Scalable Algorithms for Sequential Decision Making Under Uncertainty in Multiagent Systems

by

Ekhlas S. Sonu

(Under the direction of Prashant Doshi)

Abstract

Decision making or planning is a crucial part of AI research. In recent times, deploying autonomous agents – such as search and rescue robots, autonomous unmanned vehicles, planetary rovers, and many others – has been proposed as a feasible approach in many real-world scenarios where sending humans is deemed either too risky or too expensive. Many times, these agents are required to operate under uncertainty, which may arise either due to the dynamics of the environment or due to the inaccuracies inherent in the sensors and actuators of the autonomous agent. In order to accomplish its goal most efficiently, the autonomous agent must compute an optimal plan for itself.

Decision making in partially observable stochastic settings is formalized by partially observable Markov decision processes (POMDPs) [35]. Interactive partially observable Markov decision processes (I-POMDPs) [20] generalize POMDPs to multiagent settings where the goal is to compute the optimal policy for an individual agent operating in the presence of other agents whose goals and preferences may not align with that of the subject agent. The solution to the problem of multiagent decision making under uncertainty suffers from several sources of intractability. These sources of intractability arise due to the uncertainty inherent

in the environment, such as the stochastic state transitions, noisy or partial observations, and uncertainty about the goals, capabilities, and beliefs of the other interacting agents. While POMDP allows an agent to act rationally in single agent settings and most POMDP approximation algorithms are generalizable to multiagent contexts such as I-POMDPs, additional effort must be made to tackle problems specific to multiagent settings.

In my dissertation, I propose several techniques that target various sources of intractability that plague multiagent decision making problems formalized by I-POMDPs. The first approach is a simple enhancement to existing POMDP algorithms that limits the observation space in certain contexts for quicker solution. The next approach is a generalized policy iteration technique that restricts the exponential growth of the model space, thereby speeding up the computation of a locally optimal policy. Third, I propose an online bimodal approach that in which the agent behaves as a POMDP treating other agents as a static noise under higher uncertainty but once it has received sufficient information regarding the current physical state of the environment, switches to full I-POMDP mode. This asymptotically leads to better runtime. Finally, I target the problem of exponentially growing complexity of I-POMDP solution with the number of interacting agents. This source of intractability is overcome by exploiting commonly found problem structures such as anonymity [58] and context-specific independence [7]. Exploiting these problem structures enables solution of problems involving thousands of interacting agents in under six hours.

INDEX WORDS: Artificial intelligence, rational decision making, interactive decision theory, multiagent planning under uncertainty, interactive partially observable Markov decision process, scalable algorithms

Scalable Algorithms for Sequential Decision Making Under Uncertainty in Multiagent Systems

by

Ekhlas S. Sonu

B. Tech., Visvesvaraya National Institute of Technology, India, 2008

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2015

Scalable Algorithms for Sequential Decision Making Under Uncertainty in Multiagent Systems

by

Ekhlas S. Sonu

Approved:

Major Professor: Prashant Doshi

Committee: Walter D. Potter
Khaled Rasheed
Pascal Poupart

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
August 2015

Dedicated to my mother for her endless and unquestioning love and support.

Scalable Algorithms for Sequential Decision Making Under Uncertainty in Multiagent Systems

Ekhlas S. Sonu

July 27, 2015

Acknowledgements

I would like to acknowledge the contributions of my advisor, Dr. Prashant Doshi for the guidance, support, motivation and countless advice he has given me over the years. Not only did he motivate me in to pursue a career as a researcher in my early graduate school days, he has also been and continues to be a mentor and a source of inspiration to me. Second, I would like to thank my committee members, Dr. Walter D. Potter, Dr. Khaled Rasheed and Dr. Pascal Poupart for taking time out of their busy schedule to read my dissertation and providing me with valuable suggestions for improving it. Third, I thank Dr. Bhandarkar for his timely guidance and inputs. I would like mention the members of the office staff of the CS department at UGA for their diligence that made life much easier for the graduate students.

I would also like to mention the members of Thinc Lab, fellow travelers in the long journey – Uthaya, Xia, Muthu, Ken, Roi, Anousha, Amir and Will – for making the life in the lab fun and enjoyable. Special thanks to Roi and Ken for their inputs on my dissertation. A special note for BJ and Brad for keeping me connected to the world outside the confines of the lab. A special note of appreciation for John Harney for being a great mentor to me in the early years as a graduate student and during my brief stint at ORNL.

Finally, I would like to thank my mother. I am what I am because of her love, support and sacrifices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous decision making under uncertainty has received much interest in recent times. Over the past few years autonomous agents have found applications in the wide variety of fields such as:

1. **Healthcare:** Autonomous agents could be used to guide patients with cognitive disabilities in performing everyday tasks [29] [48]. Also, they may be used to formulate the course of treatment in multi-treatment therapy [26].

2. **Autonomous Navigation:** The advent of self driving cars [71] [72] [41] and the use of autonomous aerial drones for delivery [4] has opened up brand new avenues for efficient autonomous planning in urban domains.

3. **Security and Defense:** The use of robotic agents such as autonomous unmanned aerial vehicles (AUAV) in defense sector for surveillance and espionage has received much attention in recent times. Autonomous agents may also be used in search and rescue missions.

4. **Space and Deep Sea Exploration:** Autonomous robots could be used to explore the regions of space and the depths of the oceans that are inaccessible to humans.

An autonomous agent must be equipped to handle various sources of uncertainty that may be characteristic of its environment. For example, in the case of self driving cars often there is some uncertainty about the traffic conditions. Also, the agent must be mindful of the behavior of other vehicles that may merge or leave dynamically. Similarly, in a search and rescue operation, the complete map of the affected area may not be available. Secondly, disasters such as fire and earthquake could dynamically alter the layout of the operation theatre or interfere with the sensor signals. Finally, the agents themselves could be equipped with faulty sensors and actuators which may affect their performance. Regardless of these shortcomings, the agent must guarantee a certain degree of performance before deployment in the real world. Computing an optimal plan is critical for the autonomous agent if such guarantees must be made.

Decision theory is the field of AI that deals with the problem of optimal plan computation for an agent. Depending on the settings many frameworks have been suggested for planning. Markov decision processes (MDPs) [54] [60] and its derivatives have garnered much popularity in this regard. Primarily, the decision theoretic planning problem consists of two components – the agent(s) and the environment. The environment consists of the physical world in which a given task (such as the search and rescue operation) must be carried out and the agents are the software or hardware devices (or in some cases humans) that carry out the task. An autonomous agent may perceive the environment, reason about its evolution (possibly brought about as a result of its own action and the actions of all other agents), and carry out the task assigned to it. Depending on the characteristics of the environment and the type of interaction between the agents several generalizations of MDPs have been proposed.

1.1 Decision Making in Partially Observable Single Agent Settings

A typical single agent setting may be illustrated as shown in figure 1.1. The possible circumstances of the environment are captured using a set of its states. At each time-step, the agent may perform an action that may stochastically alter the state of the environment. As a result the agent receives an observation about the updated state of the environment and a reward that reflects the degree of its success in achieving its goals.

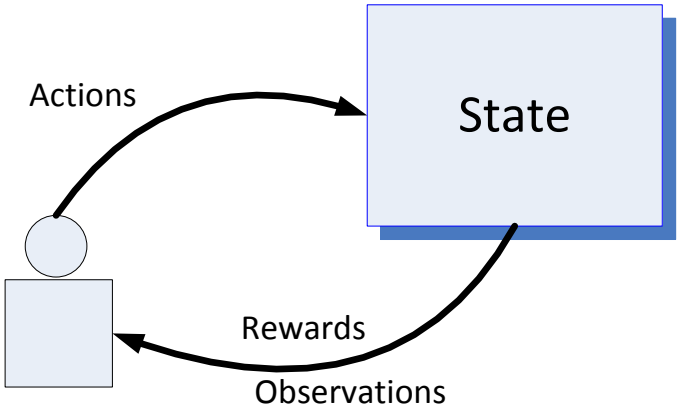


Figure 1.1: Illustration of an agent operating in single agent settings. The agents action affect the state of the environment as a result of which the agent receives its observation and reward.

The goal of a rational agent operating in a sequential setting is to select a sequence of actions that would maximize its long term rewards. In stochastic and partially observable dynamic settings, an agent must predict all possible outcomes of its actions and compute a conditional plan (commonly referred to as "*policy*") that maximizes its expected utility in the long term. Partially observable Markov decision processes (POMDPs) [35] formalize decision making in single agent settings under uncertainty. The problem of single agent planning under uncertainty as formalized by POMDPs has been studied quite intensively in recent times.

Formally a POMDP is defined as the following tuple:

$$\text{POMDP} = \langle S, A, T, \Omega, O, R, OC \rangle$$

where:

- S is the set of *states* of the environment.
- A, is the set of actions that the agent may perform.
- $T : S \times A \times S \to [0, 1]$, is the transition function which gives the distribution over the next physical states given the current state and agent's action.
- Ω is the set of observations that the agent may receive.
- $O : S \times A \times \Omega_i \to [0, 1]$, is the observation function which is the probability with which the agent receives an observation conditioned on a its action and the resulting state.
- $R : S \times A \to \mathbb{R}$, is the reward function which is the reward the agent receives given its action and the start state.
- OC is the optimality criterion which may be the discounted sum of the rewards obtained over a fixed number of steps (called horizon) or the converged sum.

In partially observable domains where the complete information about the current state is not available, the agent bases its plan on the belief it has over the states of the environment. The belief of an agent is a probability distribution that it assigns over the states of the environment, i.e. $b \in \Delta(S)$ (where $\Delta(\cdot)$ is the set of probability distributions over a set). The computation of an optimal plan requires two major operations: belief update and value iteration.

1.1.1 Belief Update

At time-step t, the agent performs an action a^t based on its belief b^t, and receives an observation ω^{t+1} conditioned on the action and the altered state of the environment s^{t+1}. The agent's action at the next step is based on its updated belief b^{t+1} (also represented as $\tau(b^t, a^t, \omega^t)$). The updated belief of the agent is obtained as follows:

$$\begin{aligned} b^{t+1}(s^{t+1}) &= Pr(s^{t+1}|b^t, a^t, \omega^{t+1}) \\ &= \eta O(s^{t+1}, a^t, \omega^{t+1}) \sum_{s^t} b^t(s^t) T(s^t, a^t, s^{t+1}) \end{aligned} \tag{1.1}$$

1.1.2 Value Iteration

As I mentioned earlier, the goal of an agent in sequential settings is to maximize its long-term discounted rewards. POMDP value iteration maximizes the agent's long term rewards for horizon h and an initial belief b^t as follows:

$$\begin{aligned} V^h(b^t) &= \max_{a^t} \left\{ ER(b^t, a^t) + \gamma \sum_{\omega^{t+1}} Pr(\omega^{t+1}|b^t, a^t) V^{h-1}(\tau(b^t, a^t, \omega^{t+1})) \right\} \\ &= \max_{a^t} \left\{ \sum_{s^t} b^t(s^t) R(s^t, a^t) + \gamma \sum_{\omega^{t+1}} Pr(\omega^{t+1}|b^t, a^t) V^{h-1}(\tau(b^t, a^t, \omega^{t+1})) \right\} \end{aligned} \tag{1.2}$$

where $ER(b^t, a^t)$ is the expected immediate reward for performing a^t from belief b^t, γ is the discount factor and the term following γ is the reward that the agent expects to receive in the future. A common term used in POMDP literature is the Q-value which is defined for a belief and an action ($Q(b^t, a^t)$) as the expected reward if the agent were to perform action a^t in the first step and act optimally thereafter. Naturally, the value function could be defined as:

$$V^h(b^t) = \max_{a^t} Q(b^t, a^t) \tag{1.3}$$

Since the space of beliefs is continuous, it is impossible to compute the optimal policy for

every belief. Fortunately, the POMDP value function satisfies the piecewise linear convex (PWLC) property. That is, the complete solution to a POMDP may be represented as a set of vector Γ. For any given belief, the optimal value may be computed by computing a scalar product of the belief with all vectors in Γ and picking the optimal value.

$$V^h(b^t) = \max_{\alpha \in \Gamma^h} b^t \cdot \alpha$$

The sets of vectors are computed iteratively starting from horizon 1 up to horizon h as follows:

Step 1: First we generate the vectors for horizon 1 as follows:

$$\Gamma^1_{a,*} \overset{\cup}{\leftarrow} \alpha^1_{a,*}(s) = R(s,a)$$

Step 2: For horizon h we generate intermediate sets $\Gamma^h_{a,\omega}$

$$\Gamma^h_{a,\omega} \overset{\cup}{\leftarrow} \alpha^h_{a,\omega}(s) = \gamma \sum_{s'} T(s,a,s') O(s',a,\omega) \alpha^{h-1}(s'); \quad \forall \alpha^{h-1} \in \Gamma^{h-1}$$

Step 3: Next we compute the vectors for policies with a as the first action by computing the following cross-sum:

$$\Gamma^h_a = \Gamma^1_{a,*} \oplus \Gamma^h_{a,\omega_1} \oplus \Gamma^h_{a,\omega_2} \oplus \ldots \oplus \Gamma^h_{a,\omega_{|\Omega|}}$$

where $\Gamma_1 \oplus \Gamma_2 = \{\alpha_1 + \alpha_2 | \alpha_1 \in \Gamma_1, \alpha_2 \in \Gamma_2\}$.

Step 4: Finally we compute the set of optimal vectors at horizon h as:

$$\Gamma^h = Prune(\bigcup_a \Gamma^h_a)$$

where the operator *Prune* removes all vectors that are not optimal at any belief in the belief space because they will never represent an optimal value.

As a result of the PWLC property, the Q-value for an action and by extension the value function may be written as follows:

$$Q^h(b^t, a^t) = \sum_{s^t} b^t(s^t) R(s^t, a^t) + \gamma \sum_{\omega^{t+1}} \max_{\alpha \in \Gamma^{h-1}} \left\{ \sum_{s^t} b^t(s^t) \sum_{s^{t+1}} O(s^{t+1}, a^t, \omega^{t+1}) \right. \\ \left. T(s^t, a^t, s^{t+1}) \alpha(s^{t+1}) \right\} \tag{1.4}$$

1.1.3 Bounds on POMDP Value Function

POMDP complexity of POMDP value iteration is exponential in the number of observations (refer to cross-sum step). However, we may quickly compute upper and lower bounds on the POMDP value function which may then be used to guide quicker solutions [65] [63] [40] [53]. Two of the well known bounds on POMDP value function are *fast informed bound* which serves as an upper bound and *blind policy* which serves as a lower bound [26] [27].

Upper Bound: Fast Informed Bound

The fast informed bound on the horizon h Q-value function is computed as follows:

$$\begin{aligned} \overline{Q}^h(b^t, a^t) &= \sum_{s^t} b^t(s^t) R(s^t, a^t) + \gamma \sum_{\omega^{t+1}} \sum_{s^t} b^t(s^t) \max_{\alpha \in \overline{\Gamma}^{h-1}} \left\{ \sum_{s^{t+1}} O(s^{t+1}, a^t, \omega^{t+1}) \right. \\ &\qquad \left. T(s^t, a^t, s^{t+1}) \alpha(s^{t+1}) \right\} \\ &= \sum_{s^t} b^t(s^t) \left[R(s^t, a^t) + \gamma \sum_{\omega^{t+1}} \max_{\alpha \in \overline{\Gamma}^{h-1}} \left\{ \sum_{s^{t+1}} O(s^{t+1}, a^t, \omega^{t+1}) T(s^t, a^t, s^{t+1}) \alpha(s^{t+1}) \right\} \right] \end{aligned} \tag{1.5}$$

where $\overline{\Gamma}^{h-1}$ is the set containing the FIB upper bound vectors for horizon $h-1$.

Notice that the max term over α has moved inside the summation over the initial states. Hence the upper bound property. An interesting property of FIB is that irrespective of the horizon, we always get one vector per action [26]. The upper-bound on POMDP value function is obtained as $\overline{V}(b^t) = \max_{a^t} \overline{Q}^h(b^t, a^t)$.

Lower Bound: Blind Policy

A blind policy is obtained when the agent chooses the same future policy irrespective of the observation it receives. This loss of information resulting from ignoring the observation results in a sub-optimal value. The value of the blind policy is obtained as follows:

$$\begin{aligned}
\underline{Q}^h(b^t, a^t) &= \sum_{s^t} b^t(s^t) R(s^t, a^t) + \gamma \max_{\alpha \in \underline{\Gamma}^{h-1}} \Bigg\{ \sum_{\omega^{t+1}} \sum_{s^t} b^t(s^t) \sum_{s^{t+1}} O(s^{t+1}, a^t, \omega^{t+1}) \\
&\qquad T(s^t, a^t, s^{t+1}) \alpha(s^{t+1}) \Bigg\} \\
&= \sum_{s^t} b^t(s^t) R(s^t, a^t) + \gamma \max_{\alpha \in \underline{\Gamma}^{h-1}} \left\{ \sum_{s^t} b^t(s^t) \sum_{s^{t+1}} T(s^t, a^t, s^{t+1}) \alpha(s^{t+1}) \sum_{\omega^{t+1}} O(s^{t+1}, a^t, \omega^{t+1}) \right\} \\
&= \sum_{s^t} b^t(s^t) R(s^t, a^t) + \gamma \max_{\alpha \in \underline{\Gamma}^{h-1}} \left\{ \sum_{s^t} b^t(s^t) \sum_{s^{t+1}} T(s^t, a^t, s^{t+1}) \alpha(s^{t+1}) \right\}
\end{aligned} \tag{1.6}$$

where $\underline{\Gamma}^{h-1}$ is the set containing the blind policy lower bound vectors for horizon $h-1$.

Notice that in the first step, the max term over α has moved outside the sum over observations. Hence the lower bound property. The lower-bound on POMDP value function is obtained as $\underline{V}(b^t) = \max_{a^t} \underline{Q}^h(b^t, a^t)$.

1.2 Decision Making in Partially Observable Multiagent Settings

In multiagent settings, the environment is inhabited by multiple agents. An agent's rewards, its observations, and the state of the environment may also be affected by the actions of

other agents inhabiting its environment. For rational decision making under uncertainty in multiagent settings, POMDPs have been generalized along two lines. While decentralized POMDPs (Dec-POMDP) [6] generalize POMDPs to settings involving teams of agents operating together towards achieving a common goal, interactive POMDPs (I-POMDP) [20] formulate the multiagent decision making problem from the perspective of a self-interested agent that shares its environment with other self-interested agents with common or contradicting goals. My dissertation focuses on the later framework.

Naturally, the decision making process of the agent in multiagent settings is much more complicated compared to decision making in single agent settings because in addition to accounting for the uncertainty in the environment and the uncertainty arising from its faulty actuators and sensors, the subject agent must also predict how the other agents behave over a period of time and how their actions affect its own goals. To do so, in addition to maintaining a probability distribution over the physical states of the environment, the subject agent must also maintain a distribution over the models of other agents. The models of the other agents reflect how the subject agent views the goals, capabilities, and beliefs of the other agents and how they reason about the environment and all other agents (including the subject agent itself) present in the environment. At each step, the agent must update its belief over both the physical states and the models of other agents. This form of decision making is formalized by the interactive POMDP (I-POMDP) framework [20]. Decision making from the perspective of a single agent operating in a multiagent setting is illustrated in figure 1.2.

In recent times I-POMDP has been proposed as a viable framework for decision making in a myriad of applications across several disciplines which testifies to its growing appeal.

1. **Law Enforcement:** I-POMDPs have been proposed to explore strategies for countering money laundering [42, 46].

2. **Defense:** I-POMDPs have been enhanced to include trust levels for facilitating defense

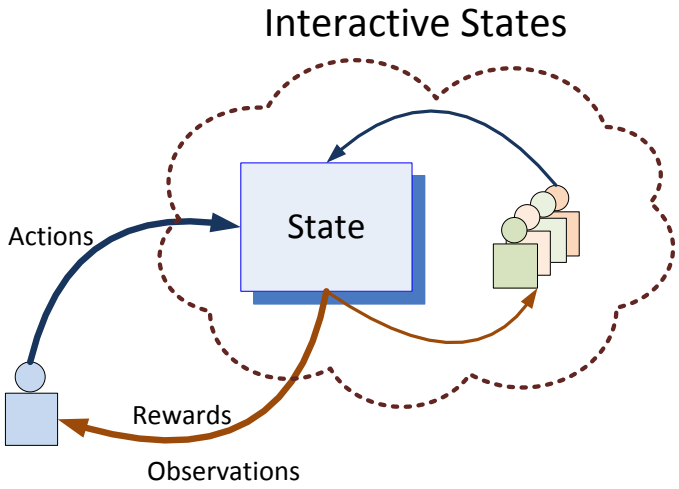


Figure 1.2: Illustration of an agent operating in multiagent settings. The state, observation, and reward are affected by the actions of all agents. Therefore, the subject agent must model the other agents to predict their behavior. These models are included in the interactive state space.

simulations [62, 61].

3. **Game Playing:** I-POMDPs have been used to produce winning strategies for playing the lemonade stand game [78], explored for use in playing Kriegspiel [11].

4. **Robot Learning:** I-POMDPs were discussed as a suitable framework for a robot learning tasks interactively from a human teacher [76, 75].

5. **Psychology:** I-POMDPs have been modified to include empirical models for simulating human behavioral data pertaining to strategic thought and action [16].

The growing appeal of I-POMDPs have necessitated research for exploring scalable solution algorithms for the framework. I illustrate the formalism for I-POMDPs in the context of the multiagent tiger problem [20] – a two-agent generalization of the well-known single agent tiger problem [35], often used for illustrating and evaluating POMDPs.

Example 1 (Multiagent tiger problem)**.** *In this problem, two agents, i and j, face two closed doors one of which hides a tiger while the other hides a pot of gold. An agent gets rewarded for opening the door that hides the gold but gets penalized for opening the door leading to the tiger. Each agent may open the left door (action denoted by OL), open the right door (OR), or listen (L). On listening, an agent may hear the tiger growling either from the left (observation denoted by GL) or from the right (GR). Additionally, the agent hears creaks emanating from the direction of the door that was possibly opened by the other agent – creak from the left (CL) or creak from right (CR) – or silence (S) if no door was opened. All observations are assumed to be noisy. If any door is opened by an agent, the tiger appears behind any of the two doors randomly in the next time step.*

While the actions of the other agent do not directly affect the reward for an agent, they may potentially change the location of the tiger thereby impacting the utility of the agent's previous observations. This motivates modeling the other agent. While other multiagent formulations of the tiger problem also exist such as the team setting of Nair et al. [44], this formulation of the problem differs from others in the presence of door creaks and that it is not cooperative.

1.2.1 Finitely Nested Interactive POMDPs

As I mentioned previously, I-POMDPs generalize POMDPs to multiagent settings by considering dynamic behavioral models of other agents as part of the state space. These models may themselves be I-POMDPs which may also be modelling the subject agent similarly (i.e. as an I-POMDP). This leads to agents recursively modeling each other infinitely through an infinitely-nested modeling space. In order to make the framework computable, the nesting is limited to a strategy level, l, thereby leading to finitely-nested I-POMDPs, which makes the framework operational. Formally, a level l I-POMDP for agent i interacting with one other

agent j is defined as the following 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* at strategy level, l, defined as, $IS_{i,l} = S \times M_{j,l-1}$, where S is the set of physical states, and $M_{j,l-1}$ is the set of models ascribed to the other agent. We describe the model space after this definition in this subsection.
- $A = A_i \times A_j$, is the set of joint actions of both agents.
- $T_i : S \times A \times S \rightarrow [0, 1]$, is the transition function which gives the distribution over the next physical states given the current state and a joint action.
- Ω_i is the set of observations agent i may receive.
- $O_i : A \times S \times \Omega_i \rightarrow [0, 1]$, is the observation function which is the probability with which agent i receives an observation conditioned on a joint action and the resulting state.
- $R_i : S \times A \rightarrow \mathbb{R}$, is the reward function which is the reward agent i receives given a joint action performed by both agents from a state.
- OC_i is the optimality criterion which may be a the discounted sum of the rewards obtained over a fixed number of steps (called horizon) or till convergence.

Dennett's intentional stance [12] offers a way to organize the space of mental models into those that are *intentional* and denoted by Θ_j, and others that are subintentional, denoted by SM_j. Intentional models ascribe beliefs, capabilities, preferences and rationality in action selection to the other agent. Examples of intentional models include the decision-theoretic formalism of POMDPs. Subintentional models include a distribution over actions,

$\Delta(A_j)$, which may be history dependent as in fictitious play [19]. Here, $\Delta(\cdot)$ denotes the set of all probability distributions over the argument random variable. A special example is the no-information model often represented by a uniform distribution. A more powerful subintentional model is the finite state automaton. We may follow a recursive bottom-up construction of the interactive state space.

$$\begin{array}{lll} IS_{i,0} = & S, & \Theta_{j,0} = \{\langle b_{j,0}, \hat{\theta}_{j,0}\rangle | b_{j,0} \in \Delta(IS_{j,0})\}, \quad M_{j,0} = \Theta_{j,0} \cup SM_j \\ IS_{i,1} = & S \times M_{j,0}, & \Theta_{j,1} = \{\langle b_{j,1}, \hat{\theta}_{j,1}\rangle | b_{j,1} \in \Delta(IS_{j,1})\}, \quad M_{j,1} = \Theta_{j,1} \cup M_{j,0} \\ \vdots & \vdots & \vdots \qquad\qquad\qquad\qquad\qquad\qquad\quad \vdots \\ IS_{i,l} = & S \times M_{j,l-1}, & \Theta_{j,l} = \{\langle b_{j,l}, \hat{\theta}_{j,l}\rangle | b_{j,l} \in \Delta(IS_{j,l})\}, \quad M_{j,l} = \Theta_{j,l} \cup M_{j,l-1} \end{array} \tag{1.7}$$

The 0-th level belief is a probability distribution over the physical states only, and the 0-th level models, $M_{j,0}$, are generally limited to be computable and consist of the set of computable intentional models of level 0, $\Theta_{j,0}$, and the subintentional models, SM_j. An intentional model, $\theta_{j,0} = \langle b_{j,0}, \hat{\theta}_{j,0}\rangle$, where $b_{j,0}$ is agent j's level 0 belief, $b_{j,0} \in \Delta(IS_{j,0})$, and $\hat{\theta}_{j,0} = \langle A_j, T_j, \Omega_j, O_j, R_j, OC_j\rangle$, is collectively labeled as j's level 0 *frame*. Here, j is assumed to be Bayesian and rational. 0-th level intentional models are the traditional POMDPs whose parameters, T_j, O_j, and R_j are specified over j's individual actions, A_j. [1] [2] An agent's first-level beliefs are joint probability distributions over the physical states and level 0 models of the other agent. First-level models are computable and include computable level 1 intentional models and level 0 models of the agent. A level 1 intentional model, $\theta_{j,1} = \langle b_{j,1}, \hat{\theta}_{j,1}\rangle$, consists of the agent's first-level belief, $b_{j,1}$, and its frame, $\hat{\theta}_{j,1} = \langle A, T_j, \Omega_j, O_j, R_j, OC_j\rangle$. Note that parameters in the level 1 frame, T_j, O_j and R_j are specified over the joint actions, A. An

[1] Note that the definition of a belief rests on first defining the underlying state space. The state space is not explicitly stated in the intentional model for brevity.

[2] One way of obtaining a POMDP at level 0 is to use a fixed distribution over the other agent's actions and fold it into T_j, O_j, and R_j as noise.

agent's second-level beliefs are distributions over the physical states and level 1 models of the other agent, and so on up to the level l.

This recursive construction of the interactive state space shares similarities with the hierarchical belief spaces as defined in game theory [3, 9, 43]. For example, with the aim of mathematically formalizing Harsanyi's abstract definition of a type as a vector containing private information [25], Brandenburger and Dekel [9] define a first-order belief as a distribution over the underlying state space. A second-order belief is a joint distribution over the state space and first-order beliefs, and so on. A type is then a hierarchy of first-order belief, second-order belief, and so on. A distinction from the construction in (1.7) is that the game-theoretic formalization assumes the other agent to jointly possess beliefs of each lower order. Thus, an agent is assumed to simultaneously exhibit a first-order belief, a second-order belief, and so on. While this construction facilitates its intended use of mathematically formalizing the type, it complicates its operational use in predicting a behavior for the other agent.

Example 2 (Beliefs)**.** *In the context of the multiagent tiger problem, the physical state space consists of whether the tiger is behind the left door (TL) or the right door (TR). Therefore, a 0-th level belief of agent j, $b_{j,0} \in \Delta(S)$, is a probability distribution over the two states. For example, $b_{i,0} = \langle 0.85, 015 \rangle$, indicates that the agent believes that the tiger is behind the left door with a probability of 0.85, and behind the other door with the remaining probability of 0.15. An example subintentional model in SM_j is a probability distribution, $\langle 0.1, 0.8, 0.1 \rangle$, over the agent's actions, L, OL and OR. 0-th level models, $M_{j,0}$, include subintentional models, which are probability distributions in this example, and POMDP formalizations of the single-agent tiger problem that constitute the frames, $\hat{\theta}_{j,0}$, with initial beliefs, $b_{j,0}$. These intentional models are members of $\Theta_{j,0}$. A level 1 belief, $b_{j,1} \in \Delta(IS_{j,1})$, is a joint distribution over the physical states and level 0 models of the other agent. It is represented as two probability distributions, each of which is over the countable 0-th level models of the other*

assumed to have identical frames, one for TL and another for TR. The marginal of these distributions gives the probability over TL and TR, respectively.

Belief Update

An agent's belief over its interactive states is a sufficient statistic, fully summarizing the agent's observation history [20]. Beliefs are updated after the agent's action and observation using Bayes rule. Compared to single agent settings, the belief update in multiagent settings are further complicated due to two factors. First, since the state of the physical environment depends on the actions performed by all agents the prediction of how it changes has to be made based on the probabilities of various actions of the other agents. Probabilities of others' actions are obtained by solving their models. Second, changes in the models of the other agents have to be included in the update. The changes reflect the others' observations and, if it is modeled intentionally, the update of other agents' beliefs. In this case, the agent has to update its beliefs about the other agents based on what it anticipates the other agents observe and how they update their model. For our two agent I-POMDP, agent i's updated belief over an interactive state, $is'_{i,l} = \langle s', \langle b'_{j,l-1}, \hat{\theta}'_j \rangle \rangle$, may be formalized using the following state estimation function:

$$\begin{aligned} b'_{i,l}(is'_{i,l}) = \; & Pr(is'_{i,l}|a_i, \omega_i, b_{i,l}) \\ & = \eta \sum_{is_{i,l}|\hat{\theta}_j = \hat{\theta}'_j} b_{i,l}(is_{i,l}) \sum_{a_j \in A_j} Pr(a_j|\theta_{j,l-1}) T_i(s, a_i, a_j, s') \times \\ & O_i(s', a_i, a_j, \omega_i) \sum_{\omega_j \in \Omega_j} O_j(s', a_j, a_i, \omega_j) Pr(b'_{j,l-1}|b_{j,l-1}, a_j, \omega_j) \end{aligned} \tag{1.8}$$

where η is the normalization constant, $b_{i,l}$ and $b'_{i,l}$ are the initial and updated level l beliefs of agent i, respectively, T_i is its transition function, O_i its observation function, and O_j is the observation function of agent j, $Pr(a_j|\theta_{j,l-1})$ is the probability that a_j is rational for a

Bayesian agent j modeled using $\theta_{j,l-1}$, and $Pr(b'_{j,l-1}|b_{j,l-1}, a_j, \omega_j)$ is 1 if $b_{j,l-1}$ updated using action, a_j, and observation, ω_j, equals $b'_{j,l-1}$, and 0 otherwise. [3]

When there are more than two agents in the environment, the models of other agent m_j and its action a_j are replaced by the joint models and the joint actions of all other agents respectively.

Value Iteration

A solution to an I-POMDP$_{i,l}$ is a *policy*, $\pi_i : \Delta(IS_{i,l}) \rightarrow \Delta(A_i)$, which maps agent i's belief to a distribution over its actions, analogous to that of a POMDP. Using the Bellman equation, each belief state in an I-POMDP has a value which is the maximum sum of future discounted rewards the agent can expect starting from that belief state. Previous approaches for solving I-POMDPs utilize value iteration to compute the value for a belief, which is represented using the following equation:

$$V_i(\langle b_{i,l}, \hat{\theta}_i \rangle) = \max_{a_i \in A_i} \left\{ \sum_{is_{i,l}} b_{i,l}(is_{i,l}) ER_i(is_{i,l}, a_i) + \gamma \sum_{\omega_i \in \Omega_i} Pr(\omega_i|a_i, b_{i,l}) \times V_i(SE_{\theta_{i,l}}(b_{i,l}, a_i, \omega_i)) \right\} \tag{1.9}$$

where, $ER_i(is_{i,l}, a_i) = \sum_{a_j} R(s, a_i, a_j) Pr(a_j|\theta_{j,l-1})$, and $SE(.)$ is the state estimator function which denotes the belief update of agent i given initial belief $b_{i,l}$, action, a_i, and its observation, ω_i, as shown in Eq. 1.8.

[3] Precluding considerations of computability, if the prior belief over $IS_{i,l}$ is a probability density function, then $\sum_{is_{i,l}|\hat{\theta}_j = \hat{\theta}'_j}$ is replaced by an integral over the continuous space. In this case, $Pr(b'_{j,l-1}|b_{j,l-1}, a_j, \omega_j)$ is replaced with a Dirac-delta function, $\delta_D(SE_{\theta_{j,l-1}(b_{j,l-1}, a_j, \omega_j)} - b'_{j,l-1})$, where $SE_{\theta_{j,l-1}}(\cdot)$ denotes state estimation involving the belief update of agent j. These substitutions also apply elsewhere as appropriate.

Analogous to POMDPs, the I-POMDP value function in Eq. 1.9 is composed of a set of linear value vectors [20]. A vector assigns an expected value to every interactive state:

$$\begin{aligned}\alpha_i(\langle s, \theta_{j,l-1}\rangle) \;\; &= \max_{a_i \in A_i} \sum_{a_j \in A_j} Pr(a_j|\theta_{j,l-1}) \Bigg\{ R_i(s,a_i,a_j) + \gamma \sum_{\omega_i \in \Omega_i} \max_{\alpha_i^{\omega_i}} \sum_{s'} \sum_{\theta'_{j,l-1}} \\ &T_i(s,a_i,a_j,s') O_i(s',a_i,a_j,\omega_i) \sum_{\omega_j} O_j(s',a_i,a_j,\omega_j) \\ &\times Pr(b'_{j,l-1}|b_{j,l-1},a_j,\omega_j)\; \alpha_i^{\omega_i}(\langle s', \theta'_{j,l-1}\rangle) \Bigg\}\end{aligned} \tag{1.10}$$

where $\alpha_i^{\omega_i}$ is a value vector in the set of value vectors for the next time step.

Approaches for solving I-POMDP$_{i,l}$ are founded on iteratively improving the value vectors until they converge pointwise for every interactive state. The improvement commonly involves a *backup operation* on the set of value vectors, Γ_i^h, of the previous iteration, which produces the set of optimal value vectors for the current iteration. This operation is called a backup because it takes as input the value vectors for the next time step and utilizes them in generating the value vectors for the current time step. It may be performed as follows, which decomposes Eq. 1.10:

$$\begin{aligned}\Gamma^h_{a_i,*} \overset{\cup}{\leftarrow} \alpha_{a_i,*}(\langle s, \theta_{j,l-1}\rangle) = \;\; &\sum_{a_j \in A_j} R_i(s,a_i,a_j) Pr(a_j|\theta_{j,l-1}) \\ \Gamma^h_{a_i,\omega_i} \overset{\cup}{\leftarrow} \alpha^{a_i,\omega_i}(\langle s, \theta_{j,l-1}\rangle) = \;\; &\gamma \sum_{s'} \sum_{\theta'_{j,l-1}} \sum_{a_j \in A_j} Pr(a_j|\theta_{j,l-1})\; T_i(s,a_i,a_j,s') \\ &\times O_i(s',a_i,a_j,\omega_i) \sum_{\omega_j} O_j(s',a_i,a_j,\omega_j) \\ &Pr(b'_{j,l-1}|b_{j,l-1},a_j,\omega_j)\; \alpha'_i(s',\theta'_{j,l-1}), \\ &\forall\; \alpha'_i \in \Gamma_i^{h-1}\end{aligned}$$

Next, we cross sum the vectors, denoted by $\oplus$, across Γ^{a_i,ω_i} for different ω_i, which involves picking vectors from the different sets and summing them. This sums out the observations, and the set of vectors for differing actions is then pruned to remove the vectors that are very

weakly dominated. A pruned vector is one that did not provide the largest value at any belief, or whose value was identical to a previously retained vector at all beliefs (obtained by performing an inner product between the vector and belief) because the two vectors overlap.

$$\begin{aligned} \Gamma^h_{a_i} &\leftarrow \Gamma^h_{a_i,*} \oplus \Gamma^h_{a_i,\omega_i^1} \oplus \Gamma^h_{a_i,\omega_i^2} \oplus ... \oplus \Gamma^h_{a_i,\omega_i^{|\Omega_i|}} \\ \Gamma^h_i &= Prune(\bigcup_{a_i} \Gamma^h_{a_i}) \end{aligned} \tag{1.11}$$

Notice that each $\Gamma^h_{a_i}$ contains an exponential number of vectors although the number reduces after pruning. Furthermore, multiple approaches exist (for example, see incremental pruning [10]) that reduce the number of vectors resulting from the cross sums.

Analogous to belief update, when there are more than one other agents in the environment, m_j and a_j are replaced by joint models and joint actions of other agents respectively.

1.3 Challenges and Contributions

Decision making in multiagent settings as formalized by I-POMDPs is more complicated than single agent decision making. Next I outline the challenges involved in solving I-POMDPs and the contributions made in this dissertation.

1.3.1 Challenges involved in solving I-POMDP

The challenges of intractibility are captured precisely in the terms of four curses:

- **Curse of dimensionality** results from the size of interactive state space. The space of beliefs grows exponentially with the number of states thereby making the solutions increasingly intractable.

- **Curse of history** is a result of partial observability and manifests as exponentially growing solution space with each application of value iteration.

- **Curse of agent modeling** is an effect of curse of history on the computable model space of the other agents and indirectly the dimensionality of the subject agents interactive state space. The size of model space grows exponentially at each time step thereby further aggravating the curse of dimensionality.

- **Curse of many agents** manifests at each time step as the size of joint model space and joint action space which grows exponentially with the number of agents in the environment.

In this dissertation, I propose approaches to mitigate the effect of each of these curses in order to solve I-POMDPs efficiently.

1.3.2 Contributions

In chapter 2, I present a simple enhancement for partially observable Markov decision processes (POMDP) backup operator that identifies actions that lead to observations which are only weakly informative. We call such actions as weak- (inclusive of zero-) information inducing. Policy subtrees rooted at these actions may be computed more efficiently. While zero-information inducing actions may be exploited without error, the quicker backup for weak but non-zero information inducing actions may introduce error. I empirically demonstrate the substantial computational savings that exploiting such actions may bring to exact and approximate solutions of POMDPs while maintaining the solution quality.

In chapter 3, I present a generalized version of a well known policy iteration technique for POMDPs, the bounded policy iteration (BPI) [51], to problems involving multiple agents. BPI is known to achieve locally optimal solutions for POMDPs while avoiding the exponential growth in the size of controllers. I propose an anytime technique that interleaves bounded policy iteration steps in a bottom up fashion starting at level 0. By doing this, we are able to check the exponential blow-up of the model space thereby alleviating the curse of

nested reasoning. Using this approach, I am able to solve much larger problems (in terms of number of states, actions, observations, and level of reasoning) than those solved by the previous algorithms. I call this approach interactive bounded policy iteration (I-BPI). I also adapt a method for biasing the policy improvement in regions of beliefs that are more likely to be reached given an initial belief. This bias comes at an increased cost of computation which I mitigate using a novel method to eliminate models of the other agents that may never be reached. Further, I utilize the algorithm to solve, for the first time, a multiagent version of the well known *tiger problem* involving up to 5 agents.

