Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes

Pascal Poupart[†], **Aarti Malhotra**[†], **Pei Pei**[†], **Kee-Eung Kim**[§], **Bongseok Goh**[§] and **Michael Bowling**[‡]
[†]David R. Cheriton School of Computer Science, University of Waterloo, Canada
[§]Department of Computer Science, Korean Advanced Institute of Science and Technology, Korea
[‡]Department of Computing Science, University of Alberta, Canada
[†]{ppoupart, pei.pei, aarti.malhotra}@uwaterloo.ca, [§]{kekim,bsgoh}@cs.kaist.ac.kr, [‡]bowling@cs.ualberta.ca

Abstract

In many situations, it is desirable to optimize a sequence of decisions by maximizing a primary objective while respecting some constraints with respect to secondary objectives. Such problems can be naturally modeled as constrained partially observable Markov decision processes (CPOMDPs) when the environment is partially observable. In this work, we describe a technique based on approximate linear programming to optimize policies in CPOMDPs. The optimization is performed offline and produces a finite state controller with desirable performance guarantees. The approach outperforms a constrained version of point-based value iteration on a suite of benchmark problems.

1 Introduction

Partially observable Markov decision processes (POMDPs) provide a natural framework for sequential decision making under uncertainty. The goal is to find a policy that maximizes the expected total return specified by a reward function. However, in many applications, there are several objectives and it may be desirable to ensure that some bounds are respected for some secondary objectives. For instance, in spoken dialog systems (Williams and Young 2007) the main objective may be to minimize the number of turns while ensuring that the probability of completing a booking task is above some threshold. In mobile assistive technologies (Hoey et al. 2012; Grześ, Poupart, and Hoey 2013a), the goal may be to maximize the probability of completing a task while bounding energy consumption. In wireless communications, the goal of opportunistic spectrum access (Zhao et al. 2007) is to maximize the utilization of wireless spectrum by allowing multiple devices to use the same wireless channel while satisfying regulatory requirements on the maximum collision rate between devices. To that effect, POMDPs have been extended to constrained POMDPs (CPOMDPs) by allowing secondary objectives to be specified with corresponding bounds on the expectation of those objectives (Isom, Meyn, and Braatz 2008).

We propose a novel approach based on approximate linear programming to optimize policies in CPOMDPs. When the reachable space of beliefs is finite, an optimal policy can be found exactly by linear programming. Since the space of reachable beliefs is often infinite (or too large to enumerate), we consider an approximate linear program that works with a subset of beliefs. Although the linear program is approximate, we show that its solution is an upper bound on total rewards. Since the resulting policy is a finite state controller, we also show how to compute the exact value of each objective by solving a system of linear equations. We also devise an iterative approach to increase the subset of beliefs and ensure that in the limit of an infinitely large belief set, the exact optimal policy is found. The approach is evaluated on a set of benchmark problems. The quality of the policies and the running time outperform the state of the art, which consists of a constrained version of point-based value iteration and a minimax quadratically constrained program (Kim et al. 2011).

2 Background

We consider a partially observable Markov decision process (POMDP) defined formally by a tuple $\langle S, A, O, T, Z, R, \gamma, b_0 \rangle$ where S is the set of states s, A is the set of actions a, O is the set of observations o, $T(s', s, a) = \Pr(s'|s, a)$ is the transition distribution, $Z(o, s', a) = \Pr(o|s', a)$ is the observation distribution, $R(s, a) \in \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor and $b_0(s_0) = \Pr(s_0)$ is the initial belief at time step 0. Since the underlying state of the system is not directly observable, the decision maker can maintain a belief $b(s_t) = \Pr(s_t)$ about the current state s_t at each time step t. The belief can be updated as time progresses based on Bayes' theorem. For instance, the belief b^{ao} obtained after executing a in b and observing o is

$$b^{ao}(s') \propto \sum_s b(s) \Pr(s'|s, a) \Pr(o|s', a) \; \forall s' \qquad (1)$$

A policy $\pi : B \rightarrow A$ is a mapping from beliefs to actions. In a POMDP, there is a single objective consisting of the reward function. The goal is to find a policy π^* that maximizes the expected discounted sum of rewards.

$$\pi^* = \text{argmax}_\pi \; E\left[\sum_t \gamma^t R(s_t, a_t) | \pi\right] \qquad (2)$$

A constrained POMDP (Isom, Meyn, and Braatz 2008) allows multiple objectives to be considered. One of the objectives is optimized, while the remaining objectives are bounded. Without loss of generality, we will assume that the primary objective is *maximized* and therefore we will denote it by the reward function, while the secondary objectives are assumed to be *upper* bounded and therefore we will denote them by cost functions.[1] Hence, a CPOMDP can be defined by a tuple $\langle S, A, O, T, Z, R, \{C_k\}_{1..K}, \{c_k\}_{1..K}, \gamma, b_0\rangle$ where $C_k(s,a) \in \mathbb{R}$ is the k^{th} cost function with upper bound c_k (among K cost functions). The algorithms described in this paper can also work with different discount factors for each objective, but to keep the exposition simple we will assume that the same discount factor is used for all the objectives. The goal is to find a policy π^* that maximizes the expected discounted sum of rewards while ensuring that the expected discounted sum of costs remains bounded.

