

Bounded-Parameter Partially Observable Markov Decision Processes

Yaodong Ni and Zhi-Qiang Liu

School of Creative Media
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China
Y.D.Ni@student.cityu.edu.hk, ZQ.LIU@cityu.edu.hk

Abstract

The POMDP is considered as a powerful model for planning under uncertainty. However, it is usually impractical to employ a POMDP with exact parameters to model precisely the real-life situations, due to various reasons such as limited data for learning the model, etc. In this paper, assuming that the parameters of POMDPs are imprecise but bounded, we formulate the framework of bounded-parameter partially observable Markov decision processes (BPOMDPs). A modified value iteration is proposed as a basic strategy for tackling parameter imprecision in BPOMDPs. In addition, we design the UL-based value iteration algorithm, in which each value backup is based on two sets of vectors called U-set and L-set. We propose four typical strategies for setting U-set and L-set, and some of them guarantee that the modified value iteration is implemented through the algorithm. We analyze theoretically the computational complexity and the reward loss of the algorithm. The effectiveness and robustness of the algorithm are revealed by empirical studies.

Introduction

The partially observable Markov decision process (POMDP) is a general framework for planning under uncertainty, which has gained much attention from AI researchers (Kaelbling, Littman, and Cassandra 1998; Pineau, Gordon, and Thrun 2006). Previous work mainly focused on POMDPs with exact parameters, in which the probability distributions of action effects and observations as well as the one-step rewards are precisely given. However, in practice, the parameters can not be absolutely precise. First, when we learn a POMDP for modeling some unknown uncertain system, the parameters learned may heavily depend on the adequacy of past data and the correctness of prior knowledge and learning method. In some cases, only limited data is available and it is expensive to acquire new data, thus the learned parameters are not guaranteed to be the true ones. Incorrect knowledge or improper learning techniques can also lead to imprecise parameters. Second, a precise POMDP is such a simplified and static model for tackling uncertainty that it can not adequately model a situation where there exist factors varying with time. Finally, modeling practical problems

usually leads to a large number of states and observations. One way of tackling large-scale problems is to aggregate the similar states and observations. The aggregation techniques provide spaces of states and observations that are implicitly specified, so that the original problem is transformed to a small-scale problem. Since we can not distinguish the real underlying state from an aggregated state, the distributions of action effects in the small-scale problem are imprecise.

In many cases, although available data are not adequate for learning a POMDP, it may be possible to estimate the bounds of parameters from the data. In addition, when aggregation techniques are employed, states (observations) are aggregated when their associated probability distributions are similar. As a result, we can obtain the bounds of the distributions. The focus of this paper is on the study of POMDPs with bounded parameters.

In this paper, we present the framework of bounded-parameter partially observable Markov decision processes (BPOMDPs). A modified value iteration is proposed as the basic strategy for tackling parameter imprecision. The modified value iteration follows the idea of dynamic programming and results in a value function for any finite-horizon problem which we refer to as a "quasi-optimal" value function. For any given initial belief, the modified value iteration provides a policy tree, which can be executed without updating belief. By properly designing the modified value iteration, we reduce the volume of the vector set representing the quasi-optimal value function which has a significant impact on computational complexity.

Moreover, we design an effective anytime algorithm called *UL-based value iteration* (ULVI). In the ULVI algorithm, each iteration is implemented based on two sets of vectors called U-set and L-set. We present four strategies of setting U-set and L-set, which can lead to different performances of the algorithm. Three of them get the modified value iteration implemented through the ULVI algorithm, while the other one makes solving some problems faster. We theoretically analyze the computational complexity of the algorithm and provide a bound of reward loss that is dependent on the extent of parameter imprecision. Empirically, we show that the reward loss is relatively small for problems with reasonably small imprecision, and the runtime of our algorithm is in general shorter than that of the exact algorithm solving the true underlying POMDP for most cases.

Related Work

There have been several studies focusing on POMDPs with imprecise parameters. Most of the studies are from the literature of completely observable Markov decision process (COMDP) (White and Eldeib 1994; Givan, Leach, and Dean 2000; Harmanec 2002). These studies deal with parameter imprecision for only a special class of POMDPs. Another work not limited to COMDPs (Cozman and Krotkov 1996), however, considers only the case that the agent's actions do not affect the state transition and the observation functions are Gaussian.

The parameter imprecision of general POMDPs was first studied in (Itoh and Nakamura 2007). The authors employed the concept of second-order belief and proposed an algorithm to obtain a belief-state MDP. However, in their work, the initial belief is considered as part of the model. As a result, both the procedure and the result of their algorithm depend heavily on the initial belief. If there are multiple possible initial belief states, the algorithm may have to run once for each belief. Furthermore, the states of the belief-state MDP may be exponential in the number of states in the original POMDP, leading to high computational complexity. The method proposed in this paper successfully tackles these problems. The ULVI algorithm is independent on the initial belief state, as it returns a policy tree for any given initial belief. Hence, it is sufficient to run the algorithm for only once. Moreover, since the resulting policy is based on policy trees, updating belief is avoided. Therefore, the difficulties of generating a huge number of belief states are avoided.

POMDP Overview

A POMDP is defined by a tuple $M = \langle S, A, \Omega, T, O, R, \gamma \rangle$, where S , A , and Ω are respectively the sets of states, actions, and observations. The state-transition function $T(s, a, s')$ gives the probability of ending in state s' , given that the agent takes action a in state s . The observation function $O(s', a, o)$ gives the probability of making observation o , given that the agent takes action a and arrives at state s' . The reward function $R(s, a)$ denotes the expected reward of taking action a at state s . $\gamma \in [0, 1)$ is the discounted factor. The goal of solving a POMDP is to maximize the expected sum of the discounted reward.

Usually, a policy is executed based on updating the belief state which is a probability distribution over the states. Given the previous belief state as well as the previous action and observation, the current belief is updated using the Bayes' rule, based on which the policy provides the current action. Thus the policy can be represented as a series of functions from the set of belief states to the set of actions. A less popular representation of policy is based on the notion of *policy tree*, in which each node denotes an action and each arc denotes an observation. A policy can be represented as a mapping from the belief space to a set of policy trees; and corresponding to each initial belief b_0 , there is a policy tree pt . At each step, the agent takes an action with respect to the current node, follows the arc representing the observation it takes, and gets to the node which indicates the next action. This representation of policy is called the *tree*

mapping representation (TMR).