In chapter 4, I introduce a novel bimodal approach for online planning. We observe that in settings where the actions of other agents are not observable directly by the subject agent but can only be inferred indirectly given their effect on the physical states, observations are more informative of the other agents' model when the subject agent is more certain about the correct physical states (i.e. the entropy of its belief over physical states is low). Hence under high entropy of belief over the physical states, an agent may act as a POMDP modeling the other agents as noise. In this phase, the agent may exploit faster POMDP solution techniques. Once the subject agent has received enough information about its physical states, it may switch to I-POMDP mode. We determine the point of switching using the bounds on the value of I-POMDP and an input parameter.

In chapter 5 we address the exponential growth in the size of joint model space and the joint action space by exploiting commonly found problem structures such as action anonymity [58] and a form of context specific independence [7] that we call frame-action independence. While traditional I-POMDP approches are able to scale only up to a few other agents (5 other agents), by exploiting these commonly found structures, we are able to solve problems involving thousands of other agents in a resonable amount of time.

I conclude with a discussion on some venues to explore in the future for further improving my work and for better scalability of I-POMDP in general in chapter 6.

Chapter 2

Identifying and Exploiting Weak Information Inducing Actions in Solving POMDPs

In POMDP literature, a large portion of research deals with deals with exploiting problem structure to approximate the solution [8, 37, 59] thereby leading to significant performance gains in the particular problem domains exhibiting the relevant structure. Consistent with this promising line of investigation, we exploit yet another problem structure by identifying a type of action often found in problem domains that lead to observations that tend to be only weakly informative of the state of the environment. Such actions could be exploited so that the related computations may be performed more efficiently. For example, observations made during movement by a robotic vehicle (typically modeled sequentially post action in a POMDP) tend to be far less informative than those resulting from an action dedicated to observing the physical states. We call such actions as weak-information inducing actions. A small subset of such actions are those actions that induce no information about the environment on observations. We provide a simple and novel definition for

weak information-inducing actions by utilizing a parameter that characterizes the weakness of the following observations. We observe that the observations following zero-information inducing actions could be ignored for the belief update leading to compressed policy tree rooted at such actions and subsequently computational savings without any error. Hence, we utilize a simplified backup operator that excludes considering observations following the zero-information inducing actions without introducing any error in the value function. We extend the operator to weak-information inducing actions. As a result, we observe significant computational savings, albeit at some performance loss in terms of optimality that we are currently unable to upper bound. We demonstrate the significant computational savings by exploiting such actions in the context of an exact solution technique – incremental pruning (IP) [10] – and an aproximate solution technique – the well-known point-based value iteration (PBVI) [49] – and empirically show that the solutions are of comparable quality.

2.1 λ-Information Inducing Actions

First we begin by formalizing a definition of such actions and motivation for distinguishing them in context of the classical tiger problem [35]. We then show how we may exploit such actions thereby reducing the complexity of the backup operation.

Example 3 (Single-agent Tiger Problem)**.** *An agent is faced with the choice of two doors. Behind one there is a pot of gold that the agent desires and behind the other is a tiger that the agent must avoid. The state of the environment at any given time could be* tiger-left *(TL) or* tiger-right *(TR). The agent may choose to open either of the doors by performing actions* open-left *(OL) and* open-right *(OR) or it may choose to* listen *(L) to gain more observation about the position of the tiger through some noisy observations. On listening, the agent may hear a growl coming from behind the door containing the tiger with a probability 0.85 and a growl from behind the other door with a probability 0.15. The observations received could*

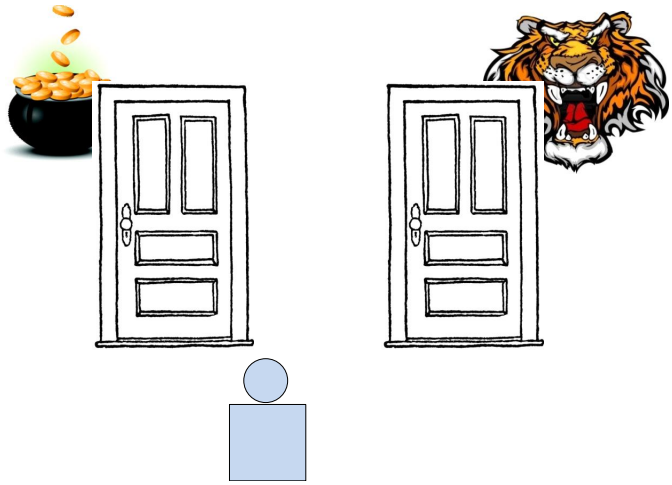


Figure 2.1: An illustration of the single agent tiger problem described in example 3.

be growl-left *(GL) or* growl-right *(GR). By opening a door, the agent resets the state of the environment randomly with a uniform probability and the game continues. The observations received on opening a door are also uniformly random.*

For the aforementioned problem, the observations probabilities on each action of the agent could be written as the following tables :

L	$Pr(GL\|s, L)$	$Pr(GR\|s, L)$
TL	0.85	0.15
TR	0.15	0.85

OL/OR	$Pr(GL\|s, OL/OR)$	$Pr(GR\|s, OL/OR)$
TL	0.5	0.5
TR	0.5	0.5

Table 2.1: Observation function for the single agent tiger problem introduced in example 3. The table on the left shows the observation probability for each end state on listening and the one on the right shows the observation probabilities on performing an action to open either door.

2.1.1 Definition

In the classical tiger problem (example 3), observations subsequent to opening a door (OL/OR) do not provide any information about the door containing the tiger. We gen-

eralize this concept to actions leading to weakly informative observations. We label such actions as λ-*information inducing*, and define them as:

Definition 1 (λ-Information Inducing Actions)**.** *An action, $a \in A$, is λ - information inducing if for all observations:*

$$1 \leq \frac{max_{s' \in S} O(s', a, \omega)}{min_{s'' \in S} O(s'', a, \omega)} \leq \lambda \quad \forall \omega \in \Omega$$

where $\lambda \in \mathbb{R}$. We denote the action using a_λ and the set of all such actions using A_λ . Let $\bar{A_\lambda} = A \backslash A_\lambda$.

In general, low values of λ are representative of actions that generate weak observations while high λ signals rich observation(s), although the actual values are subjective to the problem domain. In the tiger problem, setting λ to 1 would identify actions OL and OR as weak-information inducing actions (indeed they are zero-information inducing actions). On the other hand setting *lambda* to a value ≥ 5.67 would identify all actions as weak-information inducing. Naturally, the solution quality will be affected in the later case.

2.1.2 Approximate Solution

The POMDP belief update may be decomposed into the prediction step where the agent updates its belief based solely on the previous belief and action performed and the correction step where the agent incorporates the observation received to correct its predicted belief. We observe that for zero-information inducing actions ($\lambda = 1$ in Def. 1) incorporating the observation to update its predicted belief doesn't change its value. Hence, correction step may be considered unnecessary for such actions. We extend this approach of ignoring the observations to λ-information inducing actions in general. We hypothesize that for weak-information inducing actions, the corrected beliefs for on receiving any observation would be

close to the predicted belief and to each other. Hence, the same policy may be optimal (or near optimal) for all reachable beliefs.

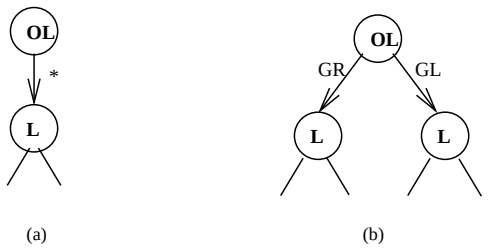


Figure 2.2: (a) An example policy tree (alpha vector) generated by our approximate backup for λ-information inducing action (OL). Here, '*' denotes any observation.(b) Analogous policy tree generated by the traditional backup; both policy subtrees rooted at L are identical and therefore storing one of them is redundant.

Our approach is to shorten the belief update process for λ- information inducing actions by ignoring observations. The abbreviated update leads to a different and quicker backup. The approximate backup process is equivalent to generating policy trees analogous to the one in Fig. 2.2(a) for actions in A_λ , in comparison to the traditional policy trees (Fig. 2.2(b))

Substituting just the prediction step within the Bellman equation leads to the following backup for all actions, $a_\lambda \in A_\lambda$. Let Γ^{h1} be the set of horizon $h1$ alpha vectors. Then,

$$\Gamma^{a_\lambda,*} \overset{\cup}{\leftarrow} \alpha^{a_\lambda,*} = R(s,a_\lambda) + \gamma \sum_{s' \in S} T(s,a_\lambda,s')\alpha(s') \quad \forall \alpha \in \Gamma^{h-1}$$

$$\Gamma_\lambda = \bigcup_{a_\lambda \in A_\lambda} \Gamma^{a_\lambda,*}$$

The backup process proceeds as in the original procedure described in section 1.1 for all other actions in $\bar{A_\lambda}$ resulting in the set Γ' . We obtain the final set of vectors for horizon h as:

$$\Gamma_\lambda^h = prune\ (\Gamma_\lambda \bigcup \Gamma')$$

Notice the absence of cross-sum operations for actions in A_λ. Consequently, we generate

$|\bar{A}_\lambda||\Gamma^{h-1}|^{|\Omega|} + |A_\lambda||\Gamma^{h-1}|$ intermediate vectors in the worst case, which could be far less than $|A||\Gamma^{h-1}|^{|\Omega|}$ vectors generated in the exact approach, if the set A_λ is not empty. The horizon h value function is then obtained as: $V_\lambda^h(b) = \underset{\alpha \in \Gamma_\lambda^h}{max}\ \alpha \cdot b$.

2.2 Experiments

We implemented the approximate solution described in Section 2.1.2 in the context of both IP and PBVI. We selected well-known benchmark problem domains often used to evaluate POMDP solution techniques. In Table 2.2, we show results for a variety of problem domains. Our methodology was to solve each problem exactly using IP and approximately using PBVI – often for longer time horizon in the latter case. We noted the maximum expected reward obtained by averaging over 1,000 or more random belief points (shown in column R). We then measured the time taken by the approaches modified to exploit λ-information inducing actions to reach the expected rewards obtained previously (including time taken to identify such actions).

2.3 Discussion

While zero-information inducing actions ($\lambda = 1$ in Def. 1) could be exploited in many problem domains for faster computation at no performance loss, for all other values of λ we see a loss in expected rewards. Currently, we are unable to bound the difference between the corrected and predicted beliefs for the action in terms of λ. As a result, the error introduced in the reward function by the approximation may not be bounded. However, our empirical results in Table 2.2 indicate that if λ is relatively low, we obtain solutions of quality comparable to the original techniques. We selected IP for demonstration because it is one of the quickest exact POMDP solution techniques, while PBVI is representative of a general class POMDP approximation techniques that utilize a small subset of belief points in backup operator for

Method	R	Time (secs)	H	$\|\Gamma\|$	Speedup%
Tiger *(2s, 3a, 2o)*					
IP	9.41	3.83 ± 0.2	226	9	
IP + λ=1	9.41	3.4 ± 0.22	226	9	∼12
PBVI	8.96	0.16 ± 0.2	30	9	
PBVI + λ=1	8.96	0.1 ± 0.01	30	9	∼23
Web-Mall *(2s, 3a, 2o)*					
IP	3.24	3.52 ± 0.08	216	27	
IP + λ=1	3.24	3.43 ± 0.03	216	27	∼3
PBVI	3.09	0.37 ± 0.07	30	13	
PBVI + λ=1	3.09	0.26 ± 0.02	30	13	∼29
Machine Maintenance *(3s, 4a, 2o)*					
IP	4.00	9.43 ± 0.20	216	27	
IP + λ=1	4.00	6.50 ± 0.18	216	27	∼31
PBVI	3.28	0.41 ± 0.01	36	12	
PBVI + λ=1	3.28	0.30 ± 0.01	36	12	∼26
Machine_256 *(256s, 4a, 16o)*					
IP	1.62	0.08	10	2	
IP + $\lambda = 1$	1.62	0.04 ± 0.01	10	2	∼47
PBVI	1.33	266.76 ± 2.06	20	1	
PBVI + $\lambda = 1$	1.33	233.98 ± 2.95	20	1	∼12
Fugitive *(9s, 5a, 4o)*					
IP	40.17	68.33 ± 0.88	3	1436	
IP + λ=1	40.17	67.11 ± 0.42	3	1436	∼2
PBVI	345.9	31.65 ± 1.07	31	20	
PBVI + λ=1	345.9	18.93 ± 0.44	31	20	∼40
Learning c2 *(12s, 8a, 3o)*					
IP	0.40	0.72	2	338	
IP + λ=10	0.39	0.03	2	27	91
PBVI	0.63	127.17 ± 3.57	6	873	
PBVI + λ=10	0.63	16.65 ± 0.07	7	201	∼87

scalability. If λ is high to the extent that all actions in a problem domain are identified and exploited, the approach may not result in good quality solutions due to high error. The

Learning c3 *(24s, 12a, 3o)*					
IP	0.39	54.22 ± 1.93	2	2680	
IP + λ=10	0.38	0.77 ± 0.01	2	54	∼99
PBVI	0.78	608.94 ± 10.5	8	880	
PBVI + λ=10	0.79	158.35 ± 1.79	10	312	74
Learning c4 *(48s, 16a, 3o)*					
IP	–	–	–	–	
IP + λ=10	–	–	–	–	
PBVI	0.78	2025.7 ± 41.8	11	896	
PBVI + λ=10	0.79	636.39 ± 10.8	12	338	∼69

Table 2.2: Significant speed ups are obtained for several problems from the POMDP repository when λ-information inducing actions are exploited for different λ. '–' indicates that the problem could not be solved by the approach for at least horizon two due to insufficient memory. The run times were obtained on a Intel dual-core 2.8GHz, 4GB RAM platform with Linux.

resulting solution would be the same as the value of the blind policy which is known to be a lower bound on POMDP value function [26]. Thus, low values of λ that identify a subset of actions are preferable. Consequently, the approach should not be used for problems where the observation functions are identical for most actions.

Chapter 3

Generalized and Bounded Policy Iteration for Finitely Nested I-POMDPs

One class of POMDP solution techniques involves searching the solution space directly to find an optimal policy. Initially proposed in the context of POMDPs [23], the technique represents the infinite horizon solution as a *finite state automaton* and iteratively improves it until it converges to an optimal value. The benefit is that the finite state controller typically converges before its value converges across all states and it is useful when the sequential decision making is to be performed over an infinite number of time steps. 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 [51]. However, although this scales POMDP solutions to larger problems 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 [5] leading to improved scalability.

I introduce a novel *generalization* of BPI to the context of finitely-nested I-POMDPs leading to a new anytime approximation technique for decision making in multiagent settings. The generalization must contend with an interactive state space that includes candidate models of the other agents making the space infinite. The intentional models include beliefs of the other agents that are updated with time, and the models themselves could be finitely-nested I-POMDPs. The generalization improves on previous approximation techniques for I-POMDPs on two important fronts thereby making it significant: we may solve significantly larger problem domains and generate solutions with significantly better quality compared to those generated by previous techniques.

In generalizing BPI to I-POMDPs, we first represent a solution of the other agent's I-POMDP at the next lower level as a finite state controller (FSC) and reformulate the interactive state space of the subject agent to include the physical states, the set of nodes in a controller ascribed to the other agent and how they transition, with possible loss of generality in practice. For domains where the other agent may be ascribed multiple candidate frames representing possibly different capabilities and preferences, we may include multiple controllers in the set that is ascribed to the other agent. For multiple other agents, we may ascribe joint controllers: one for each other agent. The solution of the I-POMDP at the current level is also a FSC which is used in lieu of its models in solving the I-POMDP of the next higher level. The presence of controllers at different levels for the agents leads to novel challenges and alternative approaches. We do not follow an approach wherein we first solve the I-POMDP of the lower level agent completely to get a fixed controller which we then use in the I-POMDP of the next higher level. Instead, in order to facilitate *anytime* behavior for the approach, we interleave the evaluation and improvement of the controllers at different levels of nesting. Each iteration may recursively involve evaluating and possibly

improving the controller of the other agent followed by improvement of the subject agent's controller. However, the dynamic embedded controller alters the state space. This motivates first evaluating the current controller on the new interactive state space, the one with possibly modified model space, before improving it. This approach differs from BPI's implementation in decentralized POMDPs where the controller for each agent is improved independently, but an optional correlation device is introduced for coordination among them. Such a shared source of randomness is usually not feasible in non-cooperative settings. Importantly, we observe that the value of the subject agent's controller can't converge unless the lower-level controllers converges first.

The benefit of this approach 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. However, the added cost is a result of the formalization of the I-POMDP framework.

Note that given a fixed controller which represents a solution of the lower-level models, we may formulate the I-POMDP at a particular level as a POMDP with a complex state space that is a join of the physical state space and the nodes of the other agent's FSC. The transition, observation, and reward function could be altered accordingly by marginalizing the actions of the other agent from its nodes. Subsequently, we could apply single-agent BPI directly to this POMDP. However, our focus on interleaving improvements at various levels means that the ascribed controller at any level may change, and the POMDPs would be repeatedly reformulated making this alternative approach inconvenient for anytime solution purpose. We call our approach *interactive BPI*. We experimentally evaluate its properties using benchmark problems. In particular, we show that the converged controller for the subject agent generates solutions of significantly better value in proportionately less time compared to results reported by the previous best I-POMDP approximation, interactive

point-based value iteration (I-PBVI) [15]. Furthermore, we provide solutions of I-POMDPs that are nested to levels as deep as four for the first time. Ultimately, this allows the application of I-POMDPs to scale to more realistic domains with reduced trade off in value of the solution compared to previous approximations, as we demonstrate by applying the technique to substantially larger problem domains, which are inspired by real-world applications.

The policy improvement step suggested in our approach replaces a node in the current controller with a new node whose value uniformly improves the value of the original node over the entire belief simplex. This approach is suitable for offline planning when the initial belief is not available. However, often an agent is equipped with an initial belief which could be used to bias the improvement step in regions of belief that are of interest for the solution. Hence, we extend interactive BPI with a method that improves the controller keeping the initial belief in mind. This additionally requires solving a system of linear equations in each iteration which could be computationally costly. We compensate for the additional computations by utilizing beliefs reachable from the initial one to iteratively prune some nodes in the other agent's controllers that are of little significance for the solution. These are nodes that cannot be reached given the initial belief and correspond to models that are implausible. However, the impossible nodes may become probable as the controller is iteratively improved. While it is difficult to estimate the future form of the controller, we selectively prune those nodes, which as a heuristic lead to a small loss in current value of the controller for any belief. This is a novel use of initial beliefs that is pertinent in multiagent settings and specific to interactive BPI. We evaluate the improved performance of the controllers and report on the run times as well.

I list the novel contributions of this article below:

- A straightforward method for extending BPI to the context of finitely-nested I-POMDPs would be to evaluate and improve the controllers of the lower level until convergence to a satisfactory local optima before moving to the next higher level. However, the

higher-level controller may not be improved for some time until the ones at the lower levels have converged. Hence, we present a more sophisticated approach, which interleaves improvements of the other agent's controllers with improvements of the subject agent's controller. This approach facilitates *anytime behavior*, with the challenge that the interactive state space may change dynamically at every iteration. Therefore it is necessary to re-evaluate the controller for the updated model space.

- The interactive state space in I-POMDPs includes candidate models of the other agents which contain the belief of the other agent over its interactive states thereby making the space uncountably infinite. Our approach solves the I-POMDPs of the other agents as finite state controllers which may be used to predict other agents' actions. Hence the controllers could replace the models of the other agents in the interactive space of the subject agent with some loss in solution quality. The benefit of this approach is that the controllers finitely and compactly represent the model space.

- We extend our base approach to exploit the agent's initial belief of the subject agent, if available, toward obtaining a more compact and improved controller for the given initial belief. This modification is significant mainly in larger problem domains. However, it incurs additional computation costs, which we compensate by removing models that are unlikely to be of much significance to the value of the initial belief.

- Our extensive evaluations demonstrate that the generalization of BPI improves on previous approximation techniques for I-POMDPs on two important fronts: we may solve significantly larger problem domains and generate solutions of much better quality.

- We extend our approach to solve, for the first time, problems involving multiple other agents and problems in which the other agents are ascribed multiple candidate frames.

- Finally, while BPI has been extended to decentralized POMDPs, the context there is strictly cooperative. Its novel generalization makes it useful in non-cooperative settings as well.

The rest of this chapter is structured as follows. I provide a brief background on BPI in the context of POMDPs, in Section 3.1. I generalize bounded policy iteration to multiagent settings as formalized by I-POMDPs in Section 3.2. I discuss the interactive BPI algorithm in Section 3.3 and analyze the computational savings that it brings. I show how we may exploit the subject agent's initial beliefs toward further improving the quality of the solution, in Section 3.4. In Section 3.5, I evaluate interactive BPI's performance on several problem domains ranging from toy benchmarks to substantially larger problems, and as other parameters of the context are varied. Finally, I conclude this article with a discussion in Section 3.7 of open concerns and future avenues of work. In the Appendix, I show the general algorithm for interactive BPI in the context where multiple agents are present, which are modeled using several candidate frames.

3.1 Policy Iteration for POMDPs

In contrast to value iteration which iterates over the value function (Eq. 1.9) till convergence or for a finite horizon, policy iteration searches directly over the policy space. The traditional representation of a policy is as a function that maps beliefs to actions. An early method by Sondik [66] for performing policy iteration in the context of single-agent POMDPs defined a canonical representation of a policy as a stationary mapping from polytopes of beliefs to actions where the polytopes were represented using linear inequalities. Hansen [23] proposed that representing the converged policy of an agent as a *finite state automaton* is more convenient for the purpose of policy iteration. The drawback being that it may not represent all optimal infinite-horizon policies, though it will get arbitrarily close in value. At any point

in the iteration, the finite state controller represents an infinite horizon policy of the agent for any belief.

We may define a simple (deterministic) controller for an agent i in a single-agent setting, 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, $\mathcal{T}_i : \mathcal{N}_i \times \mathcal{E}_i \rightarrow \mathcal{N}_i$. Based on the edge label (observation), a node transitions to another node in the deterministic controller. For convenience of presentation, 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 similar to equation 1.11. We may view each node in the controller as the root node of an infinite horizon policy such that it is associated with an action and a value vector which represents the expected reward of following the policy. As the value function is improved, new vectors may be introduced causing additional nodes to be added in the controller. In order to somewhat check the size of the controller, some nodes may be dropped if their corresponding vectors are dominated at all states by some other vector [23]. Nevertheless, the solution size still grows rapidly. Pruning vectors that are jointly dominated by multiple vectors introduces stochastic transitions between the nodes of the controller [51].

Despite the best pruning methods, controllers often grow exponentially in size at every improvement step making evaluation and further improvement intractable. Poupart and Boutilier [51] show that the controller size may be minimized and, in fact kept bounded, in two ways: First, we may prune a node whose corresponding vector is dominated by a convex combination of a subset of other vectors. In order to determine if a vector is

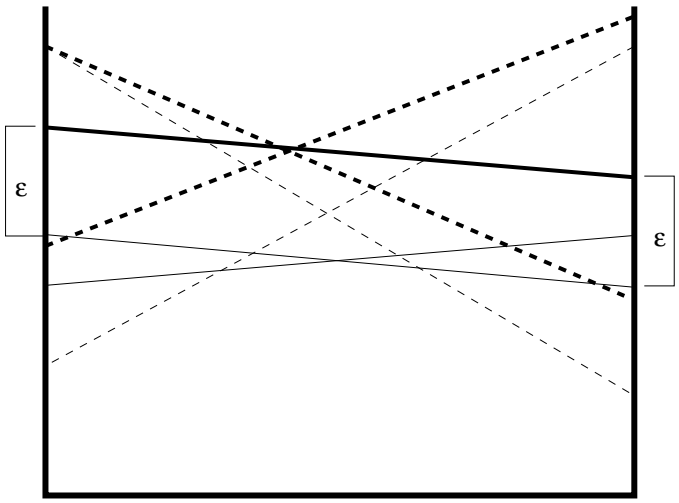


Figure 3.1: The dashed vectors constitute the optimal, backed up value function. Value vector (solid line in bold), representing a node in the improved controller, is a convex combination of the two dashed backed-up vectors in bold. It point-wise dominates a vector that constitutes the value function of the previous controller, by ϵ. Notice that none of the dashed vectors fully dominate the previous vectors by themselves.

dominated jointly by multiple vectors, a convex-combination vector passing through the point of intersection of the combined vectors and parallel to the dominated vector is computed, as illustrated in Fig. 3.1. This is equivalent to replacing multiple updated vectors with a single candidate vector and allows us to prune nodes that are completely dominated by this convex combination, which otherwise 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 to a fixed point, a backup that generates new linear vectors and corresponding nodes is guaranteed to improve it in value [23]. Thus, short of performing a full backup and risking an exponential growth in the size of the controller we may perform a partial backup to compute a convex combination of backed up vectors that uniformly improves the value of an existing vector to replace the latter in the controller. This causes the action mapping, $\mathcal{L}_i$, to be stochastic as well. Consequently, we may redefine $\mathcal{L}_i$ as,

$\mathcal{L}_i : \mathcal{N}_i \rightarrow \Delta(A_i)$, and the edge transition function as, $\mathcal{T}_i : \mathcal{N}_i \times A_i \times \Omega_i \times \mathcal{N}_i \rightarrow [0, 1]$. [1] Of course, the technique becomes susceptible to converging to local optima in our desire to keep the number of nodes fixed.

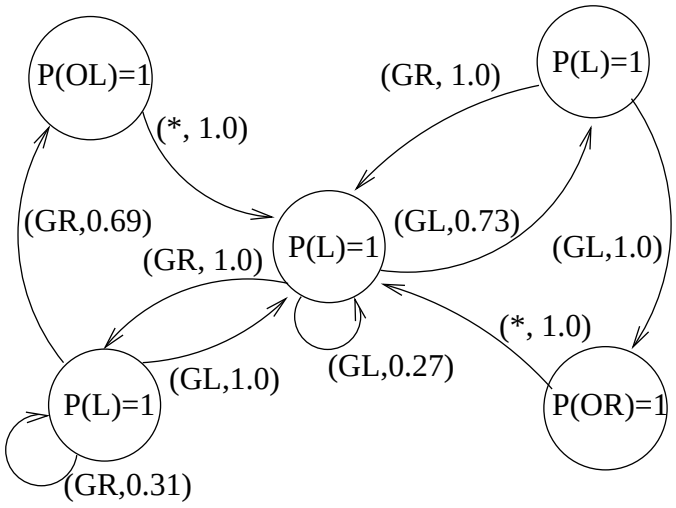


Figure 3.2: An example converged controller for the single agent tiger (Eg. 3) problem. In this controller, all nodes represent deterministic actions, although in general they could be stochastic. Note that transitions due to observations are stochastic with probabilities given on the edge labels, and '*' indicates any observation.

Example 4 (Finite state controller)**.** *We show an example stochastic finite state controller for the single agent tiger problem, which has converged, in Fig. 3.2. Notice the proliferation of edges in the controller because of stochastic transitions due to observations.*

3.2 Generalized Policy Iteration for Multiagent Settings

I generalize the bounded policy iteration technique of Section 3.1 to the context of I-POMDPs nested to a finite level, l. For simplicity of presentation, I assume that the frame of the other

[1] A node in the controller mapped to an action distribution may be split into multiple nodes. Each node is deterministically mapped to a single action and incoming edges to the original node now connect to the split nodes with the action distribution.

agent j, $\hat{\theta}_{j,l-1}$, is known and remains fixed; it need not be the same as that of agent i. Our approach generalizes to multiple candidate frames and other agents as mentioned later in this section.

The subject agent ascribes a finite state automaton to the other agent, which replaces the intentional models of the other agent. I formalize the corresponding transformation of the interactive state space including the mapping of beliefs over the transformed space in Section 3.2.1 and discuss its implications. In Sections 3.2.2 and 3.2.3, I describe the evaluation and improvement of finite state controllers, and a method for escaping from local optima in Section 3.2.4. Finally, in Section 3.2.5 I discuss the novel challenges and alternative approaches for improving the controllers at the different levels.

3.2.1 Transforming the Interactive State Space

Let $\pi_{i,l} = \langle \mathcal{N}_{i,l}, \hat{f}_{i,l} \rangle$ be a controller for the level l agent i, where $\mathcal{N}_{i,l}$ is the set of nodes in the controller and $\hat{f}_{i,l}$ groups the remaining parameters of the controller as mentioned previously in Section 3.1. Analogously, let j's level $l-1$ controller be defined as $\pi_{j,l-1}$. Next, we define a set, $\mathcal{F}_{j,l-1}$, 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 the remaining part of the controller, $\pi_{j,l-1}$; and $\hat{\theta}_{j,l-1}$ is j's frame.

Recall that an intentional model, $\theta_{j,l-1} \in \Theta_{j,l-1}$, consists of j's belief, $b_{j,l-1}$, and the frame. Additionally, recall from Section 3.1 that each node in a finite state controller is associated with a value vector, which provides the converged value of performing the action(s) associated with the node and following the controller thereafter. Consequently, each model, $\theta_{j,l-1}$, may be mapped to a node in the following way: Compute the inner product between the belief in the model, $b_{j,l-1}$, and the value vector associated with a node, for all nodes in the controller. Then, map the model to the node whose vector results in the larger inner product breaking

ties randomly. The mapped node gives the current action and subsequent behavior for the model as prescribed by the controller.

Proposition 1 asserts an important property of this mapping, which we denote as, $\mathcal{K}_{\pi_j} : \Theta_{j,l-1} \rightarrow \mathcal{F}_{j,l-1}$, and it forms the basis for our approach.

Proposition 1 (Partition)**.** *Given a finite state controller for agent j, $\pi_{j,l-1}$, the mapping, $\mathcal{K}_{\pi_j}$, induces an equivalence relation between j's intentional models thereby partitioning the intentional model space, $\Theta_{j,l-1}$, ascribed to agent j.*

Proof. As per the mapping procedure mentioned previously, each intentional model, $\theta_{j,l-1} \in \Theta_{j,l-1}$, is mapped to a node in the controller, $\pi_{j,l-1}$, and multiple models may be mapped to a single node. These models are equivalent. Furthermore, if two or more value vectors associated with different nodes lead to the largest inner product, we break ties randomly and map the model to one of the nodes. Therefore, no model is mapped to more than one node and all the models are mapped. Consequently, nodes in a controller, $\pi_{j,l-1}$, allow the map, $\mathcal{K}_{\pi_j}$, that *partitions* the intentional model space. □

In other words, for a belief in any model in $\Theta_{j,l-1}$, a node exists in the controller that will provide the corresponding action(s). Corollary 1 then follows trivially:

Corollary 1 (Compact interactive state space)**.** *If the finite-state controller, $\pi_{j,l-1}$, is optimal over the infinite horizon for the frame, $\hat{\theta}_{j,l-1}$, then without loss of generality 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 obtained as the output of $\mathcal{K}_{\pi_j}(\theta_{j,l-1})$, for some model, $\theta_{j,l-1} \in \Theta_{j,l-1}$.

Given the general undecidability of the problem of obtaining an exact infinite-horizon solution of a POMDP [47] and that not all POMDPs admit an optimal finite-state controller [23], it may be difficult to obtain an optimal, $\pi_{j,l-1}$, in practice. Therefore, in practice, the transformation of the interactive state space given in Corollary 1 may be with some loss in generality because the mapping of j's models to the nodes in its controller cannot reflect its optimal behavior in some cases, thereby leading to performance loss for agent i. The loss in performance may be reduced by obtaining an infinite-horizon solution that is near optimal. Because our approach involves progressively improving this controller, the given controller may represent an initial solution for the entire intentional model space. Given $\pi_{j,l-1}$, this transformation of the interactive state space involves finding the mapped $f_{j,l-1}$ for each intentional model. This involves determining the corresponding node of the controller. For the $|\Theta_{j,l-1}|$ many intentional models, determining $\mathcal{K}_{\pi_j}(\theta_{j,l-1})$ for all models takes $\mathcal{O}(|\Theta_{j,l-1}||\mathcal{N}_{j,l-1}||IS_{j,l-1}|)$ time.

If there are $|\hat{\Theta}_j| > 1$ frames with differing capabilities and preferences, the finite set $\mathcal{F}_{j,l-1}$ in the interactive state space will be larger with $f_{j,l-1} \in \mathcal{F}_{j,l-1}$ differing in $\hat{f}_{j,l-1}$ and $\hat{\theta}_{j,l-1}$ as well. In other words, $\mathcal{F}_{j,l-1}$ may contain as many distinct controllers as there are frames. Furthermore, if there are K other agents, 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 controller for each agent k. This is because the agents may differ in their frames and receive private observations. Consequently, their controllers may evolve or transition differently from each other.

Because the set of nodes in $\mathcal{F}_{j,l-1}$ is finite, an important benefit of the above representation is that the uncountably 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 has often been a hurdle for previous approximation techniques that operate on it, such as the interactive point-based value iteration [15]. 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. Notable among them is the quantal response model, which assumes different models as a single parameter is varied [22]. The parameterized quantal response allocates non-zero probability to each action, which is an exponential function of the action's value. However, the space of models obtained in this way does not exhaustively cover the entire space of models because the action distributions are constrained to follow the logit function.

An interesting question is, "Given the transformed interactive state space, can we represent I-POMDP$_{i,l}$ as a POMDP with a large state space?" In particular, the subject agent's anticipation of how the other agent updates its models is reflected by the transitions between nodes in the other agent's controller, $\pi_{j,l-1}$. If the controller does not change with time, then the joint distribution of the next interactive state, $\langle s, f_{j,l-1}\rangle'$, and observation, ω_i, given agent i's current state, $\langle s, f_{j,l-1}\rangle$, and action, a_i, is given by: $Pr(\omega_i, \langle s, f_{j,l-1}\rangle' | a_i, \langle s, f_{j,l-1}\rangle) = \sum_{a_j} Pr(a_j|n_{j,l-1})\ O_i(s', a_i, a_j, \omega_i) \sum_{\omega_j} O_j(s', a_i, a_j, \omega_j)\ T_i(s, a_i, a_j, s')\ Pr(n'_{j,l-1}|n_{j,l-1}, a_j, \omega_j)$, where $n_{j,l-1}$ is a part of $f_{j,l-1}$, O_i and T_i are obtained from the I-POMDP definition, and O_j is part of j's frame in $f_{j,l-1}$. Consequently, we may solve the transformed I-POMDP$_{i,l}$ given j's controller as a large POMDP. On the other hand, if the ascribed controller changes, we would need to reformulate the POMDP.

Agent i's belief over the original interactive state space, $S \times \Theta_{j,l-1}$, is transformed to a belief over the state space, $S \times \mathcal{F}_{j,l-1}$. We may obtain this belief over the transformed interactive state space as:

$$b_{i,l}(s, f_{j,l-1}) = \sum_{\theta_{j,l-1}:\mathcal{K}_{\pi_j}(\theta_{j,l-1})=f_{j,l-1}} b_{i,l}(s, \theta_{j,l-1}) \tag{3.1}$$

where $\mathcal{K}_{\pi_j}$ is the mapping as defined previously. The complexity of the belief transformation is dominated by the complexity of finding the mapped $f_{j,l-1}$ for each intentional model. This involves determining the corresponding node of the controller. For the $|\Theta_{j,l-1}|$ many

intentional models, determining $\mathcal{K}_{\pi_j}(\theta_{j,l-1})$ for all models takes $\mathcal{O}(|\Theta_{j,l-1}||\mathcal{N}_{j,l-1}||IS_{j,l-1}|)$ time. This increases to $\mathcal{O}(|\Theta_{j,l-1}|^K\ |\mathcal{N}_{j,l-1}||IS_{j,l-1}|)$ for $K > 1$ other agents in the context. [2]

Because multiple belief distributions over the original interactive state space may sum to the same belief over the transformed space, Corollary 2 states an important property of this belief.

Corollary 2 (Many-to-one mapping)**.** *The belief transformation as defined in Eq. 3.1 is a many-to-one mapping.*

Let, $\pi_{i,l}$ 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 are used to recursively update controllers at the lower levels as well thereby generalizing the iterations to multiagent settings.

3.2.2 Policy Evaluation

As I mentioned previously, each node, $n_{i,l}$, in the controller is associated with a vector of values, $\alpha_i(\cdot, n_{i,l})$, that gives the expected (converged) value, at each interactive state, of following the controller beginning from that node. In the context of I-POMDPs with one other agent whose frame is known, 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 each of which is analogous to Eq. 1.10 but with controller nodes substituting for the models:

[2]Because probability measures are countably additive, Eq. 3.1 remains mathematically well-defined although the subset of intentional models that map to some $f_{j,l-1}$ could be countably infinite. Of course, in practice we consider a finite set of intentional models for the other agent.

$$\alpha_i(s, n_{j,l-1}, n_{i,l}) = \sum_{a_i \in A_i} Pr(a_i|n_{i,l}) \sum_{a_j \in A_j} Pr(a_j|n_{j,l-1})$$
$$\times \Bigg\{ R_i(s, a_i, a_j) + \gamma \sum_{\omega_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s')\ O_i(s', a_i, a_j, \omega_i)$$
$$\times \sum_{\omega_j} O_j(s', a_i, a_j, \omega_j)\ Pr(n'_{j,l-1}|n_{j,l-1}, a_j, \omega_j) \qquad (3.2)$$
$$\times \sum_{n'_{i,l}} Pr(n'_{i,l}|n_{i,l}, a_i, \omega_i)\ \alpha_i(s', n'_{j,l-1}, n'_{i,l}) \Bigg\}$$
$$\forall s, n_{j,l-1}, n_{i,l}$$

In Eq. 3.2, 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 $Pr(a_i|n_{i,l})$, $Pr(n'_{i,l}|n_{i,l}, a_i, \omega_i)$ and $Pr(a_j|n_{j,l-1})$, $Pr(n'_{j,l-1}|n_{j,l-1}, a_j, \omega_j)$ are obtained from $\hat{f}_{i,l}$ and $\hat{f}_{j,l-1}$, respectively, and O_j is obtained from j's frame; $\hat{f}_{j,l-1}$ and j's frame are present in $f_{j,l-1}$. Equation 3.2 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, $Pr(b'_{j,l-1}|b_{j,l-1}, a_j, \omega_j)$ in Eq. 1.10, is represented using a transition from one node to another by the term, $Pr(n'_{j,l-1}|n_{j,l-1}, a_j, \omega_j)$. Equation 3.2 extends the policy evaluation step of BPI for single-agent POMDPs [51] by including the update of other agent's behavior based on its predicted action and anticipated observations. Alternately, we could apply BPI's policy evaluation directly to a complex POMDP. As we mention in Section 3.2.1, any change in the ascribed controller would require a complete reformulation of the POMDP. On the other hand, only the terms related to the ascribed controller change in Eq. 3.2.

Solution of the system, which contains $|S||\mathcal{N}_{j,l-1}||\mathcal{N}_{i,l}|$ many variables and as many equations, results in value vectors over the reformulated model space for each node in agent i's controller. The asymptotic time complexity of solving the system of linear equations is polynomial in the number of variables. This increases to $|S|(|\mathcal{N}_{j,l-1}||\hat{\Theta}_j|)^K|\mathcal{N}_{i,l}|$ many

variables in the presence of K other agents each with $|\hat{\Theta}_j|$ distinct frames. 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.

3.2.3 Policy Improvement

A straightforward way of improving the controller is to perform a backup operation on the vectors of the current controller, analogously to the backup operation shown in Section 1.2 but with the models replaced with the nodes in the controller, to create new nodes and then prune out the nodes whose corresponding vectors are dominated by those of one or more other nodes. However, as I mentioned earlier, this method of policy improvement could lead to exponential growth in the solution size at each improvement step.

