# Approximating State Estimation in Multiagent Settings Using Particle Filters

Paper ID: 875

**Abstract**

State estimation, also called filtering and monitoring, consists of updating an agent's belief given executed actions and observed evidence to date. The agent's belief is a distribution over the current state. In single agent environments, the state estimation can be formalized within the framework of partially observable Markov decision processes (POMDPs). Exact estimation can be performed in simple cases, but approximate techniques, like particle filtering, have been used in more realistic cases. This paper extends these techniques to multiagent settings. The main difficulty we tackle is that to fully represent an agent's beliefs in such environments one has to specify probability distributions over the physical state and over the beliefs of other agents. This leads to interactive hierarchical belief systems first developed in game theory. Since the update of such beliefs proceeds recursively, we extend the particle filtering technique by defining sampling procedures which involve sampling on all levels of the belief hierarchy. We present algorithms, discuss some of their properties, and illustrate the performance of our implementation using simple examples.

## I. Introduction

To act rationally, agents have to be able to track the evolution of a state of the environment over time, based on the actions they perform and on the available observations. In realistic single agent and multiagent settings, the agent faces uncertainty due to both imperfect sensors and non-deterministic action outcomes. This results in an incomplete knowledge about the true state of the world, usually represented as a probability distribution and called an agent's belief. In single agent settings, the state estimation is usually accomplished with a technique called *Bayes filter*. A Bayes filter allows the agent to maintain a belief about the state of the world at any given time, and updating this belief each time an action is performed and new sensory information arrives. The convenience of this approach lies in the fact that the update is independent of the past percepts and action sequences. Technically, this means that the agent's belief is a *sufficient statistic*, i.e., it fully summarizes all of the information in contained in past actions and observations.

The operation of a Bayes filter can be decomposed into a two-step process:

- *Prediction* When an agent performs a new action, $a^{t-1}$, its prior belief state is updated.

$$Pr(s^t|a^{t-1}, b^{t-1}) = \int_{s^{t-1}} b^{t-1}(s)T(s^t|s, a^{t-1})ds$$

- *Correction* Thereafter, when an observation, $o^t$, is received, the intermediate belief state, $Pr(\cdot|a^{t-1}, b^{t-1})$, is corrected.

$$\begin{aligned} Pr(s^t|o^t, a^{t-1}, b^{t-1}) = \; & \alpha O(o^t|s^t, a^{t-1}) \\ & \times Pr(s^t|a^{t-1}, b^{t-1}) \end{aligned}$$

  where $\alpha$ is the normalizing constant, $T$ and $O$ are the transition and observation functions, respectively.

An important application of the Bayes filter is in single agent planning. Specifically, the Bayes filter performs the belief update step in the POMDP planning framework [10]. However, performing the exact update is possible only in the simplest environments [1] – in realistic cases one is forced to rely on approximations. One popular approximation is based on representing beliefs over continuous state spaces with a finite number of discrete samples, or particles. Particle filtering propagates the samples forward in time using the prediction and correction steps above, and generates an approximate representation of the agent's updated belief.

In multiagent settings matters are more complicated since the others' actions also change the state of the system. A straightforward approach to extending the Bayes filter to multiagent case is to consider other agents' actions as exogenous events (or noise) which occur with static probability

[1] For continuous spaces strong assumptions of Gaussian beliefs and linear dynamics lead to Kalman filters and computable updates.

distributions. However, this treats other agents as random devices and does not capture the other agents' belief updates over time which, of course, causes changes in their behavior.

A more refined approach is for the agent to maintain beliefs over the other agent's state. [2] The fact that the other agent's state can be described in terms its beliefs [3] (as well as in terms of other factors, say, preferences, abilities, etc.) has two important consequences. First, the original agent's beliefs are necessarily defined over a continuous space so exact update is difficult. This suggests that sampling methods could be convenient approximations. Second, a nested hierarchy of beliefs obtains which includes other agent's beliefs over the original agent's beliefs, and so on. Such *interactive beliefs* have been the subject of much work in game theory and in theoretical computer science [**?**], [**?**], [**?**], [1], [**?**], [12], [**?**], [9]. According to this approach, the multiagent state estimation problem, therefore, consists of keeping track of other's beliefs and of the physical state changes due to actions of all agents.

We explore the second approach and explicitly include comprehensive models of the interacting rational agents in the state space. The models encompass all information influencing the agent's behavior which includes its preferences, capabilities, and beliefs, and are analogous to *types* in Bayesian games [7]. In this representation, the state estimation filter recursively updates an agent's nested belief, given the agent's action and received observation. As in case of POMDPs, the important applications of the update is in autonomous agent planning in multiagent settings [4], [5].

Of course, as we mentioned, exact filtering is not possible and we propose using the particle filtering technique for approximating the multiagent state estimation filter. Mirroring the hierarchical character of interactive beliefs, our approach involves nested sampling at each of the hierarchical

[2] For clarity, we will assume that there are only two interacting agents.

[3] Including an agent's belief in the description of its state is, technically, not required. However, since actions depend on beliefs it is necessary to keep track of agent's beliefs to predict its actions. Predicting other agent's actions is, in turn, required to predict the physical state.

levels of nested beliefs. The result is a multiagent version of the basic bootstrap filter [6], [2] often used in single agent settings.

Rest of this paper is structured in the following manner. We formally introduce our multiagent state estimation filter in Section 2. We then present the particle filtering algorithm that approximates the belief update. We empirically analyze the performance of our particle filter in two multiagent domains. We conclude with discussion and future work.

## II. MULTIAGENT STATE ESTIMATION

In this section, we first introduce the framework which we use to formalize our filter. We then present our state estimation filter for multiagent settings.

