Topological Order Based Planner for Solving POMDPs

Jilles Dibangoye, Guy Shani, Braim Chaid-Draa, Abdel-Illah Mouaddib

Topological Order Planner for POMDPs

Jilles S. Dibangoye
Laval University
gdibango@ift.ulaval.ca

Guy Shani
Microsoft Research
guyshani@microsoft.com

Brahim Chaib-draa
Laval University
chaib@ift.ulaval.ca

Abdel-Illah Mouaddib
University of Caen
mouaddib@info.unicaen.fr

Abstract

Over the past few years, point-based POMDP solvers scaled up to produce approximate solutions to mid-sized domains. However, to solve real world problems, solvers must exploit the structure of the domain. In this paper we focus on the topological structure of the problem, where the state space contains layers of states. We present here the Topological Order Planner (TOP) that utilizes the topological structure of the domain to compute belief space trajectories. TOP rapidly produces trajectories focused on the solveable regions of the belief space, thus reducing the number of redundant backups considerably. We demonstrate TOP to produce good quality policies faster than any other point-based algorithm on domains with sufficient structure.

1 Introduction

POMDPs provide a powerful framework for planning in domains involving hidden states and uncertain action effects. While exact solvers can only handle small domains [Cassandra *et al.*, 1997], approximate methods have shown the ability to handle larger problems. A well known approach for computing approximate solutions is the point-based method [Pineau *et al.*, 2003a]. Point-based algorithms compute a policy using point-based backups (updates) over a finite subset of reachable belief states, hoping that the computed policy will generalize well to other, unobserved beliefs.

In many cases, real world environments contain much structure. Indeed, several forms of structure have been investigated in the past [Shani *et al.*, 2008; Pineau *et al.*, 2003b]. Algorithms that take advantage of such structure can solve structured problems much faster than generic algorithms.

While generally it is possible for an agent to move back and forth between world states, in many cases effects of actions cannot be undone. In a manufacturing plant, for example, robots may glue together two components. After the components are glued, it is impossible to "unglue" them. Thus, transitions into the state where the components are glued cannot be traversed backwards. In such cases we can divide the problem state space into *layers* — groups of states; the agent can move between each two states of a layer; when the agent leaves a layer, it cannot return to it. We call this a *topological structure* [Dai and Goldsmith, 2007; Bonet and Geffner, 2003; Abbad and Boustique, 2003] and say that a problem has much topological structure when the problem state space has many layers. These characteristics are embodied in many real-world applications including assembly line optimization; network routing; or railway traffic control. Consider the assembly of a car that consists in multiple steps: first the car moves to the engine installation; then the engine installation crew checks for malfunctions; thereafter finishing the engine installation the car moves respectively to the hood and the wheel stations. Each transition from a station to another is preceded by a quality measurement procedure that prevents car malfunctions. Whenever malfunctions perceived by noisy sensors occurred the car is sent back to the former station. However, once a car has been properly assembled it is never disassembled. In this paper, we aim at tackling real-world applications that yield such a significant topological structure.

Given such topological structure a point-based algorithm can focus its attention on subsets of the belief space where the value function can be improved, reducing the number of redundant backups. Indeed, generic point-based algorithms execute in many cases a very large number of backups that result in no improvements to the value function [Pineau *et al.*, 2003a; Smith and Simmons, 2005; Shani *et al.*, 2007]. As the point-based backups lie at the core of a point-based algorithm, reducing the number of executed backups can speed up an algorithm considerably.

In this paper we present the Topological Order Planner (TOP) that uses the topological order of the underlying MDP to find good belief space trajectories. TOP groups together states into layers, creating an acyclic layer graph. Layers are solved in reversed topological order, starting with layers that contain goal states. Belief space trajectories are directed towards the solveable layers of the model. Once a layer has been solved, trajectories that reach that layer can be terminated, resulting in shorter trajectories and thus — less point-based backups. Before choosing TOP we should first check whether the domain contains multiple layers. When the domain does not contain such structure, choosing other point-based algorithms is reasonable. However, when the domain contains a significant topological structure, TOP will outperform other point-based algorithms.