An alternate approach was proposed by Poupart and Boutilier [51] in the context of POMDPs. Instead of first updating the value vectors using the backup operation and then checking for pointwise dominance, the approach performs a partial back up that integrates the two in a single linear program. The controller is improved by evaluating whether a node, $n_{i,l}$, in i's controller may be replaced with a new node, possibly a convex combination of the updated vectors generated directly without explicitly generating those updated vectors, whose value is better at all interactive states.

Our linear program extends that for a single-agent POMDP by involving additional terms related to j's dynamic behavior as guided by its controller embedded in the interactive state space. While BPI's improvement step could be directly used, the associated disadvantage of repeatedly reformulating the large POMDP remains. We show the linear program below:

$$
\begin{aligned}
& max \;\; \epsilon \\
& \; s.t. \; \alpha_i(s, n_{j,l-1}, n_{i,l}) + \epsilon \leq \sum_{a_i} c_{a_i} \sum_{a_j} Pr(a_j | n_{j,l-1}) \\
& \times \Bigg\{ R_i(s, a_i, a_j) + \gamma \sum_{\omega_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s') \\
& \times O_i(s', a_i, a_j, \omega_i) \sum_{\omega_j} O_j(s', a_i, a_j, \omega_j) \\
& \times Pr(n'_{j,l-1} | n_{j,l-1}, a_j, \omega_j) \sum_{n_{i,l}^{\omega_i}} c_{a_i, n_{i,l}^{\omega_i}} \alpha_i(s', n'_{j,l-1}, n_{i,l}^{\omega_i}) \Bigg\} \\
& \forall \;\; s, n_{j,l-1}; \\
& \sum_{a_i} c_{a_i} = 1; \;\; \sum_{n_{i,l}^{\omega_i}} c_{a_i, n_{i,l}^{\omega_i}} = c_{a_i} \;\; \forall a_i, \omega_i; \\
& \; c_{a_i, n_{i,l}^{\omega_i}} \geq 0 \;\; \forall a_i, \omega_i, n_{i,l}^{\omega_i}; \;\; c_{a_i} \geq 0 \;\; \forall a_i
\end{aligned}
\tag{3.3}
$$

The value function terms in Eq. 3.3 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, $n_{i,l}^{\omega_i}$, on performing action a_i and receiving observation ω_i as, $Pr(n_{i,l}^{\omega_i} | n'_{i,l}, a_i, \omega_i) = c_{a_i, n_{i,l}^{\omega_i}}$.

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. 3.2 have fixated for every node.

Proposition 2 (Value improvement)**.** *If the controller at level l, $\pi_{i,l}$, has not converged, policy evaluation followed by improvement transforms $\pi_{i,l}$ into a controller whose value function is as good for every belief state and better for some belief state(s).*

Proof. Performing policy evaluation first establishes the value vectors of $\pi_{i,l}$ over $IS_{i,l}$ given the current level $l-1$ controller of j. On improvement yielding $\epsilon > 0$, a node is replaced with a new node, $n'_{i,l}$, whose value vector, possibly a convex combination of implicitly backed up value vectors, pointwise dominates the value vector of the previous node. Because edges incoming to the previous node now point to $n'_{i,l}$, the value of some belief state(s) is improved. □

The linear program is efficient in having $|A_i||\Omega_i||\mathcal{N}_{i,l}| + |A_i| + 1$ variables and $|S||\mathcal{N}_{j,l-1}|$ constraints, which increases to $|S|(|\hat{\Theta}_j||\mathcal{N}_{j,l-1}|)^K$ constraints if there are $K > 1$ other agents and $|\hat{\Theta}_j|$ frames for each agent. As we mention later, we utilize the dual simplex method to solve the linear program whose average-case running time is polynomial in the number of variables and constraints [70] though it is possible to construct linear programs such that the worst-case complexity is exponential [38].

3.2.4 Escape from Local Optima

Because of the strategy of obtaining a value vector that is a convex combination of backed up nodes and which uniformly pointwise dominates another vector by ϵ, the iterations may converge on a peculiar *local optima* in which all the value vectors touch from below, or in other words, they are tangential to, the intersections of the backed up value function of the nodes at that step. This is a very common form of local optima. If such conditions on the value vectors are met, no further improvement using Eq. 3.3 is possible. We illustrate this phenomena in Fig. 3.3. Such local optima may prevent the approach from reaching an optimal controller for the given size. Of course, there are other conditions for a local optima to occur as well.

Poupart and Boutilier [51] mention a simple approach of potentially dislodging from the local optima caused by the tangent condition by adding new nodes, which is applicable

in the context of I-POMDPs as well. Specifically, we arbitrarily pick a tangential vector whose value we will seek to improve. The tangent point is obtained from the dual of the LP in equation 3.3. Beliefs reachable from the tangential belief point of intersection are generated followed by adding nodes to the controller that improve the value associated with the reachable beliefs, as we show in Fig. 3.3. This is equivalent to performing a full-back up on the computed reachable beliefs alone. The reachable beliefs are generated using Eq. 1.8 in which the models of j are substituted with corresponding controller nodes. This allows the value of the node representing the tangential vector to improve due to the improved values of reachable nodes.

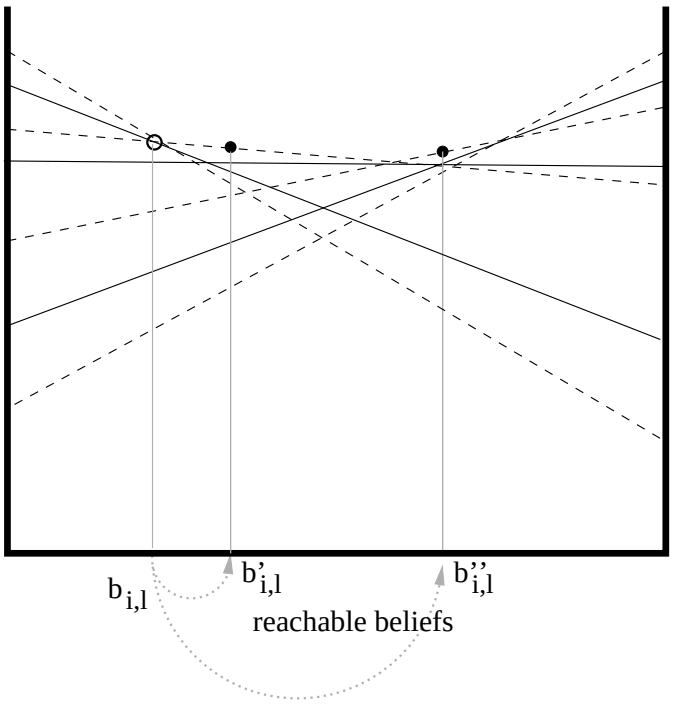


Figure 3.3: The dashed vectors constitute the optimal, backed up value function while the solid vectors constitute the current value function at an optima. Note that these vectors are *tangential* to the intersections of the dashed vectors indicating that a local optima has been reached. Solid dots represent improved values associated with the beliefs, $b'_{i,l}$ and $b''_{i,l}$, obtained from the backed up value function. These beliefs are reachable from $b_{i,l}$ whose value we seek to improve. Corresponding nodes are added to the controller.

Improving controllers while keeping the set of nodes bounded is often limited by local

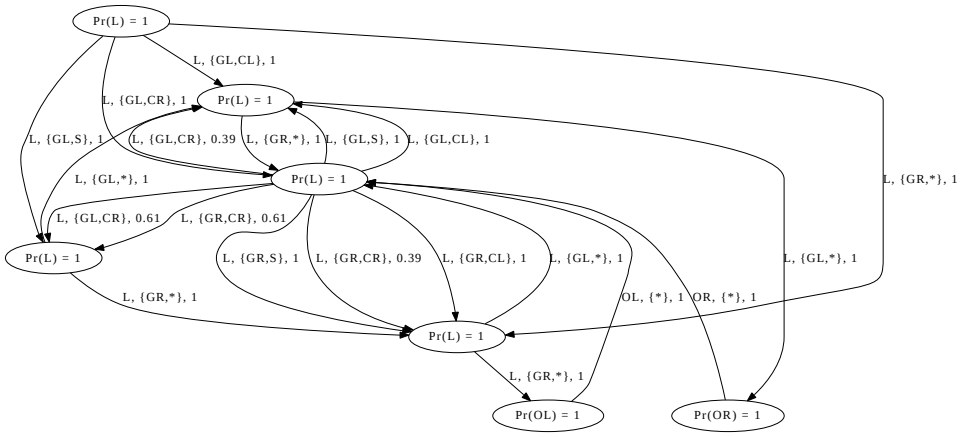
optima, whose value could be arbitrary. Subsequently, an escape technique provides an important way to rescue the approach and produce good quality controllers although the risk remains that it may converge to another local optima. While I-POMDPs permit escapes using a simple approach, we may not quickly escape local optima in the context of other multiagent frameworks such as the decentralized POMDP because it does not explicitly maintain beliefs or defines belief updates.

3.2.5 Embeded Controller at Different Levels

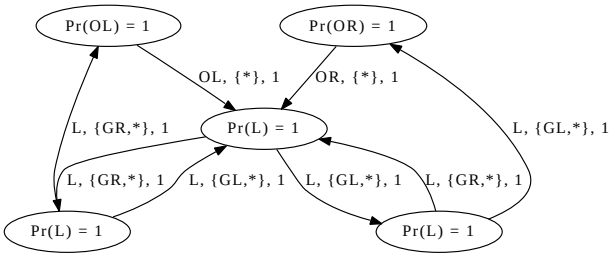
Agent i's controller at level l, $\pi_{i,l}$, is associated with a value function composed of vectors over an interactive state space in which each interactive state includes a node of j's level $l-1$ controller and the other parameters of the controller. In other words, $IS_{i,l}$ embeds j's level $l-1$ controller. Similarly, the interactive state space for agent j contains states which include nodes of i's level $l-2$ controller and the other controller parameters, i.e.$IS_{j,l-1}$ embeds i's level $l-2$ controller. This embedding of controllers terminates at level 0 where the corresponding state space consists of physical states only. In order to better understand controllers at different levels, we show an example.

Example 5 (Controllers at different levels)**.** *In Fig. 3.4, we show an example controller at level 2 that has converged for the multiagent tiger problem, and ascribed controllers at lower levels. Notice that the controllers become more sophisticated as we go up the levels in order to account for increasingly sophisticated behaviors by lower-level agents. Actions at all nodes for each controller are deterministic, though the observations result in stochastic transitions in agent i's level 2 controller.*

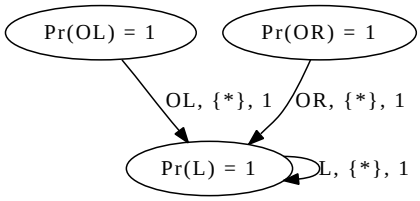
The presence of controllers at different levels leads to novel challenges. Given that the other agent's controller is embedded in agent i's interactive state space, a naive but efficient approach would be to iteratively improve i's controller while holding j's controller in the



Agent i's level 2 controller



Agent j's level 1 controller



Agent i's level 0 controller

Figure 3.4: Controllers at different levels for I-POMDP$_{i,2}$ in the context of the multiagent tiger problem. An edge label such as L, {GL,CL},1 means that the transition occurs with probability 1 due to the agent listening and hearing a growl from the left and creak from the left. '*' indicates any growl or creak as appropriate. The controller at each level has converged possibly to a local optima.

state space fixed. This approach may be reasonable given a readily available good quality controller for the other agent. However, this is rare and in general the approach is naive as the corresponding solution will likely be poor as better quality controllers may be obtained

to predict the other agent's actions. This is particularly relevant because I-POMDPs model the other agent as being rational.

Therefore, we adopt a more sophisticated approach, which interleaves improvements of the other agents' 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. An alternate and appealing technique would be to evaluate and improve the controller of the lower level until convergence before moving to the next level. 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. We illustrate this difference in the context of multiple problems in Section 3.5. Note that Proposition 2 continues to apply for both these techniques. Of course, the higher-level controllers in the two approaches additionally may not converge to the same local optima although this is not the case in our illustration. However, this doesn't guarantee that both approaches will yield the same controllers.

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, another 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 are modified by its improvement step.

Lastly, at level 0 the I-POMDP collapses into a POMDP. Consequently, we may utilize the traditional BPI for POMDPs [51] in order to evaluate and improve the level 0 agent's controller.

3.3 Algorithm and Computational Savings

Algorithm 1 outlines the procedure, labeled Interactive BPI (I-BPI), for performing BPI in the context of finitely nested I-POMDPs with one other agent j. A general version for K other agents is given in the Appendix. 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 (see Algorithm 2). Naturally, the node transitions into itself for all observations. The interactive state space is then reformulated to include the node(s) from the other agent's controller (lines 1-2) incurring the associated computational expense. Please refer to Section 3.2 for discussion on how the interactive state space is reformulated if multiple frames are ascribed to the other agent or if more than one other agent is present in the setting.

Algorithm 1 Interactive BPI for I-POMDPs

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

1: $\pi_{i,l} \leftarrow$ InitializeControllers ($\theta_{i,l}$)
2: Reformulate, $IS_{i,l}$ at each level l in $\theta_{i,l}$ to contain the physical state and the controller of other agent
3: Beginning with level, $l = 0$, perform a single-step full backup at each level, l, using the transformed $IS_{i,l}$, resulting in $|\mathcal{N}_{i,l}| \leq |A_i|$ nodes in a controller, $\pi_{i,l}$
4: **repeat**
5: **repeat**
6: $\pi_{i,l} \leftarrow$ Evaluate&Improve ($\pi_{i,l}$, $\theta_{i,l}$ with reformulated $IS_{i,l}$)
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 controller, $\pi^*_{i,l}$

We perform a single full backup at each level to expand the controller from size 1, which is likely to demonstrate poor performance, to controllers of size $|A_i|$ or $|A_j|$, as appropriate (line 3) reformulating the interactive state space and reevaluating the controller at next higher level in the process. If the frame of the other agent is unknown and the subject agent maintains belief over various frames of other agent, a controller is initialized for each

frame of the other agent. If there are more agents, then a controller is initialized for frame of each other agent and we utilize the joint controllers to represent the joint model space. Subsequent steps of evaluation and improvement are performed for each of these controllers, whose interactive state spaces may themselves contain embedded controllers of the other agent(s). This may give rise to a tree of controllers whose branching factor is the number of frames scaled by the number of other agents in the setting.

Algorithm 2 Recursively initialize controllers at all levels down to 0

InitializeControllers (I-POMDP: $\theta_{i,l}$) **returns** initial controller, $\pi_{i,l}$

1: **if** $l \geq 1$ **then**
2: $\quad \pi_{j,l-1} \leftarrow$ InitializeControllers $(\theta_{j,l-1})$
3: Construct a controller, $\pi_{i,l}$, with a single node, $|\mathcal{N}_{i,l}| = 1$, mapped to a randomly selected action

Algorithm 3, Evaluate&Improve, then recursively performs a single step of evaluation of the nested controller(s) and its bounded improvement (lines 1-2). For the lowest-level controller, the evaluation and improvement proceeds as outlined by Poupart and Boutilier [51] in the context of single-agent POMDPs, which is briefly reviewed in Section 3.1 (lines 3-5). At levels 1 and above, we evaluate the controller using Eq. 3.2 and improve it while keeping the number of nodes fixed using Eq. 3.3 (lines 6-8). The computational complexity of the algorithm is primarily determined by the complexity of the evaluation and improvement steps on the controllers at the different levels. While the number of these steps is not fixed, the complexity of the evaluation and improvement is discussed in Sections 3.2.2 and 3.2.3, respectively. These steps are performed for each agent and its frame in general.

Observe that I-BPI interleaves the evaluation and improvement of the controllers at the different levels. On convergence, Algorithm 1 attempts to push the controllers past any local optima, by escaping it (line 8). When this is no longer possible or when the algorithms exceeds a predetermined execution time, the converged controller is returned as the solution of the level l I-POMDP.

Algorithm 3 Evaluation and bounded improvement of the controllers at different levels.

Evaluate&Improve (controller: $\pi_{i,l}$, I-POMDP model: $\theta_{i,l}$) **returns** controller, $\pi'_{i,l}$

1: **if** $l \geq 1$ **then**
2: $\quad \pi_{j,l-1} \leftarrow$ Evaluate&Improve $(\pi_{j,l-1}, \theta_{j,l-1})$
3: **if** l=0 **then**
4: $\quad$ Evaluate controller, $\pi_{i,0} = \langle \mathcal{N}_i, \mathcal{E}_i, \mathcal{L}_i, \mathcal{T}_i \rangle$
5: $\quad$ Improve controller, if possible, analogously to a POMDP [51]
6: **else**
7: $\quad$ Evaluate controller, $\pi_{i,l} = \langle \mathcal{N}_{i,l}, \mathcal{E}_{i,l}, \mathcal{L}_{i,l}, \mathcal{T}_{i,l} \rangle$, using Eq. 3.2
8: $\quad$ Improve controller, if possible, while keeping $|\mathcal{N}_{i,l}|$ fixed using Eq. 3.3
9: **return** improved controller, $\pi'_{i,l}$

Example 6 (I-BPI)**.** *We illustrate a step within* I-BPI *on a level 2 I-POMDP for agent i in Fig. 3.5. Beginning with a single node at each level, we perform a full backup followed by bounded improvement. Because the controllers at each level converged, we perform an escape of the controllers from local optima.*

The complexity of solving I-POMDPs is dominated by the complexity of solving the models. In general, the space of models ascribed to the other agent could be infinite 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 (Corollary 1). However, if we limit the model space and let $|\Theta|$ be a bound on the number of models ascribed to one other agent. Then, the interactive state space for K other agents contains $(K|\Theta|)^l$ models for 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 in Section 3.5.

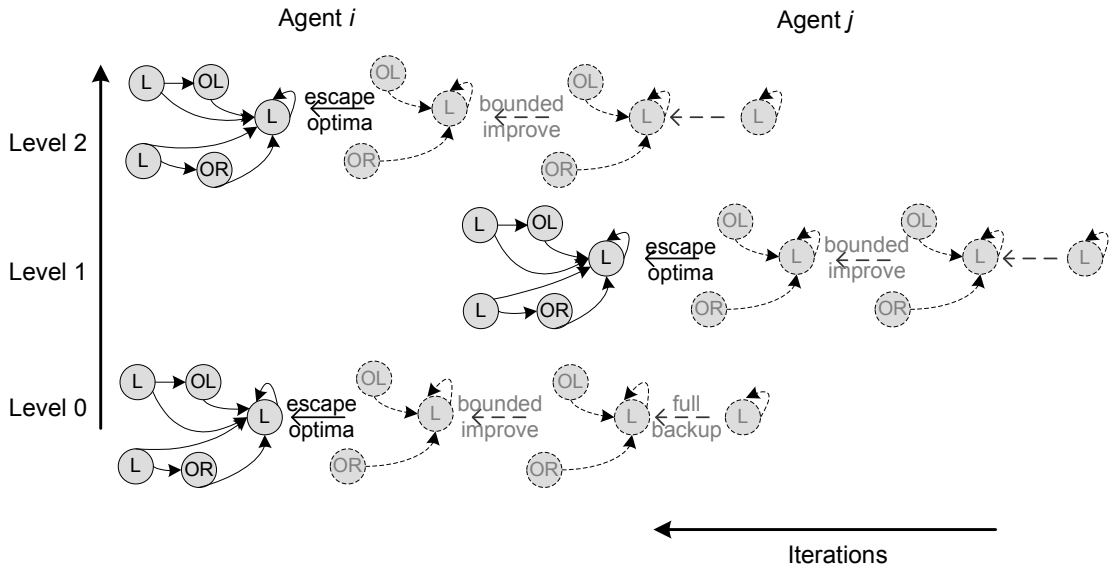


Figure 3.5: Recursive invocations lead to evaluation and improvement beginning at the bottom and up the nesting levels for I-POMDP$_{i,2}$. Controllers were initialized with a single node and a full backup takes place followed by evaluation and bounded improvement in the previous iteration, which does not improve the controllers (shown dashed and greyed out). Current iteration involves escaping the controllers from the local optima bottom up as the recursion unwinds. We demonstrate this process in the context of the multiagent tiger problem.

3.4 Accounting for Initial Beliefs

Although we may not always assume the presence of an initial belief over the interactive state space, in some problem domains such as the money laundering discussed previously in Section 3.5, initial beliefs may be known and could be utilized in order to possibly improve the solution. Subsequently, we seek to extend I-BPI to account for initial beliefs. As I mention in Chapter 1, a belief at level l is a distribution over the interactive state space. Each interactive state includes an intentional model, which may contain a candidate belief over the level $l-1$ interactive state space.

3.4.1 Occupancy Distribution

Poupart and Boutilier [52] compute the occupancy distribution over the physical state space, which assigns an aggregate probability to each state based on the set of beliefs reachable from a given initial belief over an infinite number of steps using the current controller. A fixed point of the occupancy distribution is computed. Policy improvement is then biased toward regions with high occupancy. Particularly, instead of constructing a node whose vector uniformly dominates that of one of the existing nodes, a vector is obtained which improves more in the region of state space exhibiting high occupancy.

This approach may be extended to possibly improve solutions for I-POMDPs as well with associated challenges due to the presence of the other agent and strategy levels. Given an initial belief of level l, $b^0_{i,l}$, which defines a distribution over the physical states and level $l-1$ models, each of which contains a belief over $l-2$ models and so on, we begin by mapping the models at each level up to $l-1$ to the nodes in the initial nested controller per Proposition 1. Subsequently, the initial belief over states and models is transformed to a belief over the states and nodes of the controller using Eq. 3.1 and incurring the associated computational complexity. Given this initial belief, we compute the occupancy distribution, Occ_i, over all physical states, nodes of agent j's $l-1$ controller, and those of i's controller, by solving a system of linear equations as shown below:

$$
\begin{aligned}
&Occ_i(s', n'_{j,l-1}, n'_{i,l}) = \dot{b}^0_{i,l}(s', n'_{j,l-1}, n'_{i,l}) + \gamma \sum_{n_{i,l}} \sum_{a_i} Pr(a_i|n_{i,l}) \\
&\times \sum_{s, n_{j,l-1}} \sum_{a_j} Pr(a_j|n_{j,l-1}) \sum_{\omega_i} O_i(s', a_i, a_j, \omega_i) T_i(s, a_i, a_j, s') \\
&\times \sum_{\omega_j} O_j(s', a_i, a_j, \omega_j) Pr(n'_{j,l-1}|n_{j,l-1}, a_j, \omega_j) \\
&\times Pr(n'_{i,l}|n_{i,l}, a_i, \omega_i)\ Occ_i(s, n_{j,l-1}, n_{i,l}) \\
&\forall s', n'_{j,l-1}, n'_{i,l}
\end{aligned}
\tag{3.4}
$$

Here, $\dot{b}^0_{i,l}(s', n'_{j,l-1}, n'_{i,l})$ is equal to i's initial level l belief, $b^0_{i,l}$, on the interactive state, $\langle s', n'_{j,l-1} \rangle$, if $n'_{i,l}$ is the node in i's controller that the initial belief maps to – the value vector associated with the node leads to the largest inner product with the belief over the interactive state; for any other node the probability is 0. The computed distribution is not normalized. Equation 3.4 generalizes the computation of the occupancy distribution for a single-agent POMDP [52] to the multiagent setting by predicting the other agent's action and accounting for how j updates its policy based on its private observation per the controller ascribed to it.

Analogously, we compute the occupancy distribution from the belief in each level $l-1$ model of j and using j's controller. We obtain a single aggregate occupancy distribution for level $l-1$ by performing a convex combination of the different occupancies. The combination weights are the marginal probabilities assigned to each model in i's level l initial belief summed over the physical states. Subsequently, we obtain an occupancy distribution at each level.

Given the occupancy distribution, we may alter the policy improvement step of Eq. 3.3 to allow non-uniform improvements across all interactive states, $\langle s', n'_{j,l-1} \rangle$, weighted by the respective occupancy values for each node, $n_{i,l}$, as shown in Eq. 3.5:

$$
\begin{aligned}
&max \sum_{s,n_{j,l-1}} \epsilon_{s,n_{j,l-1}} \cdot Occ_i(s, n_{j,l-1}, n_{i,l}) \\
&s.t.\ \alpha_i(s, n_{j,l-1}, n_{i,l}) + \epsilon_{s,n_{j,l-1}} \leq \sum_{a_i} c_{a_i} \sum_{a_j} Pr(a_j | n_{j,l-1}) \\
&\times \Bigg\{ R_i(s, a_i, a_j) + \gamma \sum_{\omega_i} \sum_{s'} \sum_{n'_{j,l-1}} T_i(s, a_i, a_j, s') O_i(s', a_i, a_j, \omega_i) \\
&\sum_{\omega_j} O_j(s', a_i, a_j, \omega_j) Pr(n'_{j,l-1} | n_{j,l-1}, a_j, \omega_j) \sum_{n_{i,l}^{\omega_i}} c_{a_i, n_{i,l}^{\omega_i}} \alpha_i(s', n'_{j,l-1}, n_{i,l}^{\omega_i}) \Bigg\} \\
&\forall \quad s, n_{j,l-1};
\end{aligned}
\tag{3.5}
$$

$$
\sum_{a_i} c_{a_i} = 1;\ \sum_{n_{i,l}^{\omega_i}} c_{a_i, n_{i,l}^{\omega_i}} = c_{a_i} \quad \forall a_i, \omega_i;
$$

$$
c_{a_i, n_{i,l}^{\omega_i}} \geq 0 \quad \forall a_i, \omega_i, n_{i,l}^{\omega_i};\ c_{a_i} \geq 0 \quad \forall a_i
$$

$$
\epsilon_{s,n_{j,l-1}} \geq 0 \quad \forall s, n_{j,l-1}
$$

The occupancy measures bias the value improvement in the reachable regions of the belief space. Specifically, the objective value is larger when larger improvements are obtained for those interactive states that exhibit a higher occupancy as given by Occ_i. Adding the constraint, $\epsilon_{s,n_{j,l-1}} \geq 0$ for all interactive states ensures a non-negative improvement across the entire belief simplex; it may vary for the different states. The linear program above extends an analogous linear program for single agent POMDPs [52] to our multiagent setting. Algorithm 2 now solves the linear program above at each level in order to improve the nested controller, with the occupancy distributions computed just before the improvement step using the previously updated controller.

Computing occupancy is roughly equivalent in complexity to the policy evaluation step of Eq. 3.2 with both systems containing the same number of similar equations. In particular, the system has $|S||\mathcal{N}_{j,l-1}||\mathcal{N}_i|$ many variables that increases to $|S|(|\hat{\Theta}_j||\mathcal{N}_{j,l-1}|)^K|\mathcal{N}_i|$ for K agents and $|\hat{\Theta}_j|$ possible frames, and as many equations, with the asymptotic time complexity

of solving the system being polynomial in the number of variables. Therefore, computing occupancy distributions may impact the execution time non-trivially especially for larger problems, as we demonstrate in the next section.

3.4.2 Pruning Nodes

We aim to compensate for the additional complexity that the revised policy improvement step adds to I-BPI. We may utilize the occupancy distribution to determine those models of the other agent that become impossible given the initial belief. Because the models map to the nodes in j's controller, we utilize the marginal distribution over $n_{j,l-1}$ obtained by summing out s and $n_{i,l}$ from the occupancy distribution, $Occ_i(s, n_{j,l-1}, n_{i,l})$:

$$Pr(n_{j,l-1}) = \sum_{s,n_{i,l}} Occ_i(s, n_{j,l-1}, n_{i,l})$$

Nodes with probability zero in the above marginal on the left are pruned, and these correspond to the nodes in the controller that are unreachable given the initial belief. Reasoning backwards, their incoming edges, if any, do not get traversed and nodes connected to these edges are unreachable as well and must have zero marginalized occupancy. Otherwise, any probability mass on these nodes would "flow" to the former (unreachable) node through the edge making it reachable. Therefore, both incoming and outgoing edges are also removed. By pruning these nodes, we reduce the size of the embedded controller and consequently, the size of the interactive state space by $|S|$ each time a node is pruned.

A limitation of this approach is that as the controller is iteratively improved by substituting nodes in the improvement step and adding nodes during the escape, previously unreachable nodes may become subsequently reachable. While it is difficult to estimate the form of the controller in the future, a straightforward approach to address this would be to reintroduce the pruned nodes into the updated controller before the start of the next itera-

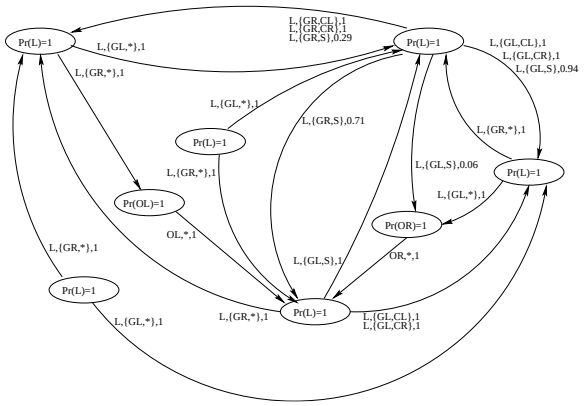
tion, compute the new occupancy distribution and utilize it to again determine the nodes of the other agent's controller with zero marginal. Obviously, this approach repeatedly brings back the pruned nodes.

We seek to avoid this repeated contraction and expansion of the controller; we partially address this challenge in two ways. (i) We prune the nodes infrequently: only after the controller has converged and before we have performed an escape from the local optima by adding new nodes. Subsequently, we do not prune nodes while the controller is iteratively improved. (ii) Although pruning the nodes does not impact the value of the controller keeping the occupancy distribution in mind, it reduces the overall value in general. Therefore, among all the nodes with a zero marginal, we selectively pick those whose negative impact on the current value of the controller in general is among the least. Specifically, for each node with a zero marginal, $\hat{n}_{j,l-1}$, we may compute the current regret due to pruning it, denoted ε, using the simple linear program below:

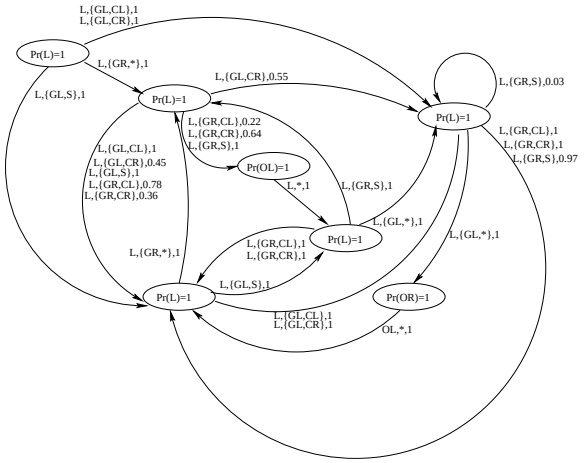
$$
\begin{aligned}
&max \;\; \varepsilon \\
&s.t. \;\; \alpha_j(s, n_{i,l-2}, \hat{n}_{j,l-1}) + \varepsilon \leq \sum_{n_{j,l-1} \in \mathcal{N}_{j,l-1} - \{\hat{n}_{j,l-1}\}} c_{n_{j,l-1}} \alpha_j(s, n_{i,l-2}, n_{j,l-1}) \;\; \forall s, n_{i,l-2}; \\
&\sum_{n_{j,l-1} \in \mathcal{N}_{j,l-1} - \{\hat{n}_{j,l-1}\}} c_{n_{j,l-1}} = 1; \quad c_{n_{j,l-1}} \geq 0 \;\; \forall n_{j,l-1} \in \mathcal{N}_{j,l-1} - \{\hat{n}_{j,l-1}\}
\end{aligned}
\tag{3.6}
$$

As $\varepsilon < 0$, the node with the maximum ε obtained from the linear program above gives the minimum regret, and is pruned. As we mentioned previously, it is difficult to estimate the future form of the controller and the impact of node removal on future values due to the presence of local optima. Therefore, this latter technique of selecting nodes for removal with minimum current regret acts as a heuristic.

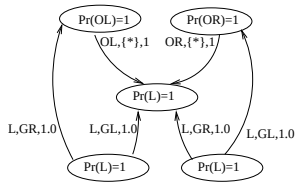
I illustrate the effect of selectively removing nodes with zero marginals in the occupancy distribution by showing a controller obtained by using the occupancy in Fig. 3.6, and the controller when the nodes are selectively pruned as well in Fig. 3.7. These controllers are



Agent i's level 2 controller



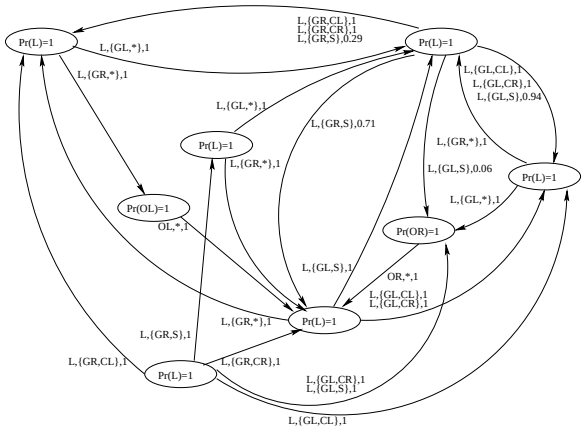
Agent j's level 1 controller



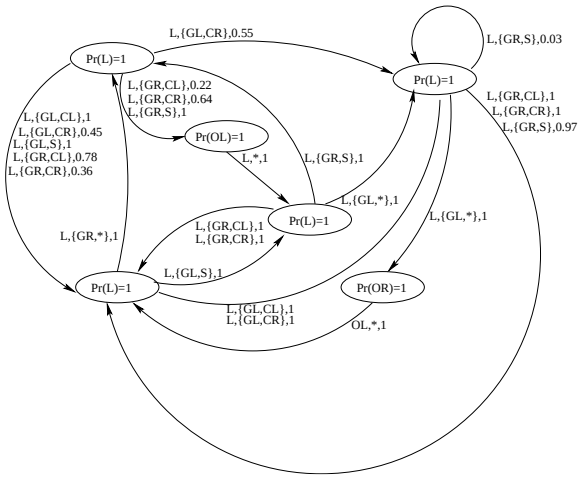
Agent i's level 0 controller

Figure 3.6: Converged controllers at different levels for I-POMDP$_{i,2}$ in the context of the multiagent tiger problem using I-BPI with occupancy computations. An initial belief of agent i is used according to which it is unaware of the tiger's location and knows that j is unaware too.

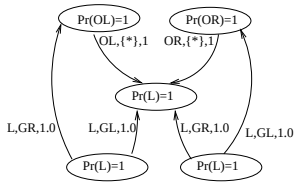
for the level 2 multiagent tiger problem with an initial belief of agent i that the tiger could be behind any one of the doors with equal probability, and i knows that j is unaware of the



Agent i's level 2 controller



Agent j's level 1 controller



Agent i's level 0 controller

Figure 3.7: A converged controller for I-POMDP$_{i,2}$ in the context of the multiagent tiger problem using I-BPI with occupancy distribution and removal of nodes for the same initial belief as in Fig. 3.6. This controller differs from the latter in having one less node at level 1, which differentiates the level 2 controller as well.

tiger's location as well. The two controllers differ at level 1 with the controller in Fig. 3.7 having one node less than the other. Observe that this in turn affects the level 2 controller, which has the same number of nodes as the other, but is structurally different. In particular, agent i performs more listen actions in a sequence according to the revised controller.

3.5 Experiments

We implemented Algorithms 1 and 3 for I-BPI shown in Section 5.4. In order to exploit the sparseness of values in the definitions of some of the problem domains, we used the open-source library IML++ [13] for solving the system of linear equations while using efficient *sparse matrix computations*, and we utilized the dual simplex method as implemented in IBM's CPLEX for solving the linear programs. We comprehensively evaluate various performance parameters of I-BPI in Section 3.5.1 and evaluate I-BPI's performance in the context of multiple frames and agents in Section 3.5.2. [3] Next, we implemented the extensions described in Section 3.4 in I-BPI to account for initial beliefs, and evaluate the impact in Section 3.5.3.

3.5.1 Anytime Property, Scalability to Deeper Levels and Larger Problems

We evaluated I-BPI's properties on two benchmark problem domains: a non-cooperative version of the multiagent tiger problem and a cooperative version of the multiagent machine maintenance (MM) problem, each of which has two agents, i and j. While the multiagent tiger problem is described in chapter 1, the multiagent MM problem generalizes Smallwood and Sondik's [64] MM problem to a two-agent setting in which both agents operate on the

[3] Example problems such as the multiagent tiger problem or user-specified problems may be solved using an implementation of I-BPI in our new online problem-solving environment using POMDP-based frameworks at `http://lhotse.cs.uga.edu/pomdp`.

machine with two components that could fail. Each agent may choose to manufacture the product, examine the product, inspect the machine or repair it before the next production cycle. On examining the product, it may be defective with a high probability if the components have failed. Doshi and Gmytrasiewicz [14] provide the 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 [14] and interactive point-based value iteration (I-PBVI) [15], both of which employ value iteration. In addition to these toy problems, we evaluate I-BPI's performance and demonstrate scalability using two significantly larger problem domains: the money laundering problem [46] and an autonomous unmanned aerial vehicle (AUAV) reconnaissance problem on a grid both of which are non-cooperative.

The money laundering problem, introduced by Ng et al. [46] and possessing realistic underpinnings, 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 possible locations of the red team's assets are: dirty pot, bank accounts, insurance, securities, offshore accounts, shell companies, trusts, corporate loan, casino accounts, real estate, and clean pot. The possible locations of the blue team's sensor are: bank accounts, insurance, securities, shell companies, trusts, corporate loan, casino accounts, real estate and none. 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. As Ng et al. mention [46], this problem is about 20

times larger than previous problem domains solved using I-POMDPs. It 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.

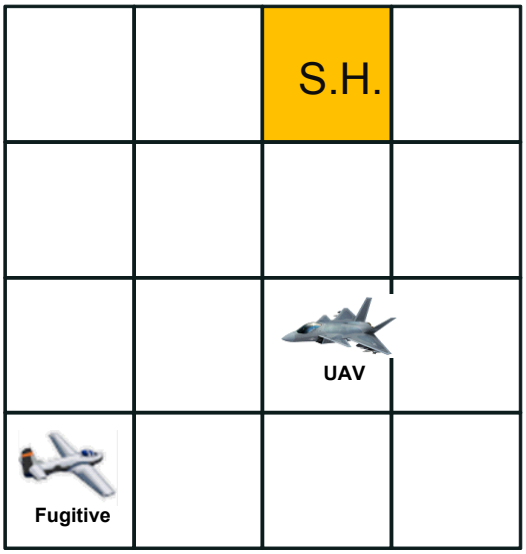
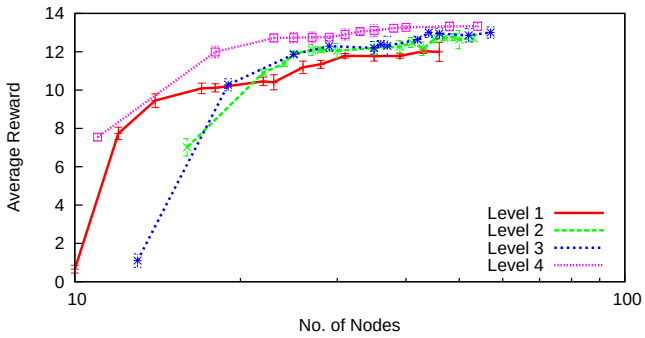
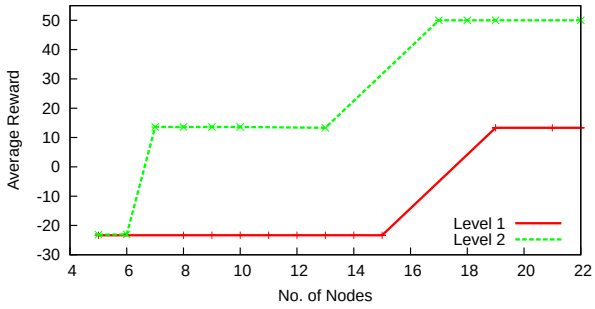


Figure 3.8: AUAV reconnaissance problem on a 4X4 grid. The goal of the fugitive is to reach the *safe house* (marked S.H.). The goal of UAV is to intercept the fugitive before it reaches the safe house.

