RAO*: An Algorithm for Chance-Constrained POMDP's

Pedro Santana* and **Sylvie Thiébaux**$^+$ and **Brian Williams***
*Massachusetts Institute of Technology, CSAIL
32 Vassar St., Room 32-224, Cambridge, MA 02139
{*psantana,williams*}*@mit.edu*
$^+$The Australian National University & NICTA
Canberra ACT 0200, Australia
Sylvie.Thiebaux@anu.edu.au

Abstract

Autonomous agents operating in partially observable stochastic environments often face the problem of optimizing expected performance while bounding the risk of violating safety constraints. Such problems can be modeled as chance-constrained POMDP's (CC-POMDP's). Our first contribution is a systematic derivation of execution risk in POMDP domains, which improves upon how chance constraints are handled in the constrained POMDP literature. Second, we present RAO*, a heuristic forward search algorithm producing optimal, deterministic, finite-horizon policies for CC-POMDP's. In addition to the utility heuristic, RAO* leverages an admissible execution risk heuristic to quickly detect and prune overly-risky policy branches. Third, we demonstrate the usefulness of RAO* in two challenging domains of practical interest: power supply restoration and autonomous science agents.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) (Smallwood and Sondik 1973) have become one of the most popular frameworks for optimal planning under actuator and sensor uncertainty, where POMDP solvers find policies that maximize some measure of expected utility (Kaelbling, Littman, and Cassandra 1998; Silver and Veness 2010).

In many application domains, however, performance is not enough. Critical missions in real-world scenarios require agents to develop a keen sensitivity to risk, which needs to be traded-off against utility. For instance, a search and rescue drone should maximize the value of the information gathered, subject to safety constraints such as avoiding dangerous areas and keeping sufficient battery levels. In these domains, autonomous agents should seek to optimize expected reward while remaining safe by deliberately keeping the probability of violating one or more constraints within acceptable levels. A bound on the probability of violating constraints is called a *chance constraint* (Birge and Louveaux 1997). Unsurprisingly, attempting to model chance constraints as negative rewards leads to models that are oversensitive to the particular penalty value chosen, and to policies that are overly risk-averse or overly risk-taking (Undurti and How 2010). Therefore, to accommodate the aforementioned scenarios, new models and algorithms for *constrained* MDPs have started to emerge, which handle chance constraints explicitly.

Research has mostly focused on fully observable constrained MDPs, for which non-trivial theoretical properties are known (Altman 1999; Feinberg and Shwarz 1995). Existing algorithms cover an interesting spectrum of chance constraints over secondary objectives or even execution paths, e.g., (Dolgov and Durfee 2005; Hou, Yeoh, and Varakantham 2014; Teichteil-Königsbuch 2012). For constrained POMDPs (C-POMDP's), the state of the art is less mature. It includes a few suboptimal or approximate methods based on extensions of dynamic programming (Isom, Meyn, and Braatz 2008), point-based value iteration (Kim et al. 2011), approximate linear programming (Poupart et al. 2015), or on-line search (Undurti and How 2010). Moreover, as we later show, the modeling of chance constraints through unit costs in the C-POMDP literature has a number of shortcomings.

Our first contribution is a systematic derivation of the *execution risk* in POMDP domains, and how it can be used to enforce different types of chance constraints. A second contributions is Risk-bounded AO* (RAO*), a new algorithm for solving chance-constrained POMDPs (CC-POMDPs) that harnesses the power of heuristic forward search in belief space (Washington 1996; Bonet and Geffner 2000; Szer, Charpillet, and Zilberstein 2005; Bonet and Geffner 2009). Similar to AO* (Nilsson 1982), RAO* guides the search towards promising policies w.r.t. reward using an admissible heuristic. In addition, RAO* leverages a second admissible heuristic to propagate execution risk upper bounds at each search node, allowing it to identify and prune overly risky paths as the search proceeds. We demonstrate the usefulness of RAO* in two risk-sensitive domains of practical interest: automated power supply restoration (PSR) and autonomous science agents (SA).

RAO* returns policies maximizing expected cumulative reward among the set of deterministic, finite-horizon policies satisfying the chance constraints. Even though optimal CC-(PO)MDPs policies may, in general, require some limited amount of randomization (Altman 1999), we follow Dolgov and Durfee (2005) in deliberately developing an approach focused on deterministic policies for several reasons. First, deterministic policies can be effectively solved through heuristic forward search (Section 4). Second, computing randomized policies for C-POMDP's generally involves intractable formulations over all reachable beliefs, and current approximate methods (Kim et al. 2011; Poupart et al. 2015) do not guarantee solution feasibility. Finally, there is the human aspect that users rarely trust stochasticity when dealing with safety-critical systems.