We describe the family of topological solvers and explain when such a solver is expected to compute an optimal policy. We then suggest a specific instance of this family, explaining the difficulties in implementing the various parts of the algorithm in a POMDP context. We provide an extensive experimental study of our algorithms in environments with different levels of topological structure, showing how our algorithm gains more power as the number of layers grows.

2 Background and Related Work

We begin with an overview of Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs). We then discuss the point-based approach to solving POMDPs, focusing on trial-based algorithms in this family.

An MDP is a tuple (S, A, T, R, γ) where: S is a discrete and finite state space; A is a discrete and finite action space; $T(s'|s, a)$ is a function of transition probabilities, *i.e.,* the probability of transiting from state s to state s' when taking action a; $R(s, a)$ is a real-valued reward function, that defines the outcomes received when taking action a in state s; γ is a discount factor.

A POMDP is a tuple $(S, A, T, R, \Omega, O, \gamma, b_0)$ where: S, A, T, R are the same as in an MDP; Ω is a discrete and finite set of observations; $O(o|s', a)$ is the probability of observing o after executing a and reaching state s; A *belief state*, describes the probability distribution over states, providing a sufficient statistic for a given history. b_0 defines the initial belief state, *i.e.,* the agent belief over its initial state. The next belief state, denoted $b' = \tau(b, a, o)$, that incorporates the latest action-observation pair (a, o) and the current belief state b, is updated as follows:

$$b'(s') = \frac{O(o|s', a) \sum_s b(s) T(s'|s, a)}{pr(o|b, a)} \quad (1)$$

$$pr(o|b, a) = \sum_s b(s) \sum_{s'} T(s'|s, a) O(o|s', a) \quad (2)$$

Given a POMDP, the goal is to find a *policy*, mapping histories or belief states to actions, that maximizes some aspect of the reward stream, such as the average discounted reward (ADR). A popular approach to computing such policies is by computing a *value function* that assigns values to belief states. A value function V can be represented as a set of α vectors, *i.e.,* $V = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$, such that: $V(b) = \max_i \ \alpha_i \cdot b$.

2.1 Point-Based Value Iteration

It is possible to compute an optimal value function for the entire belief space through iterative complete backups [Cassandra *et al.*, 1997]. However, the number of α-vectors generated by complete backups is exponential in the horizon of the problem — the number of steps required to achieve a reward. As such, exact computation of the value function does not scale beyond small toy problems.

However, V can be iteratively improved using backup operations over specific belief points, known as point-based backups [Pineau *et al.*, 2003a; Spaan and Vlassis, 2005; Smith and Simmons, 2005]:

$$g^{\alpha}_{a,o}(s) = \sum_{s'} O(o|s', a) T(s'|s, a) \alpha(s') \quad (3)$$

$$g^{b}_{a} = r_a + \gamma \sum_o \arg\max_{g^{\alpha}_{a,o} :\, \alpha \in V} (b \cdot g^{\alpha}_{a,o}) \quad (4)$$

$$backup(b) = \arg\max_{g^{b}_{a} :\, a \in A} (b \cdot g^{b}_{a}) \quad (5)$$

where $r_a(s) = R(s, a)$ is a vector representation of the reward function.

Instead of computing an optimal value function over the entire belief space, we can compute V only over a finite subset B of belief states [Lovejoy, 1991]. As a result, the backup operator is polynomial and the complexity of the value function is bounded by the number $|B|$ of belief states. This method yields good policies if an optimal value function over B may generalizes well other belief states. To collect good belief subsets, point-based algorithms collect only belief states that are reachable from the starting belief state b_0 [Pineau *et al.*, 2003a]. Techniques of this family differ on the method used to collect the reachable belief states and the way they order the point-based backups on the collected beliefs.

