Structured Parameter Elicitation

Li Ling Ko
Graduate School for Integrative Sciences & Engineering, National University of Singapore
Singapore 117456, Singapore

David Hsu and **Wee Sun Lee** and **Sylvie C. W. Ong**
Department of Computer Science, National University of Singapore
Singapore 117417, Singapore

Abstract

The behavior of a complex system often depends on parameters whose values are unknown in advance. To operate effectively, an autonomous agent must *actively* gather information on the parameter values while progressing towards its goal. We call this problem *parameter elicitation*. Partially observable Markov decision processes (POMDPs) provide a principled framework for such uncertainty planning tasks, but they suffer from high computational complexity. However, POMDPs for parameter elicitation often possess special structural properties, specifically, factorization and symmetry. This work identifies these properties and exploits them for efficient solution through a factored belief representation. The experimental results show that our new POMDP solvers outperform SARSOP and MOMDP, two of the fastest general-purpose POMDP solvers available, and can handle significantly larger problems.

Introduction

Planning with incomplete and imperfect information is an essential capability for autonomous agents interacting with the physical world. Often the behavior of a complex physical system depends on parameters whose values are unknown in advance. To operative effectively, an autonomous agent must *actively* gather information on the parameter values while progressing towards its goal. We call this problem *parameter elicitation*. Consider the examples below:

- An automated telephone booking system needs to determine a user's desired flight and issue a ticket (Williams and Young 2007). The unknown parameters may include a user's origin, destinations, and travel dates. The system must ask as few questions as possible to get reliable estimates of the parameter values and issue the ticket.
- A planetary rover explores an area for rocks with scientific value (Smith and Simmons 2004). It knows the rocks' locations, but not their value, *i.e.*, the parameters. The rover needs to find as many valuable rocks as possible and reach a final destination quickly to report the information.
- An underwater acoustic modem needs to find the best parameter settings of communication channels to transmit a data file quickly (Shankar, Chitre, and Jayasuria 2010).

In all these distinct domains, the common difficulty in planning is to gather enough information on the unknown parameters for efficient operation.

POMDPs provide a principled framework for such planning tasks. They allow for optimal trade-off between information gathering and goal achievement. However, it is computationally intractable to solve POMDPs exactly. In a discrete POMDP, the belief space $\mathcal{B}$ has dimensionality roughly equal to $|S|$, the number of states. The size of $\mathcal{B}$ thus grows exponentially with $|S|$. Despite the dramatic progress of point-based approximate POMDP solvers in recent years (Pineau, Gordon, and Thrun 2003; Smith and Simmons 2004; Kurniawati, Hsu, and Lee 2008), complex systems with large state spaces still pose significant challenges. Consider a preference elicitation task such as the automated booking system described earlier. Adding some standard preferences (airlines, seating, food restrictions, *etc.*), we easily end up with 10 preference slots, each having up to 10 values on the average. This results in a state space with roughly 10^{10} states, a size that no general-purpose POMDP solvers can handle today.

Parameter elicitation, however, possesses special structural properties. Despite being unknown, the parameter values are constant and do not change over time. Furthermore, although the parameters are interdependent, their dependencies can usually be captured in a relatively sparse graph. We show that under reasonable assumptions, this leads to a *factored belief space* over the parameter values, and the space remains factored as the state of the system changes over time. This contrasts with general dynamic Bayesian networks (DBNs), where the variables—in this case, the parameters—become coupled over time. We identify the class of POMDPs that allow a factored belief representation and exploit this for greater computational efficiency in our new POMDP solver. Finally, in many parameter elicitation problems, we are interested in the correct parameter value, but are not biased towards any particular value *a priori*. For example, an automated booking system wants to know the user's intended destination, regardless of it being Paris, New York, or Tokyo. Our solver further improves efficiency by exploiting such *symmetry*.

Preliminaries

A POMDP models an agent taking a sequence of actions under uncertainty to maximize its total reward. Formally a discrete POMDP with an infinite horizon is specified as a tuple $(S, A, O, T, Z, R, \gamma)$, where S is a set of states, A is a set of actions, and O is a set of observations.

In each time step, the agent takes an action $a \in A$ and

moves from a state $s \in S$ to $s' \in S$. Due to the uncertainty in action, the end state s' is specified by a conditional probability function $T(s, a, s') = p(s'|s, a)$, which gives the probability that the agent lies in s', after taking action a in state s. The agent then makes an observation on the end state s'. Due to the uncertainty in observation, the observation result $o \in O$ is again specified by a conditional probability function $Z(s', a, o) = p(o|s', a)$. In each step, the agent receives a real-valued reward $R(s, a)$ if it takes action a in state s. The agent's goal is to maximize its expected total reward by choosing a suitable sequence of actions. When the action sequence has infinite length, we typically specify a discount factor $\gamma \in (0, 1)$ so that the total reward is finite and the problem is well defined. In this case, the expected total reward is given by $\mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, where s_t and a_t denote the agent's state and action at time t.