The paper is organized as follows. Section 2 formulates the type of CC-POMDPs we consider, and details how RAO* computes execution risks and propagates risk bounds forward. Next, Section 3 discusses shortcomings related to the treatment of chance constraints in the C-POMDP literature. Section 4 presents the RAO* algorithm, followed by our experiments in Section 5, and conclusions in Section 6.

2 Problem formulation

When the true state of the system is hidden, one can only maintain a probability distribution (a.k.a. belief state) over the possible states of the system at any given point in time. For that, let $\hat{b}_k : \mathcal{S} \to [0,1]$ denote the *posterior* belief state at the k-th time step. A belief state at time $k+1$ that only incorporates information about the most recent action a_k is called a *prior* belief state and denoted by $\bar{b}(s_{k+1}|a_k)$. If, besides a_k, the belief state also incorporates knowledge from the most recent observation o_{k+1}, we call it a *posterior* belief state and denote it by $\hat{b}(s_{k+1}|a_k, o_{k+1})$.

Many applications in which an agent is trying to act under uncertainty while optimizing some measure of performance can be adequately framed as instances of Partially Observable Markov Decision Processes (POMDP) (Smallwood and Sondik 1973). Here, we focus on the case where there is a finite policy execution horizon h, after which the system performs a deterministic transition to an absorbing state.

Definition 1 (Finite-horizon POMDP). *A FH-POMDP is a tuple $H = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, b_0, h \rangle$, where $\mathcal{S}$ is a set of states; $\mathcal{A}$ is a set of actions; $\mathcal{O}$ is a set of observations; $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a stochastic state transition function; $O : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ is a stochastic observation function; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function; b_0 is the initial belief state; and h is the execution horizon.*

An FH-POMDP solution is a *policy* $\pi : \mathcal{B} \to \mathcal{A}$ mapping beliefs to actions. An optimal policy π^* is such that

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{h} R(s_t, a_t) \middle| \pi\right]. \qquad (1)$$

In this work, we focus on the particular case of discrete $\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$, and deterministic optimal policies. Beliefs can be recursively computed as follows:

$$\bar{b}(s_{k+1}|a_k) = \Pr(s_{k+1}|\hat{b}_k, a_k) = \sum_{s_k} T(s_k, a_k, s_{k+1})\hat{b}(s_k) \qquad (2)$$

$$\begin{aligned}\hat{b}(s_{k+1}|a_k, o_{k+1}) &= \Pr(s_{k+1}|\hat{b}_k, a_k, o_{k+1}),\\ &= \frac{1}{\eta} O(s_{k+1}, o_{k+1})\bar{b}(s_{k+1}|a_k),\end{aligned} \qquad (3)$$

where T and O are from Definition 1, and

$$\eta = \Pr(o_{k+1}|a_k, b_k) = \sum_{s_{k+1}} O(s_{k+1}, o_{k+1})\bar{b}(s_{k+1}|a_k) \qquad (4)$$

is the probability of collecting some observation o_{k+1} after executing action a_k at a belief state b_k. In addition to optimizing performance, the next session shows how one can enforce safety in FH-POMDPs by means of chance constraints.

2.1 Computing risk

A chance constraint consists of a bound Δ on the probability (chance) of some event happening during policy execution. Following (Ono, Kuwata, and Balaram 2012), we define this event as a sequence of states $s_{0:h} = s_0, s_1, \ldots, s_h$ of a FH-POMDP H violating one or more constraints in a set C. Let p (for "path") denote a sequence of states $p = s_{0:h}$, and let $c_v(p, C) \in \{0, 1\}$ be an indicator function such that $c_v(p, C) = 1$ iff one or more states in p violate constraints in C. The latter implies that states encompass all the information required to evaluate constraints. With this notation, we can write the chance constraint as

$$\mathbb{E}_p\left(c_v(p, C) | H, \pi\right) \le \Delta, \qquad (5)$$

One should notice that we make no assumptions about constraint violations producing observable outcomes, such as causing execution to halt.

Our approach for incorporating chance constraints into FH-POMDP's extends that of (Ono and Williams 2008; Ono, Kuwata, and Balaram 2012) to partially observable planning domains. We would like to be able to compute increasingly better approximations of (5) with admissibility guarantees, so as to be able to quickly detect policies that are guaranteed to violate (5). For that purpose, let b_k be a belief state over s_k, and Sa_k (for "safe at time k") be a Bernoulli random variable denoting whether the system *has not violated* any constraints at time k. We define

$$er(b_k, C|\pi) = 1 - \Pr\left(\bigwedge_{i=k}^{h} Sa_i \middle| b_k, \pi\right) \qquad (6)$$

as the *execution risk* of policy π, as measured from b_k. The probability term $\Pr\left(\bigwedge_{i=k}^{h} Sa_i \middle| b_k, \pi\right)$ can be written as

$$\Pr\left(\bigwedge_{i=k+1}^{h} Sa_i \middle| Sa_k, b_k, \pi\right) \Pr(Sa_k|b_k, \pi), \qquad (7)$$