One approach to collect belief states is the *trial-base* approach [Bonet and Geffner, 2003], in which an algorithm executes a trajectory in the belief space, usually starting from b_0. Trial-based algorithms iteratively compute such a trajectory, execute backups on the belief states in the trajectory, and then compute a new trajectory. The algorithm therefore needs only remember the beliefs of the current trajectory, but can still compute a value function for many belief points.

Of this family of algorithms, *heuristic search value iteration* (HSVI) has demonstrated an impressive performance on a number of larger domains [Smith and Simmons, 2005]. HSVI creates trajectories based on an upper bound and a lower bound over the value function, denoted $\bar{V}$ and $\underline{V}$ respectively. Each such trajectory starts with the initial belief state b_0. HSVI always executes the best action specified by the upper bound, and then selects the successor belief state, which maximizes the gap between the bounds. Once the trajectory is finished, the belief states are updated in reversed order (Algorithm 1).

HSVI generates good trajectories, especially after the bounds become reasonably tight, but the maintenance of the two bounds incur considerable overhead. Hence, the gain resulting from the reduced number of backups does not fully manifest in the total execution time.

The Forward Search Value Iteration algorithm (FSVI [Shani *et al.*, 2007]) uses the underlying MDP Q-value function as a heuristic to traverse the belief space (Algorithm 2). Through the traversal, it maintains both the underlying MDP state s and the belief state b, selecting actions based on the current state and the MDP Q-value function. FSVI proceeds then by selecting the next state s' and observation o by sampling them from $T(\cdot|s, a)$ and $O(\cdot|a, s')$ respectively. The advantage of FSVI is that the overhead of computing the MDP Q-value function is negligible and so is the action selections.

One major drawback of FSVI is its inability to recognize traversals that will improve its state information. In fact, by

Algorithm 1: The HSVI algorithm.

```
begin
    initialize V̄ and V̲
    while V̄(b) − V̲(b) > ε do
        EXPLORE(b_0, V̄, V̲)
end

EXPLORE(b, V̄, V̲)
begin
    if V̄(b) − V̲(b) > εγ^{−t} then
        a* ← arg max_a Q_V̄(b, a)
        o* ← arg max_o (V̄(τ(b, a, o)) − V̲(τ(b, a, o)))
        EXPLORE(τ(b, a*, o*), V̄, V̲)
        add(V̄, backup(b, V̲))
        V̄ ← HV(b)
end
```

Algorithm 2: The FSVI algorithm.

```
begin
    Initialize V
    repeat
        Sample s_0 from the b_0 distribution
        MDPEXPLORE(b_0, s_0)
    until V has converged
end

MDPEXPLORE(b, s)
begin
    if s is not a goal state then
        a* ← arg max_a Q(s, a)
        Sample s' from T(·|s, a)
        Sample o from O(·|a*, s')
        MDPEXPLORE(τ(b, a*, o), s')
    add(V, backup(b, V))
end
```

making use of the optimal Q-value function of the underlying MDP, FSVI limits its ability to create traversals that will visit states that may provide useful observations. Also, FSVI uses a static policy for generating traversals that ignores the current value function. As a consequence, FSVI results in a sub-optimal value function (over visited beliefs) in certain domains.

Both FSVI and HSVI backup the entire path that was explored. It is possible that many of these backups will result in redudant α vectors, mostly because the value of the successor beliefs did not change. As we will later show, on most benchmarks, the number of these redundant backups far outnumber the number of backups that result in useful α vectors.

3 Topological Structure

In many realistic domains, some actions could not be reversed. That is, once an agent has executed an action that took it from state s to state s', it can no longer go back to state s.