In POMDP planning, we compute an *optimal policy* that maximizes the agent's expected total reward. A POMDP policy $\pi \colon \mathcal{B} \to A$ maps a belief $b \in \mathcal{B}$—which is a probability distribution over S—to a prescribed action $a \in A$.

A policy π induces a value function V_π, which specifies the expected total reward $V_\pi(b)$ of executing π starting from b. It is well-known that V^*, the value function for an optimal policy π^*, can be approximated arbitrarily closely by a piecewise-linear, convex function. As a result, a value function V can be represented as a set Γ of vectors, commonly called α-*vectors*: $V(b) = \max_{\alpha \in \Gamma}(\alpha \cdot b)$.

More information on POMDPs is available in (Kaelbling, Littman, and Cassandra 1998).

Factored Parameter Elicitation as a POMDP

Parameter elicitation can be modeled as a POMDP. In addition to the usual elements, this POMDP contains one more element Θ, which is a set of parameter values. The unknown parameter value $\theta \in \Theta$ may affect the system dynamics. So we need to modify the state-transition function: $T(s, a, s', \theta) = p(s'|s, a, \theta)$. We also need actions that gather information on θ and observations that provide such information. We capture this in the observation function: $Z(s', a, o, \theta) = p(o|s', a, \theta)$. The solution to the POMDP is an optimal policy that gathers enough information on θ for the agent to operate as efficiently as possible.

To solve this POMDP efficiently, we must exploit the underlying structural properties of the parameter elicitation problem. Consider a complex system whose behavior depends on many parameters. Often these parameters are not all interdependent. Instead, they form groups, each of which affects only one system component, with limited dependencies between parameters from different groups. We can exploit this to factor the belief on parameters.

In general, we assume that the initial belief on parameters has the factored form $p(\boldsymbol{\theta}) = \prod_i f_i(\boldsymbol{\theta}_i)$, where $\boldsymbol{\theta}$ is the set of all parameters and $\boldsymbol{\theta}_i \subseteq \boldsymbol{\theta}$ is the subset of parameters in factor f_i. We can think intuitively of $\boldsymbol{\theta}_i$ as a group of parameters for a particular system component, but different parameter groups $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are not necessarily disjoint.

For the belief $p(\boldsymbol{\theta})$ to remain factored the same way over time, we need additional structural properties. First, consider the state variables. There are no restrictions on dependencies among the state variables. In Figure 1, for example, we have edges (x_1, x'_2) and (x_2, x'_3). We can also have (x_3, x'_1) and (x_1, x_2) if needed. However, all the state variables must be fully observable. Otherwise, the system dynamics eventually couples all the parameters over time, because the state variables depend on the parameters.

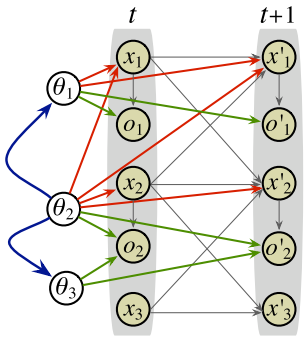

Figure 1. Factored parameter elicitation. The shaded nodes are fully observable. Only representative dependencies are shown. The belief over the parameters θ_1, θ_2, and θ_3 remain factored for all t: $p(\theta_1, \theta_2, \theta_3) = f_1(\theta_1, \theta_2) \cdot f_2(\theta_2, \theta_3)$.

Furthermore, if a state variable depends on a subset $\boldsymbol{\varphi}$ of parameters, then $\boldsymbol{\varphi} \subseteq \boldsymbol{\theta}_i$ for some i. In Figure 1, x'_1 may depend on both θ_1 and θ_2, as θ_1 and θ_2 belong to the same factor f_1. No state variable depends on both θ_1 and θ_3.

The dependencies of observation variables on parameters are subject to the same requirements as those of state variables. However, there are no restrictions on the dependencies of observation variables on state variables.

These conditions are formally stated below and illustrated in Figure 1.