where $\Pr(Sa_k|b_k, \pi)$ is the probability of the system not being in a constraint-violating path at the k-th time step. Since b_k is given, $\Pr(Sa_k|b_k, \pi)$ can be computed as

$$\Pr(Sa_k|b_k, \pi) = 1 - \sum_{s_k \in S} b(s_k) c_v(p(s_k), C) = 1 - r_b(b_k, C), \qquad (8)$$

where $p(s_k)$ is the *path* leading to s_k, and $r_b(b_k, C)$ is called the *risk* at the k-th step. Note that $c_v(p(s_k), C) = 1$ iff s_k or any of its ancestor states violate constraints in C. In situations where the particular set of constraints C is not important, we will use the shorthand notation $r_b(b_k)$ and $er(b_k|\pi)$. The second probability term in (7) can be written as

$$\begin{aligned}&\Pr\left(\bigwedge_{i=k+1}^{h} Sa_i \middle| Sa_k, b_k, \pi\right)\\ &= \sum_{b_{k+1}} \Pr\left(\bigwedge_{i=k+1}^{h} Sa_i \middle| b_{k+1}, \pi\right) \Pr(b_{k+1}|Sa_k, b_k, \pi),\\ &= \sum_{b_{k+1}} (1 - er(b_{k+1}|\pi)) \Pr(b_{k+1}|Sa_k, b_k, \pi).\end{aligned} \qquad (9)$$

The summation in (9) is over belief states at time $k+1$. These, in turn, are determined by (3), with $a_k = \pi(b_k)$ and some corresponding observation o_{k+1}. Therefore, we have $\Pr(b_{k+1}|Sa_k, b_k, \pi) = \Pr(o_{k+1}|Sa_k, \pi(b_k), b_k)$. For the purpose of computing the RHS of the last equation, it is useful to define *safe prior* belief as

$$\begin{aligned}\bar{b}^{sa}(s_{k+1}|a_k) &= \Pr(s_{k+1}|Sa_k, a_k, b_k), \\ &= \frac{\sum_{s_k:c_v(p(s_k),C)=0} T(s_k, a_k, s_{k+1})b(s_k)}{1-r_b(b_k)}, \quad (10)\end{aligned}$$

With (10), we can define

$$\begin{aligned}\Pr{}^{sa}(o_{k+1}|a_k, b_k) &= \Pr(o_{k+1}|Sa_k, a_k, b_k), \\ &= \sum_{s_{k+1}} O(s_{k+1}, o_{k+1})\bar{b}^{sa}(s_{k+1}|a_k), \quad (11)\end{aligned}$$

which is the distribution over observations at time $k+1$, assuming that the system was in a non-violating path at time k. Combining (6), (8), (9), and (11), we get the recursion

$$\begin{aligned}&er(b_k|\pi) = r_b(b_k) \\ &\quad + (1-r_b(b_k))\sum_{o_{k+1}} \Pr{}^{sa}(o_{k+1}|\pi(b_k), b_k)er(b_{k+1}|\pi), \quad (12)\end{aligned}$$

which is key to RAO*. If b_k is terminal, (12) simplifies to $er(b_k|\pi) = r_b(b_k)$. Note that (12) uses (11), rather than (4), to compute execution risk. We can now use the execution risk to express the chance constraint (5) in our definition of a chance-constrained POMDP (CC-POMDP).

Definition 2 (Chance-constrained POMDP). *A CC-POMDP is a tuple $\langle H, \mathcal{C}, \Delta\rangle$, where H is a FH-POMDP; $\mathcal{C}$ is a set of constraints defined over $\mathcal{S}$; and $\Delta = [\Delta^1, \ldots, \Delta^q]$ is a vector of probabilities for q chance constraints*

$$er(b_0, C^i|\pi) \leq \Delta^i, C^i \in 2^{\mathcal{C}}, i = 1, 2, \ldots, q. \quad (13)$$

The chance constraint in (13) bounds the probability of constraint violation over the whole policy execution. Alternative forms of chance constraints entailing safer behavior are discussed in Section 2.3. Our approach for finding optimal, deterministic, chance-constrained solutions for CC-POMDP's in this setting is explained in Section 4.