$$\pi^* = \text{argmax}_\pi \; E\left[\sum_t \gamma^t R(s_t, a_t)|\pi\right] \tag{3}$$

$$\text{subject to } E\left[\sum_t \gamma^t C_k(s_t, a_t)|\pi\right] \leq c_k \;\forall k \tag{4}$$

Note that the actual sum of discounted costs may be higher than the bound since the bound only applies to the expectation (average). It is tempting to change the definition of CPOMDPs to bound actual costs, however, in stochastic domains, this often makes the problem infeasible or leads to conservative policies that do not perform well most of the time. A bound on actual costs means that the probability that the costs exceed the bounds should be zero. For instance, disastrous states such as a robot crash or system failure can rarely be completely ruled out, which means that all policies will have a non-zero probability of exceeding desirable cost bounds. A more practical alternative is simply to bound the probability that some undesirable states are reached. This can be done within the CPOMDP framework by assigning a cost of 1 to undesirable states and 0 to the other states. The expected discounted sum of costs corresponds to the probability of reaching an undesirable state when the discount factor is 1 and the process terminates as soon as an undesirable state is reached.

The optimization of CPOMDP policies is notoriously difficult. We cannot restrict ourselves to deterministic policies since the optimal policies may all be stochastic (Altman 1999; Kim et al. 2011). This is due to the presence of multiple objectives, where it may be necessary to randomize over multiple actions in order to tradeoff between costs and rewards. An exact solution can be obtained by solving a minimax quadratically constrained optimization problem (Kim et al. 2011), however this approach is intractable. A suboptimal dynamic programming technique (Isom, Meyn, and Braatz 2008) as well as a constrained version of point-based value iteration (Kim et al. 2011) have also been proposed. However, the non-convex nature of the optimal value function complicates dynamic programming and point-based value iteration. Alternatively, we can think of the CPOMDP as a constrained belief MDP. By treating the beliefs as states, the problem becomes fully observable, however the number of reachable beliefs may be very large (even infinite). In the next sections we describe how to optimize a policy by solving an approximate linear program for a subset of reachable beliefs.

[1]The sign of each objective can be flipped to ensure that the primary objective is maximized while the secondary objectives are upper bounded.

3 Constrained Belief State MDPs

Let B be the space of beliefs. We can then treat constrained POMDPs as constrained belief state MDPs (Altman 1999). The constrained MDP that is equivalent to a constrained POMDP can be formally defined by a tuple $\langle \bar{S}, A, \bar{T}, \bar{R}, \{\bar{C}_k\}_{1..K}, \{c_k\}_{1..K}, \gamma, \bar{s}_0\rangle$. Here $\bar{S} = B$ and $\bar{s}_0 = b_0$. The transition function $\bar{T}(b', b, a) = \Pr(b'|b,a)$ can then be expressed in terms of beliefs such that

$$\Pr(b'|b,a) = \sum_{\{o|b^{ao}=b'\}} \Pr(o|b,a) \tag{5}$$

Similarly, the reward and cost functions can be defined in terms of beliefs

$$\bar{R}(b,a) = \sum_s b(s)R(s,a) \tag{6}$$

$$\bar{C}_k(b,a) = \sum_s b(s)C_k(s,a) \;\forall k \tag{7}$$

Suppose that we restrict B to the space of reachable beliefs. When there are finitely many reachable beliefs, we can solve constrained POMDPs by linear programming:

$$\max_{\{y(b,a)\}\;\forall b,a} \; \sum_{b,a} y(b,a)\bar{R}(b,a) \tag{8}$$

$$s.t. \sum_{a'} y(b',a') = \delta(b',b_0) + \gamma \sum_{b,a} y(b,a)\Pr(b'|b,a) \;\forall b'$$

$$\sum_{b,a} y(b,a)\bar{C}_k(b,a) \leq c_k \;\forall k$$

$$y(b,a) \geq 0 \;\forall b,a$$

Here, $\delta(b_1, b_2)$ is a Dirac delta that returns 1 when $b_1 = b_2$ and 0 otherwise. The variables $y(b,a)$ can be interpreted as the (discounted) occupancy frequencies of each b,a pair. Overall, the above linear program corresponds to the dual linear program for MDPs (Puterman 2009) with the addition of cost constraints (Altman 1999). We can extract an optimal stochastic policy π^* from the occupancy frequencies by normalizing them:

$$\pi^*(a|b) = \Pr(a|b) = \frac{y(b,a)}{\sum_{a'} y(b,a')} \tag{9}$$

Similarly, the optimal expected total reward and costs are

$$V_R^*(b_0) = \sum_{b,a} y(b,a)\bar{R}(b,a) \tag{10}$$

$$V_{C_k}^*(b_0) = \sum_{b,a} y(b,a)\bar{C}_k(b,a) \;\forall k \tag{11}$$

4 Approximate Linear Programming

In most applications, the space of reachable beliefs is not finite and is unknown. We propose to approximate the linear program in (8). To obtain this approximation we will consider a finite set of beliefs $\hat{B}$. This set does not have to contain reachable beliefs. It just has to be a set of beliefs that contains the *corner* beliefs as well as some other beliefs to "cover" well the space of reachable beliefs. Here, corner beliefs are degenerate beliefs corresponding to each state. As we will see shortly, the corner beliefs are needed to ensure that any belief in the simplex can be interpolated. We will describe an iterative approach in the next section to gradually define $\hat{B}$. Let's define an approximate belief state MDP with respect to $\hat{B}$. The main issue is that the beliefs that are reachable from $\hat{B}$ may not be in $\hat{B}$. We resolve this by replacing any belief outside of $\hat{B}$ by a convex combination of beliefs in $\hat{B}$. This idea is inspired from POMDP algorithms (e.g., HSVI (Smith and Simmons 2005), SARSOP (Kurniawati, Hsu, and Lee 2008) and GapMin (Poupart, Kim, and Kim 2011)) that represent upper bounds with a set of belief-value pairs and interpolate the value of any other belief as a convex combination of those belief-value pairs.

Let b be a belief that is not in $\hat{B} = \{b_i\}_{1..|\hat{B}|}$. Since there may be many convex combinations of the b_i's that equal b, we select the convex combination that minimizes the weighted Euclidean norm of the difference between b and each b_i by solving the following linear program:

$$\begin{aligned}\min_{\{w_i\}\ \forall i} \quad & \sum_i w_i ||b - b_i||_2^2 \\ s.t. \quad & \sum_i w_i b_i(s) = b(s)\ \forall s \\ & \sum_i w_i = 1 \quad \text{and} \quad w_i \geq 0\ \forall i\end{aligned} \tag{12}$$

Let's use the above linear program to interpolate all beliefs b^{ao} that can be reached from some $b \in \hat{B}$ in one step. We define $w(b_i, b^{ao}) = w_i$ to be the probability w_i associated with b_i when interpolating b^{ao}. We can then make use of this interpolation probability to define an approximate transition function $\tilde{T}$.

$$\tilde{T}(b', b, a) = \widetilde{\Pr}(b'|b, a) = \sum_o \Pr(o|b, a) w(b', b^{ao}) \tag{13}$$

In turn, we can use this approximate transition model directly in the linear program in (8) to obtain an approximately optimal policy. Alg. 1 summarizes the steps to obtain this approximately optimal policy.

5 Policy Evaluation

Since the policy returned by Alg. 1 is based on an approximate transition function, there is a need to evaluate the expected reward and costs of the policy. It turns out that the policy is a stochastic finite state controller (Hansen 1998b) for which we can easily compute the exact expected discounted total reward and costs by solving a system of linear equations. Let N be the set of nodes n in the controller such that we associate a node n_b to each belief b. The action chosen in node n_b is determined by the policy $\Pr(a|n_b) = \pi(a|b)$. The transition to the next node $n_{b'}$ is determined by the interpolation distribution $\Pr(n_{b'}|n_b, a, o) = w(b', b^{ao})$. Since the transition distribution is based on the interpolation distribution, we can't assume that the distribution over states is given by b when the system is in node n_b. However, we can compute the joint occupancy frequency $y(n_b, s)$ that the system is in node n_b when the true underlying state is s, by solving the following system of linear equations.

Algorithm 1 Approximate LP

```
1: inputs: CPOMDP, $\hat{B}$
2: outputs: $\pi, \hat{V}_R^*(b_0)$
3: for each $b \in \hat{B}, a \in A, o \in O$ do
4:     Compute $w(b', b^{ao})\ \forall b' \in \hat{B}$ by solving LP (12)
5: end for
6: $\widetilde{\Pr}(b'|b, a) \leftarrow \sum_o \Pr(o|b, a) w(b', b^{ao})\ \forall b, b' \in \hat{B}$
7: $\{y, \hat{V}_R^*(b_0)\} \leftarrow$ solve LP in (8) with $\hat{B}$ and $\widetilde{\Pr}(b'|b, a)$
8: $\pi(a|b) \leftarrow y(b, a) / \sum_{a'} y(b, a')\ \forall b, a$
```

$$y^\pi(n_{b'}, s') = \delta(b', b_0) b_0(s') + \gamma \sum_{a,b,s,o} \pi(a|n_b) \tag{14}$$
$$\Pr(s'|s, a) \Pr(o|s', a) w(b', b^{ao}) y^\pi(n_b, s)\ \forall b', s'$$