Proposition 1 Factored parameter elicitation *(FPE) can be modeled as a POMDP with a set $\boldsymbol{\theta}$ of unknown parameters. It satisfies the following conditions:*

1. *The initial belief $p(\boldsymbol{\theta})$ at time $t = 0$ has the factored form $p(\boldsymbol{\theta}) = \prod_i f_i(\boldsymbol{\theta}_i)$.*
2. *All state variables are fully observable.*
3. *If a state variable or an observation variable depends on a subset $\boldsymbol{\varphi}$ of parameters, then $\boldsymbol{\varphi} \subseteq \boldsymbol{\theta}_i$ for some i.*

Under these conditions, the belief on $\boldsymbol{\theta}$ retains the same factored form $p(\boldsymbol{\theta}) = \prod_i f_i(\boldsymbol{\theta}_i)$ for all time t.[1]

Consider again the examples in the introduction. In the automated booking system, the parameter set $\boldsymbol{\theta}$ contains the user's various preferences, initially unknown to the system. Some preferences are dependent. For example, travel dates often restrict the available flights. The system asks questions about the user's preferences. The user's answers are the observations, which are noisy due to limitations in speech recognition technology. Each question and answer deal with only a single preference. In other words, each observation depends on a single parameter. Even if a question asks for multiple preferences, we can usually decompose the answer into components, each addressing one preference only. The automated booking systems belongs to a type of preference elicitation tasks called *slot-filling dialog* (SFD), in which each slot contains a user preference and the agent's goal is to fill in the values of all slots (Williams and Young 2007).

The planetary rover domain is commonly known as Rock Sample (Smith and Simmons 2004). The parameters are binary variables, each indicating whether a particular rock is

[1] The proofs of all propositions appear in the appendix.

valuable. There is one state variable specifying the rover's position. There is another set of state variables, each specifying whether a rock has been sampled. All state variables are fully observable. The rover can take noisy long-range sensor readings to gather information on the rocks. It can also sample a rock in the immediate vicinity to determine its scientific value. Both long-range sensing and sampling occur on one rock at a time.

In Rock Sample, the state variables for the rover and the rocks do not depend on any unknown parameters. Consider a variant, in which unknown ground conditions, *e.g.*, friction, affect the rover's motion. Since the rover's position is fully observable, the information can be used to infer the unknown parameter value that characterizes ground conditions, in order to improve the rover's motion control. So this more realistic variant can also be modeled as FPE.

Slot-filling dialog appears completely unrelated to the robot exploration task in Rock Sample; however, they can both be modeled as FPE.

Algorithm

Recent point-based POMDP solvers typically represent the value function V as a set of α-vectors. In FPE, beliefs are factored, but α-vectors cannot be similarly factored, even when the reward function is additive (Koller and Parr 1999).

Another way of representing V is to compute and store the values of V at a set of sampled beliefs. We then approximate $V(b)$ at a belief b by interpolating the values of V at sampled beliefs close to b under a suitable distance metric (Bonet, Caracas, and Geffner 2009). However, a key difficulty remains for FPE. Computing the distance naively, by expanding the factored beliefs, is expensive and scales poorly with the number of parameters. It is essential to compute the distance between beliefs entirely in the factored representation.

In FPE, a belief $b \in \mathcal{B}$ is represented as a pair (s, b_Θ), where s is the state, which is fully observable, and b_Θ is the belief over parameters. For interpolation, only beliefs with the same s values are useful. To simplify the presentation, we assume here that the factored belief b_Θ can be represented as a tree T. A node i of T represents a parameter θ_i in the FPE model and has an associated conditional probability distribution $p(\theta_i|\theta_{\rho(i)})$, where $\rho(i)$ denotes the parent of a node i in T. For general factored beliefs, we can use junction trees (Lauritzen and Spiegelhalter 1988).

We now introduce the tree distance d_T between beliefs:

Definition 1 *Suppose that two beliefs* $b_\Theta(\theta) = \prod_i p(\theta_i|\theta_{\rho(i)})$ *and* $b'_\Theta(\theta) = \prod_i p'(\theta_i|\theta_{\rho(i)})$ *have the same factored form represented as a tree* T*. The tree distance* d_T *between* b_Θ *and* b'_Θ *is*

$$d_T(b_\Theta, b'_\Theta) = \sum_i \sum_{\theta_i, \theta_{\rho(i)}} |p(\theta_i|\theta_{\rho(i)}) - p'(\theta_i|\theta_{\rho(i)})|.$$

The next proposition shows that the L_1 distance between two tree-structured beliefs is upper bounded by d_T.