2.2 Propagating risk bounds forward

The approach for computing risk in (12) does so "backwards", i.e., risk propagation happens from terminal states to the root of the search tree. However, since we propose computing policies for chance-constrained POMDP's by means of heuristic forward search, one should seek to propagate the risk bound in (13) forward so as to be able to quickly detect that the current best policy is too risky. For that, let $0 \leq \tilde{\Delta}_k \leq 1$ be a bound such that $er(b_k|\pi) \leq \tilde{\Delta}_k$. Also, let o'_{k+1} be the observation associated with child b'_{k+1} of b_k and with probability $\Pr^{sa}(o'_{k+1}|\pi(b_k), b_k) \neq 0$. From (12) and the condition $er(b_k|\pi) \leq \tilde{\Delta}_k$, we get

$$\begin{aligned}er(b'_{k+1}|\pi) \leq \frac{1}{\Pr^{sa}(o'_{k+1}|\pi(b_k), b_k)}&\left(\frac{\tilde{\Delta}_k - r_b(b_k)}{1-r_b(b_k)}\right. \\ &\left. - \sum_{o_{k+1}\neq o'_{k+1}} \Pr{}^{sa}(o_{k+1}|\pi(b_k), b_k)er(b_{k+1}|\pi)\right). \quad (14)\end{aligned}$$

The existence of (14) requires $r_b(b_k) < 1$ and $\Pr^{sa}(o'_{k+1}|\pi(b_k), b_k) \neq 0$ whenever $\Pr(o'_{k+1}|\pi(b_k), b_k) \neq 0$. Lemma 1 shows that these conditions are equivalent.

Lemma 1. *One observes* $\Pr^{sa}(o_{k+1}|\pi(b_k), b_k) = 0$ *and* $\Pr(o_{k+1}|\pi(b_k), b_k) \neq 0$ *if, and only if,* $r_b(b_k) = 1$.

Proof:

- $\Leftarrow$: if $r_b(b_k) = 1$, we conclude from (8) that $c_v(s_k, C) = 1, \forall s_k$. Hence, all elements in (10) and, consequently, (11) will have probability 0.
- $\Rightarrow$: from Bayes' rule, we have

$$\Pr(Sa_k|o_{k+1}, a_k, b_k) = \frac{\Pr^{sa}(o_{k+1}|a_k, b_k)(1-r_b(b_k))}{\Pr(o_{k+1}|a_k, b_k)} = 0$$

  Hence, we conclude that $\Pr(\neg Sa_k|o_{k+1}, a_k, b_k) = 1$, i.e., the system is guaranteed to be in a constraint-violating path at time k, yielding $r_b(b_k) = 1$.

The execution risk of nodes whose parents have $r_b(b_k) = 1$ is irrelevant, as shown by (12). Therefore, it only makes sense to propagate risk bounds in cases where $r_b(b_k) < 1$.

One difficulty associated with (14) is that it depends on the execution risk of all siblings of b'_{k+1}, which cannot be computed exactly until terminal nodes are reached. Therefore, one must approximate (14) in order to render it computable during forward search.

We can easily define a necessary condition for feasibility of a chance constraint at a search node by means of an admissible execution risk heuristic $h_{er}(b_{k+1}|\pi) \leq er(b_{k+1}|\pi)$. Combining $h_{er}(\cdot)$ and (14) provides us with a necessary condition

$$\begin{aligned}er(b'_{k+1}|\pi) \leq \frac{1}{\Pr^{sa}(o'_{k+1}|\pi(b_k), b_k)}&\left(\frac{\tilde{\Delta}_k - r_b(b_k)}{1-r_b(b_k)}\right. \\ &\left. - \sum_{o_{k+1}\neq o'_{k+1}} \Pr{}^{sa}(o_{k+1}|\pi(b_k), b_k)h_{er}(b_{k+1}|\pi)\right). \quad (15)\end{aligned}$$

Since $h_{er}(b_{k+1}|\pi)$ computes a lower bound on the execution risk, we conclude that (15) gives an upper bound for the true execution risk bound in (14). The simplest possible heuristic is $h_{er}(b_{k+1}|\pi) = 0, \forall b_{k+1}$, which assumes that it is absolutely safe to continue executing policy π beyond b_k. Moreover, from the non-negativity of the terms in (12), we see that another possible choice of a lower bound is $h_{er}(b_{k+1}|\pi) = r_b(b_{k+1})$, which is guaranteed to be an improvement over the previous heuristic, for it incorporates additional information about the risk of failure at that belief state. However, it is still a myopic risk estimate, given that it ignores the execution risk for nodes beyond b_{k+1}. All these bounds can be compute forward, starting with $\tilde{\Delta}_0 = \Delta$.

2.3 Enforcing safe behavior at all times

Enforcing (13) bounds the probability of constraint violation over total policy executions, but (14) shows that unlikely policy branches can be allowed risks close or equal to 1 if that will help improve the objective, giving rise to a "daredevil" attitude. Since this might not be the desired risk-aware behavior, a straightforward way of achieving higher levels of safety is to depart from the chance constraints in (13) and, instead, impose a set of chance constraints of the form

$$er(b_k, C^i|\pi) \leq \Delta^i, \forall i, b_k \text{ s.t. } b_k \text{ is nonterminal.} \tag{16}$$

Intuitively, (16) tells the autonomous agent to "remain safe at all times", whereas the message conveyed by (13) is "stay safe overall". It should be clear that (16)⇒(13), so (16) necessarily generates safer policies than (13), but also more conservative in terms of utility. Another possibility is to follow (Ono, Kuwata, and Balaram 2012) and impose

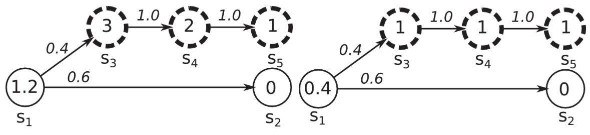
$$\sum_{k=0}^{h} r_b(b_k, C^i) \leq \Delta^i, \forall i, \tag{17}$$

which is a sufficient condition for (13) based on Boole's inequality. One can show that (17)⇒(16), so enforcing (17) will lead to policies that are at least as conservative as (16).