Example 1 *A factory may have an assembly line, where products must go through multiple stations. At each station, one or more components are being added to the product. While there is some partial order required, since some stations must precede other stations, in other cases the order of stations is not important. It is possible, that a product may reach a station with a malfunction, and gets sent back to a former station for repair, but once a station has properly assembled its components, they are never disassembled. The assembly management system can therefore balance the workload by directing products towards free stations. Assuming that components are never disassembled from the product, this domain has a significant topological structure.*

Given the topological structure we identify sets of states that we call *layers*. A layer is a strongly connected component — the agent can move back and forth between any two states that belong to the same layer, but once the agent has left a layer, it can no longer go back to it. The layers of states can be organized as a DAG, where there is an edge from layer S_i to layer S_j if there exists some state $s_i \in S_i$, an action a, and a state $s_j \in S_j$ such that $T(s_i, a, s_j) > 0$. Indeed, several successful attempts were made in the past to use this topological structure of the state space to speed up MDP solvers [Dai and Goldsmith, 2007; Bonet and Geffner, 2003; Abbad and Boustique, 2003].

A topological structure in the MDP state space induces a topological structure in the belief space. For example, let b be a belief with non-zero probabilities for some states in a layer S_i, and all the probabilities of states belonging to layers that are parents of S_i are zero. Let b' be some successor (not necessarily immediate) of b that has zero probabilities for states in S_i. Then, while the agent can move from b to b', it cannot go back from b' to b.

However, as the belief space is infinite, we cannot detect the strongly connected components (layers) of the belief space. There is also no direct mapping from the layers of the belief space into the layer of the state space. Indeed, it is possible for a belief to have non-zero probabilities over states that belong to various layers. Therefore, the belief space has more layers than the MDP state space. In some cases, it is even possible that the number of layers in the belief space will be infinite.

4 Topological Order Planning

Given the layers of the state space (or the belief space), a solver can exploit this information to update states (or beliefs) in a smart order. Cycles in the state space force value and policy iteration algorithms to repeatedly update the value of the same state, until convergence. In our case, cycles exist only within a layer. However, the layer structure is by definition acyclic. As such, we can solve each layer only once.

To define the correct order by which layers should be solved, we introduce the notion of *solved layers* and *solveable layers*. A layer S_i is solved if for each state $s \in S_i$, $V(s)$ is within ϵ of its optimal value. Thus, the value of all states in a solved layer cannot be further improved. A layer S_i is solveable if it does not have any successors, or if all of its successor layers are already solved. A topological order solver executes updates only on states in the solveable layers, because backups on solved layers cannot improve the value function. Backups on other non-solveable layers may not be

final — these layers need to be revisited after their successor layers have been solved.

When the domain has goal or terminal states — absorbing states that the agent cannot leave — these states are forming single state layers. These layers are always solveable.

Theorem 1 *A topological order solver, that updates only states in solveable layers, converges to an ϵ optimal value function if:*

1. *Each layer becomes solveable throughout the algorithm.*
2. *Each solveable layer eventually becomes solved.*

5 Topological Order Planner for POMDPs

In a POMDP, because we cannot compute the topological structure, an implementation of the pure topological solver requires some approximations. A POMDP topological solver requires a number of key components:

- Identifying that a layer is solved.
- Identifying that a layer is solveable.
- Finding beliefs that belong to solveable layers.

Below, we explain the challenges and suggest solutions to all of these components.

Our approach is motivated by the trial-based, point-based, family of POMDP solvers. We execute traversals in belief space towards the solveable layers, and update only beliefs that belong to solveable layers.

5.1 Identifying Belief Space Layers

As we explained above, it is impossible to identify the layers of the belief space in the general case. We will therefore map the layers of the POMDP onto the MDP layers. We do so by running trajectories in both belief space and state space at the same time, as was done by the FSVI algorithm. Throughout the trajectory we generate pairs of states and beliefs. We say that a pair (s, b) belongs to a layer S_i if $s \in S_i$. This approach is only an approximation of the true belief layers. First, it may be that a single belief will be a part of multiple layers, associated with different MDP states. Second, this method maps many different belief layers into the same MDP layer. However, as we will later show, this approximation is very useful in practice, and allows us to estimate whether a belief belongs to a solveable layer.

