_{i(j),l} = \langle \mathcal{N}_{i(j),l}, \mathcal{E}_{i(j),l}, \mathcal{L}_{i(j),l}, \mathcal{T}_{i(j),l} \rangle$, using Eq. 1
8:    Improve controller, if possible, while keeping $|\mathcal{N}_{i(j),l}|$ fixed using Eq. 2
9: **return** improved controller, $\pi'_{i(j),l}$

a single step of evaluation of the nested controller and its bounded improvement (lines 1-2). For the lowest-level controller, the evaluation and improvement proceeds as outlined by Poupart and Boutilier [15] in the context of POMDPs. At levels 1 and above, we evaluate the controller using Eq. 1 and improve it while keeping the number of nodes fixed using Eq. 2.

The presence of a nested controller leads to novel challenges. Observe that I-BPI interleaves the evaluation and improvement of the controllers at the different levels. The alternate technique would be to evaluate and improve the controller of the lower level until convergence. The former approach better facilitates anytime behavior in comparison to the latter in which the higher-level controller may not be improved for many iterations until the lower-level controller has converged. Of course, the higher-level controllers in the two approaches may not converge to the same local optima. Furthermore, notice that the bounded improvement of j's or i's lower-level controller while keeping the number of nodes fixed still alters the interactive state space because $\hat{f}_{j,l-1}$ or $\hat{f}_{i,l-2}$ changes. Consequently, $IS_{i,l}$ or $IS_{j,l-1}$ may dynamically change at each iteration. Therefore, an alternate technique of evaluating the controllers at all levels first followed by recursively improving them is not feasible because the previous value evaluation of a level l controller is invalidated when lower-level controllers improve.

We illustrate a step within I-BPI on a level l I-POMDP in Fig. 2. On convergence, Algorithm 1 attempts to push the nested controller past any local optima, by escaping it for the lower-level controllers first (line 8). When this is no longer possible, the converged nested controller is returned as the solution of the level l I-POMDP.

Computational Savings In general, the space of models ascribed to the other agent is continuous because each candidate model includes a possible belief as well. I-BPI reformulates the interactive state space by mapping the space of models to a finite set of nodes in the other agent's controller, without loss of generality. However, if we limit the model space and let $|\Theta|$ be a bound on the number of models acribed to one other agent. Then, the interactive state space for K other agents contains $(K|\Theta|)^l$ models at all levels of the nesting. Mapping $|\Theta|$ to $|N|$ nodes of a controller, whose size remains fixed, we obtain a set of size $(K|N|)^l$. This space is significantly smaller because usually, $|N| \ll |\Theta|$, leading to much mitigated impacts of the curses of both dimensionality and history. We empirically demonstrate the effect of these savings next.

5. EXPERIMENTS

We implemented Algorithms 1 and 2 for I-BPI shown in the previous section, and evaluated its properties on two benchmark problem domains: a non-cooperative version of the multiagent tiger problem and a cooperative version of the multiagent machine maintainence (MM) problem, each of which has two agents, i and j. Doshi and Gmytrasiewicz [4] provide details on these problem domains. While these problems have small dimensions, they have been used as benchmarks for previous I-POMDP approximation techniques, such as the interactive particle filter [4] and interactive point-based value iteration (I-PBVI) [5], which employ value iteration. In addition to these toy problems, we evaluate I-BPI's performance and demonstrate scalability using two larger problem domains: autonomous unmanned aerial vehicle (AUAV) reconnaissance problem on a 5×5 grid and the money laundering problem [13], both of which are non-cooperative.

The AUAV problem involves reconnaissance in a 5×5 grid in which an AUAV is tasked with capturing a fugitive who seeks to escape to a safe house (fixed at a predetermined grid location). We model the AUAV as a level 1 I-POMDP. The physical state of the fugitive at any given time is its relative position to the safe house and that of the UAV is its relative position w.r.t. the fugitive. This formulation leads to a physical state space of 81 states for the AUAV and 25 for the fugitive. Each agent may take one of 5 actions of moving in one of the four cardinal directions or listening to get observation about the target's location. We assumed that the actions taken by both agents on the grid are deterministic. The 4 observations for each agent allow it to sense the cardinal direction of its target relative to its own location. We assume that the observations are noisy.

The money laundering problem, introduced by Ng et al. [13], is a game between law enforcement (blue team) and the money launderers (red team) who aim to move their assets from a 'dirty' pot to a 'clean' one through a series of financial transactions while evading capture by the blue team. The blue team can place sensors at various locations such as bank accounts, trusts and real estate to detect the presence of the 'dirty' money. The physical state is defined by the joint location of the dirty money and that of the sensor. The red team may perform any of the three nondeterministic actions of placement, layering or integration to move its assets from one location to another or it could listen to gain noisy information about the location of the blue team's sensor. The blue team may place its sensors in one of eight locations or it could confiscate the assets of the red team. This problem exhibits a physical state space of 99 states for the subject agent (blue team), 9 actions for the subject agent and 4 for the opponent, and 11 observations for the subject and 4 for the other.