3 Relation to constrained POMDP's

Alternative approaches for chance-constrained POMDP planning have been presented in (Undurti and How 2010) and (Poupart et al. 2015), where the authors investigate constrained POMDP's (C-POMDP's). They argue that chance constraints can be modeled within the C-POMDP framework by assigning unit costs to states violating constraints, 0 to others, and performing calculations as usual.

There are two main shortcomings associated with the use of unit costs to deal with chance constraints. First, it only yields correct measures of execution risk in the particular case where constraint violations cause policy execution to terminate. If that is not the case, incorrect probability values can be attained, as shown in the simple example in Figure 1. Second, assuming that constraint violations cause execution to cease has a strong impact on belief state computations. The key insight here is that assuming that constraint violations cause execution to halt provides the system with an invaluable observation: at each non-terminal belief state, the risk $r_b(b_k, C)$ in (8) must be 0. The reason for that is simple: (*constraint violation* ⇒ *terminal belief*) ⇔ (*non-terminal belief* ⇒ *no constraint violation*).



(a) Incorrect execution risks computed using unit costs. (b) Correct execution risks computed according to (12).

Figure 1: Modeling chance constraints via unit costs may yield incorrect results when constraint-violating states (dashed outline) are not terminal. Numbers within states are constraint violation probabilities. Numbers over arrows are probabilities for a non-deterministic action.

Assuming that constraint violations are terminal is reasonable when undesirable states are destructive, e.g., the agent is destroyed after crashing against an obstacle. Nevertheless, it is rather limiting in terms of expressiveness, since there are domains where undesirable states can be "benign". For instance, in the power supply restoration domain described in the experimental section, connecting faults to generators is undesirable and we want to limit the probability of this event. However, it does not destroy the network. In fact, it might be the only way to significantly reduce the uncertainty about the location of a load fault, therefore allowing for a larger amount of power to be restored to the system.

4 Solving CC-POMDP's through RAO*

In this section, we introduce the Risk-bounded AO* algorithm (RAO*) for constructing risk-bounded policies for CC-POMDP's. RAO* is based on heuristic forward search in the space of belief states. The motivation for this is simple: given an initial belief state and limited resources (including time), the number of *reachable* belief states from a set of initial conditions is usually a very small fraction of the total number of possible belief states.

Similar to AO* in fully observable domains, RAO* (Algorithm 1) explores its search space from an initial belief b_0 by incrementally constructing a hypergraph G, called the *explicit* hypergraph. Each node in G represents a belief state, and a hyperedge is a compact representation of the process of taking an action and receiving any of a number of possible observations. Each node in G is associated with a Q value

$$Q(b_k, a_k) = \sum_{s_k} R(s_k, a_k) b(s_k) + \sum_{o_{k+1}} \Pr(o_{k+1}|a_k, b_k) Q^*(b_{k+1}) \tag{18}$$

representing the expected, cumulative reward of taking action a_k at some belief state b_k. The first term corresponds to the expected current reward, while the second term is the expected reward obtained by following the optimal deterministic policy π^*, i.e., $Q^*(b_{k+1}) = Q(b_{k+1}, \pi^*(b_{k+1}))$. Given an admissible estimate $h_Q(b_{k+1})$ of $Q^*(b_{k+1})$, we select actions for the current estimate $\hat{\pi}$ of π^* according to

$$\hat{\pi}(b_k) = \arg\max_{a_k} \hat{Q}(b_k, a_k), \tag{19}$$

where $\hat{Q}(b_k, a_k)$ is the same as (18) with $Q^*(b_{k+1})$ replaced by $h_Q(b_{k+1})$. The portion of G corresponding to the current estimate $\hat{\pi}$ of π^* is called the *greedy* graph, for it uses an admissible heuristic estimate $h_Q(b_k, a_k)$ of $Q^*(b_{k+1})$ to explore the most promising areas of G first.

The most important differences between AO* and RAO* lie in Algorithms 2 and 3. First, since RAO* deals with partially observable domains, node expansion in Algorithm 2 involves full Bayesian prediction and update steps, as opposed to a simple branching using the state transition function T. In addition, RAO* leverages the heuristic estimates of execution risk explained in Section 2.2 in order to perform early pruning of actions that introduce child belief nodes that are guaranteed to violate the chance constraint. The same process is also observed during policy update in Algorithm 3, in which heuristic estimates of the execution risk are used to prevent RAO* to keep choosing actions that are promising in terms of heuristic value, but can be proven to violate the chance constraint at an early stage.

