A Finite-State Controller Based Offline Solver for Deterministic POMDPs

Alex Schutz[1] , **Yang You**[2] , **Matías Mattamala**[1] , **Ipek Caliskanelli**[2] , **Bruno Lacerda**[1] and **Nick Hawes**[1]

[1]University of Oxford

[2]UK Atomic Energy Authority

{alexschutz, matias, bruno, nickh}@robots.ox.ac.uk, {yang.you, ipek.caliskanelli}@ukaea.uk

Abstract

Deterministic partially observable Markov decision processes (DetPOMDPs) often arise in planning problems where the agent is uncertain about its environmental state but can act and observe deterministically. In this paper, we propose DetMCVI, an adaptation of the Monte Carlo Value Iteration (MCVI) algorithm for DetPOMDPs, which builds policies in the form of finite-state controllers (FSCs). DetMCVI solves large problems with a high success rate, outperforming existing baselines for DetPOMDPs. We also verify the performance of the algorithm in a real-world mobile robot forest mapping scenario.

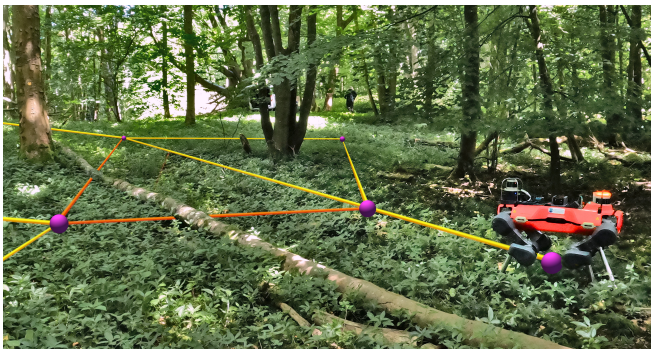


Figure 1: A topological map used for navigation in a forest where possibly obscured terrain leads to uncertain traversability.

1 Introduction

Many planning problems with environmental probabilities are naturally framed as deterministic partially observable Markov decision processes (DetPOMDPs), especially where the environment state is not fully known a-priori but can be observed during mission execution. A common example is robot navigation on a graph where the robot may not know the true traversability of the edges beforehand. Problems of this type include workplace environments where movement of workers and stock may block routes [Nardi and Stachniss, 2020; Tsang *et al.*, 2022; Lacerda *et al.*, 2019], or outdoor environments (see Figure 1) where the traversability of paths is uncertain [Huang *et al.*, 2023; Dey *et al.*, 2014].

DetPOMDPs have been under-studied in the literature, with existing approaches relying on the problems being cast as another problem type, such as a general POMDP or an AND-OR graph [Bonet, 2009]. These approaches have limited applicability on realistic problem sizes. In many domains, independent uncertainties result in a combinatorial state space. For example, in the robotic navigation domain, the number of states grows exponentially with the number of uncertain edges. Such situated AI and robotics problems are typically goal-oriented, as they involve reaching a destination or performing a task. Furthermore, resource constraints on the robot during navigation often restrict online planning. Therefore, desirable features of an algorithm for these applications include fast offline synthesis of compact policies with high goal achievement for problems with large state spaces.

In this paper, we introduce *DetMCVI*, an offline algorithm designed for goal-oriented DetPOMDPs, which achieves state-of-the-art performance on problems with large state spaces. DetMCVI is based on Monte Carlo Value Iteration (MCVI) [Bai *et al.*, 2011], adapted to goal-oriented settings as per Goal-HSVI [Horák *et al.*, 2018], and optimised for deterministic POMDPs. The algorithm builds policies as finite state controllers (FSCs), which allow for general connectivity in the policy graph. The FSC structure mitigates the failure cases of tree-based policies when planning is time limited, since it allows sub-solutions to be reused, minimising policy incompleteness. Furthermore, DetMCVI scales to domains out of reach of algorithms that require an explicit representation by sampling transitions. Our implementation is found at http://github.com/ori-goals/DetMCVI.

The contributions of this paper are: 1) the introduction of DetMCVI, a novel scalable algorithm for solving DetPOMDPs; 2) empirical analysis demonstrating that DetMCVI quickly generates compact policies which are more successful than current state-of-the-art baselines; 3) modelling of a real-world robotics problem involving topological navigation under uncertain environment conditions as a DetPOMDP.

2 Related Work

2.1 Deterministic POMDPs

A partially observable Markov decision process (POMDP) is used to model a Markov decision process (MDP) in which

the state is not fully observable. Interactions with the environment produce *observations*, which inform a *belief* about the current state based on observation probabilities. A deterministic POMDP is a restriction of a POMDP where actions and observations have deterministic outcomes [Bonet, 2009].

A POMDP can be modelled as a Belief-MDP, which is an MDP whose states are the possible beliefs of the POMDP. The number of states in the Belief-MDP of a DetPOMDP is upper bounded by $(1 + |\mathcal{S}|)^{|\mathcal{S}|}$ [Littman, 1996], making an exact approach generally computationally infeasible.

Bonet [2009] shows that DetPOMDPs have a direct relation to AND/OR graphs. These can be solved offline using search-based heuristic algorithms such as AO* [Chakrabarti, 1994], LAO* [Hansen and Zilberstein, 2001] and RTDP [Barto *et al.*, 1995], though we do not require the adaptations for cyclic graphs provided by the latter two. Anytime AO* [Bonet and Geffner, 2021] is a modification which probabilistically searches outside of the best graph, providing better solutions under early termination or with a non-admissible heuristic. AO*-based planners have been used for a number of real-world robotics problems [Guo and Barfoot, 2019; Chung and Huang, 2011; Ferguson *et al.*, 2004]. These search approaches produce policy trees, and do not leverage similarity in policy features. In our work we use a more compact policy representation to avoid repeated solving for similar states.

2.2 Related Problem Formulations

A related formulation is the POMDP-lite [Chen *et al.*, 2016], which restricts partial observability to state variables which change deterministically or are constant. Many of the POMDP-lite domains are examples of DetPOMDPs, as DetPOMDPs are a restriction of the POMDP-lite. Similarly, the Multiple-Environment MDP models problems in which the true environment may be one of many possible MDPs [Raskin and Sankur, 2014], though no distribution over possible environments is assumed. A DetPOMDP can also be framed as a Bayes-Adaptive MDP (BAMDP) [Duff, 2002], where the latent variable encodes the true realisation of the state and the initial distribution is used as the prior. Conformant planning [Bonet, 2010] and Contingent planning [Muise *et al.*, 2014; Brafman and Shani, 2021] consider problems with deterministic transitions, partial observability, and an unknown initial state. They differ from DetPOMDPs in that they do not account for a probability distribution over states.