The optimal policy can be generated from an optimal value function which returns the maximum expected reward for each given initial belief. The value function is yielded by applying the value iteration brought from dynamic programming. Usually, it is obtained by solving a Bellman equation, in which updating belief is implemented. Without updating belief, the value iteration can be implemented based on policy trees. We first compute the expected value of executing the t -step policy tree pt from state s by

$$V_{pt}(s) = R(s, a(pt)) + \gamma \sum_{s' \in S} T(s, a(pt), s') \sum_{o^i \in \Omega} O(s', a(pt), o^i) V_{o^i(pt)}(s'),$$

where $a(pt)$ is the action at the root node of pt , $o^i(pt)$ is the $(t-1)$ -step subtree following arc o^i . The value of executing pt for initial belief b is $V_{pt}(b) = \sum_{s \in S} b(s) V_{pt}(s)$, and the value function is $V_t(b) = \max_{pt \in \mathcal{PT}_t} V_{pt}(b)$, where \mathcal{PT}_t denotes the set of t -step policy trees.

For each pt , V_{pt} can be represented by a $|S|$ -dimensional vector $\alpha_{pt} = \langle V_{pt}(s^1), \dots, V_{pt}(s^{|S|}) \rangle$ and $V_{pt}(b) = b \cdot \alpha_{pt}$. V_t can be represented by a set of vectors Γ with $V_t(b) = \max_{\alpha \in \Gamma} b \cdot \alpha$. There is a unique minimum vector set Γ_t^* representing V_t , which is called a *parsimonious representation* (PR). Γ_t^* can be obtained by pruning an intermediate set Γ_t^+ that is generated based on previous PR Γ_{t-1}^* . Γ_t^+ contains $|A| |\Gamma_{t-1}^*|^{|S|}$ vectors and a vector α is in Γ_t^+ if $\forall s \in S$,

$$\alpha(s) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \sum_{o^i \in \Omega} O(s', a, o^i) \beta_{o^i}(s'), \quad (1)$$

where $a \in A$, and $\beta_{o^i} \in \Gamma_{t-1}^*$, $\forall o^i \in \Omega$. The process of generating Γ_t^* from Γ_{t-1}^* is called the one-step value backup.

In the last century, many researchers focused on the exact algorithms for solving POMDPs, and several efficient algorithms were produced such as the incremental pruning algorithm (Cassandra, Littman, and Zhang 1997), the witness algorithm (Kaelbling, Littman, and Cassandra 1998), etc. However, the computational cost of exact algorithms can be very expensive in some cases. In the worst case, the computation requires time doubly exponential in horizon. Recently, many approximate algorithms were proposed to speed up solving POMDPs, however only near-optimal reward values can be obtained (Spaan and Vlassis 2005; Pineau, Gordon, and Thrun 2006).

Bounded-Parameter POMDP model

In this section, we introduce the framework of the bounded-parameter partially observable Markov decision processes (BPOMDPs) and propose a modified value iteration as a basic strategy for dealing with parameter imprecision in a BPOMDP.

Framework

A BPOMDP is defined as a set of POMDPs, and is denoted by a tuple $\tilde{M} = \langle S, A, \Omega, \tilde{T}, \tilde{O}, \tilde{R}, \gamma \rangle$, where

- S, A, Ω , and γ are the same as those defined in POMDPs.
- $\tilde{T} = \langle \underline{T}, \overline{T} \rangle$ denotes the set of state-transition functions of POMDPs in \tilde{M} , where $\underline{T}, \overline{T}$ are functions satisfying

$$0 \leq \underline{T}(s, a, s') \leq \overline{T}(s, a, s') \leq 1, \forall s, s' \in S, a \in A.$$

A function $T : S \times A \rightarrow \Pi(S)$ is in \tilde{T} , if and only if

$$\underline{T}(s, a, s') \leq T(s, a, s') \leq \overline{T}(s, a, s'), \forall s, s' \in S, a \in A.$$

- $\tilde{O} = \langle \underline{O}, \overline{O} \rangle$ denotes the set of the observation functions of POMDPs in \tilde{M} , where $\underline{O}, \overline{O}$ are functions satisfying

$$0 \leq \underline{O}(s', a, o) \leq \overline{O}(s', a, o) \leq 1, \forall s' \in S, a \in A, o \in \Omega.$$

A function $O : S \times A \rightarrow \Pi(\Omega)$ is in \tilde{O} , if and only if

$$\underline{O}(s', a, o) \leq O(s', a, o) \leq \overline{O}(s', a, o), \forall s', a, o.$$

- $\tilde{R} = \langle \underline{R}, \overline{R} \rangle$ denotes the set of the reward functions of POMDPs in \tilde{M} , where $\underline{R}, \overline{R}$ are functions satisfying

$$\underline{R}(s, a) \leq \overline{R}(s, a), \forall s \in S, a \in A.$$

A real-valued function R is in \tilde{R} , if and only if

$$\underline{R}(s, a) \leq R(s, a) \leq \overline{R}(s, a), \forall s \in S, a \in A.$$

A POMDP $M' = \langle S', A', \Omega', T', O', R', \gamma' \rangle$ is in \tilde{M} if and only if $S' = S, A' = A, \Omega' = \Omega, T' \in \tilde{T}, O' \in \tilde{O}, R' \in \tilde{R}$, and $\gamma' = \gamma$.

In order to ensure that BPOMDPs are well-defined in the sense that any BPOMDP is a non-empty set of POMDPs, we set the following constraints on \tilde{T} and \tilde{O} :

$$\begin{aligned} \sum_{s' \in S} \underline{T}(s, a, s') \leq 1 \leq \sum_{s' \in S} \overline{T}(s, a, s'), \quad \forall s \in S, a \in A; \\ \sum_{o \in \Omega} \underline{O}(s', a, o) \leq 1 \leq \sum_{o \in \Omega} \overline{O}(s', a, o), \quad \forall s' \in S, a \in A. \end{aligned}$$

Note that the BPOMDP is a more general framework than POMDP, since a BPOMDP degenerates into a POMDP when $\underline{T} = \overline{T}, \underline{O} = \overline{O}$, and $\underline{R} = \overline{R}$. In addition, it is assumed that $\max_{s,a} \overline{R}(s, a)$ and $\min_{s,a} \underline{R}(s, a)$ are both finite values, so the one-step reward is bounded. Thus, for any infinite-horizon problem, we can instead solve a finite-horizon problem with sufficiently long horizon. In this paper, we focus on the finite-horizon BPOMDP problems.