Proposition 2 *Given two beliefs* b_Θ *and* b'_Θ *with the same factored form represented as a tree* T*, we have*

$$||b_\Theta - b'_\Theta||_1 = \sum_\theta |b_\Theta(\theta) - b'_\Theta(\theta)| \leq d_T(b_\Theta, b'_\Theta).$$

The FPE algorithm uses d_T for interpolating the value function V in a factored belief space. In this work, the parameters are assumed to be discrete. To search for nearby beliefs efficiently, we represent each (discrete) conditional probability distribution in a tree T as a table. We discretize and hash the table entries into regular cells, allowing constant-time lookup of nearby beliefs. By Proposition 2, beliefs in the same cell have similar values of V. Our algorithm thus assumes that they have the *same* value (Bonet, Caracas, and Geffner 2009).

Following the recent point-based POMDP planning approach (Smith and Simmons 2004; Kurniawati, Hsu, and Lee 2008), the FPE algorithm computes a policy by sampling beliefs from the reachable belief space $\mathcal{R} \subseteq \mathcal{B}$, the set of points reachable from a given initial belief $b_0 \in \mathcal{B}$ under arbitrary sequences of actions and observations. At each sampled belief, the algorithm maintains upper and lower bounds on V. The sampled beliefs form a search tree $T_\mathcal{R}$, where each node represents a sampled belief (s, b_Θ) in $\mathcal{R}$ and the root is the initial belief b_0. To sample new beliefs, we start from the root of $T_\mathcal{R}$ and traverse a single path down. At a node (s, b_Θ) along the path, we choose action a with the highest upper bound, as well as state s' and observation o that make the largest contribution to the gap between the upper and lower bounds at the root of $T_\mathcal{R}$. We then compute a new belief b'_Θ over θ: $b'_\Theta(\theta) = \eta Z(s', a, o, \theta) T(s, a, s', \theta) b_\Theta(\theta)$, where η is a normalization constant. The new node (s', b'_Θ) is inserted into $T_\mathcal{R}$ as a child of (s, b_Θ). The sampling path terminates when it reaches a sufficient depth to improve the bounds at the root of $T_\mathcal{R}$. We then go back up this path to the root and perform lower and upper bound backup at each node along the way. Backup uses the standard Bellman update, which is an iteration of dynamic programming that improves the value at a belief by looking ahead one step further. See, *e.g.*, (Kurniawati, Hsu, and Lee 2008) for details.

To construct the initial bounds at a belief, we may use general techniques. By assuming that all parameters are fully observable, we obtain a Markov decision process (MDP) solution, which provides an upper bound. To obtain a lower bound, we can use any arbitrary policy. Alternatively, we may construct specific bounds for a given problem. This is further described in the section on experimental results.

To execute a policy, we use the lower bound on V together with one-step look-ahead search.

FPE with Symmetries

Symmetry is common in certain classes of parameter elicitation problems, such as slot-filling dialog systems. Here, we show how to exploit symmetry within FPE.

Symmetry allows us to identify the same POMDPs by renaming states, actions and observations. Specifically, let $k\colon S \to S$ be a permutation on the states, $g\colon A \to A$ be a permutation on the actions, and $h\colon O \to O$ be a permutation on the observations. If the following conditions hold

$$\begin{aligned} T(s, a, s') &= T(k(s), g(a), k(s')) \\ R(s, a) &= R(k(s), g(a)) \\ Z(s', a, o) &= Z(k(s'), g(a), h(o)), \end{aligned}$$

then the optimal value function V^* is symmetric, in the sense that $V^*(b) = V^*(b')$ when b' is obtained from the

belief vector b by permuting the elements of b according to k (Kim 2008).

In a multiple-slot dialog system, each slot has a set of possible values (for example, Paris, New York and Tokyo are possible destinations in an automated booking system), and the states are combinations of user preferences for the values of each slot. User preferences are assumed to be static, so the first condition above holds. Typical actions include querying the value of a slot and confirming it. It is reasonable to assume that the cost of such actions is independent of the value of a slot or even of the slot identity, satisfying the second condition. Furthermore, we can reasonably assume that the observation errors due to confirmation actions are independent of the slot values, satisfying the third condition for such actions. For query actions, if we permute the values within a slot, we can permute the values of observations to satisfy the third condition as well. Under these assumptions, it follows that the optimal value function is symmetric with respect to permuting the values within a slot.

To exploit the symmetry, we permute beliefs into a canonical representation, then discretize and hash the canonical belief using the approach described in the previous section. Here, we restrict ourselves to permuting values of single parameters independently. This retains the form of the factored belief. Let each parameter variable be $\theta_i \in \boldsymbol{\theta}$, and denote Θ_i as the space of parameter values for θ_i. Let $k_i : \Theta_i \rightarrow \Theta_i$ be a permutation on the parameter values of θ_i. Without loss of generality, assume that there are n variables. We then have $V^*(b) = V^*(b')$ if $b(\theta_1, \theta_2, \cdots, \theta_n) = b'(k_1(\theta_1), k_2(\theta_2), \cdots, k_n(\theta_n))$, for all possible permutations k_i on the parameter values of each θ_i.