The exact expected total reward and costs can then be computed by the following expectations:

$$V_R^\pi(b_0) = \sum_{b,s,a} y^\pi(n_b, s) \pi(a|b) R(s, a) \tag{15}$$

$$V_{C_k}^\pi(b_0) = \sum_{b,s,a} y^\pi(n_b, s) \pi(a|b) C_k(s, a)\ \forall k \tag{16}$$

We can also show that the approximately optimal total reward $\hat{V}_R^*$ obtained by solving the approximate LP (8) based on $\hat{B}$ is an upper bound on the exact optimal total reward V_R^*. In order to show this, let's analyze the LP in (8) and its equivalent dual. As mentioned earlier, the LP (8) can be interpreted as estimating occupancy frequencies, based on which expected total costs can be bounded while maximizing the expected total reward. The dual of this LP is also very informative:

$$\min_{\{\mathcal{V}(b)\}_{\forall b}, \{z_k\}_{\forall k}} \sum_b \delta(b, b_0) \mathcal{V}(b) + \sum_k c_k z_k \tag{17}$$
$$\text{s.t. } z_k \geq 0\ \forall k$$
$$\mathcal{V}(b) \geq \bar{R}(b, a) - \sum_k \bar{C}_k(b, a) z_k + \gamma \sum_{b'} \Pr(b'|b, a) \mathcal{V}(b')\ \forall b, a$$

Here, the variables $\mathcal{V}(b)$ can be interpreted as a *penalized* value function obtained based on a weighted combination of the reward function and the cost functions. The variables z_k denote the weights of the cost functions. In other words, the dual shows that an optimal policy can be obtained by optimizing a single objective composed of the reward function from which we subtract a weighted combination of the costs. This is a natural thing to do when combining several objectives into a single objective. Since the scale of the reward and the cost functions may not be the same, it is not always clear how to set the weights to adjust the scales. The dual LP essentially shows that the weights should be optimized simultaneously with the adjusted value function. We are now ready to show that the approximate value $\hat{V}^*_R(b_0)$ is an upper bound on the optimal value $V^*_R(b_0)$

Theorem 1. *The approximately optimal value $\hat{V}^*_R(b_0)$ found by solving LP (8) or its dual (17) with the subset $\hat{B}$ is an upper bound on the exact optimal value $V^*_R(b_0)$.*

Proof. Consider the dual LP (17) with the subset $\hat{B}$. Its optimal value is $\hat{V}^*_R(b_0)$ and let its optimal weights be $\hat{z}^*$. Consider the dual LP (17) with the entire belief space B. Its optimal value is $V^*_R(b_0)$ and let its optimal weights be z^*. Instead, if we fix the weights to $\hat{z}^*$, the optimal value of this partly instantiated dual LP (let's call it $V^*_{R,\hat{z}^*}(b_0)$) is an upper bound on $V^*_R(b_0)$.

$$V^*_{R,\hat{z}^*}(b_0) \geq V^*_R(b_0) \tag{18}$$

This follows from the fact that the dual LP is a minimization problem and therefore the value of any feasible solution is an upper bound on the optimal value. Note also that by fixing the weights to $\hat{z}^*$, we are also reducing the problem to a POMDP with a single objective. Hence,

$$V^*_{R,\hat{z}^*}(b_0) = \mathcal{V}^*(b_0) + \sum_k c_k \hat{z}^*_k \tag{19}$$

where $\mathcal{V}^*$ is the optimal value function of this single objective POMDP. If we restrict the belief space to $\hat{B}$ by interpolation in this single objective POMDP, we obtain an upper bound $\hat{\mathcal{V}}^*$ on $\mathcal{V}^*$.

$$\hat{\mathcal{V}}^*(b) \geq \mathcal{V}^*(b) \ \forall b \tag{20}$$

This inequality was shown by Hauskrecht (2000) and it is used to derive upper bounds on the optimal value function in several point-based value iteration techniques (Smith and Simmons 2005; Kurniawati, Hsu, and Lee 2008; Poupart, Kim, and Kim 2011; Grzes and Poupart 2014). Intuitively, Eq. 20 follows from the fact that the optimal value function of a single objective POMDP is piece-wise linear and convex (Sondik 1971; Smallwood and Sondik 1973), and therefore belief interpolation necessarily yields values at least as high as when we do not interpolate. Finally, the optimal value of the dual LP based on $\hat{B}$ is

$$\hat{V}^*_R(b_0) = \hat{\mathcal{V}}^*(b_0) + \sum_k c_k \hat{z}^*_k \tag{21}$$