The Modified Value Iteration

The lack of precise knowledge of the underlying model presents several obstacles for solving BPOMDPs. First, The criterion of optimality in solving POMDPs is not applicable for solving BPOMDPs, because it is impossible to obtain the precise expected sum of discounted reward for any given policy and initial belief. Second, as parameters are imprecisely given, updating belief can not be implemented precisely. One method to solve these difficulties is to use the second-order belief, which is an uncertainty measure on possible POMDPs (Itoh and Nakamura 2007). However, the precise second-order beliefs are still difficult to specify.

To tackle these obstacles, we propose a modified one-step value backup, which is iterated to construct a modified value iteration. A policy with TMR is obtained in the modified value iteration, so that updating belief is avoided. Without adopting the optimality criterion based on the exact expected reward, the modified value iteration returns a quasi-optimal value function. The quasi-optimal value function can be represented by a set of vectors. We use $\hat{\Gamma}_t^*$ to denote the PR of the quasi-optimal value function when there are t steps to go. Each vector in $\hat{\Gamma}_t^*$ corresponds to one t -step policy tree and represents the estimated value of the tree. The set of policy trees corresponding to $\hat{\Gamma}_t^*$ is denoted by \mathcal{PT}_t^* . The modified one-step value backup is constituted by a generating step and a pruning step:

Generating $\tilde{\Gamma}_t^+$ Based on \mathcal{PT}_{t-1}^* , a set of t -step policy trees \mathcal{PT}_t^+ is constructed, in which each tree is produced by assigning an action to the root node and choosing the $(t-1)$ -step subtrees from \mathcal{PT}_{t-1}^* . According to (1), we can generate a vector set Γ_t^+ corresponding to \mathcal{PT}_t^+ for each POMDP $M \in \tilde{M}$. In general, the number of POMDPs in \tilde{M} is uncountable, so there are uncountably many such Γ_t^+ s. We denote the union of such Γ_t^+ s by $\tilde{\Gamma}_t^+$, i.e.,

$$\tilde{\Gamma}_t^+ = \bigcup_{M \in \tilde{M}} \{ \alpha \mid \forall s \in S, \alpha(s) \text{ satisfies (1) for } M; \\ a \in A, \beta_{o^i} \in \hat{\Gamma}_{t-1}^*, \forall o^i \in \Omega \}.$$

An example of $\tilde{\Gamma}_t^+$ is illustrated in Figure 1. $\tilde{\Gamma}_t^+$ is denoted by the shaded area, as the area is filled with uncountably many Γ_t^+ s. One such $\Gamma_t^+ = \{ \alpha_1^M, \alpha_2^M \}$ computed for some POMDP M in \tilde{M} is shown by the dotted lines.

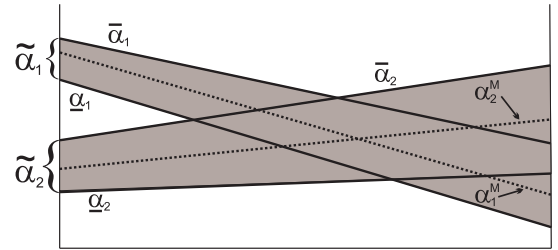


Figure 1: An example of $\tilde{\Gamma}_t^+$ and BVSSs.

From the definition of \mathcal{PT}_t^+ , each policy tree in \mathcal{PT}_t^+ corresponds to a unique $(|\Omega| + 1)$ -tuple $\langle a, \beta_{o^1}, \beta_{o^2}, \dots, \beta_{o^{|\Omega|}} \rangle$, where a is the action assigned to the root node and $\beta_{o^i} \in \hat{\Gamma}_{t-1}^*$ represents the estimated value of executing the $(t-1)$ -step subtree following o^i . In the rest of this paper, we denote a policy tree by its corresponding $(|\Omega| + 1)$ -tuple. For each policy tree $pt \in \mathcal{PT}_t^+$, we have a set of vectors,

$$\tilde{\alpha} = \{ \alpha \mid \forall s \in S, \alpha(s) \text{ satisfies (1) for } M; M \in \tilde{M} \}. \quad (2)$$

$\tilde{\alpha}$ contains uncountably many vectors, each of which represents the value of pt updated from $\hat{\Gamma}_{t-1}^*$ with some POMDP

in \widetilde{M} being the underlying model. A special property of $\widetilde{\alpha}$ that can be exploited is shown in the following theorem:

Theorem 1 Given a policy tree $pt = \langle a, \beta_{o^1}, \dots, \beta_{o^{|\Omega|}} \rangle \in \mathcal{PT}_t^+$, where $a \in A$ and $\beta_{o^i} \in \widehat{\Gamma}_{t-1}^*$, $\forall o^i \in \Omega$. Let $\widetilde{\alpha}$ be the set of $|S|$ -dimensional vectors obtained by (2). There exist two POMDPs $M_L \in \widetilde{M}$ and $M_U \in \widetilde{M}$ such that for any vector $\alpha \in \widetilde{\alpha}$ and each $s \in S$,

$$\underline{\alpha}(s) \leq \alpha(s) \leq \overline{\alpha}(s), \quad (3)$$

where $\underline{\alpha}$ and $\overline{\alpha}$ are computed by (1) with M_L and M_U as the underlying model, respectively.