For simplicity of presentation, let us consider an agent $i$, that is interacting with one other agent $j$. Our arguments generalize to a setting with more than two agents in a straightforward manner. For purposes of computability, we restrict an agent's belief to finite levels of nesting. Let $IS_{i,l}$ denote a set of **interactive** states defined as, $IS_{i,l} = S \times \Theta_{j,l-1}$, for $l \geq 1$, and $IS_{i,0} = S$, where $S$ is the set of states of the physical environment, and $\Theta_{j,l-1}$, is the set of $(l-1)^{th}$ level *intentional models* of agent $j$. A $(l-1)^{th}$ level intentional model of $j$ is, $\theta_{j,l-1} = \langle b_{j,l-1}, A, \Omega_j, T_j, O_j, R_j, OC_j \rangle$, where $OC_j$ is agent $j$'s optimality criterion, together with the assumption that agent $j$ is Bayesian rational. The $0^{th}$ level intentional model is a POMDP with the other agent's actions folded as *noise* into the $T$, $O$, and $R$ functions. For the sake of simplicity, let us rewrite $\theta_{j,l-1}$ as, $\theta_{j,l-1} = \langle b_{j,l-1}, \widehat{\theta}_j \rangle$, where $\widehat{\theta}_j \in \widehat{\Theta}_j$, is the agent $j$'s *frame*. We give a recursive bottom-up construction of the state spaces below:

$$\begin{array}{lll}
IS_{i,0} & = S, & \Theta_{j,0} = \{\langle b_{j,0}, \widehat{\theta}_j \rangle : b_{j,0} \in \Delta(IS_{j,0})\}, \\
IS_{i,1} & = S \times \Theta_{j,0}, & \Theta_{j,1} = \{\langle b_{j,1}, \widehat{\theta}_j \rangle : b_{j,1} \in \Delta(IS_{j,1})\}, \\
\vdots & & \vdots \\
IS_{i,l} & = S \times \Theta_{j,l-1}, & \Theta_{j,l} = \{\langle b_{j,l}, \widehat{\theta}_j \rangle : b_{j,l} \in \Delta(IS_{j,l})\},
\end{array}$$

A belief of agent $i$ therefore comprises of finitely nested beliefs over others' models and their beliefs about others [7], [12], [1], [9]. [4] Increasing the value of $l$ results in deeper levels of nesting.

To express $i$'s beliefs nested to some level $l$, we adopt the language of polynomials, as an example, to represent the probability density functions. We construct the belief representation in an inductive manner as follows. At $l = 0$, i's belief, $b_{i,0} \in \Delta(S)$, is a vector of length $|S| - 1$. Agent $i$'s first level belief, $b_{i,1} \in \Delta(S \times \langle B_{j,0} \times \widehat{\Theta}_j \rangle)$, is a p.d.f. represented using a $(|S|-1)$-variate polynomial for each state and $j$'s frame. The polynomial can be written in a parametric form: $f_i^1 = \langle d, c_1, c_2, \ldots, c_{(d+1)^{(|S|-1)}} \rangle$, where $d \in \mathbb{N}$ is the degree, and $c \in \mathbb{R}$ is a coefficient. Note that the area under the polynomials must sum to 1. We describe $i$'s level 2 beliefs, $b_{i,2} \in \Delta(S \times \langle B_{j,1} \times \widehat{\Theta}_j \rangle)$, by a univariate polynomial over the degree $d$ of $f_j^1$ and $(d+1)^{|S|-1}$ univariate polynomials, one over each coefficient of $f_j^1$. We specify higher levels of beliefs in an analogous way.

There are two differences that complicate state estimation in multiagent settings, when compared to single agent ones. First, since the state of the physical environment depends on the actions performed by both agents, the prediction of how the physical state changes has to be made based on the predicted actions of the other agent. The probabilities of other's actions are obtained based on its models. Thus, we do not assume that actions are fully observable by other agents. Rather, agents can attempt to infer what actions other agents have performed by sensing their results on the environment. Second, changes in the models of other agents have to be included in the update. Specifically, in the case of intentional models, update of the other agent's beliefs due to its new observation has to be included. In other words, the agent has to update its beliefs based on what it anticipates that the other agent observes and how it updates.

For ease of comparison with the traditional Bayes filter, we decompose our multiagent state estimation filter into two steps:

- *Prediction* When an agent, say $i$, performs an action $a_i^{t-1}$, and the other agent $j$ performs

[4] Implicit in the definition of interactive beliefs is the assumption of coherency [1].

$a_j^{t-1}$, the predicted belief state is,

$$\begin{array}{l} Pr(is^t|a_i^{t-1}, a_j^{t-1}, b_{i,l}^{t-1}) = \int_{is^{t-1}:\widehat{\theta}_j^{t-1}=\widehat{\theta}_j^t} b_{i,l}^{t-1}(is^{t-1}) \\ \times Pr(a_j^{t-1}|\theta_{j,l-1}^{t-1}) T(s^{t-1}, a_i^{t-1}, a_j^{t-1}, s^t) \\ \times \sum_{o_j^t} O_j(s^t, a_i^{t-1}, a_j^{t-1}, o_j^t) \\ \times \delta(SE_{\widehat{\theta}_j^t}(b_{j,l-1}^{t-1}, a_j^{t-1}, o_j^t) - b_{j,l-1}^t) d(is^{t-1}) \end{array} \tag{1}$$

where $\delta$ is the Dirac-delta function, $SE(\cdot)$ is an abbreviation for the belief update, $Pr(a_j^{t-1}|\theta_{j,l-1}^{t-1})$ is the probability that $a_j^{t-1}$ is Bayes rational for the agent described by $\theta_j^{t-1}$, and $T$ and $O$ are the transition and observation functions respectively, extended to multiagent settings.