The proof is completed by combining Eq. 18-21 as follows.

$$\begin{aligned} V^*_R(b_0) &\leq V^*_{R,\hat{z}^*}(b_0) \qquad \text{(by Eq. 18)} \\ &= \mathcal{V}^*(b) + \sum_k c_k \hat{z}^*_k \qquad \text{(by Eq. 19)} \\ &\leq \hat{\mathcal{V}}^*(b_0) + \sum_k c_k \hat{z}^*_k \qquad \text{(by Eq. 20)} \\ &= \hat{V}^*_R(b_0) \qquad \text{(by Eq. 21)} \end{aligned}$$

□

Since the exact expected total reward $V^\pi_R(b_0)$ of the policy π found by Alg. 1 is never greater than the exact optimal value $V^*_R(b_0)$, it also follows that $V^\pi_R(b_0) \leq V^*_R(b_0)$. To summarize, the value found by the approximate linear program is an upper bound on the optimal value, while the exact value of the resulting policy is a lower bound on the optimal value. The gap between these bounds is useful to determine how well $\hat{B}$ approximates the belief space. In the next section we will describe a procedure to incrementally generate $\hat{B}$ until the gap is below a desirable threshold.

We also need to analyze the expected total costs computed in the approximate LP. In principle, the bounds c_k imposed on the expected total costs ensure that the resulting policy does not yield higher costs than c_k in expectation. However, as mentioned before, the expected total costs are not exact due to $\hat{B}$. This is why it is important to compute the exact expected total costs $V^\pi_{C_k}(b_0)$ as shown in Eq. 11. If there is a cost function for which $V^\pi_{C_k}(b_0) > c_k$, then we have a problem since the solution is technically infeasible and we should not use the resulting policy. Unfortunately, Thm. 2 shows that the exact expected total costs may not satisfy the bounds.

Theorem 2. *The exact expected total costs $V^\pi_{C_k}(b_0)$ of the policy π found by solving the approximate LP based on $\hat{B}$ may be higher or lower than c_k.*

Proof. We give two examples where in one case $V^\pi_{C_k}(b_0) > c_k$ and in the other case $V^\pi_{C_k}(b_0) < c_k$. Suppose we have a single cost function such that $C(b,a) = R(b,a) \ \forall b,a$ and the approximate LP finds a suboptimal policy π for which the approximate expected total cost bound is tight, then

$$V^\pi_C(b_0) \overset{1}{=} V^\pi_R(b_0) \overset{2}{<} V^*_R(b_0) \overset{3}{\leq} \hat{V}^*_R(b_0) \overset{4}{=} c \tag{22}$$

Relation 1 follows from the assumption $C(b,a) = R(b,a) \ \forall b,a$. Relation 2 holds since π is suboptimal. Relation 3 follows from Thm. 1. Relation 4 follows from the assumption that the approximate expected total cost bound is tight and $C(b,a) = R(b,a) \ \forall b,a$. For the second case, set the cost function to be the negative of the reward function, i.e., $C(b,a) = -R(b,a) \ \forall b,a$. Then opposite relations hold based on the same explanations.

$$V^\pi_C(b_0) \overset{1}{=} -V^\pi_R(b_0) \overset{2}{>} -V^*_R(b_0) \overset{3}{\geq} -\hat{V}^*_R(b_0) \overset{4}{=} c \tag{23}$$

□

A simple approach to ensure that $V^\pi_{C_k}(b_0) \leq c_k \ \forall k$ is to (artificially) lower the bounds c_k until the exact costs satisfy the original bounds. For instance, if we have a single cost function C and after solving the approximate LP the exact expected total cost $V^\pi_C(b_0)$ of the resulting policy π exceeds the bound c, then we can do a binary search to find an artificial bound $\hat{c}$ such that the exact cost tightly matches the true bound, i.e., $V^\pi_C(b_0) = c$.

6 Belief Set Generation

Alg. 2 describes a generic approach to alternate between solving an approximate LP based on $\hat{B}$ and adding new beliefs to $\hat{B}$ until the gap between $\hat{V}_R^*(b_0)$ and $V_R^\pi(b_0)$ is less than some desirable threshold ϵ.