Proof. The proof is provided in (Ni and Liu 2008). \square

Such a set of vectors $\widetilde{\alpha}$ is called a *bounded vector set* (BVS). $\underline{\alpha}$ ($\overline{\alpha}$) is called the *lower (upper) bound vector* of $\widetilde{\alpha}$ and pt . Note that $\widetilde{\Gamma}_t^+$ can be viewed as the union of the BVSs corresponding to the policy trees in \mathcal{PT}_t^+ . In Figure 1, $\widetilde{\Gamma}_t^+$ indicated by the shaded area is the union of two BVSs denoted by two bands, $\widetilde{\alpha}_1$ and $\widetilde{\alpha}_2$. The band bounded by $\underline{\alpha}_1$ and $\overline{\alpha}_1$ denotes BVS $\widetilde{\alpha}_1$, and the other denotes $\widetilde{\alpha}_2$.

Pruning $\widetilde{\Gamma}_t^+$ to $\widehat{\Gamma}_t^*$ In order to estimate the value of a policy tree pt , the “band” should be compressed to a “line”, i.e. the corresponding BVS $\widetilde{\alpha}$ should be replaced by a single vector. Since the true model could be any POMDP in \widetilde{M} , we allow the agent to accept any vector α in $\widetilde{\alpha}$ as an estimation of the value of pt . The accepted vector α is called the *delegate* of $\widetilde{\alpha}$ and pt , and $\widetilde{\alpha}$ is called the *family* of α .

The union of a group of BVSs is called a *BVS union* (BVSU), and the union of the delegates of these BVSs is called the *delegate* of the BVSU. Note that $\widetilde{\Gamma}_t^+$ is the superset of any BVSU yielded in the current iteration. We denote the delegate of $\widetilde{\Gamma}_t^+$ by $\widehat{\Gamma}_t^+$, and prune it to $\widehat{\Gamma}_t^*$ by removing the vectors which are useless in representing the value function. An example of pruning $\widetilde{\Gamma}_t^+$ is presented in Figure 2. Note that $\widehat{\Gamma}_t^*$ is dependent on the choice of the delegate of each BVS. Any set $\widehat{\Gamma}_t^*$ yielded from the above procedure is eligible to represent the quasi-optimal value function, which is called a *PR candidate*. For any initial belief b , a policy tree is indicated by the vector $\alpha \in \widehat{\Gamma}_t^*$ maximizing $\alpha \cdot b$, thus a policy with TMR is provided.

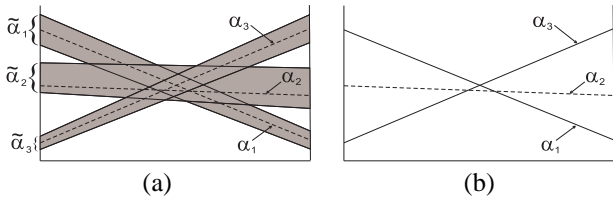


Figure 2: $\widetilde{\Gamma}_t^+$ is composed of $\widetilde{\alpha}_1$, $\widetilde{\alpha}_2$, and $\widetilde{\alpha}_3$. The procedure of pruning $\widetilde{\Gamma}_t^+$ is as follows: (a) for $i = 1, 2, 3$, a vector α_i denoted by a dashed line is selected as the delegate of $\widetilde{\alpha}_i$; (b) $\widehat{\Gamma}_t^+ = \{\alpha_1, \alpha_2, \alpha_3\}$ as the delegate of $\widetilde{\Gamma}_t^+$ is pruned to $\widehat{\Gamma}_t^* = \{\alpha_1, \alpha_3\}$ which is indicated by the solid lines.

$\widehat{\Gamma}_t^*$ with Less Vectors In addition to tackling the parameter imprecision, another motivation for proposing the modified value iteration is to alleviate the computational burden. It is known that the computational cost increases heavily with the volume of $\widehat{\Gamma}_t^*$, thus we hope to design the pruning step to keep $|\widehat{\Gamma}_t^*|$ as small as possible.

Intuitively, the delegates of all policy trees in \mathcal{PT}_t^+ should be set as the vectors updated with the same exact POMDP. This is reasonable as there is a unique real underlying model at any time. However, it is unknown which POMDP is the true underlying model. According to the definition of a PR candidate, the vectors in $\widehat{\Gamma}_t^+$ are allowed to be updated with different POMDPs, which multiplies the choices of $\widehat{\Gamma}_t^+$ and increases the chance of obtaining $\widehat{\Gamma}_t^*$ of small size.

For each iteration, among all possible PR candidates, the one containing the least vectors is called a *minimum PR*. The family of a minimum PR is called a *globally optimal BVSU*. It has been proved that (Ni and Liu 2008), a globally optimal BVSU $\widetilde{\Gamma}_t^*$ satisfies the following conditions: (a) there exists a choice of the delegate of $\widetilde{\Gamma}_t^+$, such that it can be pruned to the delegate of $\widetilde{\Gamma}_t^*$; (b) for any BVSU $\widetilde{\Gamma}_t'$ which is a strict subset of $\widetilde{\Gamma}_t^*$, any choice of the delegate of $\widetilde{\Gamma}_t'$ can not be pruned to the delegate of $\widetilde{\Gamma}_t^*$. These are called the *locally optimality* conditions in the sense that any delegate of a strict sub-BVSU of $\widetilde{\Gamma}_t^*$ can not be a PR candidate, and any delegate of a strict super-BVSU of $\widetilde{\Gamma}_t^*$ can not be a minimum PR. The BVSU satisfying the locally optimality conditions is called a *locally optimal BVSU*.

Theorem 2 Let the delegate of BVSU $\widetilde{\Gamma}_t = \bigcup_1^l \widetilde{\alpha}_i$ be $\widehat{\Gamma}_t^{u*} = \{\overline{\alpha}_1, \dots, \overline{\alpha}_l\}$, where $\overline{\alpha}_i$ is the upper bound vector of $\widetilde{\alpha}_i$, then (a) $\widehat{\Gamma}_t^{u*}$ is a minimum PR, if $\widetilde{\Gamma}_t$ is a globally optimal BVSU; (b) $\widehat{\Gamma}_t^{u*}$ is a PR candidate, if $\widetilde{\Gamma}_t$ is a locally optimal BVSU.

Proof. The proof is provided in (Ni and Liu 2008). \square

When $\widetilde{\Gamma}_t$ is a locally optimal BVSU, the $\widehat{\Gamma}_t^{u*}$ is called a *locally minimum PR* (LMPR). Since it is rather difficult to solve a minimum PR, we attempt to solve an LMPR, which makes it possible to find a minimum PR according to the above analysis. The pruning step with the goal of solving an LMPR trades off between optimality and tractability.