The proofs of soundness, completeness, and optimality for RAO* are given in Lemma 2 and Theorem 1.

Algorithm 1 RAO*

```
Input: CC-POMDP $H$, initial belief $b_0$.
Output: Optimal policy $\pi$ mapping beliefs to actions.
1: Explicit graph $G$ and policy $\pi$ initially consist of $b_0$.
2: while $\pi$ has some nonterminal leaf node do
3:     $n, G \leftarrow$ expand-policy($G, \pi$)
4:     $\pi \leftarrow$ update-policy($n, G, \pi$)
5: return $\pi$.
```

Algorithm 2 expand-policy

```
Input: Explicit graph $G$, policy $\pi$.
Output: Expanded explicit $G'$, expanded leaf node $n$.
1: $G' \leftarrow G$, $n \leftarrow$ choose-promising-leaf($G, \pi$)
2: for each action $a$ available at $n$ do
3:     $ch \leftarrow$ use (2), (3), (4) to expand children of $(n, a)$.
4:     $\forall c \in ch$, use (8), (11), (12), and (18) with admissible heuristics to estimate $Q^*$ and $er$.
5:     $\forall c \in ch$, use (15) to compute execution risk bounds
6:     if no $c \in ch$ violates its risk bound then
7:         $G' \leftarrow$ add hyperedge $[(n, a) \rightarrow ch]$
8: if no action added to $n$ then mark $n$ as terminal.
9: return $G', n$.
```

Lemma 2. *Risk-based pruning of actions in Algorithms 2 (line 6) and 3 (line 7) is sound.*

Proof: The RHS of (14) is the true execution risk bound for $er(b'_{k+1}|\pi)$. The execution risk bound on the RHS of (15) is an upper bound for the bound in (14), since we replace $er(b_{k+1}|\pi)$ for the siblings of b'_{k+1} by admissible estimates (lower bounds) $h_{er}(b_{k+1}|\pi)$. In the aforementioned pruning steps, we compare $h_{er}(b'_{k+1}|\pi)$, a lower bound on the true value $er(b'_{k+1}|\pi)$, to the upper bound (15). Verifying $h_{er}(b'_{k+1}|\pi) >$ (15) is sufficient to establish $er(b'_{k+1}|\pi) >$ (14), i.e., action a currently under consideration is guaranteed to violate the chance constraint. □

Theorem 1. *RAO* is complete and produces the optimal deterministic, finite-horizon policies meeting the chance constraints.*

Proof: a CC-POMDP, as described in Definition 2, has a finite number of policy branches, and Lemma 2 shows that RAO* only prunes policy branches that are guaranteed not to be part of any chance-constrained solution. Therefore, if no chance-constrained policy exists, RAO* will eventually return an empty policy.

Concerning the optimality of RAO* with respect to the utility function, it follows from the admissibility of $h_Q(b_k, a_k)$ in (19) and the optimality guarantee of AO*. □

5 Experiments

This section provides empirical evidence of the usefulness and general applicability of CC-POMDP's as modeling tool for risk-sensitive applications, and shows how RAO* performs when computing risk-bounded policies in two challenging domains of practical interest: automated planning for science agents (SA) (Benazera et al. 2005); and power supply restoration (PSR) (Thiébaux and Cordier 2001). All models and RAO* were implemented in Python and ran on an Intel Core i7-2630QM CPU with 8GB of RAM.

Algorithm 3 update-policy

```
Input: Expanded $n$, explicit graph $G$, policy $\pi$.
Output: Updated policy $\pi'$.
1: $Z \leftarrow$ set containing $n$ and its ancestors reachable by $\pi$.
2: while $Z \neq \emptyset$ do
3:     $n \leftarrow$ remove($Z$) node $n$ with no descendant in $Z$.
4:     while there are actions to be chosen at $n$ do
5:         $a \leftarrow$ next best action at $n$ according to (19) satisfying execution risk bound.
6:         Propagate execution risk bound of $n$ to the children of the hyperedge $(n, a)$
7:         if no children violates its exec. risk bound then
8:             $\pi(n) \leftarrow a$; break
9:     if no action was selected at $n$ then mark $n$ as terminal
```

Our SA domain is based on the planetary rover scenario described in (Benazera et al. 2005). Starting from some initial position in a map with obstacles, the science agent may visit four different sites on the map, each of which could contain new discoveries with probability based on a prior belief. If the agent visits a location that contains new discoveries, it will find it with high probability. The agent's position is uncertain, so there is always a non-zero risk of collision when the agent is traveling between locations. The agent is required to finish its mission at a relay station, where it can communicate with an orbiting satellite and transmit its findings. Since the satellite moves, there is a limited time window for the agent to gather as much information as possible and arrive at the relay station. Moreover, we assume the duration of each traversal to be uncontrollable, but bounded. In this domain, we use a single chance constraint to ensure that the event "arrives at the relay location on time" happens with probability at least $1 - \Delta$. The SA domain has size $|\mathcal{S}| = 6144$; $|\mathcal{A}| = 34$, $|\mathcal{O}| = 10$.