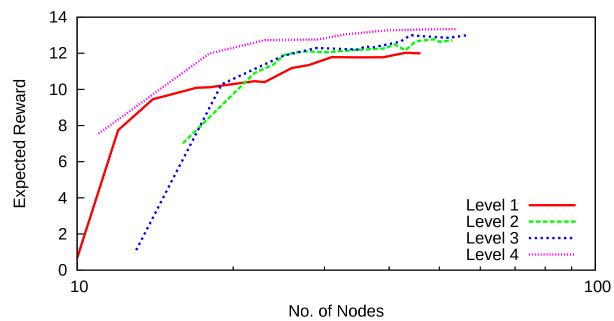

Figure 3: Average rewards for the multiagent tiger problem generally improve until convergence as we allocate more nodes to controllers to facilitate escaping local optima, in I-BPI. As we may expect, controllers generated for more strategically nested I-POMDPs eventually lead to better rewards. Although we do not show the variance for clarity, it tends to be small.

We begin by focusing on the tiger problem, and noting the average reward obtained from *simulating* converged controllers of different node sizes, and of different levels, in Fig. 3. Observe the generally increasing trend of the average rewards as the controllers increase in size on escaping from local optima. This property lends itself to an anytime behavior for I-BPI. Each reward data point is averaged over 5 trials each involving 100 initial beliefs randomly generated, and for each belief, between 100 and 1000 simulation runs were carried out.

Next, in Table 1, we report the average discounted rewards obtained from simulating the controllers that I-BPI generates along with the associated I-BPI run times, as we scale in the context of the number of nesting levels. We compare these rewards with those reported using the previous best I-POMDP approximation technique, I-PBVI [5], where the latter are obtained from actual simulation runs as well. Although I-BPI's controller is for an infinite number of time steps, we limit our runs in the simulations to a finite number of steps in order to compare with I-PBVI. Notice that I-PBVI is able to reasonably scale up to two levels only and the corresponding rewards are significantly lower than those obtained by I-BPI.

As we see from Table 1, I-BPI allows scaling solutions of I-POMDPs up to four levels deep in time duration that is

Problem	Level	Method	Time(s)	Avg. Rwd
Mult. tiger	1	I-BPI	69	11.34
		I-PBVI	2,000	5.34
	2	I-BPI	1,109	12.48
		I-PBVI	696	3.15
	3	I-BPI	3,533	13.00
		I-PBVI	—	—
	4	I-BPI	3,232	13.22
		I-PBVI	—	—
MM	1	I-BPI	15	20.22
		I-PBVI	815	4.86
	2	I-BPI	39	20.55
		I-PBVI	431	3.27
	3	I-BPI	117	21.28
		I-PBVI	—	—
	4	I-BPI	157	21.36
		I-PBVI	—	—
AUAV†	1	I-BPI	7,979	74.08
		I-PBVI	—	—
Money Laun.†	1	I-BPI	1,354	-156.21
		I-PBVI	—	—

Table 1: Average rewards of the controllers at various levels for multiple problem domains. '—' indicates that the corresponding values are not available likely because of scalability issues. †Rewards reported for these larger problem domains are the expected rewards (values) of the corresponding controllers. These results were generated on a RHEL 5 system with Xeon Core2 duo, 2.8GHz each and 4 GB of RAM.

within one hour. It also scales to larger, realistic problem domains. This improvement is primarily due to representing the model space using a finite number of nodes. Comparison with I-PBVI reveals that the quality of the controllers is significantly improved. Furthermore, previous approaches have not scaled solutions beyond two levels.

Next, we demonstrate the performance and scalability of the algorithm on the two large and realistic problems, AUAV reconnaissance and money laundering, in Table 1. Although the latter has been solved previously using the interactive particle filtering [13], the approach assumed an initial belief over the interactive state space and reported run times were more than an order of magnitude greater compared to the time taken by I-BPI. Furthermore, our approach provides a general solution valid over the entire belief space. Table 1 shows the run times for generating converged controllers of good quality for the larger problem domains. The rewards are the expected rewards (values) averaged over 100,000 randomly-generated belief points. While the AUAV problem consumes slightly more than two hours, the money laundering problem takes well within one hour. Converged controllers consisted of 45 nodes for the AUAV problem and 12 nodes for the money laundering problem. Although scaling in the nesting to level two is possible, the corresponding time taken is well beyond our cutoff of two hours. Nevertheless, the reported expected reward is competitive in comparison to those reported by Ng et al. [13] for particular initial beliefs and parameter configurations for money laundering.

An interesting empirical observation in all of these problem domains is that the level l controller, $\pi_{i,l}$, converged – it stops improving and its value vectors obtained by solving the system of linear equations given by Eq. 1 fixate – after the lower-level controller, $\pi_{j,l-1}$, converges.