Ideally, every belief should be permuted into its canonical form. Unfortunately, even for two variable beliefs, this problem is as hard as graph isomorphism, a problem for which no polynomial time solution is known. Thus, we settle for a simple but fast algorithm that maps many, but not all, symmetric beliefs into the same representation.

As before, we work with tree structured beliefs where the factor associated with parameter variable θ_i has the form $p(\theta_i|\theta_{\rho(i)})$, with $\theta_{\rho(i)}$ denoting the parent of θ_i. For each pair of parent-child, we sort the elements of the factor associated with the parent first. Then given the new ordering of the parent's values, we sort the elements of the factor associated with the child.

Specifically, for each θ_i, the factor $p(\theta_i|\theta_{\rho(i)})$ is represented as a 2-D table where the columns are indexed by the values of $\theta_{\rho(i)}$ and the rows are indexed by the values of θ_i. The factor at the root $p(\theta_0)$ is an exception, it is a 1-D list of elements and is sorted in ascending order. Now, assume that $\theta_{\rho(i)}$'s values have already been sorted. The rows of the table (values of θ_i) are then sorted in lexicographic order. Assuming that there are no ties in sorting any of the tables, this method will map any belief into its unique canonical representation. For tied values, we make a best effort at sorting by using the conditional distribution of the child. We first sort the elements of each tied column within the column itself. We then sort the resulting tied columns lexicographically.

Experimental Results

Slot-Filling Dialog. To test our algorithms, we constructed four abstract SFD problems (Figure 2) with the factorization and symmetry properties described earlier. The

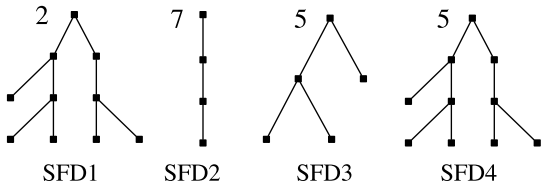

Figure 2. Four slot-filling dialog problems. Each node of a tree represents a preference slot, *i.e.*, parameter. The tree structure encodes the dependencies between the parameters – each parent-child pair in a tree belong to a factor in the belief. All slots have the same number of possible values, indicated by the number next to the tree.

agent needs to determine the values of all slots. In each time step, it asks the user a question. A "what" question queries the user for the value of a slot. A confirmation question verifies the value of a slot with the user. Each question incurs a cost. The agent may submit the result at any time. If the submission contains the correct values for all the slots, the agent receives a large reward. Otherwise, it incurs a large penalty. The agent may also choose to give up and call for human assistance. In this case, it receives a small penalty. To get high total reward, the agent must determine the correct slot values and ask as few questions as possible.

We implemented and tested the FPE and FPES algorithms on these four problems. For the initial lower bound, we compare the value of immediate submission with that of asking a single question before submission, and use the better value. For the initial upper bound, we simply use the MDP solution.

For performance comparison, we ran SARSOP (Kurniawati, Hsu, and Lee 2008), one of the fastest existing POMDP solvers, using the APPL v0.3 package. All the algorithms were implemented in C++. The tests were conducted on a 2.66GHz computer with 2GB of memory.

First we ran SARSOP until it converged or ran out of memory. We then ran our algorithms for FPE and FPES to reach or exceed the reward level that SARSOP achieved. On problems where SARSOP was unable to load, we ran FPE and FPES to their maximum reward levels. The results are shown in Table 1. For each problem, $|\Theta|$ is the total number of different parameter value combinations over all the slots, and $|S|, |A|$, and $|O|$ are the number of states, actions, and observations, respectively. They give a rough indication of the problem size. Column 2 of the table shows the expected total reward (ETR) for the policies that the three algorithms computed, and column 3 shows the corresponding times for policy computation. We estimated ETR by performing a large number of simulations. The data clearly shows that FPE outperformed SARSOP by several times. The speedup of FPES was even more dramatic. In particular, SARSOP could not even load the largest problem, SFD4, which has about 10 million parameter value combinations. In contrast, FPES computed a reasonable policy in about 2 minutes.

Column 4 reports the number of beliefs explored during policy computation. It provides some understanding of how the speedup was achieved. First compare SARSOP and FPE. To achieve a comparable reward level, SARSOP used fewer beliefs than FPE. SARSOP uses α-vectors as a value function representation, which generalizes better. Thus it does not need many beliefs. At the same time, maintaining α-vectors is computationally expensive. FPE uses nearest

Table 1. Performance comparison on SFD problems.