Algorithm 2 Iterative Approximate Linear Programming

```
1: inputs: CPOMDP, $\epsilon$
2: outputs: $\pi$
3: $\hat{B} \leftarrow \{cornerBeliefs\} \cup \{b_0\}$
4: repeat
5:     $[\pi, \hat{V}_R^*(b_0)] \leftarrow$ approximateLP(CPOMDP, $\hat{B}$)
6:     $y^\pi \leftarrow$ solve Eq. 14
7:     $V_R^\pi(b_0) \leftarrow \sum_{b,s,a} y^\pi(n_b, s)\pi(a|b)R(s,a)$
8:     $\hat{B} \leftarrow \hat{B} \cup$ beliefGeneration(CPOMDP, $\pi, \hat{B}, n$)
9: until $\hat{V}_R^*(b_0) - V_R^\pi(b_0) \le \epsilon$
```

A variety of techniques can be used to generate new beliefs that are added to $\hat{B}$ at each step (Shani, Pineau, and Kaplow 2013). Alg. 3 describes a simple approach inspired from *envelope* techniques used in planning (Dean et al. 1995). It considers the beliefs that can be reached in one step from any belief in $\hat{B}$ by executing π. Since there may be too many beliefs to be added, we can prioritize the beliefs by adding the n reachable beliefs with the largest weighted Euclidean distance to the beliefs in $\hat{B}$ based on the interpolation performed by LP (12).

Algorithm 3 Belief Generation

```
1: inputs: CPOMDP, $\pi$, $\hat{B}$, $n$
2: outputs: $\hat{B}'$ (such that $|\hat{B}'| \le n$)
3: $\hat{B}' \leftarrow \emptyset$
4: for each $b \in \hat{B}, o \in O$ do
5:     $b^o(s') \leftarrow \sum_{s,a} b(s)\Pr(s'|s,a)\Pr(o|s',a)\pi(a|b)\ \forall s'$
6:     $dist_{b^o} \leftarrow$ distance of $b^o$ to $\hat{B} \cup \hat{B}'$ by LP (12)
7:     if $dist_{b^o} > 0$ then
8:         $\hat{B}' \leftarrow \hat{B}' \cup \{b^o\}$
9:     end if
10:    if $|\hat{B}'| > n$ then        /* reduce the size of $\hat{B}'$ */
11:        for each $b' \in \hat{B}'$ do
12:            $dist_{b'} \leftarrow$ dist. of $b'$ to $\hat{B} \cup \hat{B}' \backslash \{b'\}$ by LP(12)
13:            $\hat{B}' \leftarrow \hat{B}' \setminus \{\text{argmin}_{b' \in \hat{B}'} dist_{b'}\}$
14:        end for
15:    end if
16: end for
```

The following theorem shows that the iterative approximate LP technique with envelope belief generation is guaranteed to converge to a (near) optimal policy when given enough time.

Theorem 3. *Alg. 2 is guaranteed to converge to a (near) optimal policy π of the CPOMDP, i.e., $V_R^*(b_0) - V_R^\pi(b_0) \le \epsilon$.*

Proof. We give an informal proof. We first show that Alg. 2 terminates and then we show that when it terminates it converges to a (near) optimal policy of the CPOMDP. Suppose that Alg. 2 does not terminate in the sense that $\hat{V}_R^*(b_0) - V_R^\pi(b_0)$ remains greater than ϵ. As the number of iterations increases, $\hat{B}$ will contain an increasing number of reachable beliefs since Alg. 3 is guaranteed to generate new reachable beliefs unless the current policy is evaluated accurately (error less than ϵ). As the number of iterations goes to infinity, $\hat{B}$ will eventually contain enough beliefs to accurately evaluate all policies that Alg. 1 produces infinitely often. Since Alg. 2 terminates as soon as Alg. 1 produces a policy that is evaluated accurately, we have a contradiction. Next, suppose that the algorithm terminates, but it converges to a suboptimal policy π. By Thm 1, we know that $\hat{V}_R^*(b_0) \ge V_R^*(b_0)$. Since $V_R^*(b_0) \ge V_R^\pi(b_0)$ and the algorithm terminates when $\hat{V}_R^*(b_0) - V_R^\pi(b_0) \le \epsilon$, then $V_R^*(b_0) - V_R^\pi(b_0) \le \epsilon$, which yields a contradiction. □

Table 1: Single objective POMDPs are augmented with a cost function that assigns a cost of 1 to all state-action pairs with low rewards as specified in the third column and a cost of 0 otherwise.

Problem	Reward Range	Low Reward Threshold
query.s3	1.199 to 7.48	≤ 2.75
query.s4	1.199 to 7.48	≤ 2.75
iff	-65 to 20	< 0
tiger-grid	-1 to 0.75	≤ 0
machine	-15 to 0.994	< 0
milos-aaai97	0 to 150	< 150

7 Experiments

We compared empirically our algorithm (CALP) to a constrained version of point-based value iteration (CPBVI) (Kim et al. 2011) on a set of benchmark problems. The other existing algorithms for CPOMDPs are approximate dynamic programming (Isom, Meyn, and Braatz 2008) and quadratically constrained optimization (Kim et al. 2011). Since they were shown to underperform CPBVI (Kim et al. 2011), we do not report results for them. Both CPBVI and CALP are implemented in Matlab without any explicit parallelization (other than the automated parallelization performed by Matlab's virtual machine) and were run on a 64-bit CentOS Linux machine with 24 cores (1.2 GHz), 132 Gb of RAM, Matlab R2013b and CPLEX 12.6. The toy, qcd and ncity problems are CPOMDP benchmarks from (Kim et al. 2011). The remaining problems are single objective POMDPs from http://www.pomdp.org that we augmented with costs. We set the cost to one for all state-action pairs with a reward below some threshold defined in Table 1 and zero for all other state-action pairs. The bound on the expected cumulative discounted costs ensures that the resulting policy will not visit undesirable state-action pairs too often. Hence the goal is to maximize expected total rewards while bounding the frequency of low reward events.