Formally, a layer S_i is solved, if for every state $s \in S_i$, and every belief b such that $b(s) > 0$, the value that b can gain from state s cannot be improved. That is, if V is the current value function, and V^* is the optimal value function, let $\alpha_b = \max_{\alpha \in V} b \cdot \alpha$ and $\alpha_b^* = \max_{\alpha \in V^*} b \cdot \alpha$, then $\alpha_b(s) = \alpha_b^*(s)$. We introduce $s.\text{SOLVED}$ as a means of checking whether state s and its corresponding layer S_i are solved.

While the above is correct, we cannot check every b that has a non-zero probability on a single state s because the number of such beliefs is infinite. We overcome this by maintaining an estimate on whether beliefs associated with a state can be further improved. For each state s we maintain $p(s)$ which is an estimate of the potential improvement to beliefs associated with s. We initialize $p(s) = \max_a R(s, a)$. Each time a belief associated with s is updated, we set $p(s) = 0$. When a backup for a belief b associated with s' generates a new α vector that improves the value of s' by δ, we iterate over the predecessors of s'. For each predecessor s of s', we set $p(s) = \max\{\delta \cdot T(s'|s, \cdot), p(s)\}$. Thus, a state, and all its associated beliefs, are considered solved, when all its successor states and their associated beliefs have been solved. This describes the UPDATE procedure, see Algorithm 3.

5.2 Traversing the layers

TOP is a trial-based algorithm, that is, TOP executes traversals in belief space, and afterwards updates beliefs that were observed during the traversal in reversed order. As we are interested only in the solveable layers, we need to create reachable trajectories within these layers. To achieve that, we traverse both the MDP state space and the POMDP belief space together, as was suggested by the FSVI algorithm.

Before the trial begins we choose a start state s_0 randomly and a goal state s_g reachable from s_0 for the current trajectory. When there are no solved layers, the goal state can be either a terminal state of the domain, if such a state exists, or any other state of a solveable layer otherwise. After layers have been solved, we choose a goal state from one of the solved layers that is reachable from the selected start state.

Our trials move from the start state to the goal state using the most likely path in MDP state space. We use the Floyd-Warshal algorithm, that computes maximum weight paths in graphs to find the path for us. As we do not select action outcomes stochastically, this method assures that every traversal will end at the goal state.

During the traversal we maintain the observed state-belief pairs. Once the traversal ends, we execute backups in reversed order. However, we limit the value function updates only to state-belief pairs that belong to a solveable layer.

After the update is completed, we check whether the solvable layer we have visited became solved. If all the states in that layer have a potential improvement $p(s) = 0$, we conclude that the layer is solved. Otherwise, the layer remains solvable. The algorithm is terminated when all the layers become solved.

Algorithm 3: Topological Order Planner for POMDPs.

```
TOP(b_0)
begin
    repeat
        choose s_0
        choose s_g
        TOPTRIAL(s_0, s_g, b_0)
    until V has converged
end

TOPTRIAL(s, s_g, b)
begin
    if ¬s.SOLVED then
        s' ← PICKNEXTSTATE(s, s_g)
        a ← PICKACTION(s, s')
        o ← PICKNEXTOBSERVATION(s', a)
        TOPTRIAL(s', s_g, τ(b, a, o))
        If (s, b) belongs to a solveable layer UPDATE(b, V)
end
```

6 Empirical Evaluations

We now evaluate the performance of TOP in comparison with other recent forward search algorithms such as HSVI [Smith and Simmons, 2005], and FSVI. In addition, we have also added some solvers including PVI, a prioritized POMDP solver [Shani *et al.*, 2006], PBVI [Pineau *et al.*, 2003a], and SCVI [Virin *et al.*, 2007]. Experiments have been run on an Intel Core Duo 1.83GHz CPU processor with 1Gb main memory.