The UL-based Value Iteration Algorithm

We introduce an anytime algorithm for solving BPOMDPs, which is called the *UL-based value iteration* (ULVI) as it is based on two finite sets of vectors called U-set and L-set.

The ULVI algorithm shown in Table 1 is constituted by a number of iterations of UL-based value backup which includes three elemental routines: *GenerateL*, *GenerateU*, and *UL-Pruning*. In each iteration, *GenerateL* and *GenerateU* accept $\widehat{\Gamma}_{t-1}^*$ as input, which represents the estimated value function when there are $t - 1$ steps left, and generate the L-set and the U-set respectively. The L-set and U-set are called *eligible* if both sets contain finite vectors and every vector in L-set is dominated by U-set. The UL-Pruning routine accepts only eligible U-set and L-set as its input. It prunes

U-set based on L-set and outputs $\widehat{\Gamma}_t^*$, which is accepted as input by the next iteration. The whole algorithm terminates when a given number of iterations are completed.

```

ULVI( $\Gamma_0, T$ )
 $\Gamma = \Gamma_0$ 
for  $T$  iterations
   $\Gamma_L = \text{GenerateL}(\Gamma)$ 
   $\Gamma_U = \text{GenerateU}(\Gamma)$ 
   $\Gamma = \text{UL-Pruning}(\Gamma_U, \Gamma_L)$ 
end
return  $\Gamma$ 

```

Table 1: The ULVI algorithm

Note that $\widehat{\Gamma}_t^*$ corresponds to a set of t -step policy trees, denoted by \mathcal{PT}_t^* . Hence, the ULVI algorithm provides a policy with TMR.

UL-Pruning

The UL-Pruning routine is a procedure of pruning U-set Γ_U based on L-set Γ_L . The routine accepts eligible U-set and L-set as input, and generates a vector set $\widehat{\Gamma}_t^*$ for estimating the value function. In the routine, for every vector in U-set, we check whether to remove it from U-set makes U-set and L-set ineligible. If removing a vector results in ineligible U-set and L-set, then the vector is put back in U-set and never removed; otherwise, the vector is eliminated. The routine terminates when all vectors in U-set are checked, and it outputs the resulting U-set. The routine is shown in Table 2.

```

Input: ( $\Gamma_U := \{\alpha_1, \dots, \alpha_{|\Gamma_U|}\}, \Gamma_L := \{\beta_1, \dots, \beta_{|\Gamma_L|}\}$ )
 $\Gamma = \Gamma_U$ 
for  $i = 1 : |\Gamma_U|$ 
   $\Gamma = \Gamma \setminus \{\alpha_i\}$ 
  for  $j = 1 : |\Gamma_L|$ 
    if  $\text{DominationCheck}(\beta_j, \Gamma) = \text{False}$ 
       $\Gamma = \Gamma \cup \{\alpha_i\}$ 
      break
  end
end
end
return  $\Gamma$ 

```

Table 2: The UL-Pruning routine

In Table 2, the subroutine *DominationCheck* is exactly the same with the *findRegionPoint* routine in (Cassandra 1998) except that it outputs a boolean value. It sets up and solves a linear program (LP) to determine whether a vector is dominated by a vector set.

Theorem 3 *Given that U-set and L-set are eligible, $\widehat{\Gamma}_t^*$ is generated from UL-Pruning, then (a) each vector in Γ_L is dominated by $\widehat{\Gamma}_t^*$, (b) no strict subset of $\widehat{\Gamma}_t^*$ dominates all the vectors in Γ_L .*

Proof. (a) is obvious according to Table 2. For proving (b), let Γ' be any given strict subset of $\widehat{\Gamma}_t^*$ and α a vector in $\widehat{\Gamma}_t^* \setminus \Gamma'$. We denote by Γ_α the set obtained from pruning

Γ_U before α is checked. According to Table 2, $\widehat{\Gamma}_t^* \subset \Gamma_\alpha$ and Γ_α dominates all the vectors in Γ_L . $\Gamma_\alpha \setminus \{\alpha\}$ can not dominate all the vectors in Γ_L , otherwise α should be eliminated. Thus, there exist $\beta \in \Gamma_L$ and $b \in \Pi(S)$, such that $\beta(b) > \alpha'(b)$ for each $\alpha' \in \Gamma_\alpha \setminus \{\alpha\}$. Since $\Gamma' \subset \widehat{\Gamma}_t^* \setminus \{\alpha\} \subset \Gamma_\alpha \setminus \{\alpha\}$, $\beta(b) > \alpha'(b)$ for each $\alpha' \in \Gamma'$. Thus, β can not be dominated by Γ' . The proof is complete. \square

It is notable that the $\widehat{\Gamma}_t^*$ depends on the order of the vectors in Γ_U , and the runtime of the routine is also dependent on the order of the vectors in Γ_L . In addition, if U-set and L-set are identical, then the UL-Pruning routine reduces to a regular pruning routine in exact algorithms for solving POMDPs.

GenerateL and GenerateU

Based on $\widehat{\Gamma}_{t-1}^*$, GenerateL and GenerateU produce the L-set and U-set respectively. The strategy of setting GenerateL and GenerateU not only has the crucial impact on the properties of solution, but also influences the runtime of the algorithm. When these routines are designed properly, the modified value iteration can be implemented through the ULVI algorithm.

Choices for GenerateL and GenerateU We now introduce several routines that can be adopted as GenerateL and GenerateU. The following three routines are for GenerateL:

- *Lower Bound Pruning (LBP)*

Based on \mathcal{PT}_{t-1}^* , \mathcal{PT}_t^+ is constructed as described in previous sections. The set of the lower bound vectors of all the policy trees in \mathcal{PT}_t^+ is obtained and pruned to generate L-set.