The AUAV problem involves reconnaissance in a 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 using levels 1 and 2 I-POMDP. Given a fixed location of the safe house, the physical state is the joint location of both agents thereby leading to 256 physical states for a 4×4 grid, which is much larger than any problem solved by previous I-POMDP solution techniques. Each agent may take one of 5 actions of moving in one of the four cardinal directions or listening to get an observation about the target's location. We assumed that the actions taken by both agents on the grid are deterministic. The 4 observations for UAV allow it to sense the cardinal direction of its target relative to its own location. The fugitive has 12 observations which allow it to sense the cardinal directions of its target and the pursuer. We assume that the observations are noisy.



(a)



(b)

Figure 3.9: Average rewards for the (a) multiagent tiger problem on I-POMDP$_{i,l}$ with l ranging from 1 to 4, and (b) AUAV reconnaissance on a 3×3 grid with I-POMDP$_{i,l}$ ranging from $l = 1$ to 2. The rewards generally improve and stabilize as we allocate more nodes to controllers to facilitate escaping local optima, in I-BPI. As we may expect, higher-level controllers generated for more strategic agents eventually lead to better rewards. Vertical bars indicate the standard deviation over trials, and is small.

Our experimentation in the non-cooperative context seeks to answer the following two questions: (a) How are the obtained controllers performing against the best possible adversary? (b) Is there a benefit of computing controllers that are more strategic, which in I-POMDPs translates to deeper levels? We begin by focusing on the tiger problem and the AUAV reconnaissance on a 3×3 grid, and note the average rewards obtained from *simulating* converged controllers of different node sizes, and of different levels against the highest level (most strategic) controller for the other agent, in Fig. 3.9. This not only allows a meaningful comparison of the performance of controllers of differing levels but is also indicative of the

pragmatic performance of the computed controllers against sophisticated agents. 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. Furthermore, controllers that are more strategic exhibiting a higher level, l, eventually perform better. Each reward data point is averaged over 5 trials. Each trial in the tiger problem involved 100 initial beliefs randomly generated and 3 representative initial configurations of the AUAV and the fugitive were utilized, and for each belief, 1000 simulation runs were carried out. In each simulation run, agent i's actions are guided by the converged controller of the particular level up to 4 while agent j is always acting based on a level 4 controller for the tiger problem and a level 2 controller for the AUAV problem. The latter are the highest-level converged controllers we obtain and represents the highest-valued behavior of an agent in the multiagent tiger and AUAV problems so far to the best of our knowledge. [4] We note the rewards of agent i accumulated over 20 steps as both agents act and receive observations due to which their respective controllers transition.

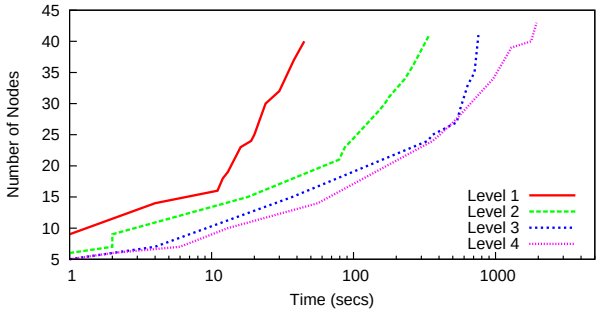


Figure 3.10: Sizes of the level l controllers that are reached in a given amount of time for the multiagent tiger problem. Expectedly, higher-level controllers take more time as the controllers at all lower levels are improved as well.

How much time is needed in obtaining the controllers of increasing sizes that demonstrate

[4] Policy iteration [23] may result in near-optimal controllers and provides an upper bound for BPI. An implementation of policy iteration for solving these controllers did not converge even for the smaller multiagent tiger problem due to the large number of nodes added at each iteration. In particular, it failed to extend beyond two steps of improvement.

the improving performance? In Fig. 3.10, we show the sizes of the level l controller reached in a given amount of time in the context of the tiger problem. Initially, time is spent in improving controllers at all levels up to l by escaping local optima. Subsequently, the lower-level controllers stop improving and time is spent on the top-level controller only with corresponding quicker increases in size.

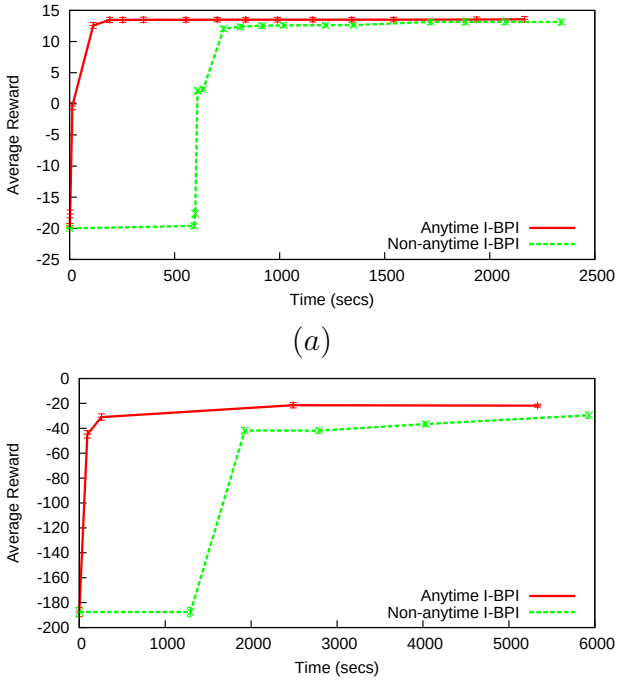


Figure 3.11: Interleaved improvement of the controllers across the different levels in I-BPI demonstrates anytime behavior for (a) I-POMDP$_{i,4}$ in the multiagent tiger context, and (b) I-POMDP$_{i,2}$ in the larger money laundering problem. It takes more than 500 seconds in the multiagent tiger problem before i's level 4 controller begins to improve when the approach is not interleaved (labeled as 'Non-interleaved'). Thereafter, it quickly improves as we may expect given the good quality predictions for the other agent. Analogously, more than 1,000 seconds elapse before the level 2 controller improves in the money laundering context.

In Fig. 3.11 we demonstrate the flexible *anytime* performance of I-BPI in the multiagent tiger and money laundering contexts and compare with the alternative technique of improving the lower-level controllers until convergence before moving to the higher-level one. Notice that the latter approach is distinctly non-anytime taking more than 500 seconds before the

lower-level controllers up to level 3 converge and the upper-level controller for the subject agent starts improving for the tiger problem. Until this time, the initial level 4 controller would be available only in comparison to the improving controller available from the anytime approach. Analogously, more than 1,000 seconds elapse before the level 2 controller for the subject agent starts improving in the money laundering problem. In order to understand the quick improvement that I-BPI brings to the controller, we perform a simple experiment in the context of the multiagent tiger problem. We improved a level 1 controller until convergence with, (a) the lower-level controller improved using a single iteration, and (b) the lower-level controller not improved beyond the seed. We obtained an average reward of 7.143 with a standard deviation of 0.32 across 5 trials for the first approach compared to 4.99 with a standard deviation of 0.17 for the second against a *common* level 1 opponent. This is demonstrative of the positive impact on the level 1 controller to attain significantly better performance when the behavior ascribed to the other agent is improved.

Next, in Table 3.1, we report the average rewards obtained from simulating the controllers that I-BPI generates along with the associated I-BPI run times, as we scale I-POMDP$_{i,l}$ in the context of the number of strategy levels, l. We compare these rewards with those reported using the previous best I-POMDP approximation technique, I-PBVI [15], where the latter are obtained from actual simulation runs as well. We report on policies obtained from I-PBVI using 100,000 belief points in times that were comparable to those of I-BPI or moderately exceeded them. We limit our runs in the simulations to a finite number of 20 steps in order to facilitate comparison between I-BPI whose controllers could be viewed as having a look ahead of an infinite number of time steps and I-PBVI policy whose look ahead is finite. Performance of both approaches is averaged on 5 trials with each trial run for 100 initial beliefs and 10,000 simulation runs per belief. While the original I-PBVI operated on an interactive state space that contained intentional models, we modify the space to include nodes that correspond to the backed up value vectors constituting the other agent's policy. This compacts the

Problem	Level	Method	Time(s)	Avg. Rwd	Property
Multiagent tiger $\|S\|$=2, $\|A_i\|$=$\|A_j\|$=3 $\|\Omega_i\|$=6, $\|\Omega_j\|$=2	1	I-BPI I-PBVI	20 81	10.96 ± 0.09 8.81 ± 0.06	$\|N_{i,1}\| = 20$ $h = 8$ $\alpha_{i,1} = 73$
	2	I-BPI I-PBVI	84 584	12.95 ± 0.10 6.50 ± 0.03	$\|N_{i,2}\| = 29$ $h = 5$ $\alpha_{i,2} = 59$
	3	I-BPI I-PBVI	758 —	13.20 ± 0.09 —	$\|N_{i,3}\| = 41$ —
	4	I-BPI I-PBVI	1,944 —	13.32 ± 0.08 —	$\|N_{i,4}\| = 43$ —
Machine Maintenance $\|S\|$=3, $\|A_i\|$=$\|A_j\|$=4 $\|\Omega_i\|$=$\|\Omega_j\|$=2	1	I-BPI I-PBVI	15 60	20.22 ± 0.13 18.71 ± 0.01	$\|N_{i,1}\| = 20$ $h = 20$ $\alpha_{i,1} = 5$
	2	I-BPI I-PBVI	39 97	20.55 ± 0.06 20.21 ± 0.02	$\|N_{i,2}\| = 20$ $h = 20$ $\alpha_{i,2} = 4$
	3	I-BPI I-PBVI	117 —	21.28 ± 0.05 —	$\|N_{i,3}\| = 20$ —
	4	I-BPI I-PBVI	157 —	21.36 ± 0.09 —	$\|N_{i,4}\| = 20$ —

3.1(a)

interactive state space analogous to I-BPI resulting in an improved performance compared to the original I-PBVI. Notice that I-PBVI is able to reasonably scale up to two levels only within the time limit and the corresponding rewards are sometimes significantly lower than those obtained by I-BPI. In the **Property** column of Table 3.1, $|N_{i,l}|$ denotes the number of nodes in the converged controller, h denotes the horizon of the policy obtained from I-PBVI, and $|\alpha_{i,l}|$ denotes the number of alpha vectors in this policy. These parameters are indicative of the sophistication of the respective policies.

As we see from Table 3.1(a), I-BPI allows scaling solutions of I-POMDPs of up to four

Problem	Level	Method	Time(s)	Avg. Rwd	Property
Money Laundering $\|S\|$=99, $\|A_i\|$=9, $\|A_j\|$=4	1	I-BPI I-PBVI	3,275 4,462	12.09 ± 0.97 0.51 ± 0.13	$\|N_{i,1}\| = 23$ $h = 5$ $\alpha_{i,1} = 29$
$\|\Omega_i\|$=11, $\|\Omega_j\|$=4	2	I-BPI I-PBVI	4,463 —	14.21 ± 1.54 —	$\|N_{i,2}\| = 42$ —
AUAV Reconnaissance $\|S\|$=81 (3 × 3), $\|A_i\|$=$\|A_j\|$=5	1	I-BPI I-PBVI	806 2,215	13.33 13.33	$\|N_{i,1}\| = 19$ $h = 4$ $\alpha_{i,1} = 63$
$\|\Omega_i\|$=4, $\|\Omega_j\|$=12	2	I-BPI I-PBVI	1,548 —	50 —	$\|N_{i,2}\| = 17$ —
AUAV Reconnaissance $\|S\|$=256 (4 × 4), $\|A_i\|$=$\|A_j\|$=5 $\|\Omega_i\|$=4, $\|\Omega_j\|$=12	1	I-BPI I-PBVI	5,695 —	-15.25 —	$\|N_{i,1}\| = 11$ —
	2	I-BPI I-PBVI	9,308 —	1.27 —	$\|N_{i,2}\| = 15$ —

3.1(b)

Table 3.1: Average rewards of the controllers obtained by solving I-POMDP$_{i,l}$ for various levels, for (a) benchmark toy problems, and (b) large domains. Rewards were obtained from executions of the controllers in simulated problem domains against the highest-level controller of the other agent. Run time for I-BPI and I-PBVI was cutoff at one hour in (a) and at two hours in (b) with all methods allowed to complete any iteration that was started before the cutoff. '—' indicates that the corresponding values were not available by the cutoff time. Standard deviation from the mean is shown and is 0 where not indicated. Student's unpaired t-test indicates that the difference in I-BPI's average rewards between levels for each problem is significant at $p \leq 0.05$ level. These results were generated on a RHEL 5 system with Xeon Core2 duo, 2.8GHz each and 4 GB of RAM.

levels deep in time duration that is within one hour. Note that previous approaches have not scaled solutions beyond two levels. It also scales to larger problem domains. This improvement is primarily due to, (i) use of BPI that improves the controller directly without explicitly backing up the vectors; and (ii) representing the model space compactly using a finite number of nodes. We additionally report the number of nodes at level l in the nested controller and note that smaller-sized controllers are preferable as they consume less

memory. Comparison with I-PBVI reveals that the quality of the controllers is significantly better in many of the problems. While we solved I-PBVI for as long a horizon as possible in the imposed time limit, Table 3.1 shows that the maximum horizon does not exceed 20 for the small problems.

A reason for the difference in reward could be due to I-BPI ascribing controllers to the other agent that are compact and of higher quality. We investigate this by cross-utilizing the same controller for agent j in I-PBVI as the one used in I-BPI in the context of the multiagent tiger problem at level 1. In 96 seconds, we obtained a policy from I-PBVI containing 68 vectors and whose average reward is 9.26 ± 0.06. At level 2, a policy with average reward of 7.88 ± 0.1 is obtained in 602 seconds. Therefore, a better quality policy obtained from I-PBVI when a lower-level controller from I-BPI is used indicates that the difference in performance between the two approaches is, in part, due to the difference in how the other agent is modeled. A second reason for the difference in reward is due to the comparatively smaller look ahead of the policies from I-PBVI when utilized in the simulations. On the other hand, I-BPI's converged controllers utilize an infinite-horizon look ahead though the controllers may not be optimal.

Next, we demonstrate the performance and scalability of the algorithm on the two large and real-world inspired problems, money laundering and AUAV reconnaissance on a grid, in Table 3.1(b). Although the former has been solved previously by Ng et al. using the interactive particle filtering [46], 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 3.1 shows the run times for generating converged controllers of good quality for the larger problem domains. Rewards obtained across 100 simulation runs each for 500 different beliefs were averaged and are reported. In the money laundering

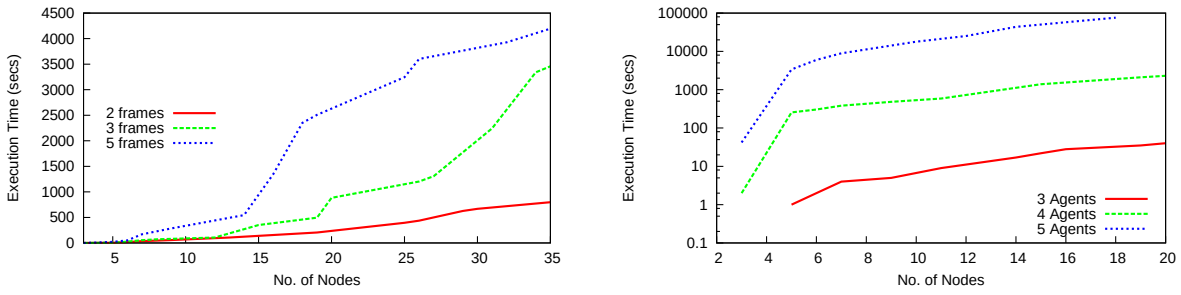
problem, we utilized random initial beliefs for the simulations while the AUAV started out in the neighborhood of the fugitive for the reconnaissance problems. While the AUAV reconnaissance on a 4×4 grid consumes slightly more than two hours at level 2, the money laundering problem takes about an hour. The sizes of the stable controllers for the different problem domains are also shown in Table 3.1. We may improve on the controllers reported in Table 3.1 by allocating more time to the algorithm in order to execute more iterations of evaluation and improvement of the stochastic controllers and escape from local optima. We point out that the reported average reward for the money laundering problem is competitive in comparison to those reported by Ng et al. [46] for particular initial beliefs and parameter configurations. Specifically, Ng et al. reported a reward of about -450 for the blue team at level 1 compared to 12.09 obtained by I-BPI for identical beliefs and configurations. They did not report rewards for level 2 agents.

An 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. 3.2 fixate – after the lower-level controller, $\pi_{j,l-1}$, converges. This is to be expected because the lower-level controller is a part of the evaluation and improvement steps for the level l controller.

3.5.2 Scalability to Multiple Frames and $K > 2$ Agents

We evaluate the performance of I-BPI when agent i is uncertain about j's frame and thinks that j might have a different model of the problem, thereby attributing *multiple frames* to the other agent. We also evaluate I-BPI when the subject agent shares its environment with *more than one* other agent. Because each new frame may contain different transition, observation and reward functions, a new controller is attributed to the other agent per frame. For $|\hat{\Theta}_j|$ frames, as many controllers are utilized and if each controller has say, $|N_{j,l-1}|$ nodes, this adds $|\hat{\Theta}_j||N_{j,l-1}|$ nodes to the interactive state space of i causing a linear increase. However,

this occurs for each controller at every level making the net increase exponential in the levels. Subsequently, I-BPI evaluates and improves each controller attributed to the other agent as well. We show the time performance of level 2 I-BPI as up to 5 frames are ascribed to the other agent in the multiagent tiger problem, in Fig. 3.12(a). The differing rates of increases in time with more nodes is due to the level 2 controller taking more time to converge in some cases because, in part, the lower-level controllers do not converge quickly compared to other cases. In simulations where the other agent's frame was unknown but from within the candidate set, we observed an increase in average reward over 10,000 runs from 9.38 to 9.7 for controllers obtained by ascribing more frames. In the context of the larger problem of AUAV reconnaissance on a 3×3 grid when the strategy level is 1, I-BPI consumed 2,285s in generating a stable controller when 2 frames were ascribed to the fugitive, and 4,174s when 3 frames were ascribed to the fugitive. Both the controllers averaged a reward of 48.33 in simulations with a fugitive at level 2.



(a) (b)

Figure 3.12: (a) Execution time of using I-BPI to solve I-POMDP$_{i,2}$ in the context of the multiagent tiger problem as multiple frames are attributed to the other agent. The number of nodes on the x-axis are those of the upper-level controller and the time is until convergence of the nested controller for that many nodes. (b) Results from using I-BPI for solving I-POMDP$_{i,1}$ in the multiagent tiger context as the setting is shared with multiple other agents.

The presence of more than one other agent in the environment leads to an exponential increase in the size of the interactive state space. This is because we form a Cartesian product of the nodes in the controllers attributed to the other agents leading to $|N_{j,l-1}|^{|\text{Agents}|}$ many

nodes in the interactive state space. We report the execution time performance of I-BPI for solving a level 1 I-POMDP with multiple agents, in Fig. 3.12(b). The initial steep increases are due to the lower-level controller also improving until it converges followed by an escape. Thereafter, the increase in time is due to adding nodes to agent i's level 1 controller from escapes and improving until local convergence. Notice that I-BPI takes about 17 minutes in generating a 15-node controller for agent i with 2 other agents (total of 3). The presence of increasingly more agents in the setting reduces the average reward of the subject agent from 9.7 (3 agents) to 5.21 (5 agents) because of increasing likelihood of some agent opening a door at any time step due to which the tiger resets. This makes the subject agent listen for observations again.

In summary, I-BPI improves on the previous best approximation technique for finitely-nested I-POMDPs by solving problems for deeper levels of strategic nesting, and problems with much larger dimensions. Simultaneously, the generated solutions are of improved value, although the presence of multiple local optima in its updates may make it difficult for I-BPI to get arbitrarily close to global optimality. While I-BPI may be utilized for agents that ascribe a small number of frames to others and to settings with up to 5 agents, further optimizations specific to these extended settings are needed for more frames and agents. Notwithstanding this, these results reflect scalability that significantly improves on previous I-POMDP techniques.

3.5.3 Improved Controllers given Initial Beliefs

I label I-BPI with occupancy computations as I-BPI+Occ, and additionally with node pruning as I-BPI+Occ+RM (RM denotes removing models). In Table 3.2, I compare the converged controllers obtained from I-BPI, I-BPI+Occ and I-BPI+Occ+RM by simulating them in the

multiagent tiger problem and in the larger problem domains of money laundering and AUAV reconnaissance on a 4×4 theater. The comparison is based on multiple metrics: simulation rewards, execution times and controller sizes. For the money laundering problem, we utilized the initial belief as used by Ng et al. [46], in which the blue team believes that the money is in the dirty pot and that there are no sensors. For the AUAV problem, we averaged over 5 realistic initial beliefs. In all of these, the AUAV is aware of its own location but is uncertain about the fugitive's location. Furthermore, it believes that the fugitive believes that the AUAV is at a Manhattan distance of between 1 to 3 places from itself. We performed 1,000 simulations of the controller for a corresponding initial belief, and the other agent was guided by the best possible controller at the highest level we could generate.

Table 3.2 demonstrates that utilizing the initial belief in I-BPI using occupancy computations for the larger problems leads to a controller that significantly improves on the one obtained from the original approach. This is because I-BPI's improvements are not guided by the initial beliefs and it may stabilize on a local optima that is worse compared to I-BPI+Occ despite escapes. While this improvement is not large for the smaller tiger problem, it is obtained from a smaller-sized corresponding controller compared to I-BPI. Note that all simulations in a domain used the same initial beliefs. Furthermore, as we mentioned previously, pruning nodes causes a drop in the quality of the controller because some of the nodes gain prominence as the controller is improved. However, other than the level 1 AUAV reconnaissance problem where the average reward dropped from 26.72 to 4.2, we did not observe large drops in the value, and importantly, the corresponding controllers continue to outperform those obtained from I-BPI in most cases.

As we may expect, the run times for I-BPI+Occ are larger than those for I-BPI for the larger problems because of the additional occupancy computation in *each* iteration, despite the lesser number of nodes in the converged controllers. In particular, the ML problem having 11 observations for each agent shows a large increase in run time of I-BPI+Occ and

Problem	Lvl	Method	Avg. Time(s)	Avg. Rwd	Property	
					$\|N_{i,l}\|$	$\|N_{j,l-1}\|$
Multiagent tiger	1	I-BPI	20	10.82 ± 0.05	20	10
		I-BPI+Occ	7	11.25 ± 0.11	13	10
		I-BPI+Occ+RM	7	11.25 ± 0.11	13	9
	2	I-BPI	84	12.45 ± 0.11	29	10
		I-BPI+Occ	101	12.54 ± 0.11	23	13
		I-BPI+Occ+RM	67	12.51 ± 0.11	23	10
	3	I-BPI	758	12.31 ± 0.08	41	29
		I-BPI+Occ	563	12.87 ± 0.11	24	24
		I-BPI+Occ+RM	216	12.81 ± 0.10	26	19
	4	I-BPI	1,944	12.46 ± 0.07	43	35
		I-BPI+Occ	1,170	12.91 ± 0.11	39	24
		I-BPI+Occ+RM	432	12.75 ± 0.06	31	14
Money Launder-ing	1	I-BPI	3,275	-6.43 ± 0.13	23	5
		I-BPI+Occ	9,891	-0.99 ± 0.21	21	6
		I-BPI+Occ+RM	4,861	-2.86 ± 0.05	19	5
	2	I-BPI	4,463	1.90 ± 0.18	42	4
		I-BPI+Occ	—	—	—	—
		I-BPI+Occ+RM	—	—	—	—
AUAV Recon. (4 × 4)	1	I-BPI	5,695	-38.37	11	5
		I-BPI+Occ	12,734	26.72	9	5
		I-BPI+Occ+RM	9,644	4.20	10	4
	2	I-BPI	9,308	-22.42	15	5
		I-BPI+Occ	18,166	28.25	9	5
		I-BPI+Occ+RM	14,151	21.78	9	4

Table 3.2: Agent's initial beliefs when available may be utilized to learn controllers that obtain better rewards. These results were generated on a RHEL 5 system with Xeon Core2 duo, 2.8GHz each and 4 GB of RAM. Run time was cutoff at five hours with iterations allowed to complete. The values for the AUAV problem is the average over the values of the 5 controllers obtained, one for each initial belief. '—' indicates that the controller did not stabilize by the cutoff time. Differences in average rewards between I-BPI and I-BPI+Occ are significant (Student's t-test, $p \leq 0.05$) except for the level 2 tiger problem.

I-BPI+Occ+RM that exceeds the cutoff. This is because occupancy computations scale with the number of joint observations (see Eq. 3.4) – a factor that is large for ML. However, these computations are not intensive for the smaller tiger problem and the associated smaller controller sizes yield run time benefits. I-BPI+Occ+RM reduces the run times significantly to more manageable levels for the larger problems. In particular, the run time for the level 1 money laundering problem is halved due to additionally pruning a single node in the agent j's controller. We note that each less node in the other agent's controller causes a reduction of $|S|$ states in the interactive state space. Furthermore, the smaller-sized controller converges more quickly, which led the level 1 controller of agent i to converge quickly. Both these together contributed to the speed up. Finally, nodes sizes of the controllers indicate the impact of selectively pruning nodes and the expressiveness of the policies. In particular, considering the initial belief generates controllers that are smaller and targeted. A node pruned from the other agent's controller implies that the intentional models mapped to that node became implausible due to the initial belief.

In summary, we provide a method for accounting for an initial nested belief of agent i, if available, within I-BPI. It generates controllers of improved value compared to I-BPI at the expense of increased computations. However, improvements are not guaranteed especially if the overall policy generated by I-BPI is already of high quality. Because our aim is to keep the complexity manageable, we introduce a novel and principled way to prune some of the nodes in the other agent's controller using the occupancy distributions. The increased computation times suggest that these extensions should be utilized if the controller from I-BPI is thought to be of poor quality. As such, we do not view I-BPI+Occ and I-BPI+Occ+RM as independent methods for solving I-POMDPs with initial beliefs.

3.6 Related Work

Tractable POMDP solution approaches specifically target the factors that aggravate the computational complexity such as curse of dimensionality which results in exponential growth in the size of the belief space over which the solution must be optimized with the increase in number of states. The other significant factor affecting the solution complexity is the curse of history. In the context of policy iteration, it is the exponential growth in the size of policy space over which the search for optimal policy should be performed with each application of the backup operator. Being a generalization of POMDPs, solutions of I-POMDPs are also affected by the curses of dimensionality and history that affect POMDPs. 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, and the number of candidate models grows over time as the agents act and observe. We term this added complexity on modeling the other agent as curse of agent modeling.

Previous approximations for finitely-nested I-POMDPs include interactive particle filtering [14] and interactive point-based value iteration [15]. 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 [50] to multiagent settings, and reduces the effect of the curse of history by optimizing the value function over a finite subset of the belief space instead of the entire space. 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. Our implementation of I-PBVI overcomes this problem by modifying the state space to include nodes that correspond to the backed up value vectors constitut-

ing the other agent's policy. This compacts the interactive state space analogous to I-BPI resulting in an improved performance compared to the original approach.

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 [56, 79]. This approach, analogously to using finite state controllers, allows a compact representation of the model space. However, computing the exact equivalence classes requires solving the models.

As we mentioned previously, BPI [51] has been extended to the context of decentralized POMDPs as well [5], which is a framework for joint decision making in cooperative settings. A stochastic finite state controller is initialized for each agent along with a correlation device, which is a shared source of randomness between the agents. Though not necessary, the correlation device provides a mechanism for the agents to coordinate their actions efficiently. Each controller and the correlation device is improved independently using the two steps of evaluation and improvement until convergence. While I-BPI differs in its use of controllers embedded at different levels which are iterated and bring its own set of unique challenges, policy iteration demonstrated significant scalability in the context of decentralized POMDPs as well, allowing problem domains of larger sizes to be solved with good quality solutions. A recent approach [1] generates optimal controllers of a fixed size in the context of both POMDPs and decentralized POMDPs using a quadratically-constrained linear program. Although the complex formulation makes the problem non-convex thereby admitting many additional local optima, empirical results verify that it generates controllers of improved value compared to BPI for the same number of nodes. However, solving a non-linear program is computationally much more intensive and often exhibits reduced stability, making it suitable for controllers of small sizes only. Furthermore, embedding controllers of the other agent in the state space in the context of I-POMDPs makes this formulation additionally computationally complex. We improve the value of the controllers by utilizing

escape techniques and exploiting initial beliefs.

Somewhat analogous to our exploit of sparseness of non-zero values in the transition and observation functions of the problem domains, Hansen [24] noted the sparseness of non-zero parameters obtained from the linear program of BPI in the context of POMDPs. While the linear program computes $|A_i| + |A_i||\Omega_i||\mathcal{N}_i|$ parameters, just a few of these typically have non-zero values for multiple problem domains. Using a series of reduced linear programs, Hansen showed that we could get the same result as solving the full linear program, but more efficiently by iteratively focusing on just the non-zero parameters.

A recent incremental policy iteration approach [21] yields small controllers with significantly improved values for POMDPs using a series of heuristic based escape techniques and a quadratic optimization program that guarantees escape. Its generalization to multiagent scenarios is yet to be explored.

3.7 Discussion

Recently emerging applications for I-POMDPs in defense and robotics necessitate the need for approaches that allow its solutions to scale. We introduced a generalized policy iteration algorithm for multiagent settings in the context of finitely-nested 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 other agents get naturally mapped to nodes in the respective controllers. The application of generalized BPI to these controllers ensures that the size of the model space doesn't increase rapidly thereby subduing the effect of the curse of agent modeling, which excessively impacts I-POMDPs. As a result, we provide solutions of I-POMDPs that are nested to levels as deep as four for the first time. Ultimately, this allows the application of I-POMDPs to scale to more realistic domains with reduced trade off in value of the solution compared to previous

approximations. We demonstrate this by applying the technique to substantially larger problem domains, which are inspired by real-world applications.

Section 3.2 discussed transforming the original interactive state space consisting of the physical states and models of the other agent into a more compact space given a finite-state controller for the other agent. If the initial controllers are improved using policy iteration [23] at all levels, we may obtain an ϵ-optimal solution of the I-POMDP$_{i,l}$ after a finite number of iterations. However, our use of the generalized BPI makes the approach an *approximation* technique because BPI is not guaranteed to reach (ϵ-)optimality often converging to local optima despite the presence of an escape technique.

Higher strategy levels usually correlate with more sophisticated behavior leading to controllers whose size increases with strategy levels, as we demonstrated in Fig. 3.6. However, the presence of local optima additionally impacts the controller sizes. Interactive BPI scales better for domains in which high quality behavior at lower levels may be represented using controllers having just a few nodes. This reduces the size of the interactive state space for higher levels and the impact of the curse of dimensionality. In this regard, uncovering properties of domains that allow smaller-sized controllers to represent high-quality behavior is an important direction of investigation.

A *limitation* of interactive BPI is its predilection of convergence to local optima leading to controllers whose quality could be unpredictable. While techniques for escaping from local optima help (Section 3.5.1), 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 pointwise uniformly greater than a previous vector leads to multiple local optima; relaxing the constraint of uniform improvement may help as we demonstrated in our use of initial beliefs.

An important limitation and line of future work is to improve the performance of this approach on problem domains having more than two agents. As we mentioned previously, a

separate controller is embedded in the state space for each other agent, and iteratively improved. Although the state space grows exponentially in the number of nodes in a controller with more agents, its size is much reduced in comparison to using models because many models map to a single node of the controller. Therefore, I-BPI obtains converged controllers for up to $K = 5$ agents (Section 3.5.2), although it remains to be investigated whether further optimizations that specifically target simplifying structure in problems involving many more agents are possible. Furthermore, it is easy to see that I-PBVI would suffer much more given multiple other agents. While the state space would also grow exponentially with the number of agents as in I-BPI, backups in I-PBVI often produce many more alpha vectors compared to the nodes in a bounded finite state controller. While the number of vectors are limited by the belief points considered, these are usually far more than the bound on the number of nodes, and grow over time (see Table 3.1). Consequently, the interactive state space is larger in the context of I-PBVI and grows over time.

Additionally, the finite state controllers that we use are a type of automata called Moore machines. Recently, Mealy machines [30] were utilized in the context of decentralized POMDPs to good effect [2]. Therefore, another avenue is to investigate the utility of different types of controllers including Mealy machines in the context of I-POMDPs. Finally, the non-linear program formulation for POMDPs [1] is shown to generate smaller-sized controllers with values that improve on BPI despite also being prone to local optima. As such, pursuing a generalized non-linear formulation for I-POMDPs is a viable direction for future work.

Chapter 4

Bimodal Switching for Online Planning in Multiagent Settings

Execution time planning (often referred to as *online planning*) involves computing the optimal policy for the subject agent in a limited amount of time given its initial belief (the prior) that evolves over time as the agent acts and observes. As a result of the additional constraint of limited time, online planning techniques emphasize on reduced planning time by trading off optimality and relying on approximations. Traditionally, the focus of online planning research has been on the single-agent settings [57], albeit isolated approaches for cooperative problems do exist [77].

I attend to online planning in the context of finitely-nested interactive partially observable Markov decision processes (I-POMDPs) [20, 17]. More specifically, I focus on a subset of I-POMDP problems in which the actions of the other agents are not directly observable to the subject agent but can be inferred from their effect on the state of the environment. In I-POMDP literature, interactive particle filtering [14] offer a way to plan at execution time given an initial belief. However, this approach is time consuming and does not scale to settings with more than tens of physical states or to long horizons.

The chief reason of poor scalability is that the subject I-POMDP agent must deliberate over the behavior of the other agents present in the environment by simultaneously updating its belief over both the physical states and the models of the other agents conditioned on the states based on its own actions and observations (*curse of dimensionality* and *curse of agent modeling*). The belief over the models of the other agents is updated based on their predicted actions and their observed effects on the physical states. These observations become more informative when the agent's belief over the physical state has reduced uncertainty (entropy). For example, in the context of a variant of the the well-known multiagent tiger problem [20] (example 1) in which the observation of creaks is not available to the subject agent, let the agent strongly believe that the tiger is on the left but on listening hear a growl from the right. If the observation is reliable with a high probability, the agent infers that the other agent likely opened the door causing the tiger to change its location.

We present a novel two stage online planning approach. In the first phase, the agent behaves as if it were alone in the environment by marginalizing the effect of the other agents as a fixed noise. During this phase, the goal of the agent is to reduce the uncertainty in its belief over the physical states. In this mode, the agent is modeled as a POMDP and planning in this phase could be performed efficiently fast POMDP planning techniques such as SARSOP [40], that takes orders of magnitude less time to execute as compared to the I-POMDP solver. As the agent operates in the POMDP phase and its uncertainty in its belief over physical states reduces, it switches to the I-POMDP mode combining its updated belief over the physical states and the initial belief over the models. From this point onwards, the agent performs online planning using interactive particle filtering technique.

The question then becomes, "When should the agent switch from the POMDP to the I-POMDP mode?" To facilitate the switching, we compute the upper and lower bound on the optimal value of the subject agent's current belief at each step. Switching takes place when the ratio between the difference between the current upper and lower bound values

to the difference between the maximum upper bound value and the minimum lower bound value becomes less than a parameter, ϵ. Because of the convexity property of the lower-bound value function (the POMDP value function is PWLC), the difference between the two bounds typically reduces as beliefs become less uncertain (near the vertices of the PWLC curve). Nevertheless, the bounds may not converge, and switching may not occur for very small values of ϵ.

The bimodal approach yields computational savings because during the initial steps of online planning (POMDP phase), the agent utilizes a fast and scalable single-agent solution algorithm. However, this computational saving may come at a cost. We estimate the error entailed by this approach. We illustrate the results of this approach on a persistent variant of the multiagent tiger problem where the agent can't directly observe the actions of the other agent through *creaks* and show that the total time elapsed over several steps of bimodal approach is significantly less compared to using an I-POMDP approach right from the start at the expense of reduced cumulative reward.

4.1 Bimodal Online Planning

Let $b^0_{i,l}$ be agent i's initial belief over the interactive state space (at time-step 0). Agent i initially views the problem as a single-agent POMDP and its belief is the portion of $b^0_{i,l}$ that represents the distribution over the physical states only, $b^0_{i,l}(s)$. In the first phase, the agent only updates the distribution over $b^0_{i,l}(s)$. Conditional beliefs over the models given the state, $b^0_{i,l}(\cdot|s)$, are held fixed. After some steps, in the second phase, the agent switches to updating the conditionals as well.

A key observation motivating the bimodal approach is that in problem domains where direct (including noisy) observations about the other agent's actions are not available and its actions are inferred by sensing the next state, perhaps noisily, knowing the current state

provides greater information about the true model of the other agent. We prove this claim next.

Definition 2 (Unobservable actions)**.** *Actions of agent j are directly unobservable to agent i iff observations of i are conditionally independent of j's action, $O_i(s, a_i, a_j, \omega_i) = O_i(s, a_i, \omega_i)$; $\forall\ a_j \in A_j,\ \omega_i \in \Omega_i,\ a_i \in A_i, s \in S$.*

Let $Pr(\dot{\theta}^{t+1}_{j,l-1}|\omega^{t+1}_i, a^t_i, b^t_{i,l})$ be i's probability of the true model of the other agent, $\dot{\theta}^{t+1}_{j,l-1}$, on observing, ω^{t+1}_i, and performing action, a^t_i, given belief, $b^t_{i,l}$. We may write it as:

$$Pr(\dot{\theta}^{t+1}_{j,l-1}|\omega^{t+1}_i, a^t_i, b^t_{i,l}) = \sum_{s^{t+1}} Pr(s^{t+1}, \dot{\theta}^{t+1}_{j,l-1}|\omega^{t+1}_i, a^t_i, b^t_{i,l})$$

Notice that the term on the right is agent i's updated belief over a state and true model of j obtained using the I-POMDP belief update. Under Def. 2 applied to both agents,

$$\begin{aligned} Pr(\dot{\theta}^{t+1}_{j,l-1}|\omega^{t+1}_i, a^t_i, b^t_{i,l}) = \sum_{s^{t+1}} \sum_{s^t} b^t_{i,l}(s^t) \sum_{\theta^t_{j,l-1}} b^t_{i,l}(\theta^t_{j,l-1}|s^t) \sum_{a^t_j} Pr(a^t_j|\theta^t_{j,l-1})\ T_i(s^t, a^t_i, a^t_j, s^{t+1}) \\ O_i(s^{t+1}, a^t_i, \omega^{t+1}_i) \sum_{\omega^{t+1}_j} O_j(s^{t+1}, a^t_j, \omega^{t+1}_j)\ \tau(\theta^t_{j,l-1}, a^t_j, \omega^{t+1}_j, \dot{\theta}^{t+1}_{j,l-1}) \end{aligned} \tag{4.1}$$

where τ is an indicator function that is 1 when model, $\theta^t_{j,l-1}$, updates to $\dot{\theta}^{t+1}_{j,l-1}$ on performing action, a^t_j, and receiving observation, ω^{t+1}_j; otherwise 0. Equation 4.1 shows that i's belief over j's true model at $t+1$ is influenced by j's predicted actions from its models, the observations that j may likely receive and agent i's transition and observation functions.

Let agents i and j perform actions, a^t_i and a^t_j respectively, due to which the state transitions from s^t to s^{t+1}. As j's actions are unobservable, state transitions allow valuable inference of j's actions.