	Reward	Time (s)	$\lvert B\rvert$ (10^3)
SFD1 ($\lvert\Theta\rvert$=1024, $\lvert S\rvert$=2, $\lvert A\rvert$=1054, $\lvert O\rvert$=23)			
SARSOP	32.2 ± 0.5	2,275	1.0
FPE	32.4 ± 0.3	126	2.0
FPES	32.7 ± 0.3	66	0.7
SFD2 ($\lvert\Theta\rvert$=2401, $\lvert S\rvert$=2, $\lvert A\rvert$=2433, $\lvert O\rvert$=31)			
SARSOP	11.3 ± 0.7	7,549	0.3
FPE	25.8 ± 0.4	1,870	1.0
FPES	27.5 ± 0.4	2	0.04
SFD3 ($\lvert\Theta\rvert$=3125, $\lvert S\rvert$=2, $\lvert A\rvert$=3155, $\lvert O\rvert$=28)			
SARSOP	-0.2 ± 0.8	12,236	0.3
FPE	5.9 ± 0.4	898	2.0
FPES	1.7 ± 0.3	51	0.2
SFD4 ($\lvert\Theta\rvert$=$5^{10}\approx 1.0\times10^7$, $\lvert S\rvert$=2, $\lvert A\rvert\approx 1.0\times10^7$, $\lvert O\rvert$=53)			
SARSOP	—	—	—
FPE	0.8 ± 0.4	8,364	2.0
FPES	6.7 ± 0.4	132	0.1

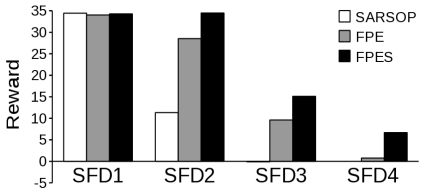

Figure 3. Highest expected total reward achieved.

neighbor interpolation for value function representation. It does not generalize as well. However, without α-vectors, FPE can explore the belief space much faster. The trade-off seems beneficial to FPE in this case. By exploiting symmetries, FPES recognizes that many beliefs are "equivalent" and avoids exploring such beliefs repeatedly, thus achieving further gain in efficiency. During the exploration, the addition of a crucial belief can sometimes result in a dramatically improved policy with a resultant discontinuous jump in the reward level. In SFD2 and SFD3, the ETRs reported for FPE and FPES are at the point of such discontinuities.

We also ran all three algorithms sufficiently long to determine the highest ETR that they can achieve in a maximum of 4 hours of computation time. The results, shown in Figure 3 are consistent with those in Table 1. In the smallest problem, SFD1, the three algorithms achieved similar ETR. On the larger problems, FPE and FPES were much better.

Rock Sample. Following the same protocol as that for SFD, we further compared the algorithms on the standard Rock Sample problem with a 5×5 grid map and N rocks, denoted as RS[5,N]. Unlike SFD, the parameters in Rock Sample are independent. There is also no symmetry in Rock Sample. On the other hand, the presence of fully observable state variables makes the problem a mixed observability Markov decision process (MOMDP) (Ong et al. 2009). Like SARSOP, the MOMDP algorithm uses α-vectors to represent the value function. However, it speeds up computation by exploiting the fully observable state variables to represent the belief space as a union of lower-dimensional subspaces and calculating the value function in these subspaces only. We compared FPE with both SARSOP and MOMDP.

In FPE, the initial lower bound is obtained from an open-loop policy that greedily samples the rock that currently looks the best in terms of distance and likelihood of being valuable. The initial upper bound is obtained by a further relaxation of the MDP solution to single-rock problems. For each rock, we compute an MDP solution to the corresponding single-rock problem RS[5,1] and then sum up the value functions for all the MDP solutions to derive an upper bound for the original problem RS[5,N].

Table 2. Performance comparison on Rock Sample problems.