- *Backup with Given Model (BGM)*

In this routine, we first take a guess of the underlying model among the POMDPs in \widetilde{M} . Assuming that the underlying POMDP is the guess model, we implement the exact one-step value backup based on $\widehat{\Gamma}_{t-1}^*$, by means of some exact algorithm for solving POMDPs such as the incremental pruning (Cassandra, Littman, and Zhang 1997). We set the yielded set as the L-set. In this routine, it is not necessary to generate \mathcal{PT}_t^+ of size exponential in $|\Omega|$.

- *BGM-based Lower Bound Pruning (BLBP)*

In this routine, we first run BGM routine and obtain a set of vectors. A set of policy trees corresponding to the vector set is then obtained. We prune the set comprising the lower bound vectors of these policy trees and generate the L-set. In some sense, BLBP produces a set “lower” than the one yielded in BGM. Further, it is possible that the L-set yielded in BLBP contains less vectors than that yielded in BGM. These properties may lead to generating smaller $\widehat{\Gamma}_t^*$ with less computation.

In addition, we introduce two routines that can be adopted as the GenerateU routine:

- *Upper Bound Pruning (UBP)*

In this routine, we construct \mathcal{PT}_t^+ based on \mathcal{PT}_{t-1}^* , and prune the set of the upper bound vectors of the policy trees

in \mathcal{PT}_t^+ . The resulting set is set to be the U-set. In some sense, the generated U-set is the “highest” PR candidate.

- *L-based Upper Bound Pruning (LUBP)*

LUBP is implemented after L-set is generated. We first find the set of policy trees corresponding to the L-set. We construct the set comprising the upper bound vectors of these trees and prune it to generate the U-set. Obviously, the generated U-set contains no more vectors than L-set.

Strategies for Setting L-set and U-set We properly pair GenerateL and GenerateU to produce four typical strategies for setting L-set and U-set. The strategies are presented in Table 3. We find that the generated L-set and U-set are eligible no matter which strategy is employed.

Strategies		$\hat{\Gamma}_t^*$	
GenerateL	GenerateU	PR candidate	LMPR
LBP	UBP	Yes	Yes
LBP	LUBP	Yes	Yes
BGM	LUBP	Yes	No
BLBP	LUBP	No	No

Table 3: The typical strategies for setting L-set and U-set

In Table 3, we characterize for each strategy the set $\hat{\Gamma}_t^*$ obtained in each iteration. By “No”, we mean that the attribute is not guaranteed. Each of the first three strategies guarantees $\hat{\Gamma}_t^*$ to be a PR candidate, thus the modified value iteration is implemented. We have proved that $\hat{\Gamma}_t^*$ is a LMPR when either of the first two strategies is employed (Ni and Liu 2008). However, both strategies require to enumerate trees in \mathcal{PT}_t^+ , which leads to extra computation. The other two strategies avoid enumerating trees in \mathcal{PT}_t^+ ; however a LMPR is not guaranteed.

Computational Complexity

In this section, we analyze the computational complexity of the UL-based value backup.

GenerateL and GenerateU We have known the complexities of the elementary operations: it requires $O(|S|(|\Omega| \log|\Omega| + |S|))$ time to compute a lower/upper bound vector; pruning a vector set Γ_{in} to a set Γ_{out} requires $|\Gamma_{in}|$ LPs with $O(|\Gamma_{out}||\Gamma_{in}|)$ constraints in the worst case and $O(|\Gamma_{out}|^2)$ constraints in the best case (Cassandra 1998); and the witness algorithm and the incremental pruning algorithm consume polynomial time in $|S|$, $|A|$, $|\Omega|$, $|\hat{\Gamma}_{t-1}^*$ and $\sum_a |\Gamma_t^a|$, where Γ_t^a is an intermediate set (Cassandra 1998).

The runtime of LBP and UBP is exponential in $|\Omega|$, because the policy trees in \mathcal{PT}_t^+ are enumerated. For BGM and BLBP, the runtime is bounded by a polynomial in $\sum_a |\Gamma_t^a|$, however, for some problems, $|\Gamma_t^a|$ can be exponentially large. Besides, for problems with small observation space, LBP may be better than BGM, as BGM produces a “higher” L-set which makes less vectors be pruned away from U-set. It is expected that the “BLBP+LUBP” strategy consumes the least time, since “BLBP” need not construct \mathcal{PT}_t^+ and generates a lower and smaller L-set than “BGM”.

UL-Pruning In general, it requires $O(|\Gamma_U||\Gamma_L|)$ LPs with $O(|\Gamma_U|^2|\Gamma_L|)$ constraints to implement UL-Pruning. In fact, the specific numbers of LPs and total constraints are polynomials in $|\Gamma_U|$, $|\Gamma_L|$, and $|\hat{\Gamma}_t^*|$. One can refer to (Ni and Liu 2008) for detail.

The Bounds of Reward Loss

For any t -horizon BPOMDP problem, a value function represented by a set of vectors is generated by the ULVI algorithm and a policy with TMR is obtained. Let V_t^E be the true value of executing the obtained policy, and let V_t^* be the optimal value function computed if the true underlying model is known. The reward loss is defined as

$$Loss = \sup_{b \in \Pi(S)} |V_t^*(b) - V_t^E(b)| = \|V_t^* - V_t^E\|_\infty.$$

We have the following theorem on the bound of reward loss:

Theorem 4 For any given t ,

$$\|V_t^* - V_t^E\|_\infty \leq \frac{k((1-\gamma)\delta_R + \gamma\eta\delta_{\bar{R}})}{(1-\gamma)^2},$$

where

$$k := \begin{cases} 3, & \text{for “BLBP+LUBP”,} \\ 2, & \text{for the other three strategies introduced,} \end{cases}$$

$$\delta_R := \max_{s,a} (\bar{R}(s,a) - \underline{R}(s,a)),$$

$$\delta_{\bar{R}} := \max_{s,a} \bar{R}(s,a) - \min_{s,a} \bar{R}(s,a),$$

$$\eta := \max_a (\eta_a^O + \eta_a^T) \wedge 1,$$

in which

$$\eta_a^O := \max_{s'} [(1 - \sum_o \underline{Q}(s', a, o)) \wedge (\sum_o \bar{O}(s', a, o) - 1)],$$

$$\eta_a^T := \max_{s'} [(1 - \sum_{s'} \underline{T}(s, a, s')) \wedge (\sum_{s'} \bar{T}(s, a, s') - 1)].$$

Proof. The proof is quite extensive and is provided in (Ni and Liu 2008). \square

The theorem shows that the bound on reward loss is dependent on the degree of parameter imprecision which is indicated by δ_R and η . When the BPOMDP degenerates into a POMDP, the imprecision disappears and both δ_R and η are zero, thus the bound reduces to zero and the ULVI algorithm becomes an exact algorithm for solving the POMDP. In addition, to compute the bound only costs $O(|S||A|(|S|+|\Omega|))$ time.