Definition 3 (Maximally informative transition)**.** *The above state transition is maximally informative about j's action iff $T_i(s^t, a_i^t, a_j^t, s^{t+1}) \geq T_i(\bar{s}^t, a_i^t, a_j^t, s^{t+1})$ for any other state, $\bar{s}^t$, and $T(s^t, a_i^t, a_j^t, s^{t+1}) > T(s^t, a_i^t, \bar{a}_j^t, s^{t+1})$ for all other actions, $\bar{a}_j^t$, of j.*

Consider domains where the other agent's actions are unobservable (Def. 2). In such settings, observations are more informative about the other agent's models if the uncertainty over the physical state is mitigated. In order to show this, let the transition that occurs from the current state, $\dot{s}^t$, due to joint actions be maximally informative (Def. 3). Let j's performed action solely lead to its true model. Agent i then receives the observation, ω_i^{t+1}, that is most likely. Then, the following proposition holds:

Proposition 3. *If $b_{i,l}^{t,1}$ is a belief which assigns probability 1 on the current state, $\dot{s}^t$, then, $Pr(\dot{\theta}_{j,l-1}^{t+1}|\omega_i^{t+1}, a_i^t, b_{i,l}^{t,1}) \geq Pr(\dot{\theta}_{j,l-1}^{t+1}|\omega_i^{t+1}, a_i^t, b_{i,l}^{t,2})$, for any other belief, $b_{i,l}^{t,2}$, under the assumption that the conditional distributions over the models of j in both beliefs are identical and do not change with state.*

Proof. For convenience, we may rewrite Eq. 4.1 as,

$$Pr(\dot{\theta}_{j,l-1}^{t+1}|\omega_i^{t+1}, a_i^t, b_{i,l}^t) = \sum_{s^t} b_{i,l}^t(s^t)\ \mathcal{X}(s^t, a_i^t, \omega_i^{t+1}, \dot{\theta}_{j,l-1}^{t+1})$$

where $\mathcal{X}(s^t, a_i^t, \omega_i^{t+1}, \dot{\theta}_{j,l-1}^{t+1})$ denotes the remaining terms of Eq. 4.1 as a function of s^t, a_i^t, ω_i^{t+1}, and $\dot{\theta}_{j,l-1}^{t+1}$. Under the assumption that i's conditional distribution over the models is identical for any state, if the current state is $\dot{s}^t$, then $\mathcal{X}(\dot{s}^t, a_i^t, \omega_i^{t+1}, \dot{\theta}_{j,l-1}^{t+1})$ is greater than $\mathcal{X}(\bar{s}^t, a_i^t, \omega_i^{t+1}, \dot{\theta}_{j,l-1}^{t+1})$ for any other state, $\bar{s}^t$. This is because, $\sum_{s^{t+1}} T_i(\bar{s}^t, a_i^t, a_j^t, s^{t+1})\ O_i(s^{t+1}, a_i^t, \omega_i^{t+1}) \leq \sum_{s^{t+1}} T_i(\dot{s}^t, a_i^t, a_j^t, s^{t+1})\ O_i(s^{t+1}, a_i^t, \omega_i^{t+1})$, as any transition to a possible next state, s^{t+1}, is maximally informative about j's action given i's action, from the current state, $\dot{s}^t$. While some other action of j could result in a next state, s^{t+1}, from some $\bar{s}^t$, it does not lead to the true model of j as per τ. As $b_{i,l}^{t,1}$ puts a probability 1 on $\dot{s}^t$, it follows

that $\mathcal{X}(\dot{s}^t, a_i^t, \omega_i^{t+1}, \dot{\theta}_{j,l-1}^{t+1})$ is greater than $\sum_{s^t} b_{i,l}^{t,2}(s^t)\ \mathcal{X}(s^t, a_i^t, \omega_i^{t+1}, \dot{\theta}_{j,l-1}^{t+1})$, for any other belief, $b_{i,l}^{t,2}$, which differs from $b_{i,l}^{t,1}$ in its distribution over the physical states only. □

While we sought to reduce some of the possibilities due to uncertainty, Proposition 3 formalizes the intuition that in domains where behavioral information about the other agent must be inferred by sensing the dynamic state, received observations (that are not noise) tend to be more informative about the other's model when the uncertainty over the current physical state is as less as possible.

If agent i's conditional belief over models is decoupled from its belief over the physical state, the proposition becomes useful as the beliefs, $b_{i,l}^{t,2}$ and $b_{i,l}^{t,1}$, could be those that are in the sequence of beliefs that i may have as it acts and observes. Consequently, in problems where j's actions are unobservable, it motivates that we update the distributions over the models as late as possible in the game. Of course, the trade off is that the conditional distributions over models are held fixed for a longer time affecting early predictions.

4.1.1 Lower Bound: POMDP model with noise

Our lower bound is a POMDP for agent i, POMDP$_i$, that we adapt from the its level l I-POMDP, I-POMDP$_{i,l}$. For POMDP$_i$, the state space is the set of physical states, S, in I-POMDP$_{i,l}$. The set of actions and observations for POMDP$_i$ is the same as the set of actions and observation for agent i in I-POMDP$_{i,l}$. Since we focus on the subset of I-POMDP problems where the observation function is independent of the actions of the other agent, i.e. other agent's actions are unobservable, the observations function of POMDP$_i$ is the same as the observation function of I-POMDP$_{i,l}$. The transition and reward function in POMDP$_i$ are the marginals of their counterparts in I-POMDP$_{i,l}$. They are obtained by summing out agent j's actions in the respective functions using a marginalization factor that we compute next. The optimality criterion remains a sum of discounted rewards over the finite horizon.

In order to compute the marginalization factor for actions of agent j used to sum them out in the transition and reward functions of I-POMDP$_{i,l}$, we first solve the I-POMDP of agent j by performing bounded policy iteration (BPI) [51] on the level 0 I-POMDP and using *interactive* BPI [67] (chapter 3) otherwise. The solution thus obtained is in the form of a finite state controller (FSC). Each node in the FSC corresponds to a distribution over j actions. Also, each node in the FSC is associated with a value vector that gives the expected reward of performing the action(s) corresponding to the node from each state, and then following the remaining controller until the values converge. Any model of agent j could be mapped to the node that maximizes the value of the inner product between the corresponding belief $b_{j,l-1}$ with the associated value vector, i.e. nodes in $\mathcal{N}_{j,l-1}$ partition the continuous model space. Doing so allows us to map the model space of j to the set of nodes in the FSC for agent j, $\mathcal{N}_{j,l-1}$. Note that multiple models may be mapped to a single node. A benefit of this approach of mapping models to nodes is that the uncountably infinite model space is reduced to a finite set of nodes, $\mathcal{N}_{j,l-1}$. We may obtain the distribution over j's actions for summing out as:

$$Pr(a_j|s) = \sum_{n_{j,l-1} \in \mathcal{N}_{j,l-1}} b^0_{i,l}(n_{j,l-1}|s)\ Pr(a_j|n_{j,l-1}) \tag{4.2}$$

where $Pr(a_j|n_{j,l-1})$ is the probability assigned to action, a_j, by the node, $n_{j,l-1}$, and $b^0_{i,l}(n_{j,l-1}|s)$ is the conditional probability mass in i's initial belief over models that get transferred to $n_{j,l-1}$ due to the mapping: $b^0_{i,l}(n_{j,l-1}|s) = \sum_{\theta_{j,l-1} \mapsto n_{j,l-1}} b^0_{i,l}(\theta_{j,l-1}|s)$.

Equation 4.2 provides the distribution over the other agent's actions used for formulating the marginalization factors. This distribution is static and the resulting POMDP$_i$ models the other agent as noise in the environment. Solution of this POMDP is obtained by using SARSOP [40], which is a fast and scalable POMDP solution technique that produces a *policy graph*.

Next, we show that the expected reward from our formulation of POMDP_i is a lower bound to the expected reward from $\text{I-POMDP}_{i,l}$ in which the model space, $\Theta_{j,l-1}$ has been substituted with the FSC for j, $\mathcal{F}_{j,l-1}$. Here, $f_{j,l-1} \in \mathcal{F}_{j,l-1}$ is, $f_{j,l-1} = \langle n_{j,l-1}, \hat{f}_{j,l-1}, \hat{\theta}_j \rangle$, where $n_{j,l-1}$ is a node in the set of nodes in j's level $l-1$ controller, $n_{j,l-1} \in \mathcal{N}_{j,l-1}$; $\hat{f}_{j,l-1}$ includes the set of edge labels, distributions of actions for each node and the edge transition function of the controller; and $\hat{\theta}_j$ is j's fixed frame. In other words, treating the other agent as noise is less valuable than correctly modeling it.

In order to compare between the two value functions, we obtain the value of i's marginal belief over the physical state in the $\text{I-POMDP}_{i,l}$ as: $V(b_{i,l}(s)) = \max_{\hat{\alpha}} \sum_{s \in S} b_{i,l}(s)\hat{\alpha}(s)$. Here,

$$\hat{\alpha}(s) = \sum_{n_{j,l-1}} b^0_{i,l}(n_{j,l-1}|s)\ \alpha(s, n_{j,l-1}) \tag{4.3}$$

where $b^0_{i,l}(n_{j,l-1}|s)$ is the initial conditional belief over nodes as defined previously, and $\alpha(s, n_{j,l-1})$ is an alpha vector that composes the value function for $\text{I-POMDP}_{i,l}$.

Proposition 4 (Lower bound)**.** *Let V denote the value function of $\text{I-POMDP}_{i,l}$. Let $\underline{H}$ and H be the backup operators for POMDP_i and $\text{I-POMDP}_{i,l}$, respectively. Then, it holds that $HV \geq \underline{H}V$.*

Proof. Let $b_{i,l}(s)$ be a belief over the physical states, and $b_{i,l}(s, n_{j,l-1}) = b_{i,l}(s)\ b^0_{i,l}(n_{j,l-1}|s)$, where $b^0_{i,l}(n_{j,l-1}|s)$ is the initial conditional distribution.

We begin by showing that for horizon 1, for any $b_{i,l}(s)$, value of this belief given by POMDP_i is the same as the value of the belief, $b_{i,l}(s, n_{j,l-1})$, shown previously:

$$\underline{V}(\hat{b}_{i,l}) = \max_{a_i \in A_i} \sum_s b_{i,l}(s)\hat{R}_i(s, a_i)$$

where $\hat{R}_i(s, a_i)$ is agent i's reward function in POMDP$_i$. Introducing a_j gives,

$$\begin{aligned}
&\underline{V}(b_{i,l}) = \max_{a_i \in A_i} \sum_s b_{i,l}(s) \sum_{a_j} R_i(s, a_i, a_j) Pr(a_j|s) \\
&= \max_{a_i \in A_i} \sum_s b_{i,l}(s) \sum_{a_j} R_i(s, a_i, a_j) \sum_{n_{j,l-1}} b^0_{i,l}(n_{j,l-1}|s)\ Pr(a_j|n_{j,l-1}) \qquad \text{(from Eq. 4.2)} \\
&= \max_{a_i \in A_i} \sum_{s, n_j} b_{i,l}(s, n_{j,l-1}) \sum_{a_j} R_i(s, a_i, a_j)\ Pr(a_j|n_{j,l-1}) \\
&= V(b_{i,l})
\end{aligned}$$

Next, we move to the case where the horizon is 2, and apply the I-POMDP$_{i,l}$ backup operator to the value function, V:

$$\begin{aligned}
&HV(b_{i,l}) = \max_{a_i \in A_i} \sum_{s, n_{j,l-1}} b_{i,l}(s, n_{j,l-1}) ER_i(s, a_i) + \sum_{\omega_i} Pr(\omega_i|a_i, b_{i,l})\ \max_{\alpha}\ b'_{i,l} \cdot \alpha \\
&= \max_{a_i \in A_i} \sum_s \sum_{n_{j,l-1}} b_{i,l}(s) b^0_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \Big\{ R_i(s, a_i, a_j) + \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i, a_j) \\
&\sum_{\omega_j} O_j(s', a_j, \omega_j) \sum_{n'_{j,l-1}} Pr(n'_{j,l-1}|n_{j,l-1}, a_i, \omega_j)\ \alpha^k(s', n'_{j,l-1}) \Big\}
\end{aligned}$$

where k is the index of the alpha vector that provides the maximal value at the updated belief, $b'_{i,l}$. We may rewrite the above dynamic programming update by noting that $\sum_{\omega_j} O_j(s', a_j, \omega_j) \sum_{n'_{j,l-1}} Pr(n'_{j,l-1}|n_{j,l-1}, a_i, \omega_j)$ represents the updated belief over the models conditioned on the updated state, $b'_{j,l-1}(n'_{j,l-1}|s')$.

$$\begin{aligned}
HV(b_{i,l}) = \max_{a_i \in A_i} \sum_s \sum_{n_j} b_{i,l}(s) b^0_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \Big\{ R_i(s, a_i, a_j) \\
+ \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i, a_j) \sum_{n'_j} b'_{j,l-1}(n'_{j,l-1}|s') \alpha^k(s', n'_{j,l-1}) \Big\}
\end{aligned}$$

$$\begin{aligned}
\geq \max_{a_i \in A_i} \sum_s \sum_{n_{j,l-1}} b_{i,l}(s) b^0_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \Big\{ R_i(s, a_i, a_j) \\
+ \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i, a_j) \sum_{n_{j'}} b^0_{j,l-1}(n'_{j,l-1}|s') \hat{\alpha}^k(s', n'_{j,l-1}) \Big\}
\end{aligned}$$

The above holds because $\alpha^k(s', n'_{j,l-1})$ is maximal at $b'_{i,l} = b'_{i,l}(s')b'_{i,l}(n'_{j,l-1}|s')$. For the belief in the above equation, $b'_{i,l}(s')b^0_{i,l}(n'_{j,l-1}|s')$, it may continue to remain maximal if $b'_{i,l}(n'_{j,l-1}|s')$ and $b^0_{i,l}(n'_{j,l-1}|s')$ are identical, otherwise it's suboptimal. Using Eq. 4.3, above equation may be rewritten.

$$
\begin{aligned}
HV(b_{i,l}) &\geq \max_{a_i \in A_i} \sum_s \sum_{n_j} b_{i,l}(s) b^0_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \Big\{ R_i(s,a_i,a_j) \\
&\quad + \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i, a_j) \hat{\alpha}^k(s') \Big\} \\
&= \max_{a_i \in A_i} \sum_s b_{i,l}(s) \sum_{a_j} \sum_{n_{j,l-1}} b^0_{i,l}(n_{j,l-1}|s) Pr(a_j|n_{j,l-1}) \Big\{ R_i(s,a_i,a_j) \\
&\quad + \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i, a_j) \hat{\alpha}^k(s') \Big\} \\
&= \max_{a_i \in A_i} \sum_s b_{i,l}(s) \sum_{a_j} Pr(a_j|s) \Big\{ R_i(s,a_i,a_j) + \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i, a_j) \hat{\alpha}^k(s') \Big\} \quad \text{(Using Eq. 4.2)} \\
&= \max_{a_i \in A_i} \sum_s b_{i,l}(s) \Big\{ \sum_{a_j} R_i(s,a_i,a_j) Pr(a_j|s) + \sum_{\omega_i} \sum_{s'} \sum_{a_j} Pr(s', \omega_i|s, a_i, a_j) Pr(a_j|s) \hat{\alpha}^k(s') \Big\} \\
&= \max_{a_i \in A_i} \sum_s b_{i,l}(s) \Big\{ \hat{R}_i(s,a_i) + \sum_{\omega_i} \sum_{s'} Pr(s', \omega_i|s, a_i) \hat{\alpha}^k(s') \Big\} \\
&= \underline{H} V(b_{i,l})
\end{aligned}
$$

Here, $\hat{R}$ is the reward function in POMDP$_i$ as defined previously, $Pr(s', \omega_i|s, a_i)$ is the joint observation and transition functions, and $\underline{H}$ is the corresponding backup operator.

As $V = \underline{V}$, we also get, $H\underline{V} \geq \underline{H}\underline{V}$ from the above proof, and furthermore, $H(\underline{H}\underline{V}) \geq \underline{H}(\underline{H}\underline{V})$. Because the I-POMDP$_{i,l}$ backup operator is isotonic, $H(H\underline{V}) \geq H(\underline{H}\underline{V})$. This implies, $H(HV) \geq \underline{H}(\underline{H}\underline{V})$. Thus, repeatedly applying the two backup operators maintains the lower bound. □

This is intuitive and demonstrates the benefit of closely tracking the other agent's dynamic models.

4.1.2 Upper Bound: I-POMDP$_{i,l}$ with perfectly observable state

In the first phase, agent i utilizes the policy resulting from solving POMDP$_i$ using the fast and scalable POMDP solution technique SARSOP to guide its actions. The value of the policies thus obtained for any belief over the physical states is guaranteed to be a lower bound to the value of the I-POMDP obtained from I-POMDP$_{i,l}$ where the models of j in the the interactive states are replaced by $\mathcal{F}_{j,l-1}$. Eventually, the agent switches to using an online solution of I-POMDP$_{i,l}$ for acting. In order to facilitate the switching, we utilize an upper bound on the value of I-POMDP$_{i,l}$ for its belief over the physical states.

If an observation reveals the physical state perfectly to agent i in an I-POMDP$_{i,l}$, the proposition below shows that the resulting value of $b_{i,l}$ is an upper bound to the general I-POMDP value. We redirect you to [68] for the proof. Notice that despite the physical state being perfectly observable, the resulting model does not collapse into an MDP because the model of the other agent continues to remain uncertain. Let $\overline{V}$ be the value function of this model, which we denote as I-POMDP$^S_{i,l}$. The value function is composed of possibly multiple vectors for each state, s. For each state, we obtain the maximal value of the inner product between the initial conditional belief, $b_{i,l}(n_{j,l-1}|s)$, and the updated alpha vectors for that state. These values form a single alpha vector over the physical states.

$$\begin{aligned}
&\overline{V}(b_{i,l}) = \textstyle\sum_s b_{i,l}(s) \max_{\alpha^s} \sum_{n_{j,l-1}} b_{i,l}(n_{j,l-1}|s)\alpha^s(n_{j,l-1}) \\
&= \textstyle\sum_s b_{i,l}(s) \max_{a_i \in A_i} \sum_{n_{j,l-1}} b_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \Big\{ R_i(s,a_i,a_j) + \\
&\textstyle\sum_{s'} Pr(s'|s,a_i,a_j) \sum_{\omega_j} O_j(s',a_j,\omega_j) \sum_{n'_{j,l-1}} Pr(n'_{j,l-1}|n_{j,l-1},a_i,a_j)\alpha^{s',k'}(n'_{j,l-1}) \Big\}
\end{aligned} \tag{4.4}$$

Proposition 5 (Upper bound). *Let $\overline{H}$ be the backup operator for the value function of I-POMDP$^S_{i,l}$ as defined in Eq. 4.4. Then, it holds that $H\overline{V} \leq \overline{HV}$, where H is the backup operator for I-POMDP$_{i,l}$ as defined previously.*

Proof. Value of a belief, $b_{i,l}$, for horizon 1 in I-POMDP$_{i,l}$ is,

$$
\begin{aligned}
V(b_{i,l}) &= \max_{a_i \in A_i} \sum_{s,n_j} b_{i,l}(s, n_{j,l-1}) R_i(s, a_i, a_j) Pr(a_j | n_{j,l-1}) \\
&= \max_{a_i \in A_i} \sum_{s} b_{i,l}(s) \sum_{n_j} b_{i,l}(n_{j,l-1}|s) R_i(s, a_i, a_j) Pr(a_j | n_{j,l-1}) \\
&\leq \sum_{s} b_{i,l}(s) \max_{a_i \in A_i} \sum_{n_j} b_{i,l}(n_{j,l-1}|s) R_i(s, a_i, a_j) Pr(a_j | n_{j,l-1}) \\
&= \overline{V}(b_{i,l})
\end{aligned}
$$

For a horizon greater than one, we obtain,

$$
\begin{aligned}
&H\overline{V}(b_{i,l}) = \max_{a_i \in A_i} \textstyle\sum_s b_{i,l}(s) \sum_{n_{j,l-1}} b_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \{R_i(s, a_i, a_j) + \\
&\textstyle\sum_{\omega_i} \sum_{s'} Pr(s', \omega_i | s, a_i, a_j) \sum_{\omega_j} O_j(s', a_j, \omega_j) \sum_{n'_{j,l-1}} Pr(n'_{j,l-1} | n_{j,l-1}, a_i, a_j) \alpha^{s',k'}(n'_{j,l-1}) \Big\}
\end{aligned}
$$

The alpha vector, $\alpha^{s',k'}(n'_{j,l-1})$, is the one in the set of vectors for the next state, s', that gives the largest value for the updated conditional belief over the models. Its selection from the set does not depend on the observation, ω_i, unlike in I-POMDP$_{i,l}$. Therefore, the equation above simplifies to,

$$
\begin{aligned}
&H\overline{V}(b_{i,l}) = \max_{a_i \in A_i} \textstyle\sum_s b_{i,l}(s) \sum_{n_{j,l-1}} b_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \big\{R_i(s, a_i, a_j) \\
&+ \textstyle\sum_{s'} Pr(s'|s, a_i, a_j) \sum_{\omega_j} O_j(s', a_j, \omega_j) \sum_{n'_{j,l-1}} Pr(n'_{j,l-1} | n_{j,l-1}, a_i, a_j) \alpha^{s',k'}(n'_{j,l-1}) \Big\} \\
&\leq \textstyle\sum_s b_{i,l}(s) \max_{a_i \in A_i} \sum_{n_{j,l-1}} b_{i,l}(n_{j,l-1}|s) \sum_{a_j} Pr(a_j|n_{j,l-1}) \big\{R_i(s, a_i, a_j) \\
&+ \textstyle\sum_{s'} Pr(s'|s, a_i, a_j) \sum_{\omega_j} O_j(s', a_j, \omega_j) \sum_{n'_{j,l-1}} Pr(n'_{j,l-1} | n_{j,l-1}, a_i, a_j) \alpha^{s',k'}(n'_{j,l-1}) \Big\} \\
&= \overline{HV}
\end{aligned}
$$

Under the isotonicity property of the I-POMDP backup operator and $V \leq \overline{V}$, we obtain, $H(HV) \leq H(H\overline{V})$. Similarly, $H(H\overline{V}) \leq H(\overline{HV})$. We may reapply the above proof and

assert that, $H(\overline{HV}) \leq \overline{H}(\overline{HV})$. This implies that, $H(HV) \leq \overline{H}(\overline{HV})$, which means that the upper bound is maintained over any number of applications of the two backup operators to their respective value functions. □

4.1.3 Bimodal switching

Let the difference between the upper and lower bound values for a belief, $b_{i,l}$, over the physical states be, $\mathsf{Diff} = \overline{V}_i(b_{i,l}) - \underline{V}_i(b_{i,l})$. Let R_{min} and R_{max} be the smallest and highest rewards in agent i's reward function, R_i. Subsequently, $R_{min}\frac{1-\gamma^H}{1-\gamma}$ and $R_{max}\frac{1-\gamma^H}{1-\gamma}$ are the minimum and maximum rewards that agent i could obtain over a finite horizon of H with a discount factor of γ. These may be easily calculated from the model definition.

Because of the piecewise linear and convexity property of the value function of POMDP_i, and the relatively flat value function of $\text{I-POMDP}^S_{i,l}$, we expect Diff to reduce as the uncertainty in agent i's belief over the state space reduces and the belief approaches the edges of the belief simplex. Our approach switches from online planning using POMDP_i to planning using $\text{I-POMDP}_{i,l}$ when $\frac{\mathsf{Diff}\cdot(1-\gamma)}{(1-\gamma^H)\cdot(R_{max}-R_{min})}$ drops to below a parameter, $\epsilon \in [0, 1]$. In other words, ϵ is the fraction of the largest possible difference in value, which triggers the switch. However, not all values of ϵ may be reached. Specifically, there is no guarantee that the upper and lower bounds converge near the vertices of the belief simplex. Moreover, extremely small values of ϵ may not ever be reached and hence may never cause a switch.

4.1.4 Computational Savings and Error Bound

Instead of solving a multiagent planning problem from the start, our approach exploits single-agent planning in the early stages subsequently switching to multiagent planning on reaching a belief distribution over the physical states that admits reduced error bound. Hence, computational savings are obtained in the early steps when fast and scalable single-agent planning

is performed. In order to obtain an estimate of the savings, let us suppose that we use an exact POMDP-based approach for online planning that generates a perfect reachability tree of $H-1$ steps whose branching factor is $(|A_i||\Omega_i|)$, from a belief. The tree contains $(|A_i||\Omega_i|)^{H-1}$ nodes each of which is associated with a single real number whose computation takes time $\mathcal{O}(|A_i||S|)$ if the node is a leaf node, otherwise it takes $\mathcal{O}(|A_i||\Omega_i||S|^2)$. On the other hand, let us suppose that we use an exact I-POMDP$_{i,l}$-based approach for the planning. The reachability tree from a belief over the interactive state space would continue to have a branching factor of $(|A_i||\Omega_i|)$ and as many nodes as mentioned previously. However, calculating the value of the belief associated with each leaf node takes time $\mathcal{O}(|A_i||S||N_{j,l-1}||A_j|)$ and the time for calculating the value at a non-leaf node takes $\mathcal{O}(|A_i||S|^2|N_{j,l-1}||A_j|\Omega_i||\Omega_j|)$. The difference in computation time at each node is due to modeling the other agent, and the savings at all the nodes accumulates over the number of steps for which planning uses POMDP$_i$, which may vary.

Our choice of SARSOP – a state of the art approach – minimizes the time taken to perform the POMDP-based planning. SARSOP generates policy trees that are near-optimal for a given horizon from any belief, and the initial computation time is amortized over the multiple steps for which the graph is used. The I-POMDP$_{i,l}$-based online planning utilizes I-PF and generates a reachability tree for a given horizon from a given belief resulting in an approximate action.

If an exact approach is utilized for online multiagent planning after the switch, error is incurred until the approach switches when ϵ is achieved. At this point, the difference between the upper and lower bounds is, $\epsilon \cdot \frac{(1-\gamma^H)\cdot(R_{max}-R_{min})}{1-\gamma}$, which bounds the error as well. If T steps were performed before switching, then the error is at least, $T \cdot \epsilon \cdot \frac{(1-\gamma^H)\cdot(R_{max}-R_{min})}{1-\gamma}$. This also serves as a reasonable estimate of the error because our bounds are tight resulting in small ϵ values, as I illustrate next.

4.2 Experimental Evaluation

To illustrate the effectiveness of the bimodal approach, we used a version of the multiagent tiger problem (example 1) which is modified to make the actions of agent j unobservable. Also, we use a persistent version of the problem where on opening a door the tiger remains behind the same door with a probability 0.75 and moves to behind the other door with the remaining probability. The reward function remains the same. The modified observation function is as shown in table 4.1.

$a_i = L$	$O_i(s', a_i, GL)$	$O_i(s', a_i, GR)$
TL	0.85	0.15
TR	0.15	0.85

$a_i = OL/OR$	$O_i(s', a_i, GL)$	$Pr(s', a_i, GR)$
TL	0.5	0.5
TR	0.5	0.5

Table 4.1: Observation function for the modified version of multiagent tiger problem where the actions of agent j are unobservable to agent i (definition 2). Notice that the observation probabilities for agent i are independent of the actions of j.

Next, we modify the transition function to let the agent persist behind its original door with probability 0.75 when either agent opens a door. The modified transition function is illustrated in table 4.2.Furthermore, we mitigate the amount of noise in j's observations of the state as modeled by i to 0.05.

In Figure 4.1, I illustrate the lower and upper bound values for changing beliefs of agent i over the physical states for the multiagent tiger problem [20]. Beginning at a belief of $b_{i,1}(TR) = 0.5$ indicating that the tiger is believed to be behind the right door with a probability of 0.5, the beliefs are updated as the agent listens and receives observations. We obtained j's controller using BPI, which has 5 nodes. Agent i's initial distribution over these nodes is obtained by mapping a distribution over level 0 models of j to a distrbution over the nodes. Notice that the fraction, ϵ, becomes smaller as the beliefs show less uncertainty although not monotonically. The increase from steps 2 to 3 occurs due to i opening the left door causing the uncertainty in its beliefs to increase and reducing the lower bound value.

$a_i = L$, $a_j = L$	$T_i(s, a_i, a_j, TL)$	$T_i(s, a_i, a_j, TR)$
TL	1.0	0.0
TR	0.0	1.0

$a_i = OL/OR$, $a_j = *$	$T_i(s, a_i, a_j, TL)$	$T_i(s, a_i, a_j, TR)$
TL	0.75	0.25
TR	0.25	0.75

$a_i = L$, $a_j = OL/OR$	$T_i(s, a_i, a_j, TL)$	$T_i(s, a_i, a_j, TR)$
TL	0.75	0.25
TR	0.25	0.75

Table 4.2: Transition function for the modified version of multiagent tiger problem. Notice that the tiger remains behind the same door with a probability 0.75 when either agent opens the door. Allowing the tiger to persist ensures that the information gained from the observations till the point when a door is opened is not completely lost by the act of opening the door.

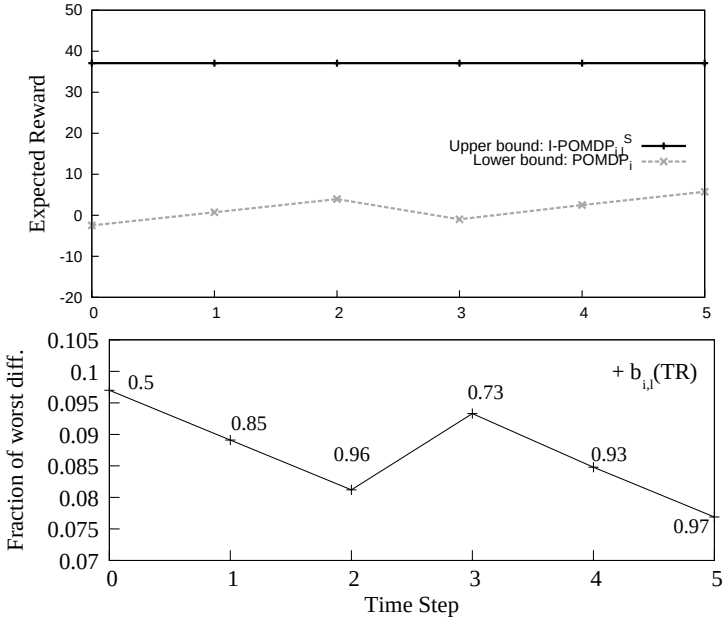


Figure 4.1: **(top)** Beginning with $b_{i,l}(TR) = 0.5$, we show the lower and upper bound values obtained from POMDP$_i$ and I-POMDP$^S_{i,l}$, respectively, for a run of the multiagent persistent tiger problem. **(bottom)** The fraction of the largest difference in bounds is shown as agent i acts, observes and its beliefs update, in simulation.

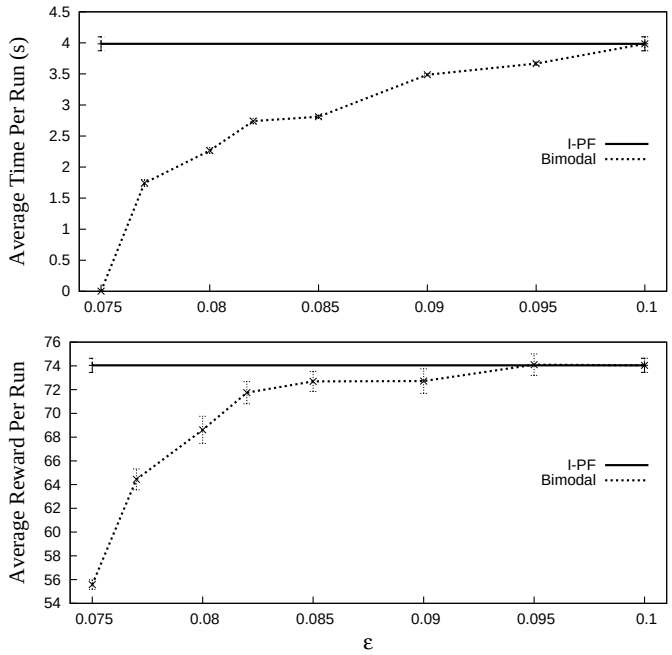


Figure 4.2: **(top)** Average time taken per simulation run for different values of ϵ (Xeon i3 2.6GHz, 4GB with Linux). **(bottom)** Cumulative rewards averaged over the 300 runs with differing ϵ.

Taking an online planning horizon of $H = 5$, we simulated agent i's play of the tiger problem for 30 steps. We vary ϵ and show the time and cumulative reward averaged over 5 trials of 300 simulation runs each. Notice the low values of ϵ indicating that our bounds are tight. The feasible range of ϵ is [0.075,0.1], with the approach unable to reach $\epsilon < 0.075$ and degenerating into POMDP$_i$ for all the steps, while satisfying $\epsilon < 0.1$ at the first step itself thereby running using I-PF for all the steps. For the smallest ϵ value which causes a bimodal switch, the average time is about 50% less than I-PF albeit obtaining average reward that is significantly lower. However, as ϵ increases, POMDP$_i$ runs for less steps and both the time and rewards increase approaching that of I-PF.

4.3 Conclusion

I presented a new approach for online planning in multiagent settings where actions of the other agent are not directly observable and must be inferred from the state transitions. For typical initial beliefs with high uncertainty over the physical states, our approach utilizes POMDP-based planning and switches over to online multiagent planning. The mode changes when the fraction of the difference between the upper and lower bounds reduces to less than a parameter. This technique of utilizing bounds is analogous to previous approaches such as HSVI [65] and SARSOP [40], although these compute bounds relevant to single agent settings. Our demonstration on a toy problem domain indicates that the bimodal approach is flexible and could trade-off computation time with the rewards received.

Chapter 5

Individual Planning in Agent Populations: Exploiting Anonymity and Frame-Action Hypergraphs

Finally, I address the intractability arising due to the number of interacting agents sharing the same environment as the subject agent by exploiting common problem structures. While the previous approaches for solving I-POMDPs such as interactive particle filtering [14], point-based value iteration [15] and interactive bounded policy iteration (I-BPI) [69] focus on mitigating the curses of dimensionality, history, and agent modeling, there has been no directed effort to solve I-POMDPs involving many – say, *a population of more than a thousand* – interacting agents. To the best of my knowledge, the most scalability reported in terms of number of agents for I-POMDPs was for five agent tiger problem [20] by I-BPI [69]. We exploit some problem structures that are inherent in many real world problem domain and demonstrate the scalability to problems involving thousands of agents.

For illustration, consider the decision-making problem of the police when faced with monitoring and controlling protests that could take place at multiple sites. The degree of

police response (the size of police force deployed) at any site is often determined by the number of protesters and the type (formally referred to as frame) of the protesters (peaceful or disruptive) participating at that location. The individual identity of the protester within each type seldom matters. This key observation of *frame-action anonymity* motivates us in how we model the agent population in the planning process. Furthermore, the size of police troop deployed at any site is decided in response to the number of disruptive and peaceful protesters predicted to converge at that particular site and much less by the number of protesters joining protest at any other site. Therefore, police actions depend on just a few actions of note for each type of agent.

The example above illustrates two known and powerful types of problem structure in domains involving many agents: *action anonymity* [58] and *context-specific independence* [7]. Action anonymity allows the exponentially large joint action space to be substituted with a much more compact polynomial space of action *configurations* where a configuration is a tuple representing the number of agents performing each action. Secondly, context-specific independence – wherein given a context such as the state and agent's own action, not all actions performed by other agents are relevant – permits the space of configurations to be further compressed by considering counts only over a subset of other agents' actions and grouping all other irrelevant actions as a dummy *null* action. We extend the definition of both action anonymity and context-specific independence to allow considerations of an agent's *frame* as well. [1] The specific contributions of this paper are listed below:

1. I-POMDP solutions are severely challenged in settings involving multiple other agents. The space complexity for representing the I-POMDP problem and the time complexity for solving the problem grows exponentially due to the exponential growth in the space of joint models and joint actions. By exploiting commonly found problem structures

[1] I-POMDPs distinguish between an agent's frame and type with the latter including beliefs as well. Frames are similar in semantics to the colloquial use of types.

such as frame-action anonymity and context-specific independence, we present a new method for considerably scaling the solution of I-POMDPs to many agents. We call our approach Many-agent I-POMDP.

2. We reformulate the I-POMDP definition in terms of multiple contexts and present a systematic way of modeling the context-specific independence in transition, observation and reward functions using frame-action hypergraphs. We integrate our new formulation in a simple method for solving I-POMDPs that models other agents using finite-state machines, builds reachability trees given an initial belief, and computes the value in a bottom-up manner.

3. We prove that the Bellman equation modified to include frame-action configurations and frame-action context-specific independence continues to remain optimal.

4. We theoretically verify the improved savings in computational time and memory, and empirically demonstrate it on two new problem domains – one that of policing protest at multiple locations and the other a gaming scenario based on the popular mobile game *Clash of Clans* (`http://clashofclans.com`) – and show results for both domains in settings involving hundreds of interacting agents.

5. While the exploitation of the aforementioned problem structures mitigates the curse of many agents, it does not lessen the impact of the curse of history which could still be prominent. To address this curse, first I describe how we adapt bounds on POMDP value function described in section 1.1.3 – *blind policy* lower bound and *fast informed bound* (FIB) upper bound [27] [26] – to efficiently compute novel bounds for I-POMDP involving many agents. Next, I introduce a *branch and bound* approach that utilizes these novel bounds for pruning reachability subtrees without the need of evaluating them first thereby significantly diminishing the effects of curse of history.

We demonstrate the efficiency of the branch and bound method by exactly solving aforementioned problems for up to 2000 agents in less than 6 hours.

5.1 Related Work

Building on graphical games [36], *action graph games (AGG)* [32] utilize problem structures such as action anonymity and context-specific independence to concisely represent single shot complete-information games involving multiple agents and to scalably solve for Nash equilibrium. The context-specific independence in the utility function is modeled using a directed graph known as action graph whose nodes are actions and an edge between two nodes indicates that the reward of an agent performing an action indicated by one node is affected by other agents performing action of the other node. Lack of edges between nodes encodes the context-specific independence. The context in such games is the action performed by the agent. Action anonymity is useful when the action sets of agents overlap substantially. In action graph games, the rewards on performing a certain action depend on the number of agents performing each of its neighboring actions in the action graph. Significant savings are obtained because the set of the vectors of counts over the set of distinct actions, called a *configuration*, is much smaller than the set of action profiles.

We substantially build on AGGs by extending anonymity and context-specific independence to include agents' frames, and generalizing their use to a partially observable stochastic game solved using decision-theoretic planning as formalized by I-POMDPs. Indeed, Bayesian AGGs [31] extend the original formulation to include agent types. These result in type-specific action sets with the benefit that the action graph structure does not change although the number of nodes grows with types: $|\hat{\Theta}||A|$ nodes for agents with $|\hat{\Theta}|$ types each having same $|A|$ actions. If two actions from different type-action sets share a node, then these actions are interchangeable. A key difference in our representation is that we explicitly model

frames in the graphs due to which context-specific independence is modeled using frame-action *hypergraphs*. Benefits are that we naturally maintain the distinction between two similar actions but performed by agents of different frames, and we add less additional nodes: $|\hat{\Theta}| + |A|$. However, a hypergraph is a more complex data structure for operation. Finally another extension of AGGs, temporal AGGs [33] extend AGGs to a repeated game setting and allow decisions to condition on chance nodes. These nodes may represent the action counts from previous step (similar to observing the actions in the previous game). TAGGs come closest to multiagent influence diagrams [39] although they can additionally model the anonymity and independence structure. *Overall, I-POMDPs with frame-action anonymity and context-specific independence significantly augment the combination of Bayesian and temporal AGGs further by utilizing the structures in a partially observable stochastic game setting with agent types.*

Varakantham et al. ([73]) building on previous work [74] recently introduced a decentralized MDP that models a simple form of anonymous interactions: rewards and transition probability specific to a state-action pair are affected by the number of other agents regardless of their identities. The interaction influence is not further detailed into which actions of other agents are relevant as in action anonymity due to which both configurations and hypergraphs are not used. Furthermore, agent types are not considered. Finally, the interaction hypergraphs in networked-distributed POMDPs [45] model complete reward independence between agents – analogous to graphical games – which differs from the hypergraphs we use (and action graphs) that model independence in reward (and transition, observation probabilities) along a different dimension: actions.