Table 2 summarizes the results. The algorithms ran until convergence of the optimal value to 3 significant digits or a time limit of 1000 seconds (which ever occured first).

Since the algorithms may take extra time to complete the last iteration, the reported times are often slightly above 1000 seconds. The expected reward and cost were computed by running 1000 simulations of 1000 steps for CPBVI. In the case of CALP, the expected reward and cost were computed by iteratively solving a system of linear equations with precision set to 3 significant digits. CPBVI did not find a policy that respects the cost bound c for several problems, which was denoted by n.a. in the table. This happened when CPBVI was not able to do enough iterations in 1000 seconds to plan over a sufficiently long planning horizon. When CALP found a policy that does not respect the cost bound due to the approximate nature of the linear program, the cost bound was artificially lowered by binary search (see the end of Section 5) until the exact expected cost tightly satisfies the bound. The time taken by the binary search is included in the results. Overall, CALP found policies that are better than or as good (when taking the standard deviation into account) as the policies found by CPBVI in less time than CPBVI. We also report the upper bound on the optimal value found by CALP in the last column. The difference between the last two columns gives an upper bound on the loss in value due to the interpolation with respect to $\hat{B}$. Since the difference is 0 for toy, qcd and ncity, CALP found an optimal policy for those CPOMDPs.

8 Related Work

This work can be viewed as an extension to the literature on fully observable constrained MDPs, where it is common to use dual linear programming (Altman 1999). Dual formulations have also been developed for dynamic programming (Wang et al. 2007) and reinforcement learning (Yang, Li, and Schuurmans 2009) in unconstrained MDPs. This work is also related to approximate linear programming (ALP) techniques for infinite state spaces in unconstrained MDPs (Schweitzer and Seidmann 1985; de Farias and Van Roy 2003). They use basis functions to restrict the space of value functions while we use interpolation to reduce the number of belief states. This approach is commonly used in point-based value iteration techniques for unconstrained POMDPs (Pineau et al. 2003; Shani, Pineau, and Kaplow 2013) where interpolation also leads to an upper bound on the optimal value function (Smith and Simmons 2005; Kurniawati, Hsu, and Lee 2008; Poupart, Kim, and Kim 2011; Grzes and Poupart 2014).

Our work is also related to the literature on finite state controllers for unconstrained POMDPs (Hansen 1998b). Instead of optimizing controllers by policy iteration (Hansen 1998a; Poupart and Boutilier 2003) or non-linear programming (Amato, Bernstein, and Zilberstein 2006), we use approximate linear programming, which allows constraints on secondary objectives to be taken into account. Finally, we focus on the offline optimization of CPOMDPs while Undurti and How (2010) developed an online forward search technique for CPOMDPs.

9 Conclusion

We proposed a new approach to optimize policies for constrained POMDPs. The technique is based on approximate linear programming. It outperforms the state of the art in terms of both solution quality and time. The policies found are finite state controllers which are also advantageous for deployment in resource constrained applications such as embedded systems as well as smartphones (Grześ, Poupart, and Hoey 2013b). In future work, it would be interesting to extend this work to constrained decentralized POMDPs (Wu, Jennings, and Chen 2012) and to explore reinforcement learning techniques for CPOMDPs.

10 Acknowledgments

The work done by Pascal Poupart, Aarti Malhotra and Pei Pei was supported by NSERC discovery and engage grants. The work done by Kee-Eung Kim and Bongseok Goh was supported by ICT R&D program of MSIP/IITP 14-824-09-014.

References

Altman, E. 1999. *Constrained Markov Decision Processes*. Chapman & Hall/CRC.

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2006. Solving POMDPs using quadratically constrained linear programs. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 341–343. ACM.

de Farias, D. P., and Van Roy, B. 2003. The linear programming approach to approximate dynamic programming. *Operations Research* 51(6):850–865.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76.

Grzes, M., and Poupart, P. 2014. POMDP planning and execution in an augmented space. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 757–764. International Foundation for Autonomous Agents and Multiagent Systems.

Grześ, M.; Poupart, P.; and Hoey, J. 2013a. Isomorph-free branch and bound search for finite state controllers. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*.

Grześ, M.; Poupart, P.; and Hoey, J. 2013b. Controller compilation and compression for resource constrained applications. In *Algorithmic Decision Theory*. Springer. 193–207.