6.1 Results

We begin by demonstrating the advantage of TOP as the number of layers in the domain increases. To show that, we modified the well known maze navigation domains Hallway and Hallway2, introducing "one way doors" that work only in the direction of the goal. As we can see in Table 1, while the performance of HSVI and FSVI remains roughly fixed for a fixed target ADR as the number of layers grows, TOP performance improves significantly.

As we explain above, TOP reduces the number of redundant backups. Such backups can originate from a number of cases — backups over beliefs that already achieved their maximal possible value, backups on beliefs whose successors have not yet improved, or backups on beliefs that improve the value, but are too early. In the last case, additional backups will later be needed on the same belief. To estimate the first two types of redundant backups by counting the number of backups that did not provide an improving α vector, which we call *useless backups*. To estimate the last type of redundant backups, we can compare the number of useful backups, by subtracting the useless backups from the total number of backups.

We continue to compare the performance of TOP to other algorithm over well known benchmarks. In non-layered domains, there is no reason to expect TOP to outperform FSVI, as both methods use a similar strategy in selecting their trajectories. We therefore created modified layers of non-layered domains by introducing one-way doors towards the goal. The well known RockSample benchmarks are very structured, indeed when a rock is sampled it cannot be unsampled. In these domains TOP outperforms all other algorithms. We can see that this improvement is mainly because TOP executes much less backups, and almost no useless backups in all problems.

7 Discussion

Other researchers have suggested to leverage different types of structure. For example, [Shani *et al.*, 2008] explain how to solve factored POMDPs using Algebraic Decision Diagrams and [Pineau *et al.*, 2003b] suggested to create hierarchies of POMDPs to solve large problems. Topological structure is orthogonal to such approaches. It is possible that a factored POMDP would also possess a significant topological structure. As such, we believe that our approach is complimentary to most other types of structure and can be integrated into other structured solvers. However, as we work with the state space, some modifications should be made to TOP in order to handle such domains. We intend to investigate such improvements in the future.

Method	ADR	$V(b_0)$	$\|V\|$	Time (sec)	#Backups	Useless #Backups	Scale
Hallway-layered $\#layers: 10$							
HSVI	0.62	1.46	143	27	199	35	×6.75
FSVI	0.62	2.08	337	57	612	143	×14.2
PBVI	0.60	2.18	249	155	3659	1289	×38.7
SCVI	0.62	1.88	693	58	1420	104	×14.5
PVI	0.50	0.36	64	288	120	12	×72
TOP	**0.62**	**1.43**	**25**	**4**	**127**	**0**	**×1**
Hallway-layered $\#layers: 15$							
HSVI	0.62	1.54	57	11	123	10	×11
FSVI	0.62	2.21	266	49	612	139	×49
PBVI	0.62	2.24	127	87	2581	1045	×87
SCVI	0.62	2.0	609	56	1430	109	×56
PVI	0.50	0.36	64	288	120	11	×288
TOP	**0.62**	**0.47**	**12**	**1**	**34**	**0**	**×1**
Hallway2-layered $\#layers: 10$							
HSVI	0.48	0.60	357	200	473	64	×10.5
FSVI	0.48	0.60	450	127	408	293	×6.68
PBVI	0.47	0.64	32	94	795	23	×4.94
SCVI	0.48	0.65	1448	435	2230	148	×22.9
TOP	**0.48**	**0.34**	**63**	**19**	**135**	**6**	**×1**
Hallway2-layered $\#layers: 15$							
HSVI	0.57	2.42	368	244	1114	258	×10.2
FSVI	0.57	1.51	290	71	357	60	×2.95
SCVI	0.57	1.61	482	120	1280	127	×5
TOP	**0.57**	**0.95**	**48**	**24**	**164**	**0**	**×1**
Hallway2-layered $\#layers: 20$							
HSVI	0.57	2.73	296	201	1102	133	×14.4
FSVI	0.57	1.55	218	65	357	38	×4.64
SCVI	0.57	1.58	388	48	1240	109	×3.43
TOP	**0.57**	**0.77**	**38**	**14**	**104**	**8**	**×1**
cit-layered $\#layers: 12$							
HSVI	0.83	0.66	492	2170	873	370	×50
FSVI	0.83	0.68	403	1278	1122	718	×29.7
SCVI	0.83	0.72	535	224	550	5	× 5
TOP	**0.83**	**0.16**	**36**	**43**	**134**	**0**	**×1**
Mit-layered $\#layers: 11$							
HSVI	0.90	0.85	731	1515	1076	214	×39
FSVI	0.90	0.80	164	494	1275	248	×13
SCVI	0.90	0.63	496	101	520	4	×2.65
TOP	**0.90**	**0.53**	**47**	**38**	**196**	**0**	**×1**
Rock Sample (4,4) $\#layers: 17$							
HSVI	18.04	17.92	173	11	239	33	×5
FSVI	18.04	13.63	79	7	98	12	×3
SCVI	16.80	15.93	405	520	16350	797	×260
TOP	**18.04**	**13.60**	**37**	**2**	**63**	**0**	**×1**
Rock Sample (5,7) $\#layers: 129$							
HSVI	24.2	22.96	208	2754	461	124	×5
FSVI	22.2	14.97	353	4057	384	12	×8
SCVI	21.3	20.22	432	535	800	147	×1.09
TOP	**24.3**	**14.51**	**68**	**488**	**95**	**0**	**×1**
Rock Sample (7,8) $\#layers: 257$							
HSVI	20.1	19.69	178	7641	226	31	×6.87
FSVI	19.7	18.43	105	4503	185	16	×4.04
TOP	**20.1**	**17.68**	**73**	**1112**	**72**	**0**	**×1**