5.2 Background

In this section, I generalize the two agent definition of a finitely-nested interactive I-POMDP defined in chapter 1 for an agent (say agent 0) of strategy level l operating in a setting inhabited by any general number of agents as the following tuple:

$$\text{I-POMDP}_{0,l} = \langle IS_{0,l}, A, T_0, \Omega_0, O_0, R_0, OC_0 \rangle$$

where:

- $IS_{0,l}$ denotes the set of *interactive states* defined as, $IS_{0,l} = S \times \prod_{j=1}^{N} M_{j,l-1}$, where S and $M_{j,l-1}$ are defined in the same manner as in section 1.2.1.
- $A = A_0 \times A_1 \times \ldots \times A_N$ is the set of joint actions of all agents;
- $T_0 : S \times A_0 \times \prod_{j=1}^{N} A_j \times S \rightarrow [0, 1]$ is the transition function.
- Ω_0 is the set of agent 0's observations;
- $O_0 : S \times A_0 \times \times \prod_{j=1}^{N} A_j \times \Omega_0 \rightarrow [0, 1]$ is the observation function.
- $R_0 : S \times A_0 \times \prod_{j=1}^{N} A_j \rightarrow \mathbb{R}$ is the reward function.
- OC_0 is the optimality criterion, which is identical to that for POMDPs.

Besides the physical state space, the I-POMDP's interactive state space contains all possible models of other agents. Therefore, agent 0's belief is a distribution over its interactive states, i.e. $b_{0,l} \in \Delta(IS_{0,l})$. In its belief update, an agent has to update its belief about the physical states as well as about the other agents' models based on an estimation about the other agents' observations and how they update their models. As the number of agents sharing the environment grows, the size of the joint action and joint model spaces increases

exponentially. Therefore, the memory requirement for representing the transition, observation, and reward functions grows exponentially as does the complexity of performing belief update over the interactive states and value iteration. In the context of N agents, interactive bounded policy iteration [69] generates good quality solutions for an agent interacting with 4 other agents (total of 5 agents) without exploiting any problem structure. To the best of our knowledge, this result illustrates the best scalability in terms of number of agents.

5.3 Many-Agent I-POMDP

To facilitate understanding and experimentation, I introduce a pragmatic running example that also forms one of our evaluation domains.

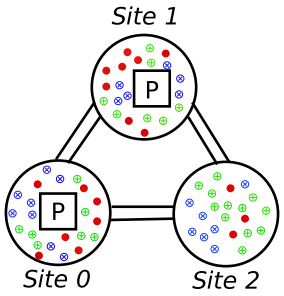


Figure 5.1: protesters of different frames (colors) and police troops at one of 3 sites in the policing protest domain. The state space of police decision making is factored into the protest intensity levels at the sites.

Example 7 (Policing Protest)**.** *Consider a policing scenario where police (agent* 0*) must maintain order in 3 geographically distributed and designated protest sites (labeled 0, 1, and 2) as shown in Fig. 5.1. A population of N agents are protesting at these sites. Police may dispatch one or two riot-control troops to either the same or different locations. Protests with differing intensities,* low, medium *and* high*, occur at each of the three sites. The goal of the police is to deescalate protests to the low intensity at each site. Protest intensity at any site is influenced by the number of protesters and the number of police troops at that*

location. In the absence of adequate policing, we presume that the protest intensity escalates. On the other hand, two police troops at a location are adequate for deescalating protest of any intensity.

5.3.1 Factored Beliefs and Update

As I mentioned previously, the subject agent in the I-POMDP framework maintains a belief over the physical state and joint models of other agents, $b_{0,l} \in \Delta(S \times \prod_{j=1}^{N} M_{j,l-1})$, where $\Delta(\cdot)$ is the space of probability distributions. For settings where N is large, the size of the interactive state space is exponentially larger, $|IS_{0,l}| = |S||M_j|^{*^N}$, where $|M_j|^*$ is the largest size of the model space among all other agents. Hence, the belief representation quickly becomes unwieldy as the number of agents grows. For example, if $|M_j|^* = 2$, a problem domain with $N > 30$ would require many gigabytes of memory to store the belief in the aforementioned joint form. For problem domains involving thousands of agents such as the one described in Example 7, it is impossible to represent beliefs in this form. However, the representation becomes manageable for large N if the belief is factored:

$$b_{0,l}(s, m_{1,l-1}, m_{2,l-1}, \ldots, m_{N,l-1}) = Pr(s)\ Pr(m_{1,l-1}|s) \times Pr(m_{2,l-1}|s) \times \ldots \times Pr(m_{N,l-1}|s) \tag{5.1}$$

This factorization assumes conditional independence of models of different agents given the physical state. Consequently, beliefs that correlate agents may not be directly represented, although correlation could be alternately supported by introducing models with a correlating device.

The memory required to store belief in factored form is $\mathcal{O}(|S|+N|S||M_j|^*)$. This is linear in the number of agents, which is much less than the exponentially growing memory required to represent the belief as a joint distribution over the interactive state space, $\mathcal{O}(|S||M_j|^{*^N})$.

Given agent 0's belief at time t, $b^t_{0,l}$, its action a^t_0 and the subsequent observation it receives, ω^{t+1}_0, the updated belief at time step $t+1$, $b^{t+1}_{0,l}$, may be obtained as:

$$\begin{aligned} b^{t+1}_{0,l}(s^{t+1}, m^{t+1}_{1,l-1}, \ldots, m^{t+1}_{N,l-1}) &= Pr(s^{t+1}, m^{t+1}_{1,l-1}, \ldots, m^{t+1}_{N,l-1} | b^t_{0,l}, a^t_0, \omega^{t+1}_0) \\ &= Pr(s^{t+1} | b^t_{0,l}, a^t_0, \omega^{t+1}_0) Pr(m^{t+1}_{1,l-1} | s^{t+1}, m^{t+1}_{2,l-1}, \ldots, m^{t+1}_{N,l-1}, b^t_{0,l}, a^t_0, \omega^{t+1}_0) \\ &\times \ldots \times Pr(m^{t+1}_{N,l-1} | s^{t+1}, b^t_{0,l}, a^t_0, \omega^{t+1}_0) \end{aligned} \tag{5.2}$$

Each factor in the product of Eq. 5.2 may be obtained as follows. The update over the physical state is:

$$\begin{aligned} &Pr(s^{t+1} | b^t_{0,l}, a^t_0, \omega^{t+1}_0) = \frac{1}{Pr(\omega^{t+1}_0 | b^t_{0,l}, a^t_0)} Pr(s^{t+1}, \omega^{t+1}_0 | b^t_{0,l}, a^t_0) \quad \text{(Bayes Rule)} \\ &\propto Pr(s^{t+1}, \omega^{t+1}_0 | b^t_{0,l}, a^t_0) \\ &= \sum_{s^t} \sum_{\mathbf{m}^t_{-0}} \sum_{\mathbf{a}^t_{-0}} Pr(s^{t+1}, \omega^{t+1}_0, s^t, \mathbf{m}^t_{-0}, \mathbf{a}^t_{-0} | b^t_{0,l}, a^t_0) \\ &= \sum_{s^t} b^t_{0,l}(s^t) \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(m^t_1 | s^t) \times \ldots \times b^t_{0,l}(m^t_N | s^t) \times \sum_{\mathbf{a}^t_{-0}} Pr(a^t_1 | m^t_{1,l-1}) \times \ldots \times Pr(a^t_N | m^t_{N,l-1}) \\ &\times O^{t+1}_0(s^{t+1}, \langle a^t_0, \mathbf{a}^t_{-0} \rangle, \omega^{t+1}_0)\ T_0(s^t, \langle a^t_0, \mathbf{a}^t_{-0} \rangle, s^{t+1}) \end{aligned} \tag{5.3}$$

and the update over the model of each other agent, $j = 1 \ldots N$, conditioned on the state at $t+1$ is:

$$\begin{aligned} &Pr(m^{t+1}_{j,l-1} | s^{t+1}, m^{t+1}_{j+1,l-1}, \ldots, m^{t+1}_{N,l-1}, b^t_{0,l}, a^t_0, \omega^{t+1}_0) \\ &= \sum_{s^t} b^t_0(s^t) \sum_{\mathbf{m}^t_{-j,l-1}} b^t_{0,l}(m^t_{j+1,l-1} | s^t) \times \ldots \times b^t_{0,l}(m^t_{N,l-1} | s^t) \sum_{a^t_j} \sum_{\mathbf{a}^t_{-j}} Pr(a^t_1 | m^t_{j+1,l-1}) \\ &\times \ldots \times Pr(a^t_n | m^t_{N,l-1}) \times \sum_{\omega^{t+1}_j} O_j(s^{t+1}, \langle a_j, \mathbf{a}^t_{-j} \rangle, \omega^{t+1}_j) Pr(m^{t+1}_j | m^t_j, a^t_j, \omega^{t+1}_j) \end{aligned} \tag{5.4}$$

Derivations of Eqs. 5.3 and 5.4 are straightforward. In particular, note that models of agents other than j at $t+1$ do not impact j's model update in the absence of correlated behavior. Thus, under the assumption of a factored prior as in Eq. 5.1 and absence of model correlations, the I-POMDP belief update may be decomposed into an update of the physical state and the models of the N agent conditioned on the state.

5.3.2 Frame-Action Anonymity

As noted by Jiang et al.[32], many non-cooperative and cooperative problems exhibit the structure where given the agent's own action, its payoffs depend on the number of agents performing certain actions rather than which agent is performing what action. This problem structure is particularly evident in Example 7 where the outcome of policing at any given site depends only on the number of peaceful and disruptive protesters converging at that site rather than their identities. Building on this idea, we observe that in partially observable, dynamic, and stochastic settings such as the one described in example 7, the state transitions and the observations of the police at a site are also influenced by the number of peaceful and disruptive protesters converging to protest at that site. This is noted in the example below:

Example 8 (Frame-action anonymity of protesters)**.** *The transient state of protests reflecting the intensity of protests at each site depends on the previous intensity at a site and the* **number** *of peaceful and disruptive protesters entering the site. Police (noisily) observes the intensity of protest at each site which is again largely determined by the* number *of peaceful and disruptive protesters at a site. Finally, the outcome of policing at a site is contingent on whether the protest was largely peaceful or disruptive. Consequently, the identity of the individual protesters beyond their frame and action is unnecessary and hence can be disregarded.*

Here, peaceful and disruptive nature of the protesters are captured by different *frames* of others in agent 0's I-POMDP, and the above definition may be extended to any number of

frames. Frame-action anonymity is an important attribute of the above domain. Formally, frame-action anonymity is defined in the context of agent 0's transition, observation and reward functions as follows:

Definition 4 (Frame-action anonymity)**.** *Let* $\mathbf{a}^p_{-0}$ *be a joint action of all peaceful protesters and* $\mathbf{a}^d_{-0}$ *be a joint action of all disruptive ones. Let* $\dot{\mathbf{a}}^p_{-0}$ *and* $\dot{\mathbf{a}}^d_{-0}$ *be permutations of the two joint action profiles, respectively. An* I-POMDP *models frame-action anonymity iff for any* a_0, s, s', $\mathbf{a}^p_{-0}$ *and* $\mathbf{a}^d_{-0}$ *:*

$T_0(s,\ a_0, \mathbf{a}^p_{-0}, \mathbf{a}^d_{-0}, s') = T_0(s, a_0, \dot{\mathbf{a}}^p_{-0}, \dot{\mathbf{a}}^d_{-0}, s')$,

$O_0(s', a_0, \mathbf{a}^p_{-0}, \mathbf{a}^d_{-0}, \omega_0) = O_0(s', a_0, \dot{\mathbf{a}}^p_{-0}, \dot{\mathbf{a}}^d_{-0}, \omega_0)$, *and*

$R_0(s, a_0, \mathbf{a}^p_{-0}, \mathbf{a}^d_{-0}) = R_0(s, a_0, \dot{\mathbf{a}}^p_{-0}, \dot{\mathbf{a}}^d_{-0})$

$\forall\ \dot{\mathbf{a}}^p_{-0},\ \dot{\mathbf{a}}^d_{-0}$.

Recall the definition of an action configuration, $\mathcal{C}$, as a tuple where each value represents the action count for an action of the other agent. A permutation of joint actions of other agents of the same frame, say $\dot{\mathbf{a}}^p_{-0}$, assigns different actions to individual agents of that frame. Despite this, the fact that the transition and observation probabilities, and the reward remains unchanged indicates that the identity of the agent performing the action is irrelevant (beyond the frame identity). Importantly, the action configuration of the joint actions and its permutation stay the same: $\mathcal{C}(\mathbf{a}^p_{-0}) = \mathcal{C}(\dot{\mathbf{a}}^d_{-0})$. This combined with Def. 4 allows redefining the transition, observation and reward functions to be over configurations as: $T_0(s, a_0, \mathcal{C}(\mathbf{a}^p_{-0}), \mathcal{C}(\mathbf{a}^d_{-0}), s')$, $O_0(s', a_0, \mathcal{C}(\mathbf{a}^p_{-0}), \mathcal{C}(\mathbf{a}^d_{-0}), \omega)$, and $R_0(s, a_0, \mathcal{C}(\mathbf{a}^p_{-0}), \mathcal{C}(\mathbf{a}^d_{-0}))$.

Let $A^p_1, \ldots, A^p_n$ be the sets of actions of n peaceful protesters, and A^p_{-0} is the Cartesian product of these sets. Let $\mathcal{C}(A^p_{-0})$ be the set of all action configurations for A^p_{-0}. *Observe that multiple joint actions from* A^p_{-0} *may result in a single configuration; these joint actions are configuration equivalent.* Consequently, the equivalence partitions the joint action set A^p_{-0} into $|\mathcal{C}(A^p_{-0})|$ classes. Furthermore, when other agents of same frame have overlapping sets of actions, the number of configurations could get much smaller than the number of joint

actions. Therefore, definitions of the transition, observation and reward functions involving configurations could be more compact.

For the sake of simplicity, let $\mathcal{C}$ be a tuple representing configuration over the actions performed by agents of all types. Then, we may represent transition, observation, and reward function as $T_0(s, a_0, \mathcal{C}, s')$, $O_0(s', a_0, \mathcal{C}, \omega)$, and $R_0(s, a_0, \mathcal{C})$.

5.3.3 Frame-Action Hypergraphs

In addition to frame-action anonymity, domains involving sizable agent populations often exhibit context-specific independences. This is a broad category and includes the context-specific independence found in conditional probability tables of Bayesian networks [7] and in action-graph games. It offers significant additional structure for computational tractability. We begin by illustrating this in the context of Example 7.

Example 9 (Context-specific independence in policing)**.** *At a protest site, payoff for policing is independent of the movement of the protesters to other sites. Similarly, the transient intensity of the protest at a site given the level of policing at the site as context is independent of the movement of protesters between other sites.*

The context-specific independences above builds on the similar independence in action graphs in two ways: (*i*) We model such partial independence in the transitions of factored states and in observation function as well, in addition to the reward function. (*ii*) We allow the context-specific independence to include the frames of other agents in addition to their actions. For example, the reward from passive policing (deploying only one troop) at a site is independent of the number of *peaceful* protesters, instead influenced primarily by the number of *disruptive* protesters.

The latter difference generalizes the action graphs into *frame-action hypergraphs*, specifically 3-uniform hypergraphs where each edge is a set of 3 nodes.

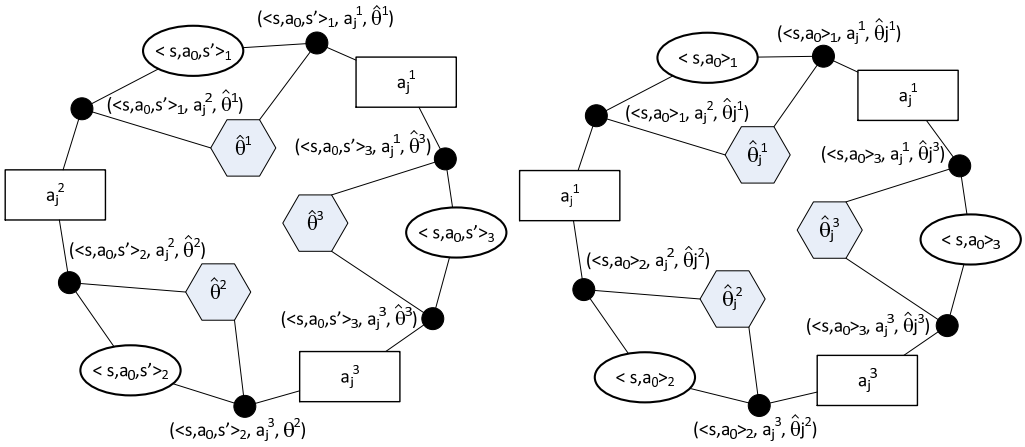


Figure 5.2: Levi (incidence) graph representation of a generic frame-action hypergraph for (a) the transition function, and (b) the reward function. The shaded nodes represent edges in the hypergraph. Each edge has the context, ψ, denoted in bold, agent's action, a, and its frame, $\hat{\theta}$, incident on it. For example, the reward for a state and agent 0's action, $\langle s, a_0 \rangle_1$ is affected by others' actions a_j^1 and a_j^2 performed by any other agent of frame $\hat{\theta}_j^1$ only.

Definition 5 (Frame-action hypergraph)**.** *A frame-action hypergraph for agent* 0 *is a 3-uniform hypergraph* $\mathcal{G} = \langle \Psi, A_{-0}, \hat{\Theta}_{-0}, E \rangle$*, where* Ψ *is a set of nodes that represent the context,* A_{-0} *is a set of action nodes with each node representing an action that any other agent may take;* $\hat{\Theta}_{-0}$ *is a set of frame nodes, each node representing a frame ascribed to an agent, and* E *is a set of 3-uniform hyperedges where each hyperedge contains one node from each set* Ψ*,* A_{-0}*, and* $\hat{\Theta}_{-0}$*, respectively.*

The context nodes differ based on whether the hypergraph applies to the transition, observation or reward functions:

- For the transition function, the context is a set containing a pair of states between which a transition may occur and an action of agent 0, $\Psi = S \times A_0 \times S$, and the action nodes includes actions of all other agents, $A_{-0} = \bigcup_{j=1}^{N} A_j$. Neighbors of a context node $\psi = \langle s, a_0, s' \rangle$ are all the frame-action pairs that affect the probability of the transition.

An edge ($\langle\ s, a_0, s'\rangle$, $\mathbf{a}_{-0}$, $\hat{\theta}$) indicates that the probability of transitioning from s to s' on performing a_0 is affected (in part) by the number of other agents of frame $\hat{\theta}$ performing the particular action in $\mathbf{a}_{-0}$.

- The context for agent 0's observation function is the state-action-observation triplet, $\Psi = S \times A_0 \times \Omega_0$, and the action nodes are identical to those in the transition function. Neighbors of a context node, $\langle s, a_0, \omega_0 \rangle$, are all those frame-action pairs that affect the observation probability. Specifically, an edge $(\langle s, a_0, \omega_0 \rangle, \mathbf{a}_{-0}, \hat{\theta})$ indicates that the probability of observing ω_0 from state s on performing a_0 is affected (in part) by the number of other agents performing action, $\mathbf{a}_{-0}$, who possess frame $\hat{\theta}$.

- For agent 0's reward function, the context is the set of pairs of state and action of agent 0, $\Psi = S \times A_0$, and the action nodes the same as those in transition and observation functions. An edge ($\langle\ s, a_0 \rangle$, $\mathbf{a}_{-0}$, $\hat{\theta}_{-0}$) in this hypergraph indicates that the reward for agent 0 on performing action a_0 at state s is affected (in part) by the agents of frame $\hat{\theta}_{-0}$ who perform action in $\mathbf{a}_{-0}$.

I illustrate a general frame-action hypergraph for context-specific independence in a transition function and a reward function as Levi graphs in Fig. 5.2(a) and (b), respectively. I point out that the hypergraph for the reward function comes closest in semantics to the graph in action graph games [32] although the former adds the state to the context and frame nodes. Hypergraphs for the transition and observation functions differ substantially in semantics and form from action graphs.

To use these hypergraphs in our algorithms, I first define the general *frame-action neighborhood* of a context node.

Definition 6 (Frame-action neighborhood)**.** *The frame-action neighborhood of a context node $\psi \in \Psi$, $\nu(\psi)$, given a frame-action hypergraph $\mathcal{G}$ is defined as a subset of $A \times \hat{\Theta}$ such that $\nu(\psi) = \{(\mathbf{a}_{-0}, \hat{\theta}) | \mathbf{a}_{-0} \in A_{-0}, \hat{\theta} \in \hat{\Theta}, (\psi, \mathbf{a}_{-0}, \hat{\theta}) \in E\}$.*

As an example in Fig. 5.2(b) , the frame-action neighborhood of a state-action pair, $\langle s, a_0 \rangle$ in a hypergraph for the reward function is the set of all action and frame nodes incident on each hyperedge anchored by the node $\langle s, a_0 \rangle$.

I move toward integrating frame-action anonymity introduced in the previous subsection with the context-specific independence as modeled above by introducing frame-action configurations.

Definition 7 (Frame-action configuration)**.** *A configuration over the frame-action neighborhood of a context node, ψ, given a frame-action hypergraph is a vector,*

$$\mathcal{C}^{\nu(\psi)} \triangleq \langle\ \mathcal{C}(\mathbf{a}_{-0}^{\hat{\theta}_1}), \mathcal{C}(\mathbf{a}_{-0}^{\hat{\theta}_2}), \ldots, \mathcal{C}(\mathbf{a}_{-0}^{\hat{\theta}_{|\hat{\Theta}|}}), \mathcal{C}(\phi)\ \rangle$$

where each a included in $\mathbf{a}_{-0}^{\hat{\theta}}$ is an action in $\nu(\psi)$ with frame $\hat{\theta}$, and $\mathcal{C}(\mathbf{a}_{-0}^{\hat{\theta}})$ is a configuration over actions by agents other than 0 whose frame is $\hat{\theta}$. All agents with frames other than those in the frame-action neighborhood are assumed to perform a dummy action, ϕ.

Definition 7 allows us to further compress the representation of the transition, observation and rewards functions of the I-POMDP using context-specific independence. Specifically, we may redefine these functions one more time (see previous redefinition in section 5.3.2) to limit the configurations only over the frame-action neighborhood of the context as, $T_0(s, a_0, \mathcal{C}^{\nu(s,a_0,s')}\rangle, s')$, $O_0(s', a_0, \mathcal{C}^{\nu(s',a_0,\omega_0)}, \omega_0)$ and $R_0(s, a_0, \mathcal{C}^{\nu(s,a_0)})$. [2]

[2]Context in our transition function is $\langle\ s, a_0, s'\ \rangle$ compared with the context of just $\langle\ s, a_0\ \rangle$ in Varakantham et al's [73] transition function.

5.3.4 Framework with Anonymity and Context-Specific Independence

In order to benefit from structures of anonymity and context-specific independence, we switch to a factored representation and redefine I-POMDP for agent 0 as follows:

$$\text{I-POMDP}_{0,l} = \langle IS_{0,l}, A, \Omega_0, \mathcal{T}_0, \mathcal{O}_0, \mathcal{R}_0, OC_0 \rangle$$

where:

- $IS_{0,l}$, A, Ω_0 and OC_0 remain the same as before. The physical states are factored as, $S = \prod_{k=1}^{K} X_k$.
- $\mathcal{T}_0$ is the transition function, $\mathcal{T}_0(x, a_0, \mathcal{C}^{\nu(x,a_0,x')}, x')$ where $\mathcal{C}^{\nu(x,a_0,x')}$ is the configuration over the frame-action neighborhood of context $\langle x, a_0, x' \rangle$ obtained from a hypergraph that holds for the transition function. This transition function is significantly more compact than the original that occupies space $\mathcal{O}(|X|^2|A_0||A_{-0}|^N)$ compared to the $\mathcal{O}(|X|^2|A_0|(\frac{N}{|\nu^*|})^{|\nu^*|})$ of $\mathcal{T}_0$, where the fraction is the complexity of $\binom{N+|\nu|^*+1}{|\nu^*|+1}$, $|\nu^*|$ is the maximum cardinality of the neighborhood of any context, and $(\frac{N}{|\nu^*|})^{|\nu^*|} \ll |A_{-0}|^N$. The value $\binom{N+|\nu|^*}{|\nu|^*}$ is obtained from combinatorial compositions and represents the number of ways $|\nu^*|+1$ non-negative values can be weakly composed such that their sum is N.
- The redefined observation function is $\mathcal{O}_0(x', a_0, \mathcal{C}^{\nu(x',a_0,\omega_0)}, \omega_0)$ where $\mathcal{C}^{\nu(x',a_0,\omega_0)}$ is the configuration over the frame-action neighborhood of context $\langle x', a_0, \omega_0 \rangle$ obtained from a hypergraph that holds for the observation function. Analogously to the transition function, the original observation function consumes space $\mathcal{O}(|X||\Omega||A_0||A_{-0}|^N)$, which is much larger than space $\mathcal{O}(|X||\Omega||A_0|(\frac{N}{|\nu^*|})^{|\nu^*|})$ occupied by this redefinition.

- $\mathcal{R}_0$ is the reward function defined as $\mathcal{R}_0(x, a_0, \mathcal{C}^{\nu(x,a_0)})$ where $\mathcal{C}^{\nu(x,a_0)}$ is defined analogously to the configurations in the previous parameters. The reward for a state and actions may simply be the sum of rewards for the state factors and actions (or a more general function if needed). As with the transition and observation functions, this reward function is compact occupying space $\mathcal{O}(|X||A_0|(\frac{N}{|\nu^*|})^{|\nu^*|})$ that is much less than $\mathcal{O}(|X||A_0||A_{-0}|^N)$ of the original.

Belief Update with Anonymity and Context-Specific Independence

For this extended I-POMDP, we compute the updated belief over a physical state as a product of its factors using Eq. 5.5 and belief update over the models of each other agent using Eq 5.6 as shown below:

$$
\begin{aligned}
&Pr(\mathbf{s}^{t+1}|b^t_{0,l}, a^t_0, \omega^{t+1}_0) \propto Pr(\mathbf{s}^{t+1}, \omega^{t+1}_0|b^t_{0,l}, a^t_0) \\
&= Pr(\omega^{t+1}_0|s^{t+1}, b^t_{0,l}, a^t_0) \times Pr(s^{t+1}|b^t_{0,l}, a^t_0) \\
&= \left\{ \prod_{k=1}^{K} Pr(\omega^{t+1}_{0,k}|x^{t+1}_k, b^t_{0,l}, a^t_0) \right\} \times \left\{ \prod_{k=1}^{K} Pr(x^{t+1}_k|b^t_{0,l}, a^t_0) \right\} \\
&= \left\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) Pr(\omega^{t+1}_{0,k}|x^{t+1}_k, b^t_{0,l}, a^t_0, \mathbf{s}^t) \right\} \times \left\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) Pr(x^{t+1}_k|b^t_{0,l}, a^t_0, x^t) \right\} \\
&= \left\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t) \sum_{\mathbf{a}^t_{-0}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0}) Pr(\omega^{t+1}_{0,k}|x^{t+1}_k, a^t_0, \mathbf{a}^t_{-0}) \right\} \times \\
&\left\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t) \sum_{\mathbf{a}^t_{-0}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0}) Pr(x^{t+1}_k|x^t, a^t_0, \mathbf{a}^t_{-0}) \right\}
\end{aligned}
$$

Let's introduce a projection function $\delta^{\nu(\psi)}$ that maps joint actions to the corresponding frame-action configurations as defined in definition 7. Formally $\delta^{\nu(\psi)} : \mathbf{a} \rightarrow \mathbf{C}^{\nu(\psi)}$, where $\mathbf{C}^{\nu(\psi)}$ is the set of all possible configurations $\mathcal{C}$ for the context ψ such that for all agents j

with frame $\hat{\theta}$, $\mathcal{C}(a,\hat{\theta}) = |\{j : a_j = a, \hat{\theta}_j = \hat{\theta}, (a_j,\hat{\theta}) \in \nu(\psi)\}|$. Next we partition the set of joint action of all other agents $\mathbf{A_{-0}}$ into smaller subsets $\mathbf{A^1_{-0}}$, ..., $\mathbf{A^{|\mathbf{C}^{\nu(\psi)}|}_{-0}}$ such that the projection function $\delta^{\nu(\psi)}$ maps all joint actions belonging to any given partition $\mathbf{A^c_{-0}}$ to the same configuration. Hence, we may rewrite the above equation as:

$$= \Bigg\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{c=1}^{|\mathbf{C}^{\nu(x_k^{t+1},a_0^t,\omega_{0,k}^{t+1})}|} \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t) \sum_{\mathbf{a}^t_{-0} \in \mathbf{A^c_{-0}}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0}) O_0(x_k^{t+1}, a_0^t, \mathbf{a}^t_{-0},$$
$$\omega_{0,k}^{t+1}) \Bigg\} \times \Bigg\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{c=1}^{|\mathbf{C}^{\nu(x_k^t,a_0^t,x_k^{t+1})}|} \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t) \sum_{\mathbf{a}^t_{-0} \in \mathbf{A^c_{-0}}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0})$$
$$T_0(x_k^t, a_0^t, \mathbf{a}^t_{-0}, x_k^{t+1}) \Bigg\}$$

Under frame-action anonymity (Definition 4) given a context, all $\mathbf{a}^t_{-0}$ that map to the same configuration yield the same value. Hence $\forall \mathbf{a}^t_{-0} \in \mathbf{A^c_{-0}}$ $T_0(x^t, a_0^t, \mathbf{a}^t_{-0}, x_k^{t+1}) = \mathcal{T}_0(x_k^t, a_0^t,$ $\mathcal{C}^{\nu(x_k^t,a_0^t,x_k^{t+1})}, x_k^{t+1})$ where $\mathcal{C}^{\nu(x_k^t,a_0^t,x_k^{t+1})} = \delta^{\nu(\mathcal{C}^{\nu(x_k^t,a_0^t,x_k^{t+1})})}(\mathbf{a}^t_{-0})$. Similarly, $O_0(x_k^{t+1}, a_0^t, \mathbf{a}^t_{-0}, \omega_{0,k}^{t+1})$ $= \mathcal{O}_0(x_k^{t+1}, a_0^t, \mathcal{C}^{\nu(x_k^{t+1},a_0^t,\omega_{0,k}^{t+1})}, \omega_{0,k}^{t+1})$. Therefore the above equation becomes:

$$= \Bigg\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{c=1}^{|\mathbf{C}^{\nu(x_k^{t+1},a_0^t,\omega_{0,k}^{t+1})}|} \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t) \sum_{\mathbf{a}^t_{-0} \in \mathbf{A^c_{-0}}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0})$$
$$O_0(x_k^{t+1}, a_0^t, \mathcal{C}^{\nu(x_k^{t+1},a_0^t,\omega_0^{t+1})}, \omega_{0,k}^{t+1}) \Bigg\} \times \Bigg\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{c=1}^{|\mathbf{C}^{\nu(x_k^t,a_0^t,x_k^{t+1})}|} \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t)$$
$$\sum_{\mathbf{a}^t_{-0} \in \mathbf{A^c_{-0}}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0}) T_0(x_k^t, a_0^t, \mathcal{C}^{\nu(x_k^t,a_0^t,x_k^{t+1})}, x_k^{t+1}) \Bigg\}$$

Also we may compute the probability of a given configuration using the probability of the models of other agents and the probability of their actions using a dynamic program that computes configurations and their corresponding probabilities by successively adding the actions of each agent at a time (described later in section 5.4). Therefore, we may rewrite the equation as:

$$
\begin{aligned}
& Pr(\mathbf{s}^{t+1}|b^t_{0,l}, a^t_0, \omega^{t+1}_0) \\
& \propto \Bigg\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{\mathcal{C}^{\nu(x_k^{t+1}, a_0^t, \omega_0^{t+1})}} Pr(\mathcal{C}^{\nu(x_k^{t+1}, a_0^t, \omega_0^{t+1})}|b^t_{0,l}(M_{1,l-1}|\mathbf{s}^t), \ldots, b^t_{0,l}(M_{N,l-1}|\mathbf{s}^t)) \\
& \mathcal{O}_0(x^{t+1}_k, a^t_0, \mathcal{C}^{\nu(x_k^{t+1}, a_0^t, \omega_0^{t+1})}, \omega^{t+1}_0) \Bigg\} \times \Bigg\{ \prod_{k=1}^{K} \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{\mathcal{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}} Pr(\mathcal{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}| \\
& b^t_{0,l}(M_{1,l-1}|\mathbf{s}^t), \ldots, b^t_{0,l}(M_{N,l-1}|\mathbf{s}^t)) \mathcal{T}_0(x^t_k, \mathcal{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}, x^{t+1}_k) \Bigg\} \qquad (5.5)
\end{aligned}
$$

Here, the term, $Pr(\mathcal{C}^{\nu(x_k^{t+1}, a_0^t, \omega_0^{t+1})}|b^t_{0,l}(M_{1,l-1}|s^t), \ldots, b^t_{0,l}(M_{N,l-1}|s^t))$, is the probability of a frame-action configuration (see Def. 7) that is context specific to the triplet, $\langle x^{t+1}_k, a_0, \omega^{t+1}_{0,k} \rangle$. It is computed using the factored beliefs over the models of each individual other agents and the probability of their individual actions conditioned on their models. I discuss this computation in the next section. The second configuration term has an analogous meaning and is computed similarly.

The factored belief update over the models of every other agent, $j = 1 \ldots N$, conditioned on the state at $t + 1$ becomes:

$$
\begin{aligned}
&Pr(m^{t+1}_{j,l-1}|s^{t+1}, \mathbf{m}^{t+1}_{-j,l-1}, b^t_{0,l}, a^t_0) \\
&= \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{m^t_{1,l-1}} b^t_{0,l}(m^t_{1,l-1}|\mathbf{s}^t) \sum_{a^t_1} Pr(a^t_1|m^t_{1,l-1}) \ldots \sum_{m^t_{N,l-1}} b^t_{0,l}(m^t_{N,l-1}|\mathbf{s}^t) \\
&\sum_{a^t_N} Pr(a^t_N|m^t_{N,l-1}) \sum_{\omega^{t+1}_j} \Big\{ \prod_{k=1}^{K} \mathcal{O}_j(x^{t+1}_k, a^t_j, a^t_{-j}, \omega^{t+1}_{j,k}) \Big\} Pr(m^{t+1}_{j,l-1}|m^t_{j,l-1}, a^t_j, \omega^{t+1}_j) \\
&= \sum_{\mathbf{s}^t} b^t_0(\mathbf{s}^t) \sum_{m^t_j} b^t_0(m^t_j|s^t) \sum_{a^t_j} Pr(a^t_j|m^t_j) \sum_{\omega^{t+1}_j} \Big\{ \prod_{k=1}^{K} \sum_{\mathcal{C}^{\nu(x^{t+1}_k, a^t_j, \omega^{t+1}_{j,k})}} Pr(\mathcal{C}^{\nu(x^{t+1}_k, a^t_j, \omega^{t+1}_{j,k})}| \\
&a^t_0, b^t_{0,l}(M_{1,l-1}|\mathbf{s}^t), \ldots, b^t_{0,l}(M_{j-1,l-1}|\mathbf{s}^t), b^t_{0,l}(M_{j+1,l-1}|\mathbf{s}^t), \ldots, b^t_{0,l}(M_{N,l-1}|\mathbf{s}^t)) \\
&\mathcal{O}_j(x^{t+1}_k, a^t_j, \mathcal{C}^{\nu(x^{t+1}_k, a^t_j, \omega^{t+1}_{j,k})}, \omega^{t+1}_{j,k}) \Big\} \; Pr(m^{t+1}_j|m^t_j, a^t_j, \omega^{t+1}_j)
\end{aligned} \tag{5.6}
$$

The term for computing the probability of configuration in the last line of equation 5.6 is derived by summing out the probability of joint action and joint models in a similar manner as in the derivation of equation 5.5.

Value Function

The finite-horizon value function of the many-agent I-POMDP continues to be the sum of agent 0's immediate reward and the discounted expected reward over the future:

$$
V^h(b^t_{0,l}) = \max_{a^t_0} \; ER_0(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega^{t+1}_0} Pr(\omega^{t+1}_0|b^t_{0,l}, a^t_0) V^{h-1}(b^{t+1}_{0,l}) \tag{5.7}
$$

where $ER_0(b^t_{0,l}, a^t_0)$ is the expected immediate reward of agent 0 and γ is the discount factor. In the context of the redefined reward function of the I-POMDP framework in this section, the expected immediate reward is obtained as:

$$ER_0(b^t_{0,l}, a^t_0) = \sum_{\mathbf{s}^t} b^t_{0,l}(s^t)\Bigg(\sum_{k=1}^{K} \sum_{\mathcal{C}^{\nu(x^t_k,a^t_0)}} Pr(\mathcal{C}^{\nu(x^t_k,a^t_0)}| \\ b^t_{0,l}(M_{1,l-1}|s^t), \ldots, b^t_{0,l}(M_{N,l-1}|s^t)) R_0(x^t_k, a^t_0, \mathcal{C}^{\nu(x^t_k,a^t_0)})\Bigg) \tag{5.8}$$

where the inner sum is over all the state factors, $\mathbf{s}^t = \langle x^t_1, \ldots, x^t_K \rangle$, and the term, $Pr(\mathcal{C}^{\nu(x^t_k,a^t_0)}|$ $b^t_{0,l}(M_{1,l-1}|s^t), \ldots, b^t_{0,l}(M_{N,l-1}|s^t))$ denotes the probability of a frame-action configuration that is context-specific to the factor x^t_k and action a^t_0. Importantly, Proposition 6 establishes that the Bellman equation above is exact.

Proposition 6 (Optimality)**.** *The Bellman equation in Eq. 5.7 provides an exact computation of the value function for the many-agent I-POMDP.*

Proof. We provide a mathematical induction based proof for proposition 6. For horizon 1 the value function may be written as:

$$V^1(b^t_{0,l}) = \max_{a^t_0} \; ER_0(b^t_{0,l}, a^t_0)$$
$$= \max_{a^t_0} \; \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{\mathbf{m}^t_{-0}} b^t_{0,l}(\mathbf{m}^t_{-0}|\mathbf{s}^t) \sum_{\mathbf{a}^t_{-0}} Pr(\mathbf{a}^t_{-0}|\mathbf{m}^t_{-0}) \sum_{k=1}^{K} R_0(x^t_k, a^t_0, \mathbf{a}^t_{-0})$$
$$= \max_{a^t_0} \; \sum_{\mathbf{s}^t} b^t_{0,l}(\mathbf{s}^t) \sum_{k=1}^{K} \Bigg\{ \sum_{\mathcal{C}^{\nu(x^t_k,a^t_0)}} Pr(\mathcal{C}^{\nu(x^t_k,a^t_0)}|b^t_{0,l}(M_{1,l-1}|\mathbf{s}^t), \ldots, b^t_{0,l}(M_{N,l-1}|\mathbf{s}^t)) \\ R_0(x^t_k, a^t_0, \mathcal{C}^{\nu(x^t_k,a^t_0)})\Bigg\}$$

Here the probability of configuration is computed using the probability of each individual agent's models given the state and the probability of its actions given the model in a similar manner as in equation 5.5. Next let's assume that for horizon $h-1$ the value function is exact.

$$V^h(b^t_{0,l}) = \max_{a^t_0} \; ER_0(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega^{t+1}_0} Pr(\omega^{t+1}_0 | b^t_{0,l}, a^t_0) V^{h-1}(b^{t+1}_{0,l})$$

The first term is the horizon one value function which is exact and the term $V^{h-1}(b^{t+1}_{0,l})$ is exact by inductive hypothesis. Finally the term $Pr(\omega^{t+1}_0 | b^t_{0,l}, a^t_0)$ is merely the normalization constant for the belief update which is also exact. As seen in derivations of equations 5.5 and 5.6, the belief update is also exact. Hence, $V^h(b^t_{0,l})$ is exact. □

5.4 Algorithms

In this section, I present an algorithm that computes the distribution over frame-action configurations using the conditional belief over the models of each individual agent and the probability of their individual actions given the model. This value is used in equations 5.5, 5.6, and 5.8. Next, I outline our simple exhaustive search based method for solving the many-agent I-POMDP defined previously. Finally, I present efficiently computable bounds on the value function of many-agent I-POMDPs and describe an efficient branch and bound algorithm that utilizes these bounds to mitigate the curse of history.