- *Correction* When agent $i$ perceives an observation, $o_i^t$, the corrected belief state is a weighted sum of the predicted belief states for each possible action of $j$,

$$\begin{array}{l} Pr(is^t|o_i^t, a_i^{t-1}, b_{i,l}^{t-1}) = \alpha \sum_{a_j^{t-1}} O_i(s^t, a_i^{t-1}, a_j^{t-1}, o_i^t) \\ \times Pr(is^t|a_i^{t-1}, a_j^{t-1}, b_{i,l}^{t-1}) \end{array} \tag{2}$$

where $\alpha$ is the normalizing constant,

In our multiagent state estimation presented above, when $j$ is modeled using an intentional model, $i$'s belief update will invoke $j$'s belief update (via the term $SE_{\hat{\theta}_j^t}(b_{j,l-1}^{t-1}, a_j^{t-1}, o_j^t)$), which in turn will invoke $i$'s belief update and so on. This recursion in belief nesting will bottom out at the $0$-th level. At this level, belief update of the agent reduces to a single agent Bayes filter.

In a manner similar to the single agent Bayes filter, the following proposition holds for our multiagent state estimation filter. The proposition results from noting that Eq. 2 expresses the belief in terms of parameters of the previous time step only. A complete proof of our state estimation filter and this proposition is given in [5].

**Proposition 1 (Sufficiency).** *Agent $i$'s current belief, i.e. probability distribution over the set $S \times \Theta_{j,l-1}$ is a sufficient statistic for the past history of $i$'s observations.*

## III. Example: Multiagent Tiger Game

We analyze the operation of our multiagent state estimation filter using a simple example.

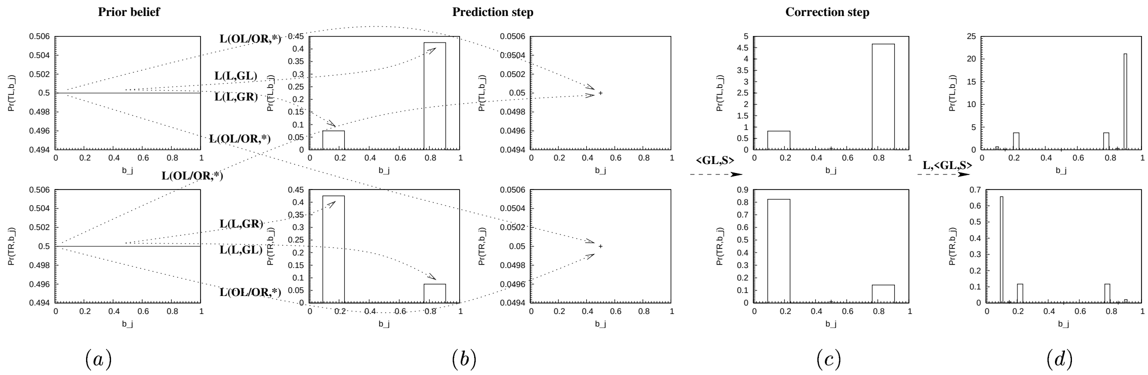



Fig. 1. Operation of our multiagent filter on the multiagent tiger game. $(a)$ is the prior. $(b)$ is the result of *prediction* given $i$'s listening action, and a pair denoting $j$'s action and observation. $(c)$ is the result of *correction* after $i$ observes $\langle GL, S\rangle$. $(d)$ depicts the result of another belief update after $i$ listens and receives $\langle GL, S\rangle$.

**Non-cooperative Multiagent Tiger Problem** The multiagent Tiger game resembles a game-show like situation in which there are two doors, behind one is a tiger ready to eat anyone who opens the door, and behind another is a pot of gold. The game is played by two agents,$i$ and $j$, who are unaware of where the gold is, and each can either open any one of the doors, or listen. The tiger emits a growl periodically, which reveals its position but only with some certainty. Additionally, each agent can also hear a creak with some certainty, if the other agent opens a door. We will assume that neither agent can perceive other's observations nor actions. The transition, reward, and observation functions describing the game are given in [5]. In such a situation, what is the sequence of actions conditioned on observations that each agent should take to maximize its gains over a period of fixed number of steps?

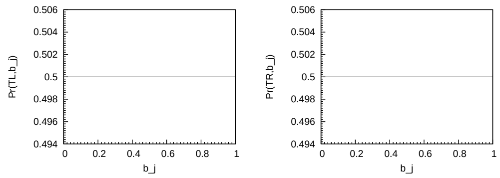



Fig. 2. Example level 1 belief of agent $i$. Here, $i$ is uninformed about $j$'s beliefs. $b_j$ is agent $j$'s level 0 belief which is $Pr_j(TL)$.

When an agent makes its choice in the multiagent tiger game, it has to consider what it believes about the location of the tiger, as well as whether the other agent will listen or open a door, which in turn depends on the other agent's beliefs. For the purpose of illustration, let us now make a simplifying assumption: Agent $i$ is aware of all of agent $j$'s properties except its belief. Therefore, $i$ is uncertain only about $j$'s beliefs. Additionally, the frames of agent $i$ and $j$ are identical, and are as shown in [5].

In Fig. 2, we show an example level 1 belief of agent $i$, expressed using the polynomial $f_i^1 = \langle 0, 0.5 \rangle$ for each state. According to this belief state, $i$ is uninformed about $j$'s beliefs. Additionally, agent $i$'s belief over the location of the tiger is computed by marginalizing over $j$'s beliefs. In Fig. 2, $i$ believes with equal probability that the tiger is behind one of the two doors. An example level 2 belief of $i$ is expressed using the polynomial, $\alpha(1 + \sum_{n=1}^{\infty}(-1)d^n)$ where $d$ is the degree of $f_j^1$ and $\alpha$ is the normalizing constant, along with $d+1$ polynomials of degree 0 defined over each coefficient of $f_j^1$, for each $d$. The same set of polynomials is used for each state. According to this belief, $i$ considers increasingly complex level 1 beliefs of $j$ (i.e. $f_j^1$ with increasing degrees) less likely.

**Operation of our state estimation filter** Let us trace the filtering process using the multiagent tiger game introduced previously. In Fig. 1, we display an example trace of the state estimation that starts with a prior belief (identical to the one in Fig. $2(ii)$), and then assumes that agent $i$ listens and hears a growl from the left and no creaks.