	Reward	Time (s)	$\lvert B\rvert$ (10^3)
RS[5,5] ($\lvert\Theta\rvert$=32, $\lvert S\rvert$=26, $\lvert A\rvert$=10, $\lvert O\rvert$=2)			
SARSOP	18.7 ± 0.4	1	0.20
MOMDP	18.7 ± 0.4	0.2	0.35
FPE	18.7 ± 0.4	1	0.03
RS[5,9] ($\lvert\Theta\rvert$=512, $\lvert S\rvert$=26, $\lvert A\rvert$=14, $\lvert O\rvert$=2)			
SARSOP	24.5 ± 0.5	66	0.8
MOMDP	24.5 ± 0.4	6	2.1
FPE	24.5 ± 0.5	11	39
RS[5,12] ($\lvert\Theta\rvert$=4096, $\lvert S\rvert$=26, $\lvert A\rvert$=17, $\lvert O\rvert$=2)			
SARSOP	25.1 ± 0.4	1307	1
MOMDP	25.2 ± 0.5	810	16
FPE	25.2 ± 0.5	201	544
	26.1 ± 0.5	601	1636
RS[5,16] ($\lvert\Theta\rvert$=65,536, $\lvert S\rvert$=26, $\lvert A\rvert$=21, $\lvert O\rvert$=2)			
SARSOP	—	—	—
MOMDP	—	—	—
FPE	28.4 ± 0.5	601	1000

Table 2 shows that FPE again outperformed SARSOP substantially. MOMDP performed better than FPE on the two smaller problems, RS[5,5] and RS[5,9], but it could not match FPE on the larger problems. In fact, SARSOP and MOMDP could not even load the largest problem, RS[5,16], which has $65,536$ parameter value combinations. When the algorithms were run long enough, they achieved roughly the same highest ETR on RS[5,5] and RS[5,9]. On RS[5,12] however, FPE achieved an ETR level of 26.1 ± 0.5, while SARSOP and MOMDP could not reach this level before running out of memory.

The experiments on SFD and Rock Sample show a clear advantage of FPE and FPES on large problems, where the overhead of maintaining α-vectors in SARSOP and MOMDP significantly slows down belief space exploration.

Related Works

Parameter elicitation is closely related to preference elicitation (Boutilier 2002) and Bayesian reinforcement learning (Poupart et al. 2006). Here we focus on discrete parameter values only, but allow dependencies among parameters.

Factored POMDPs (Boutilier and Poole 1996) refer to POMDPs with structured transition function representations. In general, they do not factor beliefs, but exploit the structure in transition functions using representations such as algebraic decision diagrams. In contrast, we identify structural properties that allow beliefs to be factored. Factored-belief POMDPs have been studied in an online search setting (Paquet, Tobin, and Chaib-draa 2005), but we show how to approximate the value function in an offline setting.

Earlier work has exploited symmetry to accelerate POMDP solution, but is only applicable to "flat" beliefs (Doshi and Roy 2008; Kim 2008). Our algorithm works on factored beliefs, with dependencies among parameters.

Conclusion

This paper introduces structured parameter elicitation. Apparently unrelated domains such as slot-filling dialog and Rock Sample are both parameter elicitation in essence. We model parameter elicitation as a POMDP and show how to solve it efficiently by exploiting the underlying structural properties, specifically, factorization and symmetry. Experiments show that the resulting POMDP solvers outperform SARSOP and MOMDP, two of the fastest general-purpose POMDP solvers, and can handle large domains that are impossible for general-purpose solvers to even load.

We plan to extend our algorithm for FPE to handle the more general case of beliefs that are approximately factored. Although the beliefs in a factored POMDP (Boutilier and Poole 1996) do not factor in general, we can approximate them by projection onto a factored belief space (McAllester and Singh 1999). An application of approximate factorization is the acoustic-modem file transmission domain described at the beginning of the paper.

Acknowledgements. This work is supported in part by MoE AcRF grant R-252-000-327-112 and MDA Singapore-MIT GAMBIT Game Lab grant R-252-000-398-490.

Appendix

Proof. (Proposition 1)
For simplicity, we refer to the observed states as well as the observations up to the current time as $\mathbf{z}$. We have $p(\boldsymbol{\theta}|\mathbf{z}) = \eta p(\boldsymbol{\theta})p(\mathbf{z}|\boldsymbol{\theta})$, where η is the normalizing constant. The term $p(\boldsymbol{\theta})$ has a factored form $\prod_i f_i(\boldsymbol{\theta}_i)$. Let z_k be a component of $\mathbf{z}$ and let $\rho(z_k)$ be the parents of z_k that do not include variables in $\boldsymbol{\theta}$. Under the assumptions, the conditional distribution of z_k has the form $p(z_k|\rho(z_k), \boldsymbol{\theta}_i)$ for some $\boldsymbol{\theta}_i$. The term $p(\mathbf{z}|\boldsymbol{\theta})$ is formed by the product of functions of the form $p(z_k|\rho(z_k), \boldsymbol{\theta}_i)$; hence, it has the same factored form as $p(\boldsymbol{\theta})$. As both $p(\boldsymbol{\theta})$ and $p(\mathbf{z}|\boldsymbol{\theta})$ have the form $\prod_i f_i(\boldsymbol{\theta}_i)$, their product also has the same factored form. □