2.3 Offline POMDP Solutions

While DetPOMDP solution methods have received relatively little attention, more well-researched POMDP methods can be applied to DetPOMDPs. The value function of a POMDP can be represented using a finite set of α-vectors [Shani *et al.*, 2013]. Point-based methods generate and optimise a set of α-vectors to approximate the value function. Heuristic Search Value Iteration (HSVI) [Smith and Simmons, 2005] bounds the values of beliefs in the belief tree to inform heuristics for a depth-first search, updating α-vectors in the backup operation. SARSOP [Kurniawati *et al.*, 2009] improves on HSVI by focusing on reachable beliefs under optimal policies. Horák *et al.* [2018] adapt HSVI for use on goal-oriented POMDPs by addressing the lack of convergence guarantees for non-discounted problems, adding a depth bound and preventing re-exploration of histories. Point-based approaches require evaluating all α-vectors in each state, making planning difficult in large state spaces. Each of these methods requires explicit knowledge of the transition and observation functions to calculate value estimates, thus cannot be used if these functions are unknown or too large to encode.

As an alternative to α-vectors, many approaches directly compute an FSC, which represents policies using action nodes and observation edges that lead to the subsequent action node. Andriushchenko *et al.* [2022] search for the best FSC from a set of candidates before expanding the search space in an iterative process, though the approach is limited in scalability as the size of the FSC increases. Other approaches construct FSCs via non-linear programming [Amato *et al.*, 2010], parametric Markov chains [Junges *et al.*, 2018], Anderson acceleration [Ermis *et al.*, 2021] and belief-integrated FSCs [Wray and Zilberstein, 2019]. However, each of these approaches requires an enumeration of states, which presents scalability barriers for very large state spaces.

Monte Carlo Value Iteration (MCVI) [Bai *et al.*, 2011] iteratively builds an FSC while searching a belief tree using a similar method to SARSOP, calculating value estimates using Monte Carlo simulations. This approach requires only samples of transitions, and works on continuous-state POMDPs, thus being suitable for large finite state spaces. In fact, the size of the state space need not be known in advance as states are only accessed from sampled beliefs, in contrast to point-based approaches which evaluate over entire the state space. These properties make MCVI favourable for specialisation to large DetPOMDPs, which we describe in Section 4.

For problems with large state-spaces, prior works typically plan online [Bonet and Geffner, 2021; Eyerich *et al.*, 2010; Silver and Veness, 2010; Chatterjee *et al.*, 2020]. Direct offline application is limited by memory inefficiency. The QMDP heuristic considers uncertain observations only at the root belief, and assumes a fully observable MDP for child beliefs [Littman *et al.*, 1995]. Bai *et al.* [2013] propose a recursive QMDP-based offline policy tree algorithm for solving continuous-state robotics problems with deterministic dynamics. We use QMDP Trees as a baseline for comparison.

3 Background

3.1 POMDPs and DetPOMDPs

Following [Bonet, 2009], we consider the goal-oriented formulation of a DetPOMDP, thus also define POMDPs in the goal-oriented setting.

Definition 1. A goal-oriented POMDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, b_0, \mathcal{G}, \mathcal{T}, \mathcal{Z}, c \rangle$ where: i) $\mathcal{S}$ is the set of states; ii) $\mathcal{A}$ is the set of actions; iii) $\mathcal{O}$ is the set of observations; iv) $b_0 \in \Delta(\mathcal{S})$ is the initial state distribution; v) $\mathcal{G} \subseteq \mathcal{S}$ is a set of absorbing goal states; vi) $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function; vii) $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ is the observation probability function; viii) $c : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ is the immediate cost of applying action a in state s, with $c(s, a) = 0 \iff s \in \mathcal{G}$.

A belief b for a POMDP is a probability distribution over its state space. We denote the set of all beliefs over state space

$\mathcal{S}$ as $\Delta(\mathcal{S})$. Hereafter, we use $\text{Supp}(b)$ to indicate the set of states in the support of belief b, that is $\{s \in \mathcal{S} \mid b(s) > 0\}$.

A solution to a POMDP is a policy π which maps an action-observation history $h_t = (a_0, o_1, a_1, \ldots, a_{t-1}, o_t)$ to an action a_t. For a policy π, the value of belief b is given by:

$$V^\pi(b) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} c(s_t, a_t) \mid b_0 = b \right]. \quad (1)$$

We seek to minimise the expected cost of the policy given the initial belief b_0, i.e., find the policy π which minimises $V^\pi(b_0)$. We denote the value of the belief for the optimal policy as V^*. The action-value function Q relates the expected cost of executing action a in belief b and subsequently following the policy π:

$$Q^\pi(b, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} c(s_t, a_t) \mid b_0 = b,\ a_0 = a \right]. \quad (2)$$

The value iteration backup equation constructs a new estimate of the value function from a previous estimate $\tilde{V}$:

$$\tilde{V}'(b) = \min_{a \in \mathcal{A}} \left\{ \sum_{s \in \mathcal{S}} c(s, a) b(s) + \sum_{o \in \mathcal{O}} \Pr(o|b, a) \tilde{V}(b') \right\}. \quad (3)$$

Here, b' is the subsequent belief which can be calculated using Bayes' rule:

$$b'(s') = \tau(b, a, o)(s') = \zeta \mathcal{Z}(s', a, o) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s') b(s), \quad (4)$$

where ζ is a normalisation constant.

In goal-oriented POMDPs, the value function for π (Equation 1) is finite if and only if π reaches a goal state with probability 1 (π is called *proper*). For convergence, we assume the existence of one such policy [Mausam and Kolobov, 2012].

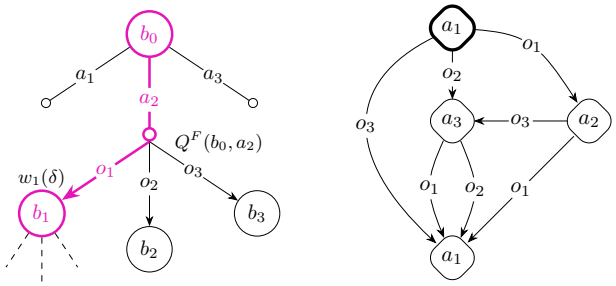
We represent reachable beliefs in a POMDP as a *belief tree*.