As we mentioned, state estimation for agent $i$ involves solving $\theta_{j,0}$ which is a traditional POMDP in which we assume that $i$'s actions are "folded" in with a probability of 0.33, 0.33, and 0.33 for OL, L, and OR, respectively (akin to ascribing a *no-information* model to agent $i$). The corresponding two steps to go policy of $j$ prescribes OL when $b_{j,0} < 0.1$, L when $0.1 \leq b_{j,0} < 0.9$, and OR when $0.9 \leq b_{j,0} \leq 1$.

Fig. $1(a)$ depicts the prior according to which $i$ is uninformed about $j$'s beliefs. Fig. $1(b)$ is the result of the *prediction* step after $i$'s listening action (L). The first column of $(b)$ shows $i$'s belief after it has listened and given that $j$ also listened ($j$ listens when its belief is between 0.1 and 0.9). The two observations $j$ can make, GL and GR, each with probability dependent on the tiger's location, give rise to flat portions representing what $i$ knows about $j$'s belief in each case. The second column of $(b)$ shows $i$'s belief after $i$ has listened and $j$'s has opened the left ($j$'s belief is less than 0.1) or right door ( $j$'s belief is greater than or equal to 0.9). The plots are identical for each action and only one of them is shown. In this case, $i$ knows that $j$ has no information about the tiger's location in this case. Fig. $1(c)$ is the result of *correction* after $i$ observes tiger's growl on the left and no creaks $\langle$GL,S$\rangle$. The plots in $(c)$ are obtained by performing a weighted summation of the plots in $(b)$. The probability $i$ assigns to TL (0.85) is now greater than TR (0.15), and information about $j$'s beliefs allows $i$ to refine its prediction of $j$'s action in the next time step. In particular, while agent $i$ in column $(a)$ believed that $j$ would listen, open-left, and open-right with probabilities 0.8,0.1,and 0.1 respectively, in column $(c)$, $i$ believes these probabilities to be 0.95, 0.013, and 0.037 respectively. Fig. $1(d)$ illustrates the posterior after another step of the belief update during which $i$ again listens and hears a $\langle$GL,S$\rangle$.

## IV. Particle Filters

A continuum of intentional models of an agent exist. Since an agent is unaware of the true models of interacting agents *ex ante*, it must maintain a belief over all possible candidate models. This precludes practical implementations of our multiagent state estimation filter, and approximate implementations are critically required.

### A. Algorithm

Several approaches to approximate non-linear recursive Bayesian estimation exist. Amongst these, the extended Kalman filter (EKF) [14], is most popular. The EKF linearises the estimation problem so that the Kalman filter can be applied. The required p.d.f. is approximated by a

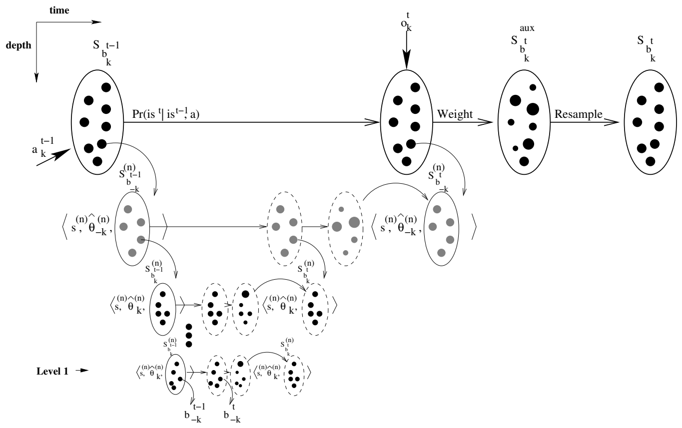



Fig. 3. An illustration of our nested particle filtering implementation. Colors black and gray distinguish filtering for the two agents.

Gaussian, which may lead to filter divergence, and therefore an increase in the error. Other approaches include the Gaussian sum filter [15], and superimposing a grid over the state space with the belief being evaluated only over the grid points [11]. In the latter approach, the choice of an efficient grid is non-trivial, and the number of grid points that must be considered is exponential in the dimensions of the state space. Recently, techniques that utilize Monte Carlo (MC) sampling for approximating the Bayesian state estimation problem have received much attention. These techniques are general enough, in that, they are applicable to both linear, as well as, non-linear problem domains. Amongst the spectrum of MC techniques, two that have been particularly well-studied in sequential settings are Markov chain Monte Carlo (MCMC) [8], and particle filters [2], [6]. Approximating our multiagent state estimation using the former technique, may turn out to be computationally exhaustive. Specifically, MCMC algorithms [8] utilize rejection sampling, that often causes a large number of intentional models to be sampled, solved, and rejected, before one is utilized for transition. However, particle filters do not employ rejection sampling for transition, and typically produce reasonable approximations of the posterior while being computationally feasible.

The general algorithm for the basic particle filter appears in [2]. We shall concentrate on a specific implementation of this algorithm, that has previously been studied under various names