In the *PSR* domain (Thiébaux and Cordier 2001), the objective is to reconfigure a faulty power network by switching lines on or off so as to resupply as many customers as possible. One of the safety constraints is to keep faults isolated at all times, to avoid endangering people and enlarging the set of areas left without power. However, fault locations are hidden, and more information cannot be obtained without taking the risk of resupplying a fault. Therefore, the chance constraint is used to limit the probability of connecting power generators to faulty buses. Our experiments focused on the semi-rural network from (Thiébaux and Cordier 2001), which was significantly beyond the reach of (Bonet and Thiébaux 2003) even for single faults. In our experiments, there were always circuit breakers at each generator, plus different numbers of additional circuit breakers depending on the experiment. Observations correspond to circuit breakers being open or closed, and actions to opening and closing switches. The PSR domain is strongly combinatorial, with $|\mathcal{S}| = 2^{61}$; $|\mathcal{A}| = 68$, $|\mathcal{O}| = 32$.

We evaluated the performance of RAO* in both domains under various conditions, and the results are summarized in Tables 1 (higher utility is better) and 2 (lower cost is better). The runtime for RAO* is always displayed in the *Time* column; *Nodes* is the number of hypergraph nodes expanded during search, each one of them containing a belief state with one or more particles; and *States* is the number of evaluated belief state particles. It is worthwhile to mention that constraint violations in PSR *do not cause execution to terminate*, and the same is true for scheduling violations in SA. The only type of terminal constraint violation are collisions in SA, and RAO* makes proper use of this extra bit of information to update its beliefs. Therefore, PSR and SA are examples of risk-sensitive domains which can be appropriately modeled as CC-POMDP's, but not as C-POMDP's with unit costs. The heuristics used were straightforward: for the execution risk, we used the admissible heuristic $h_{er}(b_k|\pi) = r_b(b_k)$ in both domains. For Q values, the heuristic for each state in PSR consisted in the final penalty incurred if only its faulty nodes were not resupplied, while in SA it was the sum of the utilities of all non-visited discoveries.

As expected, both tables show that increasing the maximum amount of risk Δ allowed during execution can only improve the policy's objective. The improvement is not monotonic, though. The impact of the chance constraint on the objective is discontinuous on Δ when only deterministic policies are considered, since one cannot randomly select between two actions in order to achieve a continuous interpolation between risk levels. Being able to compute increasingly better approximations of a policy's execution risk, combined with forward propagation of risk bounds, also allow RAO* to converge faster by quickly pruning candidate policies that are guaranteed to violate the chance constraint. This can be clearly observed in Table 2 when we move from $\Delta = 0.5$ to $\Delta = 1.0$ (no chance constraint).

Another important aspect is the impact of sensor information on the performance of RAO*. Adding more sources of sensing information increases the branching on the search hypergraph used by RAO*, so one could expect performance to degrade. However, that is not necessarily the case, as shown by the left and right numbers in the cells of Table 2. By adding more sensors to the power network, RAO* can more quickly reduce the size of its belief states, therefore leading to a reduced number of states evaluated during search. Another benefit of reduced belief states is that RAO* can more effectively reroute energy in the network within the given risk bound, leading to lower execution costs.

Finally, we wanted to investigate how well a C-POMDP approach would perform in these domains relative to a CC-POMDP. Following the literature, we made the additional assumption that execution halts at all constraint violations, and assigned unit terminal costs to those search nodes. Results on two example instances of PSR and SA domains were the following: I) in SA, C-POMDP and CC-POMDP both attained an utility of 29.454; II) in PSR, C-POMDP reached a final cost of 53.330, while CC-POMDP attained 36.509. The chance constraints were always identical for C-POMDP and CC-POMDP. First, one should notice that both models had the same performance in the SA domain, which is in agreement with the claim that they coincide in the particular case were all constraint violations are terminal. The same, however, clearly does not hold in the PSR domain, where the C-POMDP model had significantly worse performance than its corresponding CC-POMDP with the exact same parameters. Assuming that constraint violations are terminal in order to model them as costs greatly restricts the space of potential solution policies in domains with non-destructive constraint violations, leading to conservatism. A CC-POMDP formulation, on the other hand, can potentially attain significantly better performance while offering the same safety guarantee.