5.4.1 Computing Distribution Over Frame-Action Configurations

Algorithm 4 generalizes an algorithm by Jiang and Lleyton-Brown [32] for computing configurations over actions given mixed strategies of other agents to include frames and conditional beliefs over models of other agents. It computes the probability distribution of configurations over the frame-action neighborhood of an action given the belief over the agents' models: $Pr(\mathcal{C}^{\nu(x^{t+1}_k, a_0, \omega^{t+1}_{0,k})} | b^t_{0,l}(M_{1,l-1}|s^t), \ldots, b^t_{0,l}(M_{N,l-1}|s^t))$ and $Pr(\mathcal{C}^{\nu(x^t_k, a_0, x^{t+1}_k)} | b^t_{0,l}(M_{1,l-1}|s^t), \ldots, b^t_{0,l}(M_{N,l-1}|s^t))$ in Eq. 5.3, $Pr(\mathcal{C}^{\nu(x^{t+1}_k, a^t_j, \omega^{t+1}_{j,k})} | b^t_{0,l}(M_{1,l-1}|s^t), \ldots, b^t_{0,l}(M_{j-1,l-1}|s^t),$

$b^t_{0,l}(M_{j+1,l-1}|s^t)$, ... $b^t_{0,l}(M_{N,l-1}|s^t)$) in Eq. 5.4, and $Pr(\mathcal{C}^{\nu(x^t_k,a_0)}|b^t_{0,l}(M_{1,l-1}|s^t),\ldots,$ $b^t_{0,l}(M_{N,l-1}|s^t))$ in Eq. 5.8.

Algorithm 4 Computing $Pr(\mathcal{C}^{\nu(\cdot)}|b_{0,l}(M_{1,l-1}|s),\ldots,b_{0,l}(M_{N,l-1}|s))$

Input: $\nu(\cdot)$, $\langle b_{0,l}(M_{1,l-1}|s),\ldots,b_{0,l}(M_{N,l-1}|s)\rangle$
Output: A trie P_n representing distribution over the frame-action configurations over $\nu(\cdot)$

1: Initialize $c_0 = (0,\cdots,0)$ one value for each frame-action pair in $\nu(\cdot)$ and one for dummy action ϕ representing all other actions. Insert it into an empty trie P_0
2: Initialize $P_0[c_0] = 1.0$
3: **for** $j = 1$ to N **do**
4: Initialize P_j to be an empty trie
5: **for all** c_{j-1} from P_{j-1} **do**
6: **for all** $m_{j,l-1} \in M_{j,l-1}$ **do**
7: **for all** $a_j \in A_j$ such that $Pr(a_j|m_j) > 0$ **do**
8: $c_j = c_{j-1}$
9: **if** $\langle \hat{m_{j,l-1}}, a_j\rangle \in \nu(\cdot)$ **then**
10: $c_j[a_j] += 1$
11: **else**
12: $c_j[\phi] += 1$
13: **if** $P_j[c_j]$ does not exist **then**
14: Initialize $P_j[c_j] = 0$
15: $P_j[c_j] += P_{k-1}[c_{j-1}] \times Pr(a_j|m_{j,l-1}) \times b_0^{t,s^t,j}(m_{j,l-1})$
16: **return** P_n

Algorithm 4 successively adds the actions of each agent at a time. We utilize a trie data structure to store the probabilities of configurations. This enables efficient insertion and access of the configuration probabilities in the algorithm. We begin by initializing the configuration space for 0 agents (P_0) to contain one tuple of integers (c_0) with $|\nu| + 1$ 0s and assign its probability to be 1 (lines 1-2). Using the configurations at the previous step, we construct the configurations over the actions performed by j agents by adding 1 to a relevant element depending on the agent j's action and its frame (lines 3-15). If an action a_j performed by an agent j ascribed frame $\hat{m}_j$ is in the frame-action neighborhood $\nu(\cdot)$ the we increment its corresponding count by 1. Otherwise, it is considered as a dummy action and the count of ϕ is incremented (lines 9-12). Similarly, we update the probability

of a configuration using the probability of a_j and that of the base configuration c_{j-1} (line 15). This algorithm may be invoked multiple times for different values of $\nu(\cdot)$ as needed for computing belief update and value function.

Computational Savings

The complexity of accessing an element in a ternary search trie is $\Theta(\nu)$. The maximum number of configurations encountered at any iteration is upper bounded by total number of configurations for N agents, i.e. $\mathcal{O}((\frac{N}{|\nu^*|})^{|\nu^*|})$. The complexity of Algorithm 4 is *polynomial* in N, $\mathcal{O}(N|M_j^*||A_j^*||\nu^*|(\frac{N}{|\nu^*|})^{|\nu^*|})$ where M_j^* and A_j^* are largest sets of models and actions for any agent.

For the traditional I-POMDP belief update, the complexity of computing Eq. 5.3 is $\mathcal{O}(|S||M_j^*|^N|A_j^*|^N)$ and that for computing Eq. 5.4 is $\mathcal{O}(|S||M_j^*|^N|A_j^*|^N|\Omega_j^*|)$ where * denotes the maximum cardinality set for any agent. For a factored representation, belief update operator invokes equation 5.3 for each value of all state factors and it invokes equation 5.4 for each model of each agent j and for all values of updated states. Hence the total complexity of belief update is $\mathcal{O}(N|M_j^*||S|^2|M_j^*|^N|A_j^*|^N|\Omega_j^*|)$. The complexity of computing updated belief over state factor x^{t+1} using Eq. 5.5 is $\mathcal{O}(|S|NK|M_j^*||A_j^*||\nu^*|(\frac{N}{|\nu^*|})^{|\nu^*|})$ (recall the complexity of Algorithm 4). Similarly, the complexity of computing updated model probability using Eq. 5.6 is $\mathcal{O}((|S|N|M_j^*||A_j^*||\nu^*| + |\Omega_j^*|)(\frac{N}{|\nu^*|})^{|\nu^*|})$. These complexity terms are polynomial in N for small values of $|\nu^*|$ as opposed to exponential in N as in Eqs. 5.3 and 5.4. The overall complexity of belief update is also polynomial in N.

Complexity of computing the immediate expected reward in the absence of problem structure is $\mathcal{O}(|S|K|M_j^*|^N|A_j^*|^N)$. On the other hand, the complexity of computing expected reward using Eq. 5.8 is $\mathcal{O}(|S|KN|M_j^*||A_j^*||\nu^*|(\frac{N}{|\nu^*|})^{|\nu^*|})$, which is again polynomial in N for low values of $|\nu^*|$.

5.4.2 Many-Agent I-POMDP Solution Algorithm

We first utilize a simple method for solving the many-agent I-POMDP given an initial belief, each other agent is modeled using a finite-state controller which is included as part of the interactive state space as described in chapter 3. A reachability tree of beliefs as nodes is projected for as many steps as the horizon (using Eqs. 5.5 and 5.6) and value iteration (Eq. 5.7) is performed on the tree in a bottom up manner.

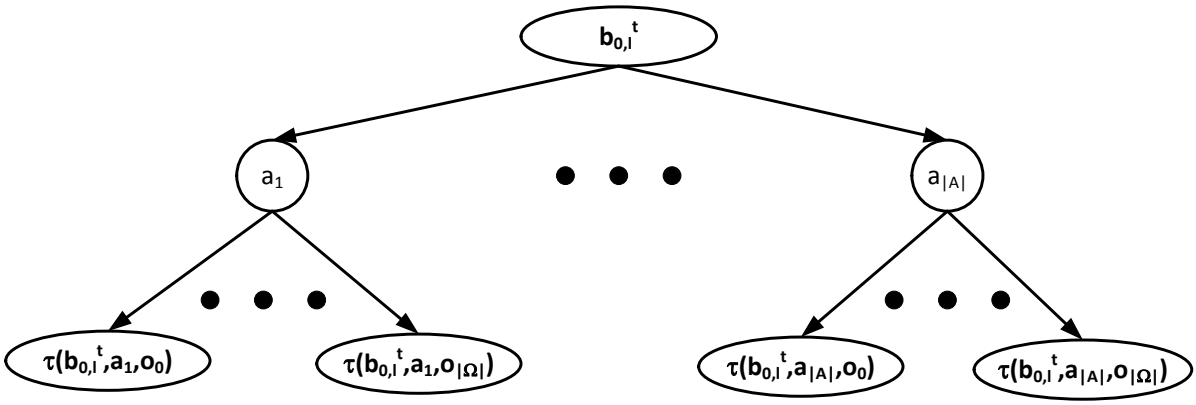


Figure 5.3: Computing beliefs reachable from a given belief in one step. One belief is reachable for each action-observation pair $\langle a_0^t, \omega_0^{t+1} \rangle$. In order to compute the horizon h reachable tree, the same process is applied repeatedly to all inner nodes. The leafs represent the horizon 1 beliefs.

Figure 5.3 illustrates computation of belief reachable from initial belief $b_{0,l}^t$ in one step. For each action a_0^t and observation ω_0^{t+1}, the updated belief $\tau(b_{0,l}^t, a_0^t, \omega_0^{t+1})$ is reachable. A horizon h reachability tree is projected by computing 1 step reachability for the root node and all inner nodes. Note that the number of beliefs in the reachability tree is $\mathcal{O}((|A_0||\Omega_0|)^h)$ which affects the computational complexity exponentially. Therefore, in order to mitigate the curse of history, we utilize a branch and bound method that utilizes novel upper and lower bounds on the value function of many-agent I-POMDPs to prune action nodes of the reachability tree without the need of exactly evaluating the subtree below them first. I derive these bounds and outline the branch and bound algorithm in this section.

5.4.3 Upper and Lower Bounds for ManyAgent I-POMDP

We adapt POMDP bounds from section 1.1.3, namely *blind policy* lower bound and *fast informed bound* (FIB) upper bound [27], to obtain quickly computable bounds for many-agent I-POMDP.

Pessimistic Blind Policy Lower Bound

In POMDP, a blind policy is one where an agent chooses the same policy irrespective of the observation it receives. In other words, it ignores the information contained in the observation. This loss of information leads to an underestimation of the expected value. We extend this concept to many-agent I-POMDPs. Next, in the interest of quicker computation, instead of iterating over configurations for any given context we take a pessimistic approach by utilizing the configuration that would lead to the least value.

Proposition 7 (Pessimistic Blind Policy)**.** *The pessimistic blind policy yields a lower bound on the value of many-agent I-POMDP.*

Proof. We begin with the horizon 1 value function and obtain a pessimistic lower bound for it.

$$
\begin{aligned}
&Q^1_{0,l}(b^t_{0,l}, a^t_0) \\
&= \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \Bigg\{ \sum_{k=1}^{K} \sum_{C^{\nu(x^t_k, a^t_0)}} Pr(C^{\nu(x^t_k,a^t_0)}|m^t_{-0}) R_0(x^t_k, a^t_0, C^{\nu(x^t_k,a^t_0)}) \Bigg\} \\
&\geq \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \Bigg\{ \sum_{k=1}^{K} \sum_{C^{\nu(x^t_k, a^t_0)}} Pr(C^{\nu(x^t_k,a^t_0)}|m^t_{-0}) \min_{\underline{C}^{\nu(x^t_k,a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k,a^t_0)}) \Bigg\} \\
&= \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \Bigg\{ \sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k,a^t_0)}) \sum_{C^{\nu(x^t_k,a^t_0)}} Pr(C^{\nu(x^t_k,a^t_0)}|m^t_{-0}) \Bigg\} \\
&= \sum_{s^t} b^t_{0,l}(s^t) \Bigg\{ \sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k,a^t_0)}) \Bigg\} \\
&= \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0)
\end{aligned}
$$

Therefore the pessimistic lower bound for horizon 1 value of many-agent I-POMDP is:

$$\underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) = \sum_{s^t} b^t_{0,l}(s^t)\left\{ \sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0)}) \right\}$$

Analogous to POMDP value function, we may define the lower bound value in vector form for each action as $\underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) = \sum_s \underline{\alpha}^1(s) \cdot b^t_{0,l}(s)$.

$$\underline{\alpha}^1(s) = \left\{ \sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0)}) \right\} \tag{5.9}$$

Let's assume that for horizon $h-1$, the pessimistic blind policy provides a lower bound to the value function and could be represented using a set of value vectors similar to equation 5.9. We derive the lower bound value vectors for horizon h. For the sake of simplicity in derivation, we temporarily forgo the changes made to accommodate the factored state representation and context specific independence. Also unlike equation 5.8 where we directly compute the probability of a configuration given the conditional distribution over the models of each agent, we write the value functions in a form where the joint models are explicitly enumerated and the probability over configuration is computed using a given joint model.

$$\begin{aligned} Q^h_{0,l}(b^t_{0,l}, a^t_0) = Q^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega^{t+1}_0} \max_{\alpha^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \\ \left\{ \sum_C Pr(C|m^t_{-0}) O_0(s^{t+1}, a^t_0, C, \omega^{t+1}_0) \right\} \left\{ \sum_C Pr(C|m^t_{-0}) T_0(s^t, a^t_0, C, s^{t+1}) \right\} \\ \sum_{m^{t+1}_{-0}} Pr(m^{t+1}_{-0}|m^t_{-0}, a^t_0, s^{t+1}) \cdot \alpha^{h-1}(s^{t+1}, m^{t+1}_{-0}) \Bigg] \end{aligned}$$

Replacing Q^1 and α^{h-1} by their respective lower bounds we have:

$$\geq \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \sum_C Pr(C|m^t_{-0}) O_0(s^{t+1}, a^t_0, C, \omega_0^{t+1}) \Bigg\} \Bigg\{ \sum_C Pr(C|m^t_{-0}) T_0(s^t, a^t_0, C, s^{t+1}) \Bigg\} \sum_{m^{t+1}_{-0}} Pr(m^{t+1}_{-0}|m^t_{-0}, a^t_0, s^{t+1}) \cdot \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

At this point, we make the blind policy modification. The agent chooses the same policy irrespective of the observations. Therefore, we may take the term maximizing over $\underline{\alpha}^{h-1}$ outside the summation over observations.

$$\geq \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{\omega_0^{t+1}} \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \sum_C Pr(C|m^t_{-0}) O_0(s^{t+1}, a^t_0, C, \omega_0^{t+1}) \Bigg\} \Bigg\{ \sum_C Pr(C|m^t_{-0}) T_0(s^t, a^t_0, C, s^{t+1}) \Bigg\} \sum_{m^{t+1}_{-0}} Pr(m^{t+1}_{-0}|m^t_{-0}, a^t_0, s^{t+1}) \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

$$= \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \sum_C Pr(C|m^t_{-0}) T_0(s^t, a^t_0, C, s^{t+1}) \Bigg\} \underline{\alpha}^{h-1}(s^{t+1}) \sum_{\omega_0^{t+1}} \Bigg\{ \sum_C Pr(C|m^t_{-0}) O_0(s^{t+1}, a^t_0, C, \omega_0^{t+1}) \Bigg\} \sum_{m^{t+1}_{-0}} Pr(m^{t+1}_{-0}|m^t_{-0}, a^t_0, s^{t+1}) \Bigg]$$

The model update probability sums to 1. Next, by interchanging the position of $\sum_{\omega_0^{t+1}}$ and the following $\sum_C$, we have:

$$= \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \sum_C Pr(C|m^t_{-0}) T_0(s^t, a^t_0, C, s^{t+1}) \Bigg\} \underline{\alpha}^{h-1}(s^{t+1}) \Bigg\{ \sum_C Pr(C|m^t_{-0}) \sum_{\omega_0^{t+1}} O_0(s^{t+1}, a^t_0, C, \omega_0^{t+1}) \Bigg\} \Bigg]$$

The nested summation after the term $\underline{\alpha}^{h-1}(s^{t+1})$ sums to 1. Therefore the above term becomes:

$$= \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \sum_{C} Pr(C|m^t_{-0}) \\ T_0(s^t, a^t_0, C, s^{t+1}) \Bigg\} \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

At this point, we may reintroduce the necessary modifications to handle factored state representation and context-specific independence:

$$= \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \prod_{k=1}^{K} \sum_{C^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} \\ Pr(C^{\nu(x^t_k, a^t_0, x^{t+1}_k)}|m^t_{-0}) T_0(x^t_k, a^t_0, C^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k) \Bigg\} \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

$$\geq \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \sum_{s^{t+1}} \Bigg\{ \prod_{k=1}^{K} \sum_{C^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} \\ Pr(C^{\nu(x^t_k, a^t_0, x^{t+1}_k)}|m^t_{-0}) \min_{\underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} T_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k) \Bigg\} \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

(Pessimistic step.)

$$= \underline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \sum_{s^{t+1}} \Bigg\{ \prod_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} T_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k) \Bigg\} \\ \cdot \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

$$= \sum_{s^t} b^t_{0,l}(s^t) \Bigg\{ \sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\} + \gamma \max_{\underline{\alpha}^{h-1}} \Bigg[\sum_{s^t} b^t_{0,l}(s^t) \\ \sum_{s^{t+1}} \Bigg\{ \prod_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} T_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k) \Bigg\} \cdot \underline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

Notice that after the *pessimistic step*, the sum over joint models is moved to the end of the equation and sums to 1. Therefore, the computation of the lower bound isn't affected by the exponential size of joint model space.

Let $\underline{\alpha}^{h-1^*}$ be the vector the maximizes the future reward of the pessimistic blind policy. Substituting it in the equation we have:

$$\underline{Q}^h_{0,l}(b^t_{0,l}, a^t_0) = \sum_{s^t} b^t_{0,l}(s^t)\Bigg\{\sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0)})\Bigg\} + \gamma\Bigg[\sum_{s^t} b^t_{0,l}(s^t)$$
$$\sum_{s^{t+1}} \Bigg\{\prod_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} T_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k)\Bigg\} \cdot \underline{\alpha}^{h-1^*}(s^{t+1})\Bigg]$$

$$\Rightarrow \underline{Q}^h_{0,l}(b^t_{0,l}, a^t_0) = \sum_{s^t} b^t_{0,l}(s^t)\Bigg[\Bigg\{\sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0)})\Bigg\} + \gamma \sum_{s^{t+1}} \Bigg\{\prod_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}}$$
$$T_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k)\Bigg\} \cdot \underline{\alpha}^{h-1^*}(s^{t+1})\Bigg] \leq Q^h_{0,l}(b^t_{0,l}, a^t_0)$$

□

Finally, the lower bound vectors for horizon h may be obtained for all actions by using lower bound vectors from horizon $h-1$ as follows:

$$\underline{\alpha}^h(s^t) = \Bigg[\Bigg\{\sum_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0)})\Bigg\} + \gamma \sum_{s^{t+1}} \Bigg\{\prod_{k=1}^{K} \min_{\underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}}$$
$$T_0(x^t_k, a^t_0, \underline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k)\Bigg\} \cdot \underline{\alpha}^{h-1}(s^{t+1})\Bigg] \tag{5.10}$$

Note that the vectors computed for lower bounds are over the physical states of the environment and not over the interactive state space. Hence, they are immune to the *curse of many agents*.

Given agent 0's belief $b^t_{0,l}$, the lower bound on the horizon h value function could be obtained as:

$$\underline{V}^h_{0,l}(b^t_{0,l}) = \max_{a^t_0} \underline{Q}_{0,l}(b^t_{0,l}, a^t_0) \tag{5.11}$$

Optimistic Fast Informed Bound Upper Bound

Fast informed bound [26] [27] provide a tighter upper bound to the POMDP value function than is obtained by using the MDP based method. One may imagine the FIB value to be obtained when the initial state of the environment is perfectly observable but in the subsequent steps the states become partially observable. Yet again in our adaptation to many-agent I-POMDPs, we utilize optimistic values of configurations in the interest of faster computation while maintaining the upper bound property.

Proposition 8 (Optimistic Fast Informed Bound)**.** *The optimistic fast informed bound gives an upper bound to many-agent I-POMDP value function.*

Proof. We begin by computing optimistic upper bounds on horizon 1 Q-value for each action a_0 as follows:

$$
\begin{aligned}
&Q^1_{0,l}(b_{0,l}, a^t_0) \\
&= \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \Bigg\{ \sum_{k=1}^{K} \sum_{C^{\nu(x^t_k, a^t_0)}} Pr(C^{\nu(x^t_k, a^t_0)}|m^t_{-0}) R_0(x^t_k, a^t_0, C^{\nu(x^t_k, a^t_0)}) \Bigg\} \\
&\leq \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \Bigg\{ \sum_{k=1}^{K} \sum_{C^{\nu(x^t_k, a^t_0)}} Pr(C^{\nu(x^t_k, a^t_0)}|m^t_{-0}) \max_{\overline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\} \\
&= \sum_{s^t} b^t_{0,l}(s^t) \sum_{m^t_{-0}} b^t_{0,l}(m^t_{-0}|s^t) \Bigg\{ \sum_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\} \\
&= \sum_{s^t} b^t_{0,l}(s^t) \Bigg\{ \sum_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\} \\
&= \overline{Q}^1_{0,l}(b_{0,l}, a^t_0)
\end{aligned}
$$

Therefore optimistic upper bound for horizon 1 value may be obtained as follows:

$$
\overline{Q}^1_{0,l}(b_{0,l}, a^t_0) = \sum_{s^t} b^t_{0,l}(s^t) \Bigg\{ \sum_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\}
$$

These upper bounds may be represented as a set of vectors one for each action a_0.

$$\overline{\alpha}^1(s) = \left\{ \sum_{k=1}^{K} \max_{\overline{C}^{\nu(x_k^t, a_0^t)}} R_0(x_k^t, a_0^t, \overline{C}^{\nu(x_k^t, a_0^t)}) \right\}$$

Next we derive the upper bound on horizon h value function.

$$Q_{0,l}^h(b_{0,l}^t, a_0^t) = Q_{0,l}^1(b_{0,l}^t, a_0^t) + \gamma \sum_{\omega_0^{t+1}} \max_{\alpha^{h-1}} \Bigg[\sum_{s^t} b_{0,l}^t(s^t) \sum_{m_{-0}^t} b_{0,l}^t(m_{-0}^t|s^t) \sum_{s^{t+1}}$$
$$\left\{ \prod_{k=1}^{K} \sum_{C^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}} Pr(C^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}|m_{-0}^t) O_0(x_k^{t+1}, a_0^t, C^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}, \omega_{0,k}^{t+1}) \right\}$$
$$\left\{ \prod_{k=1}^{K} \sum_{C^{\nu(x_k^t, a_0^t, x_k^{t+1})}} Pr(C^{\nu(x_k^t, a_0^t, x_k^{t+1})}|m_{-0}^t) T_0(x_k^t, a_0^t, C^{\nu(x_k^t, a_0^t, x_k^{t+1})}, x_k^{t+1}) \right\} \sum_{m_{-0}^{t+1}}$$
$$Pr(m_{-0}^{t+1}|m_{-0}^t, a_0^t, s^{t+1}) \cdot \alpha^{h-1}(s^{t+1}, m_{-0}^{t+1}) \Bigg]$$

By replacing $Q_{0,l}^1$ and α^{h-1} with corresponding upper bound value and inserting fast informed bound approximation by moving max over $\overline{\alpha}^{h-1}$ inside the summation over s^t we have:

$$\leq \overline{Q}_{0,l}^1(b_{0,l}^t, a_0^t) + \gamma \sum_{\omega_0^{t+1}} \sum_{s^t} b_{0,l}^t(s^t) \max_{\overline{\alpha}^{h-1}} \Bigg[\sum_{m_{-0}^t} b_{0,l}^t(m_{-0}^t|s^t) \sum_{s^{t+1}} \left\{ \prod_{k=1}^{K} \sum_{C^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}} \right.$$
$$\left. Pr(C^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}|m_{-0}^t) O_0(x_k^{t+1}, a_0^t, C^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}, \omega_{0,k}^{t+1}) \right\}$$
$$\left\{ \prod_{k=1}^{K} \sum_{C^{\nu(x_k^t, a_0^t, x_k^{t+1})}} Pr(C^{\nu(x_k^t, a_0^t, x_k^{t+1})}|m_{-0}^t) T_0(x_k^t, a_0^t, C^{\nu(x_k^t, a_0^t, x_k^{t+1})}, x_k^{t+1}) \right\}$$
$$\sum_{m_{-0}^{t+1}} Pr(m_{-0}^{t+1}|m_{-0}^t, a_0^t, s^{t+1}) \overline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

Utilizing the optimistic values of the observation and transition probabilities we have:

$$\leq \overline{Q}_{0,l}^1(b_{0,l}^t, a_0^t) + \gamma \sum_{\omega_0^{t+1}} \sum_{s^t} b_{0,l}^t(s^t) \max_{\overline{\alpha}^{h-1}} \Bigg[\sum_{m_{-0}^t} b_{0,l}^t(m_{-0}^t|s^t) \sum_{s^{t+1}} \left\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}} O_0(x_k^{t+1}, a_0^t, \right.$$
$$\left. \overline{C}^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}, \omega_{0,k}^{t+1}) \right\} \left\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}} T_0(x_k^t, a_0^t, \overline{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}, x_k^{t+1}) \right\} \overline{\alpha}^{h-1}(s^{t+1}) \Bigg]$$

Now, summation over the joint models may be moved to the end and marginalized. Therefore, the optimistic fast informed bound is:

$$\begin{aligned}
\overline{Q}^h_{0,l}(b^t_{0,l}, a^t_0) = \sum_{s^t} b^t_{0,l}(s^t) \Bigg[\Bigg\{ \sum_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\} + \gamma \sum_{\omega_0^{t+1}} \max_{\overline{\alpha}^{h-1}} \Bigg[\sum_{s^{t+1}} \\
\Bigg\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x^{t+1}_k, a^t_0, \omega^{t+1}_{0,k})}} O_0(x^{t+1}_k, a^t_0, \overline{C}^{\nu(x^{t+1}_k, a^t_0, \omega^{t+1}_{0,k})}, \omega^{t+1}_{0,k}) \Bigg\} \Bigg\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} \\
T_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k) \Bigg\} \overline{\alpha}^{h-1}(s^{t+1}) \Bigg] \Bigg] \\
\geq Q^h_{0,l}(b^t_{0,l}, a^t_0)
\end{aligned}$$

□

Finally, analogous to POMDP fast informed bound, irrespective of the horizon, we would obtain one unique value vector for each action a^t_0 which may be computed as follows:

$$\begin{aligned}
\overline{\alpha}^h_{0,l}(s^t) = \Bigg\{ \sum_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0)}} R_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0)}) \Bigg\} + \gamma \sum_{\omega_0^{t+1}} \max_{\overline{\alpha}^{h-1}} \Bigg[\sum_{s^{t+1}} \\
\Bigg\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x^{t+1}_k, a^t_0, \omega^{t+1}_{0,k})}} O_0(x^{t+1}_k, a^t_0, \overline{C}^{\nu(x^{t+1}_k, a^t_0, \omega^{t+1}_{0,k})}, \omega^{t+1}_{0,k}) \Bigg\} \Bigg\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}} \\
T_0(x^t_k, a^t_0, \overline{C}^{\nu(x^t_k, a^t_0, x^{t+1}_k)}, x^{t+1}_k) \Bigg\} \overline{\alpha}^{h-1}(s^{t+1}) \Bigg]
\end{aligned} \tag{5.12}$$

The upper bound on the value function is obtained as:

$$\overline{V}_{0,l}(b^t_{0,l}) = \max_{a^t_0} \overline{Q}_{0,l}(b^t_{0,l}, a^t_0) \tag{5.13}$$

As in the case of lower bound, the vectors are over the physical states and we don't need to utilize joint models or joint actions for their computation. Hence, these bounds aren't affected by *curse of many agents*.

5.4.4 Updating Upper and Lower Bounds

Let's compute $\overline{\dot{Q}}^h_{0,l}$ as the value by using exact I-POMDP value for 1 step followed by upper bound value for $h-1$ steps. Formally, $\overline{\dot{Q}}^h_{0,l} = Q^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) \max_{a_0^{t+1}} \overline{Q}^{h-1}(b^{t+1}_{0,l}, a^{t+1}_0)$.

Proposition 9 (Updating Upper Bound)**.** *$\overline{\dot{Q}}^h_{0,l}$ is a tighter upper bound on $Q^h_{0,l}$ than $\overline{Q}^h_{0,l}$.*

Proof. First I prove that $\overline{\dot{Q}}^h_{0,l}$ is an upper bound of $Q^h_{0,l}$.

$$
\begin{aligned}
&Q^h_{0,l}(b^t_{0,l}, a^t_0) = Q^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) V^{h-1}(b^{t+1}_{0,l}) \\
&= Q^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) \max_{a_0^{t+1}} Q^{h-1}(b^{t+1}_{0,l}, a^{t+1}_0) \\
&\leq Q^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) \max_{a_0^{t+1}} \overline{Q}^{h-1}(b^{t+1}_{0,l}, a^{t+1}_0) \\
&= \overline{\dot{Q}}^h_{0,l}(b^t_{0,l}, a^t_0)
\end{aligned}
$$

Next I show that $\overline{\dot{Q}}^h_{0,l}$ is tighter than $\overline{Q}^h_{0,l}$.

$$
\begin{aligned}
&\overline{\dot{Q}}^h_{0,l}(b^t_{0,l}, a^t_0) \\
&= Q^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) \max_{a_0^{t+1}} \overline{Q}^{h-1}(b^{t+1}_{0,l}, a^{t+1}_0) \\
&\leq \overline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) \max_{a_0^{t+1}} \overline{Q}^{h-1}(b^{t+1}_{0,l}, a^{t+1}_0) \\
&= \overline{Q}^1_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1}|b^t_{0,l}, a^t_0) \max_{\overline{\alpha}^{h-1}} \tau(b^t_{0,l}, a^t_0, \omega_0^{t+1})(S) \cdot \overline{\alpha}^{h-1}
\end{aligned}
$$

where $\tau(b^t_{0,l}, a^t_0, \omega_0^{t+1})(S)$ represents the portion of the updated belief that is over the physical states.

Substituting the belief update with equations 5.5 and 5.6 followed by the optimistic upper bound estimate, we get:

$$\leq \overline{Q}^{1}_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} \max_{\overline{\alpha}^{h-1}} \left[\sum_{s^{t+1}} \left\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}} O_0(x_k^{t+1}, a_0^t, \overline{C}^{\nu(x_k^{t+1}, a_0^t, \omega_{0,k}^{t+1})}, \omega_{0,k}^{t+1}) \right\} \right.$$
$$\left. \left\{ \prod_{k=1}^{K} \max_{\overline{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}} T_0(x_k^t, a_0^t, \overline{C}^{\nu(x_k^t, a_0^t, x_k^{t+1})}, x_k^{t+1}) \right\} \overline{\alpha}^{h-1}(s^{t+1}) \right] \right]$$
$$= \overline{Q}^{h}_{0,l}(b^t_{0,l}, a^t_0)$$

□

Similarly, we define $\underline{\dot{Q}}^{h}_{0,l} = Q^{1}_{0,l}(b^t_{0,l}, a^t_0) + \gamma \sum_{\omega_0^{t+1}} Pr(\omega_0^{t+1} | b^t_{0,l}, a^t_0) \max_{a_0^{t+1}} \underline{Q}^{h-1}(b^{t+1}_{0,l}, a_0^{t+1})$.

Proposition 10 (Updating Lower Bound)**.** *$\underline{\dot{Q}}^{h}_{0,l}$ is a tighter upper bound on $Q^{h}_{0,l}$ than $\underline{Q}^{h}_{0,l}$.*

Proof. The proof proceeds in a similar manner as the proof or proposition 9. □

Since the I-POMDP value function is *isotonic*, successively updating the bounds at non-root node of the reachability tree using $\overline{\dot{Q}}$ and $\underline{\dot{Q}}$ operator, makes the bound at root increasingly tighter.

5.4.5 Efficient Solution using Branch and Bound Based Method

Despite its remarkable effect on the curbing the curse of many agents, the solution approach of computing the entire reachability tree and performing value iteration bottom up is still plagued by the curse of history. The number of nodes in the reachability tree is exponential in the number of horizons and the dynamic program described in algorithm 4 must be invoked for each context at each belief to perform belief update and value iteration. This is a cumbersome and time-consuming operation. Hence, to address this issue, we utilize the bounds presented in section 5.4.3 to prune portions of reachability tree that are guaranteed to be suboptimal. Note that pruning an inner node of the reachability tree without expanding

the subtree rooted at it saves us the cost of computing the value of all the nodes in the subtree. We update the bounds according to the process described in section 5.4.4.

Next, I present our approach for computing the exact horizon h value of a belief $b^t_{0,l}$ using *branch and bound* method. We begin by initializing the root node of the reachability tree using algorithm 5. The algorithm takes the belief and horizon as an input. Next, for each action of agent 0, a_0, the algorithm initializes the upper and lower bound values for policies with a_0 as the first action ($\overline{Q}(b^t_{0,l}, a_0)$ and $\underline{Q}(b^t_{0,l}, a_0)$ respectively) as described in section 5.4.3. The upper and lower bound on the overall value of the node is initialized to be maximum values of these bounds as $\overline{V} = \max_{a_0} \overline{Q}(b^t_{0,l}, a_0)$ and $\underline{V} = \max_{a_0} \underline{Q}(b^t_{0,l}, a_0)$ respectively. All actions for which the upper bound on the Q-value is smaller than the lower bound of the overall value function are guaranteed to produce sub-optimal policy. Hence they may be pruned without affecting the solution quality.

Algorithm 5 InitializeNode

Input: Current belief ($b^t_{0,l}$), horizon (h)
Output: A node (n) in the reachability tree with initial value of upper bound and lower bound values for all actions

1: Initialize a new node n and set its belief to be $b^t_{0,l}$ and horizon to h
2: **for all** $a_0 \in A_0$ **do**
3: $\quad UB[a_0] \leftarrow \overline{Q}^h(b^t_{0,l}, a_0)$ (Initialize to optimistic FIB value)
4: $\quad LB[a_0] \leftarrow \underline{Q}^h(b^t_{0,l}, a_0)$ (Initialize to pessimistic blind policy value)
5: $\overline{V} = \max_{a_0} UB[a_0]$
6: $\underline{V} = \max_{a_0} LB[a_0]$
7: **for all** a'_0 **do**
8: $\quad$ **if** $UB(a'_0) < \underline{V}$ **then**
9: $\quad\quad$ Prune reachability subtree with action a'_0 at root node

Next we recursively tighten the upper and lower bounds on the value of the root node using the branch and bound approach outlined in algorithms 6, 7, and 8. Particularly, we repeatedly run algorithm 6 for the root node of the reachability tree. The termination condition is reached when the upper and lower bound have the same value.

Algorithm 6 UpdateBounds

Input: A node (n) of the reachability tree

Output: Boolean value representing if the node has been exactly solved

1: **if** $\underline{V} = \overline{V}$ **then**
2: return **true**
3: $a_0 \leftarrow$ Pick action according to a heuristic
4: **if** $\neg$branched$[a_0]$ **then**
5: Branch(n, a_0)
6: branched$[a_0] \leftarrow$ **true**
7: **else**
8: **if** $h > 1$ **then**
9: **for all** ω_0 **do**
10: UpdateBounds(NextNode$[a_0][\omega_0]$)
11: Bound(n, a_0)
12: $\overline{V} = \max_{a_0} UB(a_0)$
13: $\underline{V} = \max_{a_0} LB(a_0)$
14: **for all** a'_0 **do**
15: **if** $UB(a'_0) < \underline{V}$ **then**
16: Prune reachability subtree with action a'_0 at root node
17: return **false**

The algorithm partially unfolds the reachability tree for some action a_0 and computes tighter bounds on the Q-value based on the unfolded reachability tree. We base the choice of action on which to expand the reachability tree on a heuristic function (line 3 in algorithm 6). Both the upper bound and lower bound could provide equally good guesses as to which action is likely to be optimal given the current belief. Hence, the heuristic function may pick an action that maximizes either bound (i.e. $arg\max_{a_0} UB[a_0]$ or $arg\max_{a_0} LB[a_0]$). Depending on the choice of the heuristic function used, algorithm 6 tightens at least one of the bounds. However, choosing the action based on upper bound has one major advantage. As we update the upper bound for an action a_0, we may discover its suboptimality whenever its value drops below the upper bound value for some other action. On the other hand updating bounds based on lower bound will always pick the same action, whose value will increase monotonically, without the possibility of discovering its suboptimality till the entire reachability subtree below that action's node has been expanded and evaluated. Therefore, our heuristic function picks the action that maximizes the value of the upper bound ($arg\max_{a_0} UB[a_0]$). This heuristics is also referred to as IE-Max heuristics [34] and is utilized in a number of heuristics based POMDP algorithms [65] [63] [40].

Algorithm 7 Branch

Input: A node (n) of the reachability tree and action (a_0)

1: **if** $h > 1$ **then**
2: **for all** ω_0 **do**
3: NextNode$[a_0][\omega_0] \leftarrow$ InitializeNode$(\tau(b_{0,l}, a_0, \omega_0),\ h-1)$

Our approach partially expands the reachability tree for the action chosen by the heuristic function one horizon at a time. This is referred to as the *branch step* and is detailed in algorithm 7. For any given node and action, the *branch function* needs to be invoked only once. For the non-leaf nodes, the *branch function* computes the updated belief for all possible observations and initializes their respective nodes in the reachability tree using equation 5.

Algorithm 8 Bound

Input: A node (n) of the reachability tree and action (a_0)

1: $ER_0 \leftarrow$ Compute expected immediate reward for $b_{0,l}$
2: $UB[a_0] \leftarrow ER_0$
3: $LB[a_0] \leftarrow ER_0$
4: **if** $h > 1$ **then**
5: **for all** ω_0 **do**
6: $UB[a_0] \leftarrow \gamma\ Pr(\omega_0|b_{0,l}, a_0) \times \text{NextNode}[a_0][\omega_0].\overline{V}$
7: $LB[a_0] \leftarrow \gamma\ Pr(\omega_0|b_{0,l}, a_0) \times \text{NextNode}[a_0][\omega_0].\underline{V}$

If the node has already been branched for the chosen action, we make a recursive call to update the bounds of each updated belief.

Next, the *bound step* described in algorithm 8 tightens its upper and lower bound for the Q-value of the chosen action using the method outlined in section 5.4.4. The overall value function of the node are computed using the updated Q-value bounds. Although, for the action chosen according to the IE-max algorithm, only the upper bound is guaranteed to be updated. If the action chosen also happens to be the one that maximizes the lower bound value, the lower bound value is updated as well.

After each step of updating the bounds, we prune all actions whose upper bound values have become smaller than the (possibly) updated lower bound. Note that once a subtree has been fully expanded for an action a_0 and all its subtrees have been exactly evaluated, the upper and lower bound value for a_0 are the same (i.e. $UB[a_0] = LB[a_0]$). At this point, either the lower bound value for a_0 is greater than upper bound value for all other actions or there exists some other action a_0' such that its upper bound value is greater than $UB[a_0]$ (i.e. either $LB[a_0] \geq UB[a_0']$, $\forall a_0'$ or $\exists a_0'$ such that $UB[a_0'] > UB[a_0]$). In the first case, we may conclude that a_0 is the optimal action and its corresponding value is the exact solution

for the subtree. In the latter case, the heuristics will pick a'_0 the next time algorithm 6 is invoked for the node.