**Function** PARTICLEFILTER($S_{b_k^{t-1}}, a_k^{t-1}, o_k^t, l$) **returns** $\mathrm{S}_{b_k^t}$

1. $\mathrm{S}_{b_k^t}^{aux} \leftarrow \phi$, $\mathrm{S}_{b_k^t} \leftarrow \phi$
   <u>*Importance Sampling*</u>
2. **for all** $is_k^{(n),t-1} = \langle s^{(n),t-1}, \theta_{-k}^{(n),t-1} \rangle \in S_{b_k^{t-1}}$ **do**
3. $\quad Pr(A_{-k}|\theta_{-k}^{(n),t-1}) \leftarrow$ SOLVEMODEL($\theta_{-k}^{(n),t-1}, l-1$)
4. $\quad$ Sample $a_{-k}^{t-1} \sim Pr(A_{-k}|\theta_{-k}^{(n),t-1})$
5. $\quad$ Sample $s^{(n),t} \sim T_k(S^t|a_k^{t-1}, a_{-k}^{t-1}, s^{(n),t-1})$
6. $\quad$ **for all** $o_{-k}^t \in \Omega_{-k}$ **do**
7. $\quad\quad$ **if** ($l$ = 1) **then**
8. $\quad\quad\quad b_{-k}^{(n),t} \leftarrow$ LEVEL0BELIEFUPDATE($b_{-k}^{(n),t-1}, a_{-k}^{t-1}, o_{-k}^t$)
9. $\quad\quad\quad \theta_{-k}^{(n),t} \leftarrow \langle b_{-k}^{(n),t}, \widehat{\theta}_{-k}^{(n)} \rangle$
10. $\quad\quad\quad is_k^{(n),t} \leftarrow \langle s^{(n),t}, \theta_{-k}^{(n),t} \rangle$
11. $\quad\quad$ **else**
12. $\quad\quad\quad \mathrm{S}_{b_{-k}^{(n),t}} \leftarrow$ PARTICLEFILTER($\mathrm{S}_{b_{-k}^{(n),t-1}}, a_{-k}^{t-1}, o_{-k}^t, l-1$)
13. $\quad\quad\quad \theta_{-k}^{(n),t} \leftarrow \langle\, \mathrm{S}_{b_{-k}^{(n),t}}, \widehat{\theta}_{-k}^{(n)} \rangle$
14. $\quad\quad\quad is_k^{(n),t} \leftarrow \langle s^{(n),t}, \theta_{-k}^{(n),t} \rangle$
15. $\quad\quad$ Weight $is_k^{(n),t}$: $w_t^{(n)} = O_{-k}(o_{-k}^t|s^{(n),t}, a_{-k}^{t-1}, a_k^{t-1})$
16. $\quad\quad$ Adjust weight: $w_t^{(n)} = w_t^{(n)} \times O_k(o_k^t|s^{(n),t}, a_{-k}^{t-1}, a_k^{t-1})$
17. $\quad\quad \mathrm{S}_{b_k^t}^{aux} \stackrel{\cup}{\leftarrow} (is^{(n),t}, w_t^{(n)})$
18. Normalize all $w_t^{(n)}$ so that $\sum_{n=1}^{N} w_t^{(n)} = 1$
    <u>*Selection*</u>
19. Resample with replacement $N$ particles $\{is^{(n),t}, n = 1...N\}$
    from the set $\mathrm{S}_{b_k^t}^{aux}$ according to the importance weights.
20. $\mathrm{S}_{b_k^t} \leftarrow \{is^{(n),t}, n = 1...N\}$
21. **return** $\mathrm{S}_{b_k^t}$

**end function**

Fig. 4. Particle filtering for approximating the multiagent state estimation filter. A nesting of particle filters is used to update all levels of belief.

such as Monte Carlo localization, survival of the fittest, and bootstrap filter. We extend this implementation to our multiagent state estimation filter. Our extension to the multiagent case, similar to basic particle filtering, requires the key steps of *importance sampling* and *selection*. The resulting algorithm, inherits the convergence properties of the original algorithm [2].

Our algorithm described in Fig. 4 requires an initial set of $N$ particles, $S_{b_k^{t-1}}$, each representing the agent's possible interactive state, along with the action, $a_k^{t-1}$, the observation, $o_k^t$, and the level of belief nesting, $l \geq 1$. $S_{b_k^{t-1}}$ may be generated using the PRIORSAMPLE function that expresses beliefs using the language of polynomials mentioned previously. To facilitate implementation, we

**Function** PRIORSAMPLE($\{f_{k,1}, \ldots, f_{k,I}\}, l$) **returns** $\mathrm{S}_{b_k}$

1. **for** $n$ **from** $1$ **to** $N$ **do**
2. Sample $s^{(n)} \sim \mathcal{U}(1, 2, \ldots, |S_k|)$, $\widehat{\theta}_{-k}^{(n)} \sim \mathcal{U}(1, 2, \ldots, |\widehat{\Theta}_{-k}|)$
3. **if** (l=1) **then**
4. Sample $b_{-k}^{(n)} = \langle Pr(s_1), Pr(s_2), \ldots, Pr(s_{|S_{-k}|-1}) \rangle \sim f_{k,1}$
5. $is_k^{(n)} \leftarrow \langle s^{(n)}, \langle b_{-k}^{(n)}, \widehat{\theta}_{-k}^{(n)} \rangle \rangle$
6. **else if** (l=2) **then**
7. Sample $d \sim f_{k,1}$
8. **for** $i$ **from** $2$ **to** $1 + (d+1)^{|S_{-k}|-1}$ **do**
9. Sample $c_i \sim f_{k,i}$
10. $f_{-k,1} \leftarrow \langle d, c_1, \ldots, c_{(d+1)^{|S_{-k}|-1}} \rangle$
11. $\mathrm{S}_{b_{-k}}^{(n)} \leftarrow$ PRIORSAMPLE($\{f_{-k,1}\}, l-1$)
12. $is_k^{(n)} \leftarrow \langle s^{(n)}, \langle \mathrm{S}_{b_{-k}}^{(n)}, \widehat{\theta}_{-k}^{(n)} \rangle \rangle$
13. **else**
14. **for** $i$ **from** $1$ **to** $(1 + (d_{max}+1))^{l-2} \cdot (1 + (d_{max}+1)^{|S_{-k}|-1})$ **step** $1 + (d_{max}+1)$ **do**
15. Sample $d \sim f_{k,i}$
16. **for** $j$ **from** $2$ **to** $1 + (d+1)$ **do**
17. Sample $c_j \sim f_{k,i+j}$
18. $f_{-k} \overset{\cup}{\leftarrow} \{\langle d, c_1, \ldots, c_{d+1} \rangle\}$
19. $\mathrm{S}_{b_{-k}}^{(n)} \leftarrow$ PRIORSAMPLE($f_{-k}, l-1$)
20. $is_k^{(n)} \leftarrow \langle s^{(n)}, \langle \mathrm{S}_{b_{-k}}^{(n)}, \widehat{\theta}_{-k}^{(n)} \rangle \rangle$
21. $\mathrm{S}_{b_k} \overset{\cup}{\leftarrow} \{is_k^{(n)}\}$
22. **return** $\mathrm{S}_{b_k}$