Unfortunately, the bound is not tight enough, especially for cases that γ is closed to 1; and information of \bar{T} and \bar{O} can not be contained in the bound unless η_a^O and η_a^T are relatively small. An open problem is whether there exists tighter bound for the ULVI algorithm which is more informative but still compact.

Empirical Results

We apply the ULVI algorithm to several BPOMDPs to show its performance empirically. All results are obtained by C code on a Pentium4 PC.

We construct the BPOMDPs based on three typical POMDPs including *Tiger*, *Network*, and *Shuttle* (Cassandra 1998), which are called “original” POMDPs. For

each POMDP, each $\epsilon^P \in \{0.001, 0.005, 0.01, 0.025, 0.05, 0.1, 0.2, 0.5\}$, and each $\epsilon^R \in \{0, 0.02\}$, a BPOMDP is constructed as follows: $\forall a, s, s', \underline{T}(s, a, s') := (T(s, a, s') - \epsilon^P) \vee 0$, $\overline{T}(s, a, s') := (T(s, a, s') + \epsilon^P) \wedge 1$; $\forall a, s', o$, $\underline{O}(s', a, o) := (O(s', a, o) - \epsilon^P) \vee 0$, $\overline{O}(s', a, o) := (O(s', a, o) + \epsilon^P) \wedge 1$; $\forall a, s$, $\underline{R}(s, a) := R(s, a) - \epsilon^R \delta$, $\overline{R}(s, a) := R(s, a) + \epsilon^R \delta$, where $\delta := \max_{a,s} R(s, a) - \min_{a,s} R(s, a)$, and T, O, R are the parameter functions in the original POMDP.

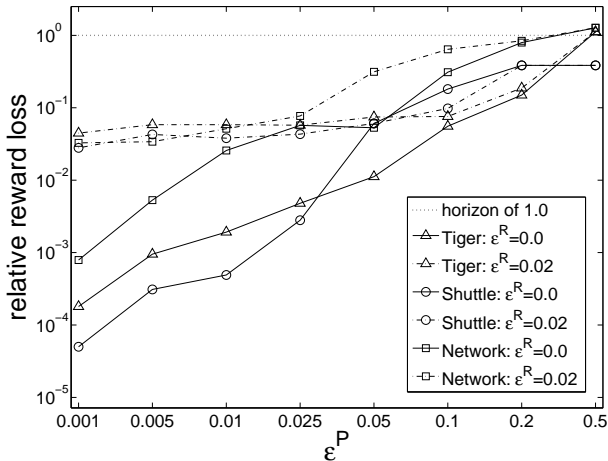


Figure 3: The relative reward losses of the ULVI algorithm.

First, we study how the parameter imprecision influences the reward loss of the ULVI algorithm. We adopt ‘‘BGM+LUBP’’ for setting U-set and L-set, as it is the best trade-off between tractability and optimality. We assume that the original POMDP is the underlying model, and we set the sparsest POMDP in the BPOMDP as the guess model in BGM, which contains the least non-zero parameters among the POMDPs in the BPOMDP. The incremental pruning algorithm is adopted in BGM. Figure 3 shows the *relative reward losses* for different degrees of parameter imprecision. The relative reward loss is defined as $\max_{b \in B} [(V_t^*(b) - V_t^E(b)) / (V_t^*(b) - V_t^{rand}(b))]$, where B is a set of initial belief states, V_t^{rand} is the expected sum of reward when the agent chooses actions randomly. In this section, for all problems, we set B as a set of 10^4 randomly generated belief states and the planning horizon $t = 400$.

We conclude from Figure 3 that, in general, the relative reward loss decreases with the decrease of ϵ^P and ϵ^R . Although Figure 3 shows that reducing parameter imprecision may result in raising the loss at some point, it only happens when ϵ^P is relatively large. Therefore, it is believable that the ULVI algorithm generates a near-optimal policy for BPOMDPs with reasonably small parameter imprecision. Note that, when $\epsilon^P = 0.5$, the ULVI algorithm generates for Tiger and Shuttle policies worse than randomly choosing actions. This is due to the huge imprecision.

Second, we illustrate in Figure 4 the runtime of the ULVI algorithm. The algorithm is set as in the previous exper-

iment. In general, the runtime of the algorithm decreases with the increase of ϵ^P and ϵ^R . The runtime is relatively short compared with the time of running incremental pruning algorithm for solving the exact original POMDPs. For each problem, the time of solving the original POMDP, denoted by a discrete point, is exceeded by the runtime of our algorithm only for the cases that the parameters are nearly precise. Thus, in general, the ULVI algorithm helps to save computational cost. In this sense, the ULVI algorithm can be viewed as an approximate algorithm for solving the regular POMDPs. Moreover, from Figure 4, it is believable that it generally costs more time to solve a BPOMDP by means of solving an arbitrary chosen POMDP by an exact algorithm.

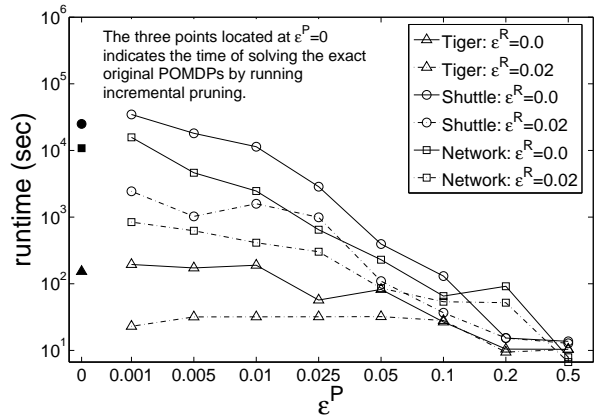


Figure 4: The runtime of the ULVI algorithm.