Definition 2. A *belief tree* rooted at b_0 is a tree in which nodes represent beliefs $b \in \Delta(\mathcal{S})$, with edges defined by action-observation pairs $e \in \mathcal{A} \times \mathcal{O}$. A child node b' is computed from a parent node b via belief update, i.e., if b' is the child of b and the edge from b to b' is (a, o), then $b' = \tau(b, a, o)$, as defined in Equation 4.

It is not feasible to build a dynamic programming algorithm directly from Equation 3, because the belief space of a POMDP is infinite. Many approaches have been proposed to approximate solutions to POMDPs, and in this paper we focus on MCVI, which we will present next. Before doing so, we define the DetPOMDP specialisation of a POMDP.

Definition 3. A DetPOMDP is a POMDP in which the transition and observation functions are deterministic. We denote the transition function as $f_\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, returning the subsequent state after taking action a in state s; and the observation function as $f_\mathcal{Z} : \mathcal{S} \times \mathcal{A} \to \mathcal{O}$, giving the observation after entering state s' using action a.

The uncertainty in a DetPOMDP is only in the initial belief. Thus, if the state is known exactly at any point in the decision process, the problem is reduced to a deterministic shortest path problem from that point forward.



(a) Belief tree (b) FSC

Figure 2: A partial belief tree and FSC built during MCVI iteration.

Example 1. The robot navigation problem from the introduction can be posed as a *Canadian Traveller Problem (CTP)* [Papadimitriou and Yannakakis, 1989], a topological navigation problem in which some edges are potentially blocked (with a known probability), and the true traversability of the edge can only be observed by the agent at one of the edge's terminal nodes. The CTP is a DetPOMDP with costs for edge traversal, a goal node, and an initial belief given by the start node and the edge traversability probabilities.

3.2 Policy Representations

Policies for POMDPs can be represented in several ways. In this paper, we are interested in policy trees and FSCs.

Definition 4. A *finite-state controller (FSC)* is a tuple $F = \langle \mathcal{V}, \eta, \psi \rangle$, where i) $\mathcal{V}$ is a finite set of nodes, with start node v_0; ii) $\psi : \mathcal{V} \to \mathcal{A}$ is the action selection function, where $a = \psi(v)$ is the action selected when in node v; iii) $\eta : \mathcal{V} \times \mathcal{O} \to \mathcal{V}$ is the node transition function, where $v' = \eta(v, o)$ is the node transitioned into after observing o in node v.

Definition 5. A *policy tree* is an FSC with no cycles and each node v' having at most one node-observation pair (v, o) such that $\eta(v, o) = v'$.

3.3 Monte Carlo Value Iteration

Monte Carlo Value Iteration (MCVI) [Bai *et al.*, 2011] is an offline method for computing FSCs, designed for continuous-state POMDPs and therefore also suitable for solving large discrete-state POMDPs. The main algorithm for MCVI is a belief tree search, described in Algorithm 1. In this process, a belief tree T is traversed by choosing child beliefs to expand using a guiding heuristic, and then the tree is traversed in reverse order while the bounds at each belief are refined and an FSC F is updated. The search terminates when the bounds at b_0 are within a suitable convergence threshold ϵ.

During the tree traversal, actions are chosen to minimise the Q function. Observations are chosen which maximise the weighted excess uncertainty of the child belief (line 21), as per HSVI [Smith and Simmons, 2005]. Child beliefs are generated via particle filtering, with an initial set of N samples (line 17). The belief tree traversal is illustrated in Figure 2a. Upon reaching a terminal state, or when the excess uncertainty of a belief is negative, the tree is traversed in reverse order, performing a backup operation at each node (line 8).

Algorithm 1: Belief Tree Search

Function SEARCHBELIEFTREE(b_0, ϵ, h)
  Initialise an FSC F with an empty set of nodes.
  Initialise belief tree T with root b_0.
  $\overline{V}(b_0) \leftarrow \infty, \underline{V}(b_0) \leftarrow \mathcal{H}(b_0)$.
  while $\overline{V}(b_0) - \underline{V}(b_0) > \epsilon$ **do**
    $(b_i)_{i=0}^{k} =$TRAVERSEBELIEFS(T, b_0, ϵ, h, F)
    for $i \in k, k-1, \ldots, 0$ **do**
      $F \leftarrow$BACKUP(F, b_i).
      $\overline{V}(b_i) \leftarrow V^F(b_i)$. *from* (6)
      $\underline{V}(b_i) \leftarrow \underline{V}'(b_i)$. *from* (3)
  return F
Function TRAVERSEBELIEFS(T, b_0, ϵ, h, F)
  $k \leftarrow 0$.
  do
    $a^\star \leftarrow \arg\min_{a \in \mathcal{A}} Q^F(b_k, a)$.
    for $o \in \mathcal{O}$ **do**
      $b^o \leftarrow \tau(b_k, a^\star, o)$.
      if $b^o \notin T$ **then**
        Add b^o to T with parent b_k and edge $(a^\star, o)$.
        $\overline{V}(b^o) \leftarrow \alpha_{F,v_0}(b^o), \underline{V}(b^o) \leftarrow \mathcal{H}(b^o)$.
      $\delta_{b^o} \leftarrow \overline{V}(b^o) - \underline{V}(b^o) - \epsilon$.
    $o^\star \leftarrow \arg\max_{o \in \mathcal{O}} \Pr(o|a^\star, b_k)\delta_{b^o}$.
    $b_{k+1} \leftarrow \tau(b_k, a^\star, o^\star)$.
    $k \leftarrow k + 1$.
  while $\delta_{b_k} > 0$
  return $(b_i)_{i=0}^{k}$.

Given an FSC F with starting node v, the expected cost of executing F from state s is $\alpha_{F,v}(s)$:

$$\alpha_{F,v}(s) = \mathbb{E}_F \left[\sum_{t=0}^{\infty} c(s_t, a_t) \mid s_0 = s \right]. \tag{5}$$

We will write $\alpha_{F,v}(b)$ to mean $\sum_{s \in \text{Supp}(b)} b(s)\alpha_{F,v}(s)$. From (1), the value of the belief b under F is:

$$V^F(b) = \min_{v \in \mathcal{V}} \alpha_{F,v}(b). \tag{6}$$

In MCVI, $\alpha_{F,v}(s)$ is calculated using Monte Carlo simulations of F, given sample-based access to the transition function for subsequent states and observations, with a default policy provided where F is undefined. An upper bound $\overline{V}(b)$ for $V^*(b)$ is given by $V^\pi(b)$ for any policy π, so we choose $V^F(b)$ to determine the upper bound using (6). The backup process updates F by adding a new node which improves this upper bound. Details of the backup process can be found in the appendix. An example of an FSC built using the MCVI backup process is shown in Figure 2b.