Table 1: Performance measurements on layered domains.

The idea of using the topological structure of the state space has been studied repeatedly in the context of MDPs [Dai and Goldsmith, 2007; Bonet and Geffner, 2003]. In an MDP, once the layers (strongly connected components) have been identified, we can execute value iteration only over state in a single solvable layer, until it becomes solved.

As we have discussed above, a direct application of these ideas to POMDPs is challenging. First, the belief space MDP has an infinite state space, making the identification of strongly connected components difficult. Also, beliefs are not continuous — in general we cannot expect that a transition to a close belief (using standard distance metrics) will be more probable than a transition to farther belief state. As such, identifying belief partitions using distance metrics will not honor the topological structure.

Our algorithm also bears some similarity to prioritized solvers. [Wingate and Seppi, 2005] has thoroughly discussed prioritization in the context of MDPs, using the Bellman error. In that context, the priority of the state is the poten-

tial increase in its value function given a backup. [Shani *et al.*, 2006] have extended these ideas into the POMDP space. They also used the Bellman error, but over beliefs rather than states. One problem with this approach is that it is time consuming to update the priorities of beliefs. We note, however, that a high Bellman error does not indicate that a belief belongs to a solveable layer. A belief b, such that $b(s) > 0$ and $R(s, a) > 0$ may initially have a high Bellman errors, even though s may belong to a layer far from the solved layers, and TOP will update b only late in the process.

A second prioritization approach was suggested by [Virin *et al.*, 2007]. Their SCVI algorithm partitions the state space into clusters of states with similar MDP value. They collect a finite set of beliefs and assign cluster weights to beliefs based on the probabilities of states in each cluster. Then, they iterate over clusters by decreasing cluster value and update the beliefs associated with the cluster. It is possible to think of SCVI as if it induces layers in domains with no a-priori topological structure. SCVI is not a trial-based algorithm, but it may be that their ideas could be brought into TOP in order to handle generic domains by introducing artificial layers.