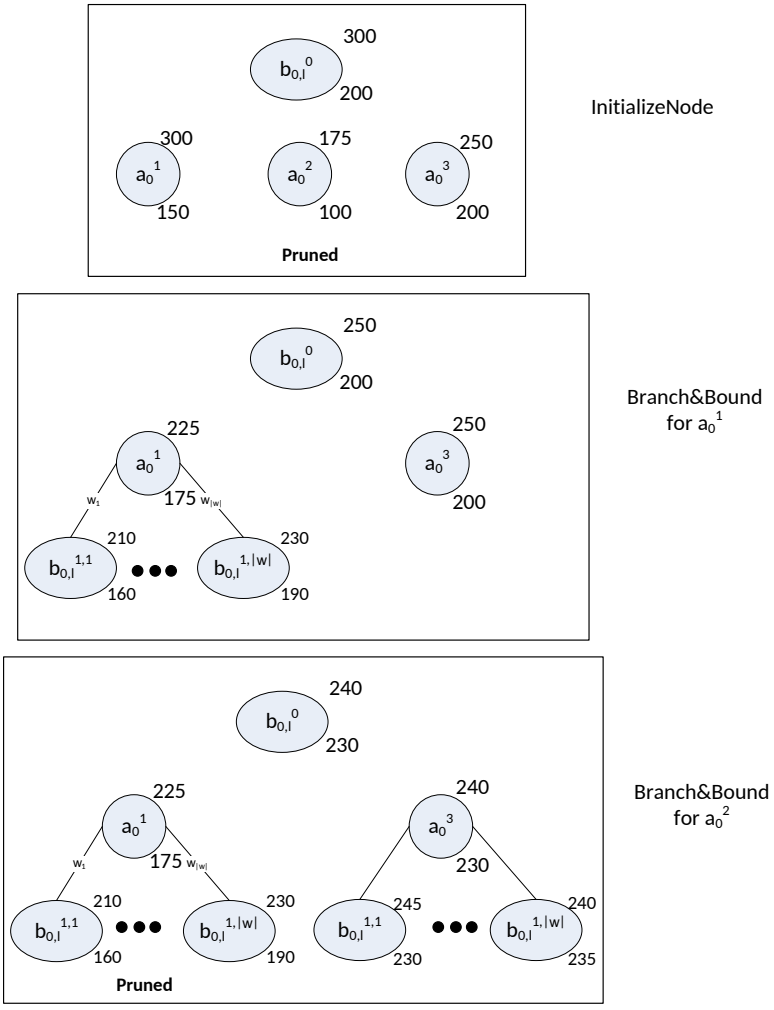


Figure 5.4: An illustration of the branch and bound algorithm performed on an example problem with 3 actions. Note the numbers beside the action nodes are the upper and lower bounds on Q-value for the action. Similarly, the number beside the belief nodes are the upper and lower bounds on the value of the belief. Note that the upper and lower bounds on the value of the belief are the maximum upper and lower bound on the Q-value for any action.

Figure 5.4 illustrates the branch and bound approach for an example problem domain. For simplicity of representation, I consider a domain with 3 actions for agent 0. In the first step, we initialize the node for the initial belief, $b^0_{0,l}$. In this step, we compute the upper and lower bounds on the Q-value for each action. The bounds on the value of $b^0_{0,l}$ are the

maximum values if upper and lower bound for any action (300 and 200 respectively). Next, notice that the upper bound value for action a_0^2 is less than the lower bound on the value of $b_{0,l}^0$. Hence space of policy trees with action a_0^2 at root node could be pruned without further expanding the reachability tree below it.

In the second step, we pick action a_0^1 using the IE-max heuristic function (notice that action a_0^1 has the maximum value of the upper bound). Next, the *branch* step computes the updated belief for a_0^1 and all observations and initializes the value of the nodes reachable in 1 step. This is followed by the *bound* step that uses the value of upper and lower bounds on the value of children node to update the Q-value for initial belief and action a_0^1.

Similarly, the bounds are updated for action a_0^3 in the third step. At this point, notice that the value of lower bound on the value of $b_{0,l}^0$ (230) is greater than the value of the upper bound on the Q-value for a_0^1 (225). Hence the entire space of policy trees with action a_0^1 could be pruned without further exploring it. The algorithm continues by expanding remaining beliefs at the next time step and running branch and bound till the values of the upper and lower bounds for $b_{0,l}^0$ converge.

5.5 Experiments

We first implemented the simple and systematic I-POMDP solving technique that computes reachable beliefs over the finite horizon and then calculates the optimal value at the root node using the Bellman equation for the Many-Agents I-POMDP framework in a bottom up fashion starting at the leaf nodes. We evaluate its performance in the aforementioned non-cooperative policing protest scenario ($|S| = 27, |A_0| = 9, |A_j| = 4, |O_j| = 8, |O_i| = 8$). We model the other agents as POMDPs and solve them using bounded policy iteration [51], representing the models as finite state controllers. This representation enables us to have a compact model space. We set the maximum planning horizon to 5 throughout the ex-

periments. The frame-action hypergraphs are encoded into the transition, observation and reward functions of the Many-Agent I-POMDP (Fig. 5.5). All computations are carried out on a RHEL platform with 2.80 GHz processor and 4 GB memory.

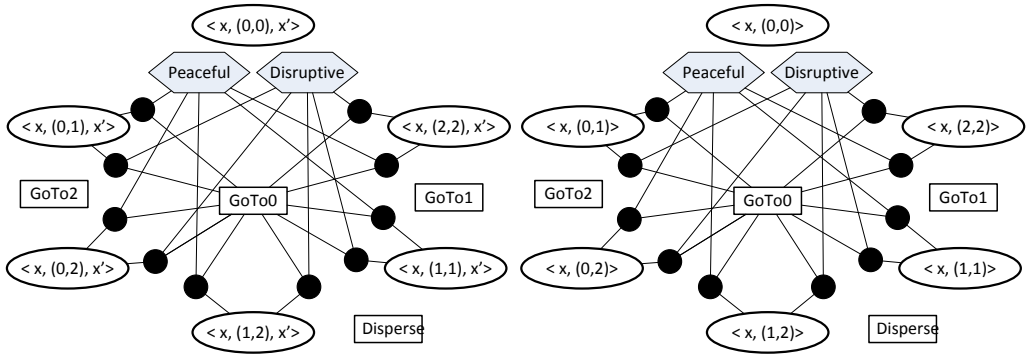


Figure 5.5: A compact levi graph representation of a the *Policing Protest* scenario as a frame-action hypergraph for (a) the transition function, and (b) the reward function for site 0. The variables x and x' represent the start and end intensity of the protest at location 0 and the action shows the location of the two police troops. Since two police troops are sufficient to deescalate any protest, the contexts in which both troops are at location 0 are independent of the actions of other agents. All other contexts depend only on the agents belonging to either frame choosing to protest at site 0.

To evaluate the computational gain obtained by exploiting problem structures, we implemented a solution algorithm similar to the one described earlier that does not exploit any problem structure. A comparison of the Many-Agent I-POMDP with the original I-POMDP yields two important results: (i) When there are few other agents, the Many-Agent I-POMDP provides exactly the same solution as the original I-POMDP but with reduced running times by exploiting the problem structure. (ii) Many-Agent I-POMDP scales to larger agent populations, from 100 to 1,000+, and the new framework delivers promising results within reasonable time.

In the first set of experiments, we consider up to 5 protesters of different frames. As shown in table 5.1, both the traditional and the Many-Agent I-POMDP produce policies with the same expected value. However, as Many-Agent I-POMDPs losslessly compress joint

protesters	H	I-POMDP	Many-Agent	Exp. Value
2	2	1 s	0.55 s	77.42
	3	19 s	17 s	222.42
3	2	3 s	0.56 s	77.34
	3	38 s	17 s	222.32
4	2	39 s	0.57 s	76.96
	3	223 s	17 s	221.87
5	2	603 s	0.60 s	76.88
	3	2,480 s	18 s	221.77

Table 5.1: Comparison between traditional I-POMDP and Many-Agent I-POMDP both following same solution approach of computing the reachability tree and performing backup in a bottom up fashion. Notice that the rate of change of time taken with number of agents is much slower for Many-Agents I-POMDP.

actions to configurations, the Many-Agent I-POMDP requires much less running time. Also, note that the rate of increase in runtime with increase in number of agents is much slower for Many-Agent I-POMDPs. This result suggests that Many-Agent I-POMDPs may scale to significantly larger number of agents.

In the second set of experiments, we test settings involving many more agents. The traditional I-POMDP doesn't not scale to such settings. Hence, we first scale up the exact solution using Many-Agent I-POMDP. This allows us to solve problems involving up to a few hundred other agents. Although the exploitation of the problem structures reduces the curse of many agents that plagues I-POMDPs, the curse of history is unaffected by such approaches. To mitigate the curse of history we implement the more efficient branch and bound based solution technique presented in section 5.4.5. We evaluate our algorithm on two problem domains (i) the non-cooperative policing domain and (ii) a gaming scenario based on a popular mobile game. We describe the second problem domain next.

5.5.1 Gaming Problem Domain

We base our second problem domain on the popular mobile game *Clash of Clans* (`http://clashofclans.com`) which is a massively multiplayer online (MMO) strategy game. The game has two aspects, the first is to gather resources through various activities (including raiding settlements of other players using own armies) and the second is to build a strong settlement and to defend own resources against the invasion of the other players' armies. For our purpose, we focus on the latter aspect of the game.

The objective of the subject agent (agent 0) is to defend it resources against the invasion by the armies of the other player. To do so it may erect a boundary walls in the four cardinal directions around its settlement and install cannons along side each wall that may ward off the attack by invading armies. We illustrate the setting for the game using a screen shot of the game shown in figure 5.6.

Figure 5.6: A screen shot of the game Clash of Clans. The image shows a simple settlement in which the resources are stored in the central structure which is surrounded by walls on four sides. The walls are guarded using cannons which are situated on the outside of the walls.

The physical state is factored into four factors, one for each direction. The state of each factor represents the defense status in the respective cardinal direction, i.e. is it completely

unprotected in that direction, just *walled*, or is it walled and *guarded* using canons. Each value representing increasing level of protection. Therefore $|S| = 81$ (3 values for each cardinal direction). At any time step, the agent may fortify its defenses on any one cardinal direction ($|A_0| = 4$) and receive observation providing information about the direction with weakest defense status ($|\Omega_0| = 4$). The action of fortifying defenses in any direction raises the level of protection from *unprotected* to *walled* and from *walled* to *guarded* with a certain probability. When attacked by an opponent's army, the subject agent may suffer losses and its defenses may be weakened or even completely destroyed.

The army of the opponent consists of N other agents that may have one of two frames: *tier 1* attacker or *tier 2* attacker. The game defines tier 1 to be a class of weaker soldiers (consisting of barbarians, archers, and goblins) that are injured easily by the canons but could recuperate quickly as well. Tier 2 attackers (consisting of giants and wizards) are more resilient to attack by the canons and cause more damage than the tier 1 attackers. But when injured, they recover slowly. The other agents may attack from one of the four cardinal direction or they may recover when injured($|A_j| = 5$). A common strategy while attacking a settlement is to attack from one of the sides with least protection. Once an agent picks a side to attack from, it continues to attack from the same side until it has been injured by a canon. When injured the agent retreats to recover. Once it has recovered, it may pick another side to attack from. The state of defenses in any direction depends on the the action of the subject agent and the number of agents of each type attacking from that direction.

5.5.2 Performance of Many-Agent I-POMDPs for Problems with Large Agent Population

For settings consisting of many other interacting agents, we demonstrate the efficiency of Many-Agent I-POMDPs on the two aforementioned problem domains. We show that while

the simpler exhaustive approach of computing all reachable beliefs could be used to solve problems involving hundreds of agents, significant savings and scalability could be obtained by utilizing the branch and bound method presented in section 5.4.5. We refer to the two methods as *Exhaustive* method and *Branch&Bound* method respectively.

We present the results for the policing problem in Table 5.2. We limit the duration of the solution algorithm to 6 hours. Notice that while the exhaustive method can solve 1000 agents for 3 horizons in under 2 hours, it unable scale to longer horizons or more agents. On the other hand the branch and bound method consistently outperforms the exhaustive method and is able to solve 2000 agent problem for 4 horizons within the time limit. Also note that both approaches are exact and yield the same value.

Table 5.3 compares the performance of the exhaustive mentod and the branch and bound method on the gaming problem domain. Yet again, the branch and bound method outperforms the exhaustive method consistently. Using the branch and bound method, we are able to solve problems involving up to 2000 attackers for 3 horizon. Whereas the exhaustive method only scales to 1000 agent problem for 3 horizons. The expected rewards are negative because we focus only on the aspect of game regarding defending the resources. Therefore, while it is possible for agent to incurr some cost for fortifying its defenses and fending off enemy's attack, we do not model attacking other agents' settlements or other activity that could yield positive rewards.

5.6 Discussion

The key contribution of the Many-Agent I-POMDP is its scalability beyond 1,000 agents by exploiting problem structures. We formalize widely existing problem structures – frame-action anonymity and context-specific independence – and encode it as frame-action hypergraphs. Other real-world examples exhibiting such problem structure are found in economics

# protesters	H	Exhaustive	Branch&Bound	Exp. Value
20	3	149 s	7 s	275.125
	4	2816 s	27 s	384.468
	5	—	86 s	482.877
50	3	157 s	8 s	274.901
	4	3062 s	33 s	384.228
	5	—	118 s	482.622
100	3	193 s	10 s	274.41
	4	3651 s	44 s	383.724
	5	—	187 s	482.107
200	3	350 s	19 s	273.384
	4	6217 s	108 s	382.664
	5	—	416 s	481.017
500	3	1137 s	73 s	270.409
	4	18087 s	285 s	379.563
	5	—	1462 s	477.822
1000	3	6782 s	465 s	265.022
	4	—	2135 s	373.958
1500	3	—	1851 s	259.743
	4	—	9858 s	368.44
2000	3	—	4189 s	254.9
	4	—	21534 s	363.348

Table 5.2: Comparison between the performance of the *Exhaustive* and *Branch&Bound* methods for the policing problem.

# Attackers	H	Exhaustive	Branch&Bound	Exp. Value
20	3	4 s	<1 s	-3.25144
	4	33 s	2 s	-3.99474
	5	228 s	7 s	-4.66932
50	3	6 s	1 s	-3.30767
	4	45 s	3 s	-4.0937
	5	304	9 s	-4.82919
100	3	14 s	2 s	-3.43564
	4	97 s	7 s	-4.30115
	5	648	27 s	-5.13859
200	3	56 s	9 s	-3.65139
	4	445 s	53 s	-4.66468
	5	3359	293 s	-5.70733
500	3	349 s	93 s	-4.62299
	4	3065 s	818 s	-6.22968
	5	—	4776 s	-8.00939
1000	3	1897 s	1190 s	-6.21816
	4	—	14087 s	-8.7851
1500	3	—	5043 s	-7.89312
2000	3	—	11089 s	-8.87943

Table 5.3: Comparison between the performance of the *Exhaustive* and *Branch&Bound* methods for the gaming problem.

where the value of an asset depends on the number of agents vying to acquire it and their financial standing (frame), in real estate where the value of a property depends on its demand, the valuations of neighboring properties as well as the economic status of the neighbors because an upscale neighborhood is desirable. Compared to the previous best approach [69], which scales to an extension of the simple tiger problem involving 5 agents only, the presented framework is far more scalable in terms of number of agents.

One drawback of Many-Agent I-POMDP is that it depends entirely on the prior belief of the subject agent. The updated beliefs are necessary at each step to compute distribution over the configurations (Algorithm 4) and to evaluate the value of a policy subtree which is in turn used to compute the value of the root node. This makes it unsuitable for settings requiring a general solution in the form of policies and corresponding value vectors. In this regard, some other structures may be exploited.

Our future work includes exploring other types of problem structures and developing approximation algorithms for I-POMDP. An integration with existing multiagent simulation platforms to illustrate the behavior of agent populations may be interesting.

Chapter 6

Conclusion

The ability to make rational decisions is a defining characteristic of human beings. From an evolutionary perspective, natural selection favors individuals who could ensure their own existence and that of their progeny by any means. Therefore, we may say that we are evolutionarily driven to be self-interested beings. In the case of human beings (as in the case of numerous other species), the best strategy for survival is to form cooperative societies primarily to ensure a continued source of sustenance (such as through agriculture) and to fend off other dangers to individuals. Thus, we may conclude that cooperative behavior arose from the self-interest of individual agents.

I-POMDPs model such self-interested behavior for individual agents. While it is easy to see the application of I-POMDPs in competitive settings where an agent may need to outsmart all other competing agents, it is also important to realize its importance in settings requiring agent cooperation. While Dec-POMDPs are ideal for settings where a team of agents needs to be deployed, their utility is limited in settings where a group of heterogeneous agents (agents developed and deployed by various sources) must be used. Heterogeneous agents may not share a common reward or common prior. For example, consider a search and rescue scenario where multiple rescue robots deployed by various medical teams must

carry out their own operation. In doing so, an agent may realize that in order to complete a sub-task, the best strategy is to cooperate with other interacting agents in the environment. Such forms of dynamic cooperation may emerge when the agents are modeled as I-POMDPs.

I-POMDPs capture a crucial aspect of multiagent decision making that we encounter in everyday life. Despite their theoretical significance, their application in real world domains is precluded due to their computational complexity. In my dissertation, I have systematically addressed the various sources of intractability and provided viable solutions to overcome them. While the curses of history and dimensionality are inherited from POMDPs and could be addressed extending efficient POMDP solution techniques – such as heuristic search value iteration (HSVI) [65], forward search value iteration (FSVI) [63], SARSOP [40] and GapMin [53] – multiagent settings present their own challenges in the form of curse of agent modeling and curse of many agents. I summarize the solution I proposed for these curses and discuss future improvements on these fronts.

6.1 Curse of Dimensionality

In POMDPs, the curse of dimensionality arises from the size of state space. The complexity of belief update and value iteration is affected quadratically with the size of state space. I-POMDPs extend the physical states to interactive states by including the models of other agents. The model space for intentional models is continuous in theory. However, in practice, the set of models of other agents is assumed to be computable. Nevertheless, the model space could still be large. This further aggravates the curse of dimensionality.

In my dissertation, I proposed a simple bimodal approach that mitigates the effect of the curse of dimensionality to some extent in certain online settings (Chapter 4). In this approach under higher uncertainty, the subject agent starts by behaving as a POMDP while modeling the other agents as noise. Once the agent has sufficiently identified the current state of the

environment, it switches to full I-POMDP mode. The key idea behind this approach is that in settings where the observations don't give direct information about other agent's actions (e.g. observation of creaks in the multiagent tiger problem 1), the observations regarding the states are more informative of other agents actions when the uncertainty over the true state is low. We utilize upper and lower bounds on the I-POMDP value function and a parameter to determine the switching point. The computational savings arise from the fact that, in the POMDP phase, the curse of dimensionality is tempered by ignoring other agent's models and that faster POMDP algorithms, such as SARSOP [40], could be utilized in this phase. However, these savings come at the cost of loss in performance due to ignoring the models of other agents in the initial phase.

6.1.1 Future Work

Eck and Soh [18] presented a similar approach for solving POMDPs in online settings with high uncertainty. In their approach, the agent first picks actions that would reduce its uncertainty about states the most. Once it has gained enough information about the current physical state, it changes its objective to maximize its expected rewards. As opposed to our method which uses bounds on the I-POMDP value function, this approach takes a more direct approach to reduce the entropy of its belief. Extension of this approach to I-POMDPs may provide some interesting insights in to the framework.

6.2 Curse of Agent Modeling

In addition to the worsened effects of curse of dimensionality, as the planning horizon increases, the number of reachable models of other agents also grows exponentially due to the effect of the curse of history on the other agents' I-POMDP. Including these models in

interactive state space further aggravates the curse of dimensionality. We term the effect of curse of history on curse of dimensionality as the curse of agent modeling.

The obvious solution for this problem is to restrict the model space while maintaining the solution quality. Finite state controllers (FSCs) provide a compact representation of the other agents' policies [23] [24] [51]. Thus FSCs may be utilized to capture the approximate behavior of the other agents. Hence, it is logical to compress the interactive state space to include the FSCs of the other agents instead of their beliefs. Building on this idea, we proposed the interactive bounded policy iteration (I-BPI) algorithm (Chapter 3).

I-BPI represents the solution of agents at all levels in the form of FSCs, redefining the interactive state space at level l as a joint between the physical states and the FSC of all other agents at level $l-1$. Level 0 I-POMDP is a POMDP and is solved using BPI [51]. While it is possible to formulate the level l I-POMDP as a POMDP by including the FSC obtained by solving the level $l-1$ I-POMDP completely, in the interest of having an anytime behavior, we interleave the solution at each level by dynamically redefining the interactive state space at each iteration.

We extended our approach to utilize initial the belief to bias the solution in the region of belief space that is more likely to be reached in a manner similar to one described in VDCBPI [52]. This bias comes with an increased cost of computation which we compensate for by removing nodes from other agents' controllers that are cannot be reached. Next, we adapt the algorithm to solve I-POMDP settings involving multiple other agents and settings where the frames of the other agents may not be exactly known.

We demonstrated that I-BPI significantly outperforms the previous best I-POMDP solution algorithm, I-PBVI [15], both in terms of solution quality and the execution time. We show that I-BPI could be used to solve significantly larger problems and present the solution for five agent tiger problem for the first time to the best of our knowledge.

6.2.1 Future Work

The major drawback of I-BPI, much like the parent algorithm BPI, is that it is prone to local optima. Although in some cases the controller could be pushed out of local optima by adding nodes that improve the value of the belief reachable from the tangent belief in one step, it is not always possible. Secondly, the controller may not ever reach the global optima. Moreover, it is impossible to guarantee the quality of the controller obtained with respect to the global optima.

A recent work by Grzes and Poupart [21] presents a more principled and guaranteed approach for escaping local optima. The ideas from this work are directly applicable to I-BPI as well. Although the approach favors solution in settings where initial belief is known, the advantage of this method is that it is guaranteed to generate the best quality controller of any given size. Expectation maximization based solutions for I-POMDPs are also promising [55].

6.3 Curse of Many Agents

In multiagent settings, the transient set of the environment, the observations received by an agent, and the rewards received by an agent may depend on the actions performed by all interacting agents sharing the environment. Therefore, to predict the behavior of other agents and how it affects the attainment of its own goals, the subject agent must model all other agents. If the other agents are individual rational agents, such as humans or autonomous robots, they must be modeled as such for guaranteed optimality.

In traditional I-POMDPs, the actions and models of all other agents are captured in the form of joint actions and joint models. Naturally, such a representation is exponential in the number of agents and quickly becomes unruly. For example, we would require a few gigabytes of memory just to store the belief of an agent in a 30 agent I-POMDP. Our goal

was to present scalable solution methods that would scale to settings with thousands of interacting agents. To do so, we extended the ideas from game theory, particularly action graph games ([32]), to multiagent decision making using I-POMDPs (Chapter 5).

We extended the concept of action anonymity to *frame-action anonymity* and captured the context specific independence using *frame-action hypergraphs*. Next, we incorporated these problem structures in I-POMDP to create the Many-Agent I-POMDP framework. Using more scalable factored representation of the belief, we devised scalable methods of computing updated belief and value function. We theoretically derive the computational savings from utilizing Many-Agent I-POMDPs and demonstrate it empirically in the context of two new problem domains – the first being a domain for policing protest and the second a gaming domain – that may involve thousands of agents. We show that while traditional I-POMDPs don't scale beyond a few agents (5 other agents), Many-Agent I-POMDPs are scalable to hundreds of agents (thousands with slight improvements as summarized in the next section).

6.3.1 Future Work

Many-Agent I-POMDPs yield remarkable scalability in terms of number of agents by utilizing commonly found problem structure. However, the approach relies heavily on the availability of a prior belief which is used to compute reachable beliefs. The reachable beliefs are then in turn used by the dynamic programming approach to compute a distribution over configuration. This approach doesn't suit offline settings where the initial belief of the agent is not known. Particularly, the use of configurations instead of joint actions doesn't work with general solutions computed using equation 1.11. This issue necessitates research into alternate methods of capturing problem structures, such as anonymity and context-specific independence. Furthermore, other problem structures that could potentially aid scalability should be explored as well.

6.4 Curse of History

The final curse afflicting I-POMDP solutions is the curse of history. This curse arises due to the partial observability inherent in the problem domains. Curse of history affects the solution complexity exponentially with the number of observations. Being the more dominant curse in the context of POMDPs, the curse of history has received much attention in POMDP literature. A more popular approach to mitigate the curse of history is to use heuristics guided point based value iteration [65] [63] [40] [53], which could be adapted to I-POMDP in a straightforward manner. Isolated approaches for solving POMDPs with large observations also exist [28].

In my dissertation, I proposed two methods to mitigate the effects of this curse. First method involved identifying actions which are guaranteed to yield weakly informative observations (Chapter 2). Such actions could be leveraged in the backup step by ignoring the resulting observations, hence speeding up the computation. We demonstrated the performance gained from exploiting weak information inducing actions on some standard benchmark POMDP problem domains that contain such actions. The second method I proposed is a novel branch and bound based method that prunes portions of the reachability tree that are guaranteed to generate a suboptimal solution. We utilize this method in conjugation with the Many-Agent I-POMDP (Chapter 5). We empirically demonstrate that the branch and bound method significantly adds to the scalability of the Many-Agent I-POMDP framework making it possible to exactly solve problems involving up to 2000 agents.

6.4.1 Future Work

The curse of history may be the biggest hindrance in the application of POMDP based algorithms in real world scenario. The solution complexity is exponential in the number of observations due to partial observability. Most heuristic based methods exploit the knowl-

edge that, in many real world scenarios given the current state, only a few observations are possible. Curse of history is more prominent in settings where the initial belief is not available. Novel method of addressing this curse in such settings should be explored.

References

[1] Christopher Amato, Daniel S Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for pomdps and decentralized pomdps. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320, 2010.

[2] Christopher Amato, Blai Bonet, and Shlomo Zilberstein. Finite-state controllers based on mealy machines for centralized and decentralized pomdps. In *AAAI*, 2010.

[3] Robert J Aumann. Interactive epistemology ii: Probability. *International Journal of Game Theory*, 28(3):301–314, 1999.

[4] Roger Berkowitz. Drones and the question of the human. *Ethics & International Affairs*, 28(02):159–169, 2014.

[5] Daniel S Bernstein, Christopher Amato, Eric A Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of markov decision processes. *Journal of Artificial Intelligence Research*, 34(1):89, 2009.

[6] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[7] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth international confer-*

ence on Uncertainty in artificial intelligence, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

[8] Craig Boutilier and David Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1168–1175, 1996.

[9] Adam Brandenburger and Eddie Dekel. Hierarchies of beliefs and common knowledge. *Journal of Economic Theory*, 59(1):189–198, 1993.

[10] Anthony Cassandra, Michael L Littman, and Nevin L Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 54–61. Morgan Kaufmann Publishers Inc., 1997.

[11] Antonio Del Giudice, Piotr Gmytrasiewicz, and Josh Bryan. Towards strategic kriegspiel play with opponent modeling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1265–1266. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[12] Daniel C Dennett. Intentional systems. *The Journal of Philosophy*, pages 87–106, 1971.

[13] Jack Dongarraxz, Andrew Lumsdaine, Xinhiu Niu, Roldan Pozoz, and Karin Remingtonx. A sparse matrix library in c++ for high performance architectures. 1994.

[14] Prashant Doshi and Piotr J Gmytrasiewicz. Monte carlo sampling methods for approximating interactive pomdps. *Journal of Artificial Intelligence Research*, 34(1):297, 2009.

[15] Prashant Doshi and Dennis Perez. Generalized point based value iteration for interactive pomdps. In *AAAI*, pages 63–68, 2008.

[16] Prashant Doshi, Xia Qu, Adam Goodie, and Diana Young. Modeling recursive reasoning by humans using empirically informed interactive pomdps. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1223–1230. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[17] Prashant J. Doshi. Decision making in complex multiagent contexts: A tale of two frameworks. *AI Magazine*, 33(4):82–95, 2012.

[18] Adam Eck and Leen-Kiat Soh. Online heuristic planning for highly uncertain domains. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '14, pages 741–748, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[19] Drew Fudenberg. *The theory of learning in games*, volume 2. MIT press, 1998.

[20] Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *J. Artif. Intell. Res.(JAIR)*, 24:49–79, 2005.

[21] Marek Grzes and Pascal Poupart. Incremental policy iteration with guaranteed escape from local optima in pomdp planning. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1249–1257, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[22] Qing Guo and Piotr Gmytrasiewicz. Modeling bounded rationality of agents during interactions. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1285–1286. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[23] Eric A Hansen. Solving pomdps by searching in policy space. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc., 1998.

[24] Eric A Hansen. Sparse stochastic finite-state controllers for pomdps. In *UAI*, pages 256–263, 2008.

[25] John C Harsanyi. Games with incomplete information played by bayesian players, i-iii part i. the basic model. *Management science*, 14(3):159–182, 1967.

[26] Milos Hauskrecht. *Planning and control in stochastic domains with imperfect information*. PhD thesis, Massachusetts Institute of Technology, 1997.

[27] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, pages 33–94, 2000.

[28] Jesse Hoey and Pascal Poupart. Solving pomdps with continuous or large discrete observation spaces. In *IJCAI*, pages 1332–1338, 2005.

[29] Jesse Hoey, Pascal Poupart, Axel von Bertoldi, Tammy Craig, Craig Boutilier, and Alex Mihailidis. Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. *Computer Vision and Image Understanding*, 114(5):503–519, 2010.

[30] John E Hopcroft. *Introduction to automata theory, languages, and computation*. Pearson Education India, 1979.

[31] Albert X Jiang and Kevin Leyton-Brown. Bayesian action-graph games. In *Advances in Neural Information Processing Systems*, pages 991–999, 2010.

[32] Albert Xin Jiang, Kevin Leyton-Brown, and Navin AR Bhat. Action-graph games. *Games and Economic Behavior*, 71(1):141–173, 2011.

[33] Albert Xin Jiang, Kevin Leyton-Brown, and Avi Pfeffer. Temporal action-graph games: A new representation for dynamic games. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 268–276. AUAI Press, 2009.

[34] Leslie Pack Kaelbling. *Learning in Embedded Systems*. MIT Press, Cambridge, MA, USA, 1993.

[35] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[36] Michael Kearns, Michael L Littman, and Satinder Singh. Graphical models for game theory. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 253–260. Morgan Kaufmann Publishers Inc., 2001.

[37] Kee-Eung Kim. Exploiting symmetries in pomdps for point-based algorithms. In *AAAI*, pages 1043–1048, 2008.

[38] Victor Klee and George J Minty. How good is the simplex algorithm. Technical report, DTIC Document, 1970.

[39] Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, 2003.

[40] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. *Proc. Robotics: Science and Systems*, 2008.

[41] Matthew McNaughton, Chris Urmson, John M Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Robotics and*

Automation (ICRA), 2011 IEEE International Conference on, pages 4889–4895. IEEE, 2011.

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

[43] Jean-Francois Mertens and Shmuel Zamir. Formulation of bayesian analysis for games with incomplete information. *International Journal of Game Theory*, 14(1):1–29, 1985.

[44] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, pages 705–711, 2003.

[45] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *AAAI*, volume 5, pages 133–139, 2005.

[46] Brenda Ng, Carol Meyers, Kofi Boakye, and John Nitao. Towards applying interactive pomdps to real-world adversary modeling. In *IAAI*, 2010.

[47] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[48] Christian Peters, Thomas Hermann, Sven Wachsmuth, and Jesse Hoey. Automatic task assistance for people with cognitive disabilities in brushing teeth - A user study with the TEBRA system. *TACCESS*, 5(4):10, 2014.

[49] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[50] Joelle Pineau, Geoffrey J Gordon, and Sebastian Thrun. Anytime point-based approximations for large pomdps. *J. Artif. Intell. Res.(JAIR)*, 27:335–380, 2006.

[51] Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In *Advances in neural information processing systems*, page None, 2003.

[52] Pascal Poupart and Craig Boutilier. Vdcbpi: an approximate scalable algorithm for large pomdps. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2004.

[53] Pascal Poupart, Kee-Eung Kim, and Dongho Kim. Closing the gap: Improved bounds on optimal pomdp solutions. 2011.

[54] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[55] Xia Qu and Prashant Doshi. Improved planning for infinite-horizon interactive pomdps using probabilistic inference (extended abstract). In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1839–1840, 2015.

[56] Bharanee Rathnasabapathy, Prashant Doshi, and Piotr Gmytrasiewicz. Exact solutions of interactive pomdps using behavioral equivalence. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1025–1032. ACM, 2006.

[57] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for pomdps. *J. Artif. Intell. Res.(JAIR)*, 32:663–704, 2008.

[58] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[59] Nicholas Roy, Geoffrey J Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *J. Artif. Intell. Res.(JAIR)*, 23:1–40, 2005.

[60] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[61] Richard Seymour and Gilbert L Peterson. A trust-based multiagent system. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 109–116. IEEE, 2009.

[62] Richard S Seymour and Gilbert L Peterson. Responding to sneaky agents in multi-agent domains. In *FLAIRS Conference*, 2009.

[63] Guy Shani, Ronen I. Brafman, and Solomon E. Shimony. Forward search value iteration for pomdps. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2619–2624, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[64] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[65] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press, 2004.

[66] Edward J Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

[67] Ekhlas Sonu and Prashant Doshi. Generalized and bounded policy iteration for finitely-nested interactive pomdps: scaling up. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1039–1048. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[68] Ekhlas Sonu and Prashant Doshi. Bimodal switching for online planning in multiagent settings. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 360–366. AAAI Press, 2013.

[69] Ekhlas Sonu and Prashant Doshi. Scalable solutions of interactive pomdps using generalized and bounded policy iteration. *Autonomous Agents and Multi-Agent Systems*, pages 1–40, 2014.

[70] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[71] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[72] Chris Urmson, Chris Baker, John Dolan, Paul Rybski, Bryan Salesky, William Whittaker, Dave Ferguson, and Michael Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 30(2):17, 2009.

[73] Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Decentralized stochastic planning with anonymity in interactions. In *AAAI*, 2014.

[74] Pradeep Varakantham, Shih-Fen Cheng, Geoffrey J Gordon, and Asrar Ahmed. Decision support for agent populations in uncertain and congested environments. In *AAAI*, 2012.

[75] Fangju Wang. An i-pomdp based multi-agent architecture for dialogue tutoring. In *International conference on advanced information and communication technology for education (ICAICTE)*, pages 486–489, 2013.

[76] Mark P Woodward and Robert J Wood. Learning from humans as an i-pomdp. *arXiv preprint arXiv:1204.0274*, 2012.

[77] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487–511, 2011.

[78] Michael Wunder, Michael Kaisers, John Robert Yaros, and Michael Littman. Using iterated reasoning to predict opponent strategies. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 593–600. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[79] Yifeng Zeng and Prashant Doshi. Exploiting model equivalences for solving interactive dynamic influence diagrams. *Journal of Artificial Intelligence Research*, 43(1):211–255, 2012.

Appendix A

Appendix

A.1 I-BPI Algorithm for N>2

In settings involving multiple agents, $1 \ldots K$, with the subject agent denoted as agent 1, and uncertainty over the other agents' frames with each being ascribed multiple frames, the joint model space is the product of the model spaces of each other agent. Also, the transition, observation and reward functions of an agent depend on the joint actions of all agents. In such settings, Algorithms 1, 2 and 3 are adapted as follows.

Algorithm 9 Interactive BPI for I-POMDPs

Interactive BPI (I-POMDP: $\theta_{1,l}$) **returns** solution, $\pi^*_{1,l}$

1: Initialize Controller ($\theta_{1,l}$)
2: Reformulate, $IS_{1,l} = S \times_{k=2}^{K} \mathcal{F}_{k,l-1}$, in $\theta_{1,l}$ and analogously for each frame of each other agent at all levels l down to level 1
3: Beginning with level, $l = 0$, perform a single-step full backup for each frame at each level, l, using the transformed $IS_{k,l}$, resulting in $|\mathcal{N}_{k,l}| \leq |A_k|$ nodes in a controller, $\pi_{k,l}$
4: **repeat**
5: **repeat**
6: $\pi_{k,l} \leftarrow$ Evaluate&Improve ($\pi_{k,l}$, $\theta_{k,l}$ with reformulated $IS_{k,l}$)
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 controller, $\pi^*_{1,l}$

Algorithm 10 Recursively initialize all controllers at all levels down to 0.

InitializeControllers (I-POMDP: $\theta_{k,l}$) **returns** initial controller, $\pi_{k,l}$

1: **if** $l \geq 1$ **then**
2: **for** $j \in \{1 \ldots K\}, j \neq k$ **do**
3: **for** each frame in the set, $\hat{\Theta}_{j,l-1}$, included in $\theta_{j,l-1}$ **do**
4: $\pi_{j,l-1} \leftarrow$ InitializeControllers ($\theta_{j,l-1}$)
5: Construct a controller, $\pi_{k,l}$, with a single node, $|\mathcal{N}_{k,l}| = 1$, mapped to a random action

Algorithm 11 Evaluation and bounded improvement of the controllers at different levels.

Evaluate&Improve (controller: $\pi_{k,l}$, I-POMDP model: $\theta_{k,l}$) **returns** controller, $\pi'_{k,l}$

1: **if** $l \geq 1$ **then**
2: **for** $j \in \{1 \ldots K\}, j \neq k$ **do**
3: **for** each frame in the set, $\hat{\Theta}_{j,l-1}$, included in $\theta_{j,l-1}$ **do**
4: $\pi_{j,l-1} \leftarrow$ Evaluate&Improve ($\pi_{j,l-1}$, $\theta_{j,l-1}$)
5: **if** l=0 **then**
6: Evaluate controller, $\pi_{k,0} = \langle \mathcal{N}_k, \mathcal{E}_k, \mathcal{L}_k, \mathcal{T}_k \rangle$
7: Improve controller, if possible, analogously to a POMDP [51]
8: **else**
9: Evaluate controller, $\pi_{k,l} = \langle \mathcal{N}_{k,l}, \mathcal{E}_{k,l}, \mathcal{L}_{k,l}, \mathcal{T}_{k,l} \rangle$, analogous to Eq. 3.2 replacing a_j with joint actions of all other agents and $n_{j,l-1}$ with joint nodes of all other agents as formulated in line 2 of Alg. 9
10: Improve controller, if possible, while keeping $|\mathcal{N}_{k,l}|$ fixed analogous to Eq. 3.3
11: **return** improved controller, $\pi'_{k,l}$

Publication List:

1. Ekhlas Sonu, Yingke Chen and Prashant Doshi, Individual Planning in Agent Populations: Exploiting Anonymity and Frame-Action Hypergraphs, in *International Conference on Automated Planning and Scheduling*, 2015.

2. Ekhlas Sonu and Prashant Doshi, Scalable Solutions of Interactive POMDPs using Generalized and Bounded Policy Iteration, *Journal of Autonomous Agents and Multiagent Systems (JAAMAS)*, pp. 1-40, April 2014.

3. Ekhlas Sonu and Prashant Doshi, Bimodal Switching for Online Planning in Multiagent Settings, *In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp.360-366, 2013, Beijing, China.

4. Ekhlas Sonu and Prashant Doshi, Generalized and Bounded Policy Iteration for Finitely-Nested Interactive POMDPs: Scaling up, *In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems: volume 1*, pp. 1039–1048, 2012, Valencia Spain.

5. Ekhlas Sonu and Prashant Doshi, GaTAC: A Scalable and Realistic Testbed for Multiagent Decision Making (demonstration), *In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1507-1508, 2012, Valencia, Spain.

6. Ekhlas Sonu and Prashant Doshi, Generalized and Bounded Policy Iteration for Interactive POMDPs, *International Symposium on Artificial Intelligence and Mathematics*, 2012, Fort Lauderdale, FL.

7. Ekhlas Sonu and Prashant Doshi, Identifying and Exploiting Weak-Information Inducing Actions in Solving POMDPs (Extended Abstract), *In Proceedings of 10th Inter-*

national Conference on Autonomous Agents and Multiagent Systems, pp. 1259-1260, 2011, Taipei, Taiwan.

8. Ekhlas Sonu and Prashant Doshi, GaTAC: A Scalable and Realistic Testbed for Multiagent Decision Making , *Workshop on Multiagent Sequential Decision Making*, In the 10th International Conference on Autonomous Agents and Multiagent Systems, 2010, Toronto, Canada.