Proof. (Proposition 2)
We can upperbound the L_1 distance between beliefs $b_\Theta(\theta)$ and $b'_\Theta(\theta)$ as follows:

$$\begin{aligned}
\sum_\theta |b_\Theta(\theta) - b'_\Theta(\theta)| &= \sum_\theta \left| \prod_i p(\theta_i|\theta_{\rho(i)}) - \prod_i p'(\theta_i|\theta_{\rho(i)}) \right| \\
&= \sum_\theta \left| \prod_i p(\theta_i|\theta_{\rho(i)}) - p(\theta_1|\theta_{\rho(1)}) \prod_{i=2}^n p'(\theta_i|\theta_{\rho(i)}) \right. \\
&\quad \left. + p(\theta_1|\theta_{\rho(1)}) \prod_{i=2}^n p'(\theta_i|\theta_{\rho(i)}) - \prod_i p'(\theta_i|\theta_{\rho(i)}) \right| \\
&\leq \sum_\theta p(\theta_1|\theta_{\rho(1)}) \left| \prod_{i=2}^n p(\theta_i|\theta_{\rho(i)}) - \prod_{i=2}^n p'(\theta_i|\theta_{\rho(i)}) \right| \\
&\quad + \sum_\theta |p(\theta_1|\theta_{\rho(1)}) - p'(\theta_1|\theta_{\rho(1)})| \prod_{i=2}^n p'(\theta_i|\theta_{\rho(i)}) \\
&\leq \sum_{k=1}^n \sum_\theta \prod_{i=1}^{k-1} p(\theta_i|\theta_{\rho(i)}) \, |p(\theta_k|\theta_{\rho(k)}) \\
&\quad - p'(\theta_k|\theta_{\rho(k)})| \prod_{j=k+1}^n p'(\theta_j|\theta_{\rho(j)}) \\
&\leq \sum_{k=1}^n \sum_{\theta_k, \theta_{\rho(k)}} p(\theta_{\rho(k)}) |p(\theta_k|\theta_{\rho(k)}) - p'(\theta_k|\theta_{\rho(k)})|.
\end{aligned}$$

The last line follows from marginalizing the left and right hand side of $|p(\theta_k|\theta_{\rho(k)}) - p'(\theta_k|\theta_{\rho(k)})|$, which are both tree structured distributions assuming that k is topologically ordered. □

References

Bonet, B.; Caracas, V.; and Geffner, H. 2009. Solving POMDPs: RTDP-Bel vs. point-based algorithms. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*.

Boutilier, C., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proc. Nat. Conf. on Artificial Intelligence*.

Boutilier, C. 2002. A POMDP formulation of preference elicitation problems. In *Proc. Nat. Conf. on Artificial Intelligence*.

Doshi, F., and Roy, N. 2008. The Permutable POMDP: Fast solutions to POMDPs for preference elicitation. In *Proc. Int. Conf. on Autonomous Agents & Multiagent Systems*.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1–2):99–134.

Kim, K. 2008. Exploiting symmetries in POMDPs for point-based algorithms. In *Proc. Nat. Conf. on Artificial Intelligence*.

Koller, D., and Parr, R. 1999. Computing factored value functions for policies in structured MDPs. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*.

Kurniawati, H.; Hsu, D.; and Lee, W. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*.

Lauritzen, S. L., and Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *J. Roy. Stat. Soc. B* 50(2): 157-224.

McAllester, D., and Singh, S. 1999. Approximate planning for factored POMDPs using belief state simplification. In *Proc. Uncertainty in Artificial Intelligence*.

Ong, S.C.W.; Png, S.W.; Hsu, D.; and Lee, W. 2009. POMDPs for robotic tasks with mixed observability. In *Proc. Robotics: Science and Systems*.

Paquet, S.; Tobin, L.; and Chaib-draa, B. 2005. An online POMDP algorithm for complex multiagent environments. In *Proc. Int. Conf. on Autonomous Agents & Multiagent Systems*.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*.

Poupart, P.; Vlassis, N.; Hoey, J.; and Regan, K. 2006. An analytic solution to discrete Bayesian reinforcement learning. In *Proc. Int. Conf. on Machine Learning*.

Shankar, S.; Chitre, M.; and Jayasuria, M. 2010. Data driven algorithms to tune physical layer parameters of an underwater communication link. Submitted to IEEE/MTS Oceans.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Proc. Uncertainty in Artificial Intelligence*.

Williams, J., and Young, S. 2007. Scaling POMDPs for spoken dialog management. *IEEE Trans. on Audio Speech & Language Processing* 15(7):2116–2129.