Next, we examine the robustness of the ULVI algorithm. We adopt the ‘‘BGM+LUBP’’ strategy, and incremental pruning algorithm is employed in BGM. The guess model in BGM is set to be the original POMDP. We only consider the cases where $\epsilon^R = 0.02$. For each BPOMDP, we randomly generate 10 POMDPs in it. Each generated POMDP is set as the underlying model, for which we calculate the relative reward loss. The largest relative reward loss is reported for each BPOMDP in Figure 5. After the name of each problem, ‘‘random’’ means that the results are the largest relative reward losses calculated as above; while ‘‘original’’ indicates that the losses are calculated for cases that the underlying models are the original POMDPs. From Figure 5, we find that the difference between the results for each BPOMDP is relatively small. Moreover, the largest relative reward loss generally decreases when ϵ^P decreases. In summary, it is shown empirically that our algorithm is robust to some extent.

Finally, we study the relationship between the runtime of the ULVI algorithm and the choice of strategy for setting GenerateU and GenrateL. We consider the Shuttle problem with $\epsilon^R = 0$ for different ϵ^P s. We run the ULVI algorithm with each of the four strategies introduced in previous sections. For each BPOMDP, the guess model is set to be the so-called sparsest POMDP. Incremental pruning is embedded in BGM and BLBP. The results are shown in Figure 6.

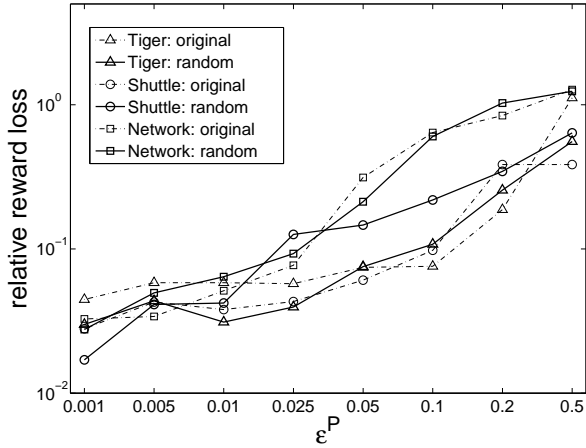


Figure 5: The largest relative reward losses of running ULVI algorithm for 10 randomly generated underlying POMDPs.

In general, the “LBP+UBP” and “LBP+LUBP” cost more time than the other two strategies. Although algorithms with “LBP+UBP” and “LBP+LUBP” can not provide a result in a reasonable time limit (36000 seconds in this experiment) for problems with small ϵ^P , they perform well for the problems with large imprecision. It is clear that “BLBP+LUBP” costs less time than “BGM+LUBP” for most cases. In future work, more experiments are needed for investigating the performance of these strategies.

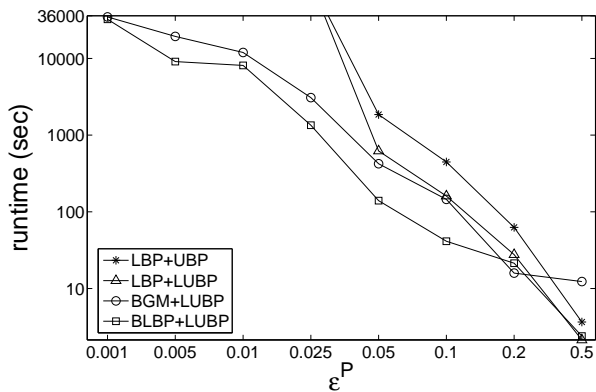


Figure 6: Runtime of ULVI with different strategies of setting U-set and L-set.

Conclusions and Future Work

In this paper, we have proposed the framework of BPOMDPs to characterize the POMDPs with imprecise but bounded parameters. A modified value iteration is introduced as a basic strategy for tackling parameter imprecision. We have designed the UL-based value iteration (ULVI) algo-

rithm for solving BPOMDPs. The modified value iteration can be implemented through the ULVI algorithm with properly designed subroutines. We have presented the computational complexity of the algorithm and provide a bound on the reward loss which depends on the extent of imprecision. The empirical experiments show that the ULVI algorithm is effective and robust.

Integrated with some state-aggregation technique, ULVI can be applied to practical large-scale problems. The performance of this application can be investigated in future work. Since the incremental pruning in BGM and BLBP can be replaced by an approximation algorithm for POMDPs such as point-based value iteration, a more general framework of ULVI algorithm deserves to be studied further.

Acknowledgments

The authors thank the anonymous reviewers for their comments and suggestions. This research is supported in part by a Hong Kong RGC CERG grant: 9041147 (CityU 117806).

References

- Cassandra, A.; Littman, M. L.; and Zhang, N. L. 1997. Incremental Pruning: A simple, fast, exact method for partially observable Markov decision processes. In *UAI-97*, 54–61.
- Cassandra, A. R. 1998. *Exact and approximate algorithms for partially observable Markov decision problems*. Ph.D. Dissertation, Department of Computer Science, Brown University.
- Cozman, F. G., and Krotkov, E. 1996. Quasi-Bayesian strategies for efficient plan generation: application to the planning to observe problem. In *UAI-96*, 186–193.
- Givan, R.; Leach, S.; and Dean, T. 2000. Bounded-parameter Markov decision processes. *Artificial Intelligence* 122(1–2):71–109.
- Harmanec, D. 2002. Generalizing markov decision processes to imprecise probabilities. *Journal of Statistical Planning and Inference* 105:199–213.
- Itoh, H., and Nakamura, K. 2007. Partially observable markov decision processes with imprecise parameters. *Artificial Intelligence* 171(8–9):453–490.
- Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1–2):99–134.
- Ni, Y., and Liu, Z.-Q. 2008. Bounded-parameter partially observable markov decision processes. Technical Report CityU-SCM-MCG-0501, City University of Hong Kong.
- Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research* 27:335–380.
- Spaan, M. T. J., and Vlassis, N. A. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research* 24:195–220.
- White, C., and Eldeib, H. 1994. Markov decision processes with imprecise transition probabilities. *Operations Research* 43:739–749.