Hansen, E. A. 1998a. An improved policy iteration algorithm for partially observable MDPs. *Advances in Neural Information Processing Systems* 1015–1021.

Hansen, E. A. 1998b. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 211–219. Morgan Kaufmann Publishers Inc.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 13.

Table 2: Benchmark comparison

domain	$\|S\|, \|A\|, \|O\|, \gamma$	c	algorithm	time (sec)	cost $V_C^\pi(B_0)$	reward $V_R^\pi(B_0)$	reward ub $\hat{V}_R^*(B_0)$
toy	3,2,1,0.9	0.95	CPBVI	0.65	0.951 ± 0.007	0.951 ± 0.007	
			CALP	0.44	0.950 ± 0.001	0.950 ± 0.001	0.950 ± 0.001
qcd	3,2,3,0.95	0.2	CPBVI	1002	0.203 ± 0.008	-0.402 ± 0.032	
			CALP	2.1	0.200 ± 0.001	-0.427 ± 0.001	-0.427 ± 0.001
ncity	1945,16,18,0.95	0.2	CPBVI	1263	0.198 ± 0.012	-2.81 ± 0.07	
			CALP	1123	0.200 ± 0.001	-2.51 ± 0.01	-2.51 ± 0.01
query.s3	27,3,3,0.99	53	CPBVI	1002	n.a.	n.a.	
			CALP	1038	52.0 ± 0.1	543 ± 1	551 ± 1
query.s4	81,4,3,0.99	60	CPBVI	1007	n.a.	n.a.	
			CALP	1153	51.6 ± 0.1	547 ± 1	607 ± 1
iff	104,4,22,0.99	4	CPBVI	1000	4.01 ± 0.13	-5.03 ± 0.97	
			CALP	959	3.99 ± 0.01	2.87 ± 0.01	9.93 ± 0.01
tiger-grid	36,5,17,0.95	18	CPBVI	1003	n.a.	n.a.	
			CALP	1019	17.8 ± 0.1	0.000 ± 0.001	2.49 ± 0.01
machine	256,4,16,0.99	9	CPBVI	1000	n.a.	n.a.	
			CALP	1154	7.73 ± 0.01	58.9 ± 0.1	65.0 ± 0.1
milos-aaai97	20,6,8,0.9	10	CPBVI	1003	n.a.	n.a.	
			CALP	710	9.85 ± 0.01	45.2 ± 0.1	52.9 ± 0.1

Hoey, J.; Yang, X.; Quintana, E.; and Favela, J. 2012. Lacasa: Location and context-aware safety assistant.

Isom, J. D.; Meyn, S. P.; and Braatz, R. D. 2008. Piecewise linear dynamic programming for constrained POMDPs. In *Proceedings of AAAI*.

Kim, D.; Lee, J.; Kim, K.-E.; and Poupart, P. 2011. Point-based value iteration for constrained POMDPs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*.

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

Pineau, J.; Gordon, G.; Thrun, S.; et al. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, 1025–1032.

Poupart, P., and Boutilier, C. 2003. Bounded finite state controllers. In *Advances in neural information processing systems*, None.

Poupart, P.; Kim, K.-E.; and Kim, D. 2011. Closing the gap: Improved bounds on optimal POMDP solutions. In *ICAPS*.

Puterman, M. L. 2009. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. John Wiley & Sons.

Schweitzer, P. J., and Seidmann, A. 1985. Generalized polynomial approximations in Markovian decision processes. *Journal of mathematical analysis and applications* 110(2):568–582.

Shani, G.; Pineau, J.; and Kaplow, R. 2013. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems* 27(1):1–51.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21(5):1071–1088.

Smith, T., and Simmons, R. 2005. Point-based POMDP algorithms: Improved analysis and implementation. In *UAI*.

Sondik, E. J. 1971. *The optimal control of partially observable Markov Decision Processes*. Ph.D. Dissertation, Stanford University, Palo Alto.

Undurti, A., and How, J. P. 2010. An online algorithm for constrained POMDPs. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 3966–3973. IEEE.

Wang, T.; Bowling, M.; Schuurmans, D.; and Lizotte, D. J. 2007. Stable dual dynamic programming. In *Advances in neural information processing systems*, 1569–1576.

Williams, J. D., and Young, S. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language* 21(2):393–422.

Wu, F.; Jennings, N.; and Chen, X. 2012. Sample-based policy iteration for constrained DEC-POMDPs.

Yang, M.; Li, Y.; and Schuurmans, D. 2009. Dual temporal difference learning. In *International Conference on Artificial Intelligence and Statistics*, 631–638.

Zhao, Q.; Tong, L.; Swami, A.; and Chen, Y. 2007. Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework. *Selected Areas in Communications, IEEE Journal on* 25(3):589–600.