**end function**

upper bound the degree, and assume that the agent is uninformed of its state and other's frame. The procedure generalizes immediately to the case where the agent believes that certain states and frames are more likely. The particle filtering proceeds by propagating each particle forward in time. However, as opposed to basic particle filtering, this is not a one-step process. In order to perform the propagation, other agent's action must be known. This is obtained by solving the other agent's model, and using its belief (contained in the particle) to find a distribution over its actions (line 3 in Fig. 4). Additionally, for each of the other agent's possible observations we must obtain its next belief state (line 6). If $l > 1$, updating the other agent's belief requires invoking the particle filter for performing its belief update (lines 12–14). This recursion in depth of the belief nesting terminates when the level of nesting becomes one, and a POMDP-like belief update is performed (lines 8–10). Though the propagation step generates $|\Omega_{-k}|N$ appropriately weighted particles, we

**Function** LEVEL0BELIEFUPDATE($b_k^{t-1}, a_k^{t-1}, o_k^t$) **returns** $b_k^t$
$Pr(a_{-k}^{t-1}) \leftarrow 1/a_{-k}^{t-1}$
**for all** $s^t \in S$ **do**
  sum $\leftarrow 0$
  **for all** $s^{t-1} \in S$ **do**
    $Pr(s^t|s^{t-1}, a_k^{t-1}) \leftarrow 0$
    **for all** $a_{-k}^{t-1} \in A_{-k}$ **do**
      $Pr(s^t|s^{t-1}, a_k^{t-1}) \stackrel{+}{\leftarrow} T_k(s^t|s^{t-1}, a_k^{t-1}, a_{-k}^{t-1})Pr(a_{-k}^{t-1})$
    sum $\stackrel{+}{\leftarrow} Pr(s^t|s^{t-1}, a_k^{t-1})b_k^{t-1}(s^{t-1})$
  $Pr(o_k^t|s^t, a_k^{t-1}) \leftarrow 0$
  **for all** $a_{-k}^{t-1} \in A_{-k}$ **do**
    $Pr(o_k^t|s^t, a_k^{t-1}) \stackrel{+}{\leftarrow} O_k(o_k^t|s^t, a_k^{t-1}, a_{-k}^{t-1})Pr(a_{-k}^{t-1})$
  $b_k^t(s^t) \leftarrow Pr(o_k^t|s^t, a_k^{t-1}) \times$ sum
Normalize the belief, $b_k^t$
**return** $b_k^t$
**end function**

resample only $N$ particles out of these (line 20). A visualization of our implementation is shown in Fig. 3.

Let us now analyze the gains in computation we will accrue using our algorithm. First, we focus on the reduction in the number of agent models that must be solved. In an $M$+1-agent setting with the number of particles bounded by $N$, each particle in $\mathrm{S}_{b_k^{t-1}}$ of level $l$ will have $M$ models of level $l-1$. Solution of each of these level $l-1$ models will require solving the lower level models recursively. The upper bound on the number of models that will be solved is $O((MN)^{l-1})$. Taking into account that there are $M$ level $l-1$ models in a particle, and $N$ such possibly distinct particles, we are required to solve $O((MN)^l)$ models. Our upper bound is polynomial in $M$, compared to $O((M|\Theta_*|^M)^l)$ models that need to be solved in the exact case, which is exponential in $M$. Here, amongst the spaces of models for each agent, $\Theta_*$ is the largest space. Typically, $N << |\Theta_*|^M$, resulting in a substantial reduction in computation. Our filtering algorithm demonstrates substantial computational gains even when we assume that solutions of models are available in constant time. Referring to Fig. 4, in a setting of $M$+1 agents, recursive calls to the algorithm will be made $O(NM|\Omega_*|)$ times, and the recursion terminates when the LEVEL0BELIEFUPDATE is executed which has time complexity $O(|S|^2|A_*|^M)$. Here, $A_*$ and

$\Omega_*$ is the largest action and observation space respectively, among all such spaces of the other agents. The total time complexity of our particle filtering algorithm is $O((NM|\Omega_*|)^l|S|^2|A_*|^M)$, compared to that of the exact case which is $O((|S||\Theta_*|^M)^2|A_*|^M M|\Omega_*|)^l|S|^2|A_*|^M)$.

## *B. Performance*

As part of our empirical investigation of the performance of the particle filter, we show, using a standard distance metric and visually, that our particle filter approximates the exact multiagent state estimation filter closely. For our analysis, we utilize the non-cooperative two-agent Tiger game, that has two physical states, as described in Section 3, and a cooperative two-agent version of the machine maintenance problem(MM) [13], that has three physical states. For both these problems, we make the simplifying assumption that models of the other agent differ only in their beliefs. We use a numerical integration implementation [5] for the exact filter as the baseline for comparison.

The lineplots in Fig. 6 show that the quality of the approximation, as measured by KL-Divergence, increases as the number of particles increases, for both the problem domains. As we may expect, level 2 belief approximations require considerably more particles as compared to level 1 approximations, to achieve similar performance. Additionally, our results show that the performance of the particle filter is adversely affected by an increase in the number of physical states in the problem domain. This is a natural result of the reduced sampling density when the number of physical states is increased, while holding the number of particles constant. [6] Each data point in the lineplots is the average of 10 runs of our particle filter. In the case of the Tiger game, the posterior used for comparison is the one that is obtained after agent $i$ listens and hears a growl left and no creaks (Fig. 1(c)). For the machine maintenance game, the posterior obtained after $i$

[5] We obtained the points for numerical integration by superimposing a high resolution regular grid on the interactive state space.
[6] The popular notion that particle filters are immune to the "curse of dimensionality" is in regard to their convergence rate, and is not a comment on their performance accuracy in comparison to exact filtering.