For trial based solvers it is crucial to balance between the identification of good trajectories, and the time that is required to compute them. HSVI uses an upper bound and a lower bound over the value function, to identify belief states where the gap between bounds can be reduced. Traversals are therefore not necessarily directed towards the regions where the lower bound, which controls the policy, can be improved. Also, the maintenance of the upper bound is extremely time consuming and therefore HSVI spends much time on computing the trajectories.

FSVI, on the other hand, is very fast in computing forward trajectories. As it only uses the static value function of the underlying MDP, computing the next state becomes very fast. However, FSVI does not change its traversal strategy as the value function gets updated. Therefore, FSVI might repeatedly revisit areas of the belief space that were already solved.

Both FSVI and HSVI execute many redundant value function updates, as they do not contain a mechanism for knowing whether a backup will be beneficial, or whether additional backups will be needed afterwards. As we have demonstrated above, TOP reduces the number of redundant backups because it considers only the solveable layers.

8 Conclusion

We have introduced a new POMDP solution method — topological order-based planning (TOP), that uses the structural properties of POMDPs to reduce the number of costly value function updates. Our method uses the layers of the MDP state space to identify layers in belief space. TOP updates only beliefs that belong to solveable layers and thus reduces the number of times that a belief is visited. We have demonstrated how TOP outperforms other POMDP solvers in domains that contain significant topological structure.

In this paper we identified the general requirements from a TOP solver, and suggested a possible implementation for POMDPs. In the future, we will investigate the integration of methods from other solvers into TOP, such as priorities, and value-directed clustering. We also intend to apply TOP to factored domains, formalizing layers in terms of state variables instead of states as it is currently assessed in a human-robot interaction scenario of sharing a mission of object transportation.

References

[Abbad and Boustique, 2003] Mohammed Abbad and Hatim Boustique. A decomposition algorithm for limiting average markov decision problems. *Oper. Res. Lett.*, 31(3):473–476, 2003.

[Bonet and Geffner, 2003] Blai Bonet and Hector Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proceedings of IJCAI*, pages 1233–1238, 2003.

[Cassandra *et al.*, 1997] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental Pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of UAI*, pages 54–61, 1997.

[Dai and Goldsmith, 2007] Peng Dai and Judy Goldsmith. Topological value iteration algorithm for Markov Decision Processes. In *Proceedings of IJCAI*, pages 1860–1865, 2007.

[Lovejoy, 1991] W. S. Lovejoy. Computationally feasible bounds for partially observable markov decison processes. *Operations Research*, 39:175–192, 1991.

[Pineau *et al.*, 2003a] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of IJCAI*, pages 1025–1032, 2003.

[Pineau *et al.*, 2003b] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Policy-contingent abstraction for robust robot control. In *Proceedings of UAI*, pages 477–484, 2003.

[Shani *et al.*, 2006] Guy Shani, Ronen I. Brafman, and Solomon Eyal Shimony. Prioritizing point-based pomdp solvers. In *Proceedings of ECML*, pages 389–400, 2006.

[Shani *et al.*, 2007] Guy Shani, Ronen I. Brafman, and Solomon Eyal Shimony. Forward search value iteration for pomdps. In *Proceedings of IJCAI*, pages 2619–2624, 2007.

[Shani *et al.*, 2008] Guy Shani, Pascal Poupart, Ronen I. Brafman, and Solomon Eyal Shimony. Efficient add operations for point-based algorithms. In *Proceedings of ICAPS*, pages 330–337, 2008.

[Smith and Simmons, 2005] Trey Smith and Reid G. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of UAI*, pages 542–547, 2005.

[Spaan and Vlassis, 2005] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[Virin *et al.*, 2007] Yan Virin, Guy Shani, Solomon Eyal Shimony, and Ronen I. Brafman. Scaling up: Solving pomdps through value based clustering. In *Proceedings of AAAI*, pages 1290–1295, 2007.

[Wingate and Seppi, 2005] David Wingate and Kevin D. Seppi. Prioritization methods for accelerating mdp solvers. *J. Mach. Learn. Res.*, 6:851–881, 2005.