Window[s]	Δ	Time[s]	Nodes	States	Utility
20	0.05	1.30	1	32	0.000
30	0.01	1.32	1	32	0.000
30	0.05	49.35	83	578	29.168
40	0.002	9.92	15	164	21.958
40	0.01	44.86	75	551	29.433
40	0.05	38.79	65	443	29.433
100	0.002	95.23	127	1220	24.970
100	0.01	184.80	161	1247	29.454
100	0.05	174.90	151	1151	29.454

Table 1: SA results for various time windows and risk levels. The *Window* column refers to the time window for the SA agent to gather information, not a runtime limit for RAO*.

Δ	Time[s]	Nodes	States	Cost
0	0.025/0.013	1.57/1.29	5.86/2.71	45.0/30.0
.5	0.059/0.014	3.43/1.29	10.71/2.71	44.18/30.0
1	2.256/0.165	69.3/11.14	260.4/23.43	30.54/22.89
0	0.078/0.043	2.0/1.67	18.0/8.3	84.0/63.0
.5	0.157/0.014	3.0/1.29	27.0/2.71	84.0/30.0
1	32.78/0.28	248.7/5.67	1340/32.33	77.12/57.03
0	1.122/0.093	7.0/2.0	189.0/12.0	126.0/94.50
.5	0.613/0.26	4.5/4.5	121.5/34.5	126.0/94.50
1	123.9/51.36	481.5/480	8590.5/2648	117.6/80.89

Table 2: PSR results for various numbers of faults (#) and risk levels. Top: avg. of 7 single faults. Middle: avg. of 3 double faults. Bottom: avg. of 2 triple faults. Left (right) numbers correspond to 12 (16) network sensors.

6 Conclusions

We have presented RAO*, an algorithm for optimally solving CC-POMDP's. By combining the advantages of AO* in the belief space with forward propagation of risk upper bounds, RAO* is able to solve challenging risk-sensitive planning problems of practical interest and size. Our agenda for future work includes generalizing the algorithm to move away from the finite horizon setting, as well as more general chance constraints, including temporal logic path constraints (Teichteil-Königsbuch 2012).

Acknowledgements

This research was partially funded by AFOSR grants FA95501210348 and FA2386-15-1-4015, the SUTD-MIT Graduate Fellows Program, and NICTA. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. We would also like to thank the anonymous reviewers for their constructive and helpful comments.

References

Altman, E. 1999. *Constrained Markov Decision Processes*, volume 7. CRC Press.

Benazera, E.; Brafman, R.; Meuleau, N.; Hansen, E. A.; et al. 2005. Planning with continuous resources in stochastic domains. In *International Joint Conference on Artificial Intelligence*, volume 19, 1244.

Birge, J. R., and Louveaux, F. V. 1997. *Introduction to stochastic programming*. Springer.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, 52–61.

Bonet, B., and Geffner, H. 2009. Solving pomdps: Rtdp-bel vs. point-based algorithms. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 1641–1646.

Bonet, B., and Thiébaux, S. 2003. Gpt meets psr. In *13^{th} International Conference on Automated Planning and Scheduling*, 102–111.

Dolgov, D. A., and Durfee, E. H. 2005. Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 1326–1331.

Feinberg, E., and Shwarz, A. 1995. Constrained discounted dynamic programming. *Math. of Operations Research* 21:922–945.

Hou, P.; Yeoh, W.; and Varakantham, P. 2014. Revisiting risk-sensitive mdps: New algorithms and results. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Isom, J. D.; Meyn, S. P.; and Braatz, R. D. 2008. Piecewise linear dynamic programming for constrained pomdps. In *Proceedings 23rd AAAI Conference on Artificial Intelligence*, 291–296.

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

Kim, D.; Lee, J.; Kim, K.; and Poupart, P. 2011. Point-based value iteration for constrained pomdps. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 1968–1974.

Nilsson, N. J. 1982. *Principles of artificial intelligence*. Springer.

Ono, M., and Williams, B. C. 2008. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 3427–3432. IEEE.

Ono, M.; Kuwata, Y.; and Balaram, J. 2012. Joint chance-constrained dynamic programming. In *CDC*, 1915–1922.

Poupart, P.; Malhotra, A.; Pei, P.; Kim, K.-E.; Goh, B.; and Bowling, M. 2015. Approximate linear programming for constrained partially observable markov decision processes. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2164–2172.

Smallwood, R., and Sondik, E. 1973. The optimal control of partially observable markov decision processes over a finite horizon. *Operations Research* 21(5):107188.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 576–583.

Teichteil-Königsbuch, F. 2012. Path-Constrained Markov Decision Processes: bridging the gap between probabilistic model-checking and decision-theoretic planning. In *ECAI*, 744–749.

Thiébaux, S., and Cordier, M.-O. 2001. Supply restoration in power distribution systems — a benchmark for planning under uncertainty. In *Proc. 6th European Conference on Planning (ECP)*, 85–95.

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

Washington, R. 1996. Incremental markov-model planning. In *Tools with Artificial Intelligence, 1996., Proceedings Eighth IEEE International Conference on*, 41–47. IEEE.