manufactures and perceives no defect in the product, is used for comparison. We selected the belief state in Fig. 2 as the prior level 1 belief of agent $i$ when playing the Tiger game, and analogously for the machine maintenance game. Prior level 2 belief of $i$ is the one mentioned in Section 3 with $d_{max} = 3$. A comparison of the run times of the exact filter implemented using numerical integration and the particle filter is shown in Fig. 7. Our particle filtering implementation significantly outperforms the numerical integration based implementation, while providing comparable performance quality. In order to assess the quality of the approximations after successive belief updates, we graphed the p.d.f.s produced by the particle filter and the exact filter [7]. The p.d.f.s arising after each of three filtering steps on the level 1 belief of agent $i$ in the Tiger game, are shown in Fig 5. Each approximate p.d.f. is the average of 10 runs of the particle filter which contained 5000 particles, and is estimated using a standard Gaussian kernel. The action/observation sequence followed was $\langle L, GL, S\rangle, \langle L, GL, S\rangle, \langle OR, GL, S\rangle$. As can be seen, our particle filter produces a good approximation of the true densities.

## V. Discussion

Bayes filter has become a cornerstone in the area of single agent state estimation for partially observable settings. It finds important applications in robot localization and tracking, as well as POMDP planning. However, no analogous state estimation filter exists for multiagent settings, in part due to the difficulty in modeling the additional uncertainty of others' decisions, that entails reasoning with a nested interactive belief system. In this paper, we have filled this gap, by proposing a novel multiagent state estimation process for a rational setting that ascribes sophisticated nested models to others, predicts their actions using these models, and tracks these models by recursively updating them.

Our realistic assumption of an agent not knowing the true models of interacting agents *a'priori*

[7] We traced the exact belief updates manually for a fair comparison.

forces it to maintain beliefs over a continuum of other agents' models, thereby precluding practical implementations of our state estimation filter. To address this problem, we extended the bootstrap filter to our multiagent state estimation process. Consequently, our multiagent state estimation filter can be implemented approximately, with a substantial reduction in computation. Our experimental results on simple problem domains are indicative of the technique's favorable performance, both in terms of quality of approximation as well as speed of computation. These positive results motivate us to analyze its performance on more substantive applications, and this is one line of future work.

Particle filters have previously been successfully applied to approximate the belief update in continuous state single agent POMDPs [16]. In the multiagent setting, particle filters have been employed in collaborative multi-robot localization [3]. In this application, the state estimation process is significantly different from ours, with emphasis on predicting the position of the robot, and not the decisions of the other robots. Additionally, to facilitate fast localization, beliefs of other robots encountered during motion are considered to be fully observable.

## References

[1] A. Brandenburger and E. Dekel. Hierarchies of beliefs and common knowledge. *Journal of Economic Theory*, 59:189–198, 1993.

[2] A. Doucet, N. D. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[3] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots on Heterogenous Multi-Robot Systems*, 8(3), 2000.

[4] P. Gmytrasiewicz and P. Doshi. Interactive pomdps: Properties and preliminary results. In *Third Intl. Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, 2004.

[5] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. In *8th Intl Symposium on AI and Math*, Ft. Lauderdale, Florida, 2004.

[6] N. Gordon, D. Salmond, and A. Smith. Novel approach to non-linear/non-gaussian bayesian state estimation. *IEEE Proceedings-F*, 140(2):107–113, 1993.

[7] J. C. Harsanyi. Games with incomplete information played by 'bayesian' players. *Mgmt. Science*, 14(3):159–182, 1967.

[8] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[9] A. Heifetz and D. Samet. Topology-free typology of beliefs. *Journal of Economic Theory*, 82:324–341, 1998.

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

[11] S. C. Kramer and H. Sorenson. Recursive bayesian estimation using piecewise constant approximations. *Automatica*, 24:789–801, 1988.

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

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

[14] H. W. Sorenson, editor. *Kalman Filtering: Theory and Application*. IEEE Press, New York, 1985.

[15] H. W. Sorenson and D. L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica*, 7:465–479, 1971.

[16] S. Thrun. Monte carlo pomdps. In *Advances in Neural Information Processing 12*, pages 1064–1070, 2000.