Lower bounds are initialised using an admissible heuristic $\mathcal{H}$. One such heuristic is given by relaxing the POMDP to a fully observable MDP and solving using a suitable MDP method. The lower bound is updated using the value iteration backup (3), given the lower bounds of the child beliefs.

Note that due to the forward-only construction of the FSC, MCVI cannot generate policies with loops, and is thus unable to support infinite horizon problems without a default policy, which is generally non-trivial to devise. In general, MCVI is not guaranteed to converge for non-discounted POMDPs [Smith, 2007]; we address this in the design of DetMCVI.

Algorithm 2: DetMCVI Backup

1 **Function** BACKUP($F = \langle \mathcal{V}, \eta, \psi \rangle, b$)
2 For each action $a \in \mathcal{A}$, $C_a \leftarrow 0$, $\mathcal{O}_{b,a} \leftarrow \emptyset$.
3 For each action $a \in \mathcal{A}$, each observation $o \in \mathcal{O}$, and each node $v \in \mathcal{V}$, $V_{a,o,v} \leftarrow 0$.
4 **for** each action $a \in \mathcal{A}$ **do**
5 **for** $s_i \in$ Supp(b) **do**
6 $s_i' \leftarrow f_\mathcal{T}(s_i, a)$.
7 $o_i \leftarrow f_\mathcal{Z}(s_i', a)$.
8 $\mathcal{O}_{b,a} \leftarrow \mathcal{O}_{b,a} \cup \{o_i\}$.
9 $C_a \leftarrow C_a + b(s_i)c(s_i, a)$.
10 **for** each node $v \in \mathcal{V}$ **do**
11 Retrieve $\alpha_{F,v}(s_i')$ from cache or calculate via 1 rollout of the policy $\pi_{F,v}$.
12 $V_{a,o_i,v} \leftarrow V_{a,o_i,v} + b(s_i)\alpha_{F,v}(s_i')$.
13 **for** each observation $o \in \mathcal{O}_{b,a}$ **do**
14 $V_{a,o} \leftarrow \min_{v \in \mathcal{V}} V_{a,o,v}$.
15 $v_{a,o} \leftarrow \arg\min_{v \in \mathcal{V}} V_{a,o,v}$.
16 $V_a \leftarrow C_a + \gamma \sum_{o \in \mathcal{O}_{b,a}} V_{a,o}$.
17 $V^{F'} \leftarrow \min_{a \in \mathcal{A}} V_a$.
18 $a^\star \leftarrow \arg\min_{a \in \mathcal{A}} V_a$.
19 Create a new FSC $F' = \langle \mathcal{V}', \eta', \psi' \rangle$. Set $\psi'(v_0') = a^\star$ and $\eta'(v_0', o) = v_{a^\star,o}$. For $k \in 1, \ldots, |\mathcal{V}|$, set $\psi'(v_k') = \psi(v_{k-1})$ and $\eta'(v_k', o) = \eta(v_{k-1}, o)$.
20 **return** F'

4 DetMCVI

There are many inefficiencies when using MCVI to solve DetPOMDPs, including re-evaluation of policy rollouts, repeated sampling of deterministic transitions, and storage of beliefs. We propose DetMCVI, an adaptation of MCVI, which solves DetPOMDPs efficiently in the goal-oriented setting.

The overall process of DetMCVI is similar to Algorithm 1, with the main differences in the backup process, presented in Algorithm 2. Here, value estimates are created for the successor states s' of a belief b by finding the best node in the FSC from which to execute the policy in s' (line 12). These value estimates are used to create a new node in the FSC (line 19), labelled with the best action, with outgoing edges connecting to the best next node in the FSC for each possible observation. We will next describe the key differences between the DetMCVI backup function and that of MCVI.

4.1 Policy Rollouts

The repeated rollouts required by MCVI when calculating the expected policy value $\alpha_{F,v}$ can be eliminated under deterministic dynamics. The modifications we make for the policy rollouts are as follows: 1) we calculate $\alpha_{F,v}(s)$ using a single rollout, which produces the exact value instead of an approximation; 2) we implement a cache for values of $\alpha_{F,v}(s)$, as the value does not change when F is updated; 3) instead of

calculating the best policy node for each element in the entire set of observations, we restrict the set to only those observations o where $\Pr(o|b,a) > 0$ for a belief b and action a. This set is calculated during the belief expansion. This greatly reduces unnecessary computations, as the size of the full set of observations can be comparatively very large.

4.2 Belief Sampling

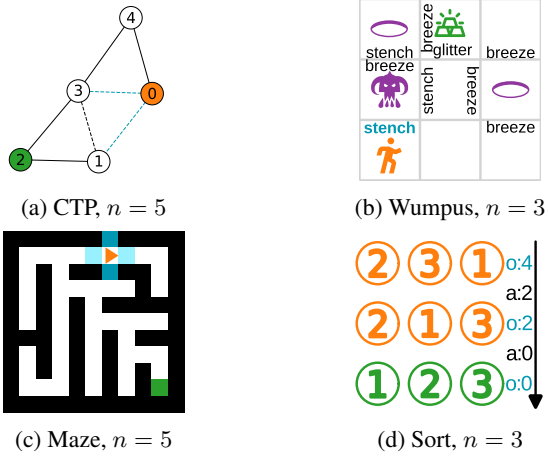
The use of Monte Carlo sampling for a deterministic problem is unnecessary, as for any action a each state only has one successor state $s' = f_{\mathcal{T}}(s,a)$ and one resultant observation $f_{\mathcal{Z}}(s',a)$. Thus, sampling multiple times will return the same result, a property we use for the following changes: 1) we sample N states and their probabilities from b_0 without replacement, instead of sampling N possibly repeated states as in MCVI. 2) In line 5, we iterate through each state in $\text{Supp}(b)$ instead of sampling each time, ensuring no duplication. 3) In a DetPOMDP, $|\text{Supp}(b')| \leq |\text{Supp}(b)|$ for any successor b' of belief b, as all states have one successor and these successors may not be unique. Thus, a limit imposed on the maximum size of the initial belief is never exceeded by any descendant beliefs, so N is not imposed in later belief sampling like in MCVI.