6. DISCUSSION

As applications emerge for I-POMDPs, approaches that allow its solutions to scale become even more crucial. We introduced a generalized policy iteration algorithm for mutiagent settings in the context of I-POMDPs. This is, to the best of our knowledge, the first policy iteration algorithm proposed for I-POMDPs. We construct a finite state controller for each differing frame of other agents, and models of the other agents get naturally mapped to nodes in the respective controllers. The application of generalized BPI to these controllers ensure that the size of the model space doesn't increase rapidly thereby subduing the effect of the curse of dimensionality, which excessively impacts I-POMDPs.

A limitation of interactive BPI is its convergence to local optima leading to controllers whose quality is unpredictable. While techniques for escaping from local optima may help, this is not guaranteed and the globally optimal value may not be achieved. In particular, the approach of seeking an improved value vector that is uniformly greater than a previous vector leads to multiple local optima; relaxing the constraint of uniform improvement may help.

Another, more practical hurdle was our use of LAPACK++ (`http://math.nist.gov/lapack++`) to solve the system of linear equations in the policy evaluation step. Although LAPACK++ is a popular linear algebra package, it's not optimized for exploiting sparseness in matrices, which we often encountered. We think that further scalability is immediately possible by exploiting the sparsity of the matrices during evaluation. Furthermore, as the number of variables in the linear programs with non-zero values is often low, sparseness may be further exploited in the policy improvement step analogously to the proposal by Hansen [10] in the context of POMDPs. It also seems possible to further improve the performance by accounting for the structure of the controllers. If the controller contains a strongly-connected component, we can evaluate it first thereby focusing on a subset of the nodes, followed by evaluating the rest of the nodes.

Another line of future work is to evaluate the performance of this approach on problem domains having more than two agents. Additionally, the finite state controllers that we use are a type of automata called Moore machines. Recently, Mealy machines were utilized in the context of decentralized POMDPs to good effect [1]. Therefore, another avenue is to investigate the utility of different types of controllers including Mealy machines in our context.

7. ACKNOWLEDGMENTS

This research is supported in part by the NSF CAREER grant, #IIS-0845036, to Prashant Doshi. We thank Kofi Boakye and Brenda Ng for providing the domain files for the money laundering problem.

8. REFERENCES

[1] C. Amato, B. Bonet, and S. Zilberstein. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In *Twenty Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1052–1058, 2010.

[2] 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.

[3] D. S. Berstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1287–1292, 2005.

[4] P. Doshi and P. Gmytrasiewicz. Monte Carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, 34:297–337, 2009.

[5] P. Doshi and D. Perez. Generalized point based value iteration for interactive POMDPs. In *Twenty Third Conference on Artificial Intelligence (AAAI)*, pages 63–68, 2008.

[6] P. Doshi, X. Qu, A. Goodie, and D. Young. Modeling recursive reasoning in humans using empirically informed interactive POMDPs. In *International Autonomous Agents and Multiagent Systems Conference (AAMAS)*, pages 1223–1230, 2010.

[7] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[8] Q. Guo and P. Gmytrasiewicz. Modeling bounded rationality of agents during interactions (extended abstract). In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 1285–1286, 2011.

[9] E. Hansen. Solving POMDPs by searching in policy space. In *Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 1998.

[10] E. Hansen. Sparse stochastic finite-state controllers for POMDPs. In *Uncertainty in Artificial Intelligence (UAI)*, pages 256–263, 2008.

[11] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[12] C. Meissner. A complex game of cat and mouse. *LLNL Science and Technology Review*, pages 18–21, March 2011.

[13] B. Ng, C. Meyers, K. Boakye, and J. Nitao. Towards applying interactive POMDPs to real-world adversary modeling. In *Innovative Applications in Artificial Intelligence (IAAI)*, pages 1814–1820, 2010.

[14] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based value iteration for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[15] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Neural Information Processing Systems*, 2003.

[16] B. Rathnasabapathy, P. Doshi, and P. J. Gmytrasiewicz. Exact solutions to interactive POMDPs using behavioral equivalence. In *Autonomous Agents and Multi-Agent Systems Conference (AAMAS)*, pages 1025–1032, 2006.

[17] R. Seymour and G. Peterson. Responding to sneaky agents in multi-agent domains. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 99–104, 2009.

[18] R. Seymour and G. L. Peterson. A trust-based multiagent system. In *IEEE International Conference on Computational Science and Engineering*, pages 109–116, 2009.

[19] R. Smallwood and E. Sondik. The optimal control of partially observable Markov decision processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[20] E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted cost. *Operations Research*, 26(2):282–304, 1978.

[21] M. Wunder, M. Kaisers, J. Yaros, and M. Littman. Using iterated reasoning to predict opponent strategies. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 593–600, 2011.