**Function** SOLVEMODEL($\theta_k, l > 0$) **returns** $Pr(A_k|\theta_k)$ [a]
S$_{b_k^0}$ $\leftarrow \{is_k^{(n)}, n = 1...N | is_k^{(n)} \sim b_{k,l} \in \theta_k\}$
<u>*Reachability Analysis*</u>
reach(0) $\leftarrow$ S$_{b_k^0}$
**for** $t \leftarrow 1$ **to** $T$ **do**
  reach($t$) $\leftarrow \phi$
  **for all** S$_{b_k^{t-1}}$ $\in$ reach($t-1$), $a_k \in A_k$, $o_k \in \Omega_k$ **do**
    reach($t$) $\stackrel{\cup}{\leftarrow}$ PARTICLEFILTER(S$_{b_k^{t-1}}$, $a_k, o_k, l$)
<u>*Value Iteration*</u>
**for** $t \leftarrow T$ **downto** 0 **do**
  **for all** S$_{b_k^t}$ $\in$ reach($t$) **do**
    U(S$_{b_k^t}$) $\leftarrow -\infty$, OPT(S$_{b_k^t}$) $\leftarrow \phi$
    **for all** $a_k \in A_k$ **do**
      U$^{a_k}$(S$_{b_k^t}$) $\leftarrow 0$
      **for all** $is_k^{(n),t} = \langle s^{(n),t}, \theta_{-k}^{(n)} \rangle \in S_{b_k^t}$ **do**
        $Pr(A_{-k}|\theta_{-k}^{(n)}) \leftarrow$ SOLVEMODEL($\theta_{-k}^{(n)}, l-1$)
        **for all** $a_{-k} \in A_{-k}$ **do**
          U$^{a_k}$(S$_{b_k^t}$) $\stackrel{+}{\leftarrow} \frac{1}{N} R(s^{(n),t}, a_k, a_{-k}) Pr(a_{-k}|\theta_{-k}^{(n)})$
      **if** ($t < T$) **then**
        **for all** $o_k \in \Omega_k$ **do**
          sum $\leftarrow 0$
          **for all** $is_k^{(n),t} = \langle s^{(n),t}, \theta_{-k}^{(n)} \rangle \in S_{b_k^t}$ **do**
            $Pr(A_{-k}|\theta_{-k}^{(n)}) \leftarrow$ SOLVEMODEL($\theta_{-k}^{(n)}, l-1$)
            **for all** $a_{-k} \in A_{-k}$, $s^{t+1} \in S_k$ **do**
              sum $\stackrel{+}{\leftarrow} O_k(o_k|s^{t+1}, a_k, a_{-k})$
              $\times T(s^{t+1}|s^{(n),t}, a_k, a_{-k}) Pr(a_{-k}|\theta_{-k}^{(n)})$
          U$^{a_k}$(S$_{b_k^t}$) $\stackrel{+}{\leftarrow} \frac{1}{N}$ $\times$ sum $\times$ U(reach($t$)[$|\Omega_k|a_k + o_k$])
      **if** (U$^{a_k}$(S$_{b_k^t}$) $\geq$ U(S$_{b_k^t}$)) **then**
        **if** (U$^{a_k}$(S$_{b_k^t}$) $>$ U(S$_{b_k^t}$)) **then**
          U(S$_{b_k^t}$) $\leftarrow$ U$^{a_k}$(S$_{b_k^t}$)
          OPT(S$_{b_k^t}$) $\leftarrow \phi$
        OPT(S$_{b_k^t}$) $\stackrel{\cup}{\leftarrow} a_k$
**for all** $a_k \in A_k$ **do**
  **if** ($a_k \in$ OPT(S$_{b_k^0}$)) **then**
    $Pr(a_k|\theta_k) \leftarrow \frac{1}{|\text{OPT}(S_{b_k^0})|}$
  **else**
    $Pr(a_k|\theta_k) \leftarrow 0$
**return** $Pr(A_k|\theta_k)$
**end function**

[a] When $l = 0$, SOLVEMODEL() reduces to solving a POMDP given an initial belief. We omit the algorithm due to lack of space.

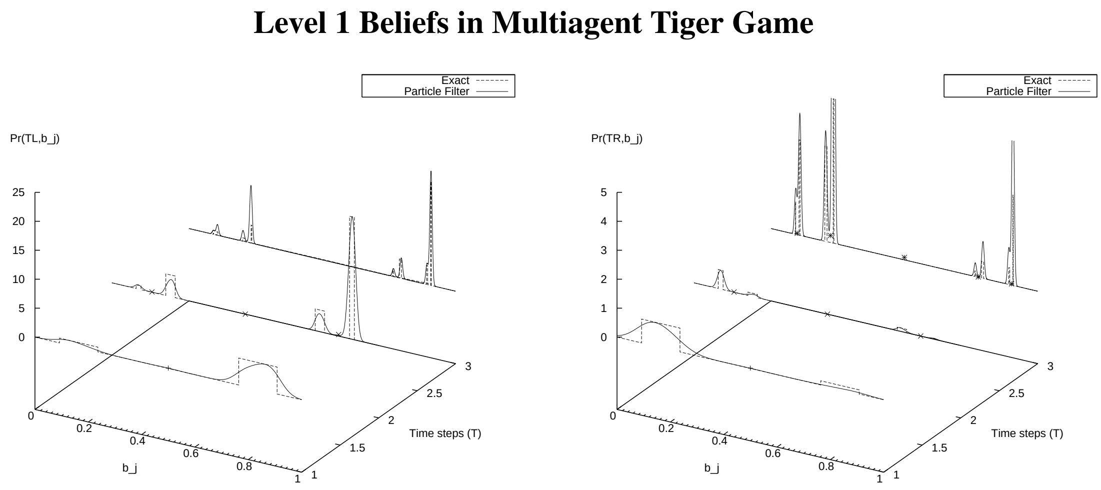
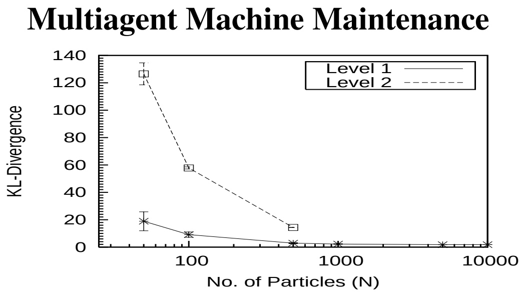



Fig. 5. The exact and approximate p.d.f.s after successive filtering steps. The peaks of the approximate p.d.f.s align correctly with those of the exact p.d.f.s, and the areas under the approximate and exact p.d.f.s are approximately equal.

**Multiagent Tiger**

$(i)$




$(ii)$

Fig. 6. Performance of the particle filter as a function of the number of particles, for $(i)$ level 1 beliefs, $(ii)$ level 2 beliefs.

| **Belief** | **Game** | **Particle Filter** | | **Numerical Integration** |
|---|---|---|---|---|
| | | **N=500** | **N=1000** | |
| Level 1 | Multiagent Tiger | 0.148s ± 0.001s | 0.332s ± 0.007s | 21.80s ± 0.036s |
| | Multiagent MM | 0.452s ± 0.009s | 0.931s ± 0.0146s | 1m 18.20s ± 0.45s |
| Level 2 | Multiagent Tiger | 2m 23.28s ± 1.1s | 11m 41.30s ± 1.52s | 25m 36.12s ± 5.66s |
| | Multiagent MM | 1m 37.59s ± 0.17s | 8m 27.29s ± 1.65s | 75m 44.74s ± 1m 55.73s |

Fig. 7. Comparison of the average running times of our numerical integration and particle filter implementations on same platform (*Pentium 4, 1.7GHz, 512MB RAM, Linux)*.