4.3 Bounds

As in MCVI, the algorithm performs a search on the belief tree, with upper and lower bounds maintained at each node, until the bounds at the root node converge with a specified tolerance. Each time a backup is performed, the upper bound $\overline{V}$ is updated according to the value of the belief in the new FSC. For leaf nodes the lower bound is calculated using an admissible heuristic, for example the full-observability MDP relaxation of the problem. In a DetPOMDP this relaxation reduces the problem to a set of deterministic shortest path problems: $\underline{V}(b) = \sum_{s \in \text{Supp}(b)} b(s)\text{dist}(s,\mathcal{G})$, where $\text{dist}(s,\mathcal{G})$ is the cost of the shortest path to the goal from state s.

4.4 Convergence

We apply the approach of Horák *et al.* [2018] to guarantee convergence of the algorithm for a goal-oriented DetPOMDP under the same conditions used by the authors, namely that the goal is reachable from all states. 1) We remove the requirement for a default policy and use uniform random action selection in the rollout calculation of $\alpha_{F,v}$ where F is undefined, with guaranteed termination [Chatterjee *et al.*, 2016]. Alternatively, we can remove the requirement for the goal to be reachable from all states so long as rollouts default to a policy which is proper. 2) We prevent re-exploration of action-observation histories by labelling each node in the belief tree with a binary flag indicating a closed belief. This flag is set when all states in the belief are terminal, or when all child beliefs of the node are closed. Closed beliefs are skipped during belief expansion. As per Horák *et al.* [2018], DetMCVI attains ϵ-optimality under the conditions when we impose a bound on the search depth $\overline{T} = \frac{\overline{C}}{c_{\min}} \frac{\overline{C}\eta\epsilon}{(1-\eta)\epsilon}$ for some $\eta < 1$ where $\overline{C}$ is the upper bound on the cost of the uniform policy, and $c_{\min}$ is the minimum per-step cost.



(a) CTP, $n = 5$ (b) Wumpus, $n = 3$

(c) Maze, $n = 5$ (d) Sort, $n = 3$

Figure 3: Selected problem instances from each domain

5 Synthetic Experiments

We evaluate the performance of DetMCVI in different DetPOMDP scenarios using the problem domains illustrated in Figure 3. 1) CTP: from Example 1. The size of the state space grows exponentially with the number of uncertain edges. 2) Wumpus World: a goal-oriented modification of that from Russell and Norvig [2021]. Due to the presence of both low- and high-cost goal states, this domain illustrates the importance of optimising the reward function in the process of seeking a goal state. 3) Maze: this domain has a long horizon but can be solved suboptimally by small FSCs. 4) Sort: this domain has a short horizon but a solution requires many branches for optimality. Full domain descriptions can be found in the appendix.

5.1 Methodology

In each domain, policies were generated using several baseline solvers. Specific to DetPOMDPs, we evaluate DetMCVI, AO* [Chakrabarti, 1994], Anytime AO* [Bonet and Geffner, 2021], and QMDP Trees [Bai *et al.*, 2013]. We also evaluate general POMDP solvers MCVI [Bai *et al.*, 2011] and SARSOP [Kurniawati *et al.*, 2009]. For MCVI we use Q-learning for the lower bound heuristic [Watkins, 1989] as we assume only sample-based access to the model, while for DetMCVI, AO* and QMDP Trees we use the bounded-depth Bellman-Ford algorithm to compute the shortest path under full observability without enumerating the entire state space. For Anytime AO* we use uniform policy rollouts, being faster than the QMDP heuristic but inadmissible. Implementation details can be found in the appendix. We impose a domain-dependent horizon T to shorten computation time for practicality. This means that convergence to an ϵ-optimal policy is not guaranteed, but we find that the policies produced by DetMCVI are of sufficient quality long before convergence.

Policies were evaluated at regular intervals using 10^5 trials from states randomly sampled from the initial belief. A trial concluded when a goal state was reached, the horizon was reached, or $\pi(h_t)$ was undefined. This may occur if an observation is made in a node which does not have provision for that observation, for example when planning does

		CTP			Wumpus			Maze			Sort	
n		20	50	100	2	3	4	10	15	20	5	7
$\lvert\mathcal{S}\rvert$		$8.60\text{x}10^4$	$8.76\text{x}10^{11}$	$1.19\text{x}10^{23}$	$3.32\text{x}10^5$	$7.55\text{x}10^7$	$4.19\text{x}10^{10}$	793	1793	3193	120	5040
$\lvert\text{Supp}(b_0)\rvert$		2790	$1.72\text{x}10^{11}$‡	$1.18\text{x}10^{21}$‡	72	$\geq 10^4$‡	$\geq 7.5\text{x}10^4$‡	792	1792	3192	119	5039
T		40	100	200	100	150	200	420	930	1640	10	14
MCVI	SR	60.8 ± 31.4									20.6 ± 31.0	0.8 ± 0.7
	$\mathcal{R}$	-	†	†	*	*	*	*	*	*	-	-
	t_{plan}	2037.6 ± 1063.2									140.00 *	18000 *
	$\lvert\pi_F\rvert$	111 ± 51									126 ± 7	721 ± 29
DetMCVI	SR	**100.0 ± 0**	**100.0 ± 0.02**	**99.7 ± 0.5**	**100.0 ± 0**	**93.9 ± 4.1**	**97.1 ± 1.0**	**100.0 ± 0**	**100.0 ± 0**	**99.5 ± 0.8**	**100.0 ± 0**	91.7 ± 0.6
	$\mathcal{R}$	1.184 ± 0.011	**1.18 ± 0.05**	1.964 ± 0.070	28.4 ± 0.4	**118 ± 1**	**182 ± 2**	16.1 ± 0.1	27.6 ± 0.1	**40.2 ± 0.1**	27.0 ± 0.2	52.5 ± 0.3
	t_{plan}	**3.8 ± 0.8**	**310 ± 99**	**2594 ± 315**	3.78 ± 0.40	1200 *	18000 *	419 ± 128	5167 ± 1383	**32223 ± 5048**	2.14 ± 0.02	18000 *
	$\lvert\pi_F\rvert$	**11 ± 2**	**24 ± 11**	**39 ± 14**	**100 ± 6**	**163 ± 61**	**636 ± 18**	493 ± 46	1098 ± 113	**1647 ± 119**	164 ± 8	**1921 ± 70**
AO*	SR	**100.0 ± 0**	97.3 ± 1.7	72.8 ± 18.4	**100.0 ± 0**	7.0 ± 1.2	4.8 ± 1.4	7.9 ± 2.0	4.8 ± 1.2	2.1 ± 0.4	**100.0 ± 0**	0.3 ± 0.1
	$\mathcal{R}$	**0.997 ± 0.010**	**1.18 ± 0.05**	**0.629 ± 0.023**	**22.8 ± 0.4**	-	-	-	-	-	**24.4 ± 0.2**	-
	t_{plan}	55.4 ± 72.6	901 ± 398	18007 ± 8602	**0.21 ± 0.01**	1200 *	18000 *	1200 *	14400 *	36000 *	59.32 ± 2.23	18000 *
	$\lvert\pi_F\rvert$	329 ± 162	2768 ± 1394	11843 ± 4598	163 ± 7	127 ± 27	97 ± 25	3645 ± 387	9005 ± 774	13434 ± 2213	324 ± 0	401 ± 38
Anytime AO*	SR	99.4 ± 1.2			**100.0 ± 0**	1.1 ± 2.2	4.3 ± 1.0	2.3 ± 0.6	1.7 ± 0.4	0.9 ± 0.2	92.4 ± 13.5	0.3 ± 0.3
	$\mathcal{R}$	**0.997 ± 0.010**	†	†	**22.8 ± 0.4**	-	-	-	-	-	28.7 ± 0.2	-
	t_{plan}	33.2 ± 718.2			2.01 ± 0	1200 *	18000 *	1200 *	14400 *	36000 *	96.79 ± 28.25	18000 *
	$\lvert\pi_F\rvert$	323 ± 151.8			163 ± 7	47 ± 22	151 ± 89	1969 ± 256	5116 ± 397	7651 ± 1051	350 ± 46	224 ± 72
QMDP Trees	SR	**100.0 ± 0.01**	97.3 ± 1.7	79.6 ± 10.9	0 ± 0	23.3 ± 0.6	15.5 ± 16.4	36.6 ± 9.7	23.8 ± 2.9	17.4 ± 2.0	**100.0 ± 0**	**97.9 ± 0.4**
	$\mathcal{R}$	**0.997 ± 0.010**	**1.18 ± 0.04**	0.817 ± 0.030	-	-	-	-	-	-	28.9 ± 0.2	**41.5 ± 0.3**
	t_{plan}	5.9 ± 0.9	631 ± 89	6135 ± 383	0.11 ± 0	324 ± 2	372 ± 41	20 ± 2	156 ± 26	586 ± 49	**0.01 ± 0.00**	**4 ± 0**
	$\lvert\pi_F\rvert$	329 ± 162	2768 ± 1394	13928 ± 6650	401 ± 0	2119 ± 0	3773 ± 729	37992 ± 4938	119560 ± 14299	260731 ± 15576	352 ± 6	15819 ± 31
SARSOP	SR							**100.0 ± 0**	**100.0 ± 0**		**100.0**	
	$\mathcal{R}$	†	†	†	†	†	†	**5.2 ± 0.0**	**6.3 ± 0.0**	†	25.0 ± 0.2	†
	t_{plan}							**8 ± 2**	**100 ± 14**		0.19	
	$\lvert\pi_F\rvert$							**402 ± 28**	**949 ± 48**		**107**	

* Computation time limit reached † Memory limit reached ‡ Belief downsampling applied

Table 1: Evaluation of offline algorithms on large DetPOMDP domains. SR = success rate (%), $\mathcal{R}$ = mean regret, t_{plan} = wall-clock planning time (s), $|\pi_F|$ = number of nodes in policy tree or FSC. DetMCVI solves the benchmarks with high success rates and small policy sizes.

not converge or when downsampling results in states not being planned for. Planning was terminated after reaching a timeout or memory limitations, or when all trials reached the goal in an evaluation. Performance is calculated over sets of 10 problem instances for the CTP and Maze problems, and over three random seeds for Wumpus and Sort.

Belief Downsampling. For problems where $|\text{Supp}(b_0)|$ is large, planning can be slow due to processes which operate on all states in the support, such as belief updates and the heuristic calculation. As described in Section 4.2, we use a belief for planning which has at most N states in the support. As the initial belief is only accessed by sampling states, we take $10N$ samples from b_0 to create a distribution of relative likelihoods, and choose the first N states from a weighted shuffle and renormalise their probabilities. We use $N = 10^5$, and we plan with the same sampled initial belief across all baselines. For the larger CTP and Wumpus problems, $N<|\text{Supp}(b_0)|$, meaning that we do not plan for all states in the support of the initial belief. The choice of N is analysed in Section 5.4.

Metrics. We define metrics for a trial beginning in state s_0 following policy π. The return R_k of a trial up to step k is given by $\sum_{t=0}^{k-1} c(s_t, \pi(h_t))$. For trials where $\pi(h_t)$ is defined for all $t \in 0, \ldots, T$, we define the full observability *regret* $\mathcal{R} = R_T - \text{dist}(s_0, \mathcal{G})$. We call a trial *successful* if the goal state is reached under π at a step $t < T$, and *failed* otherwise.

5.2 Results

Results in Table 1 demonstrate the performance of each algorithm on a selection of different problem instances. We show the regret for algorithms with a success rate greater than 70%, and the regret is calculated only for trials from initial states which were successful under all of those algorithms.

The results demonstrate the strong performance of DetMCVI in quickly finding very compact solutions which reliably achieve the goal in problems with combinatorial state spaces. Across the problem domains, DetMCVI consistently has a **high success rate metric**, outperforming the baselines in scalability. The number of nodes in the policies produced by DetMCVI were **significantly lower** than the policy tree-based baselines, for example by 360 times for CTP with $n = 100$, but still competitive in the regret metric. This difference in policy size is primarily because policy tree-based approaches are not able to reuse existing plans for new branches. In problems where $N < |\text{Supp}(b_0)|$, this can result in a drop in performance as the policy tree branch is not defined from some states, whereas an FSC policy can still be followed.

Baselines. SARSOP produced high quality solutions, but could only be applied to the smallest problems due to memory constraints, demonstrating the advantage of a sample-based algorithm. Due to the lack of a backup process, QMDP Trees prove to be effective in domains where many actions have similar value, so planning for different alternatives does not greatly benefit the solution. Despite the faster heuristic, Anytime AO* did not outperform AO* and suffered from memory constraints due to expanding extraneous parts of the belief tree. MCVI could not produce policies for most of the problems, mainly attributed to the calculation of the heuristic.

5.3 Planning with a Budget

Though offline planning is nominally performed with an infinite time allowance, real-world planning demands time constraints. It is therefore important to understand the performance of a planning algorithm when terminated early. Figure 4 shows the evolution of policies with increasing planning time for a CTP problem with $n = 25$. The success rate of the

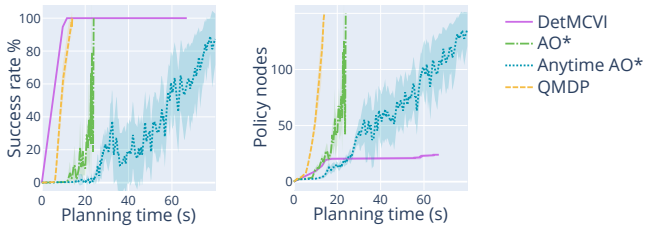


Figure 4: Success rate (left) and policy size (right) for different algorithms as evaluated on a CTP problem with $n = 25$.

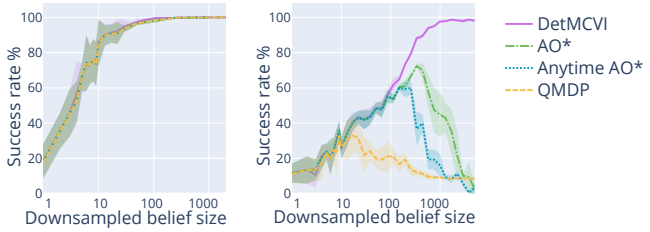


Figure 5: Success rate of policies generated with downsampled initial beliefs. Left: CTP $n = 20$, $|\text{Supp}(b_0)| = 2048$. Right: Wumpus $n = 3$, $|\text{Supp}(b_0)| \geq 10850$.

DetMCVI policy quickly improves, reaching 100% success rate within 11 seconds with a policy size of 11 nodes. In contrast, AO* and QMDP Trees take 24 and 14 seconds respectively to achieve a high success rate, and produce policies in excess of 140 nodes. Because DetMCVI chooses actions according to the upper bound, it is able to quickly find a general solution that reaches the goal from many states, and then improve the cost of the solution for specific states. Conversely, other DetPOMDP planning approaches choose actions using the lower bound, meaning that the goal is not reachable under intermediate policies until planning is complete, even using an anytime method like Anytime AO*.

5.4 Belief Downsampling

We evaluate the impact of downsampling the initial belief by planning using a range of values for N and evaluating performance over the true initial belief. Figure 5 demonstrates the success rate of policies for a CTP problem ($n = 20$) and a Wumpus problem ($n = 3$), noting the logarithmic scale. In the CTP problem, all algorithms except Anytime AO* converged within the time limit, and the planning times increased approximately linearly with N. For the Wumpus problem, DetMCVI and AO* also began to be affected by the time limit for larger values of N. QMDP Trees performance in Wumpus degrades due to bias toward safe but ineffective actions. These results show that for these domains, downsampling can offer improvements in computation time, while only affecting solution quality for smaller values of N ($\approx |\text{Supp}(b_0)|/5$).

6 Forest Experiment

The main advantage of DetMCVI is fast synthesis of compact policies with high reusability across states. This property enables us to use the algorithm on a robot, in a setting where failing to reach the goal is catastrophic, and other algorithms fail to plan sufficiently under the constraints of the real world. As shown in Figure 1, we evaluate on a robotic navigation problem involving the ANYbotics ANYmal D. The problem (further described in the appendix) is a modification of the CTP, in which edges can only be observed by attempting to traverse them, rather than being observed from a node. We use a map generated from operator-guided navigation in a forest, with shortcut edges added for the autonomous phase.

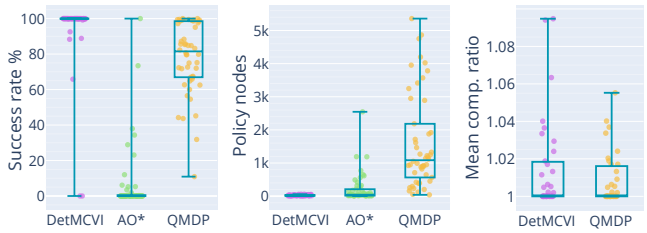


Figure 6: Success rate, policy size, and mean competitive ratio over different map realisations from the field data.

We evaluate on 50 map instances with randomised edge traversal probabilities and start and goal locations, showing results in Figure 6. Anytime AO*, MCVI, and SARSOP failed to return usable policies for the planning budget of 300 seconds. Across all instances, the average success rates for DetMCVI, AO*, and QMDP Trees respectively were 95%, 7%, and 78%; and the average policy sizes were 24, 225 and 1610. We use the canonical CTP metric of competitive ratio, defined as the ratio of achieved cost to the best cost under full observability, and we show only the trials which succeeded in both DetMCVI and QMDP Trees (AO* did not have a sufficient success rate to include). The average competitive ratios for DetMCVI and QMDP Trees were 1.014 and 1.008 respectively. Of the successful trials, DetMCVI matches or outperforms the competitive ratio of QMDP Trees on 28 of 35 maps. The results show that DetMCVI nearly always returns a 100% success rate and has very small policy sizes, with occasionally slightly higher competitive ratios for successful trials. In the two maps where DetMCVI failed to produce a viable policy, all algorithms performed poorly, indicating that the planning budget was not high enough for these instances.

7 Conclusion

We present DetMCVI, an algorithm for solving DetPOMDPs in goal-oriented environments. This algorithm is a simple yet highly effective adaptation of MCVI [Bai *et al.*, 2011] and Goal-HSVI [Horák *et al.*, 2018], able to scale to large problems as well as provide convergence guarantees under the same assumptions used by Goal-HSVI. Empirical evaluations demonstrate that our algorithm obtains competitive performance while scaling to larger problems and producing highly compact policies. These policies can be several orders of magnitude smaller than state-of-the-art approaches for DetPOMDPs, making it beneficial in computationally constrained settings such as on a mobile robot. Overall, our approach offers a promising method for solving offline, goal-driven problems with deterministic dynamics and uncertain states. Future work includes augmenting the FSC construction to allow loops, and more efficient heuristic calculations.

Acknowledgements

This work received EPSRC funding via the "From Sensing to Collaboration" programme grant [EP/V000748/1] and the UKAEA/EPSRC Fusion Grant [EP/W006839/1]. A.S. was supported by a scholarship from the General Sir John Monash Foundation.

References

[Amato *et al.*, 2010] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320, November 2010.

[Andriushchenko *et al.*, 2022] Roman Andriushchenko, Milan Češka, Sebastian Junges, and Joost-Pieter Katoen. Inductive synthesis of finite-state controllers for POMDPs. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 85–95. PMLR, August 2022.

[Bai *et al.*, 2011] Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A Ngo. Monte Carlo value iteration for continuous-state POMDPs. In *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, pages 175–191. Springer, 2011.

[Bai *et al.*, 2013] Haoyu Bai, David Hsu, and Wee Sun Lee. Planning how to learn. In *2013 IEEE International Conference on Robotics and Automation*, pages 2853–2859, Karlsruhe, Germany, May 2013. IEEE.

[Barto *et al.*, 1995] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, January 1995.

[Bonet and Geffner, 2021] Blai Bonet and Hector Geffner. Action Selection for MDPs: Anytime AO* Versus UCT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1749–1755, September 2021.

[Bonet, 2009] Blai Bonet. Deterministic POMDPs Revisited. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 59–66, Montreal, Canada, 2009. AUAI Press.

[Bonet, 2010] Blai Bonet. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174(3-4):245–269, March 2010.

[Brafman and Shani, 2021] Ronen Brafman and Guy Shani. A Multi-Path Compilation Approach to Contingent Planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1868–1874, September 2021.

[Chakrabarti, 1994] P.P. Chakrabarti. Algorithms for searching explicit AND/OR graphs and their applications to problem reduction search. *Artificial Intelligence*, 65(2):329–345, February 1994.

[Chatterjee *et al.*, 2016] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artificial Intelligence*, 234:26–48, 2016.

[Chatterjee *et al.*, 2020] Krishnendu Chatterjee, Martin Chmelík, Deep Karkhanis, Petr Novotný, and Amélie Royer. Multiple-Environment Markov Decision Processes: Efficient Analysis and Applications. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30:48–56, June 2020.

[Chen *et al.*, 2016] Min Chen, Emilio Frazzoli, David Hsu, and Wee Sun Lee. POMDP-lite for robust robot planning under uncertainty. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5427–5433, Stockholm, Sweden, May 2016. IEEE.

[Chung and Huang, 2011] Shu-Yun Chung and Han-Pang Huang. Robot Motion Planning in Dynamic Uncertain Environments. *Advanced Robotics*, 25(6-7):849–870, January 2011.

[Dey *et al.*, 2014] Debadeepta Dey, Andrey Kolobov, Rich Caruana, Ece Kamar, Eric Horvitz, and Ashish Kapoor. Gauss meets Canadian traveler: Shortest-path problems with correlated natural dynamics. In *AAMAS'14 Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 1101–1108, 2014.

[Duff, 2002] Michael O'Gordon Duff. *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*. University of Massachusetts Amherst, 2002.

[Ermis *et al.*, 2021] Melike Ermis, Mingyu Park, and Insoon Yang. On Anderson Acceleration for Partially Observable Markov Decision Processes. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 4478–4485, Austin, TX, USA, December 2021. IEEE.

[Eyerich *et al.*, 2010] Patrick Eyerich, Thomas Keller, and Malte Helmert. High-quality policies for the canadian traveler's problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 51–58, 2010.

[Ferguson *et al.*, 2004] D. Ferguson, A. Stentz, and S. Thrun. PAO for planning with hidden state. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 2840–2847 Vol.3, New Orleans, LA, USA, 2004. IEEE.

[Guo and Barfoot, 2019] Hengwei Guo and Timothy D. Barfoot. The Robust Canadian Traveler Problem Applied to Robot Routing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5523–5529, Montreal, QC, Canada, May 2019. IEEE.

[Hansen and Zilberstein, 2001] Eric A Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

[Horák *et al.*, 2018] Karel Horák, Branislav Bosanský, and Krishnendu Chatterjee. Goal-HSVI: Heuristic Search Value Iteration for Goal POMDPs. In *IJCAI*, pages 4764–4770, 2018.

[Huang *et al.*, 2023] Yizhou Huang, Hamza Dugmag, Timothy D. Barfoot, and Florian Shkurti. Stochastic Planning for ASV Navigation Using Satellite Images. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1055–1061, London, United Kingdom, May 2023. IEEE.

[Junges *et al.*, 2018] Sebastian Junges, Nils Jansen, Ralf Wimmer, Tim Quatmann, Leonore Winterer, Joost-Pieter Katoen, and Bernd Becker. Finite-state Controllers of POMDPs via Parameter Synthesis. *Conference on Uncertainty in Artificial Intelligence*, 2018.

[Kurniawati *et al.*, 2009] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. 2009.

[Lacerda *et al.*, 2019] Bruno Lacerda, Fatma Faruq, David Parker, and Nick Hawes. Probabilistic planning with formal performance guarantees for mobile service robots. *The International Journal of Robotics Research*, 38(9):1098–1123, 2019.

[Littman *et al.*, 1995] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995*, pages 362–370. Elsevier, 1995.

[Littman, 1996] Michael Lederman Littman. *Algorithms for Sequential Decision-Making*. Brown University, 1996.

[Mausam and Kolobov, 2012] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham, 2012.

[Muise *et al.*, 2014] Christian Muise, Vaishak Belle, and Sheila McIlraith. Computing Contingent Plans via Fully Observable Non-Deterministic Planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), June 2014.

[Nardi and Stachniss, 2020] Lorenzo Nardi and Cyrill Stachniss. Long-Term Robot Navigation in Indoor Environments Estimating Patterns in Traversability Changes. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 300–306, Paris, France, May 2020. IEEE.

[Papadimitriou and Yannakakis, 1989] Christos H Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. In *Automata, Languages and Programming: 16th International Colloquium Stresa, Italy, July 11–15, 1989 Proceedings 16*, pages 610–620. Springer, 1989.

[Raskin and Sankur, 2014] Jean-François Raskin and Ocan Sankur. Multiple-Environment Markov Decision Processes, December 2014.

[Russell and Norvig, 2021] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education, 2021.

[Shani *et al.*, 2013] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, July 2013.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. *Advances in neural information processing systems*, 23, 2010.

[Smith and Simmons, 2005] Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 542–549, 2005.

[Smith, 2007] Trey Smith. *Probabilistic Planning for Robotic Exploration*. Carnegie Mellon University, 2007.

[Tsang *et al.*, 2022] Florence Tsang, Tristan Walker, Ryan A. MacDonald, Armin Sadeghi, and Stephen L. Smith. LAMP: Learning a Motion Policy to Repeatedly Navigate in an Uncertain Environment. *IEEE Transactions on Robotics*, 38(3):1638–1652, June 2022.

[Watkins, 1989] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. *PhD thesis, Cambridge University*, 1989.

[Wray and Zilberstein, 2019] Kyle Hollins Wray and Shlomo Zilberstein. Generalized Controllers in POMDP Decision-Making. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7166–7172, Montreal, QC, Canada, May 2019. IEEE.