Exploiting Fully Observable and Deterministic Structures in Goal POMDPs

Håkan Warnquist[*] and **Jonas Kvarnström** and **Patrick Doherty**
Department of Computer and Information Science
Linköping University, Linköping

Abstract

When parts of the states in a goal POMDP are fully observable and some actions are deterministic it is possible to take advantage of these properties to efficiently generate approximate solutions. Actions that deterministically affect the fully observable component of the world state can be abstracted away and combined into macro actions, permitting a planner to converge more quickly. This processing can be separated from the main search procedure, allowing us to leverage existing POMDP solvers. Theoretical results show how a POMDP can be analyzed to identify the exploitable properties and formal guarantees are provided showing that the use of macro actions preserves solvability. The efficiency of the method is demonstrated with examples when used in combination with existing POMDP solvers.

Introduction

Finding optimal solutions to general POMDPs is computationally difficult except for very small problem instances, which has led to the common use of approximate methods. For example, a variety of *point-based* POMDP algorithms has allowed near-optimal solutions to be found for much larger problems than previously possible (Pineau, Gordon, and Thrun 2003; Spaan and Vlassis 2005; Smith and Simmons 2005; Shani, Brafman, and Shimony 2007; Kurniawati, Hsu, and Lee 2008).

We are interested in finding new techniques for taking advantage of specific properties of POMDPs to efficiently generate approximate solutions. We consider the case where *parts* of the state are fully observable, as in Mixed Observability MDPs, MOMDPs (Ong et al. 2010; Araya-Lòpez et al. 2010), and where *some* actions are deterministic. Furthermore, we operate on problems modeled as *goal POMDPs* (Bonet and Geffner 2009).

While these problems as a whole remain only partially observable, we show that local deterministic structures can be automatically discovered and exploited in order to take shortcuts in the search space. Specifically, deterministic actions affecting only the fully observable component of the world state can be abstracted away and combined into macro actions, permitting a planner to converge more quickly. This processing can be separated from the main search procedure, allowing us to leverage existing POMDP algorithms.

We provide formal completeness guarantees showing that the use of macro actions preserves solvability, i.e. if a problem in the targeted class of POMDPs is solvable without using macro actions, it can still be solved with them. We also demonstrate the efficiency of our method combined with the solvers HSVI2 (Smith and Simmons 2005) and FSVI (Shani, Brafman, and Shimony 2007).

Problem Formulation

We want to make a plan for achieving some goal in an environment modeled by a goal POMDP. Goal POMDPs are undiscounted POMDPs where actions only have costs (non-positive rewards) Bonet and Geffner (2009). A goal POMDP can be described by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, t, o, c, \mathcal{S}_{goal} \rangle$ where:

- $\mathcal{S}$ is the *state space*, a finite set of states.
- $\mathcal{A}$ is the *action space*, a finite set of actions.
- $\mathcal{Z}$ is the *observation space*, a finite set of observations.
- $t : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the *transition function*, where $t(s, a, s')$ is the probability that state s' is reached if action a is performed in state s, and $\sum_{s' \in \mathcal{S}} t(s, a, s') = 1$.
- $o : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{Z} \mapsto [0, 1]$ is the *observation function*, where $o(s, a, s', z)$ is the probability of observing z when s' is reached by performing action a in state s, and $\sum_{z \in \mathcal{Z}} o(s, a, s', z) = 1$,
- $c : \mathcal{S} \times \mathcal{A} \mapsto [0, \infty)$ is the *cost function*, where $c(s, a)$ is the cost of performing action a in state s,
- $\mathcal{S}_{goal} \subseteq \mathcal{S}$ is the *goal state space* consisting of states where the goal is achieved.

The model is restricted such that the goal states are *cost-free* and *absorbing*, i.e. for all $s \in \mathcal{S}_{goal}$, $a \in \mathcal{A}$, $c(a, s) = 0$ and $t(s, a, s) = 1$. In all other states actions have non-zero cost, i.e for all $s \in \mathcal{S} \setminus \mathcal{S}_{goal}$, $a \in \mathcal{A}$, $c(a, s) > 0$.

In a POMDP the true state is not known. Instead our beliefs regarding the current state are represented by a *belief state* b where $b(s)$ is the probability that we are in state s. A *goal belief state* is any belief state where all probability mass is distributed on the goal states, i.e. $\sum_{s \in \mathcal{S}_{goal}} b(s) = 1$.

[*]Scania CV AB, Södertälje, Sweden

When an action a is performed in a belief state b and the observation z is made, a new belief state b' is reached where

$$b'(s') = \frac{\sum_{s \in \mathcal{S}} o(s,a,s',z)t(s,a,s')b(s)}{\sum_{s,s'' \in \mathcal{S}} o(s,a,s'',z)t(s,a,s'')b(s)} \quad (1)$$

for all $s' \in \mathcal{S}$. Let $f : \mathcal{B} \times \mathcal{A} \times \mathcal{Z} \mapsto \mathcal{B}$ such that $f(b,a,z)$ is the new belief state obtained by applying (1) with b, a, z.

A *solution* to a goal POMDP is a *policy*, a function $\pi : \mathcal{B} \mapsto \mathcal{A}$, that guarantees that a goal state is reached. The expected cost of π is given by the function $V_\pi : \mathcal{B} \mapsto [0,\infty)$:

$$V_\pi(b) = \bar{c}(b,a) + \sum_{z \in \mathcal{Z}} \tau(b,a,z)V_\pi(f(b,a,z)) \quad (2)$$

where $a = \pi(b)$, $\bar{c}(b,a) = \sum_{s \in \mathcal{S}} b(s)c(s,a)$ is the expected cost of executing a in b and,

$$\tau(b,a,z) = \sum_{s,s' \in \mathcal{S}} o(s,a,s',z)t(s,a,s')b(s) \quad (3)$$

is the probability of observing z after performing a in b. Because only the goal states are cost-free, all policies with finite expected cost are solutions. Ideally we would find an *optimal policy* π^*, where the expected cost is minimal. For a belief state b, the minimal expected cost is

$$V^*(b) = \min_{a \in \mathcal{A}} \Big(\bar{c}(b,a) + \sum_{z \in \mathcal{Z}} \tau(b,a,z)V^*(f(b,a,z))\Big). \quad (4)$$

Representation. Many algorithms for solving POMDPs, such as the point-based algorithms, exploit the fact that the value function is convex and can be approximated with arbitrary precision by taking the minimum of a set of $|\mathcal{S}|$-dimensional hyperplanes (Sondik 1971). They therefore represent a policy π_Γ and its value function with a set of real-valued $|\mathcal{S}|$-sized vectors Γ called α-vectors. Each α-vector is associated with an action, and for all $b \in \mathcal{B}$, $\pi_\Gamma(b)$ is the action associated to the vector $\alpha^* = \arg\min_{\alpha \in \Gamma} \sum_{s \in \mathcal{S}} b(s)\alpha(s)$. The value function of π_Γ is approximated by $V_\Gamma(b) = \sum_{s \in \mathcal{S}} b(s)\alpha^*(s)$. Initially Γ consists of $|\mathcal{A}|$ α-vectors corresponding to policies where the same action is performed in every state. A non-optimal policy can be incrementally improved by performing a *backup* on selected belief states (Pineau, Gordon, and Thrun 2003). When a belief state b is backed up, a new vector is added to Γ:

$$\Gamma \leftarrow \Gamma \cup \big\{ \arg\min_{\beta_a : a \in \mathcal{A}} \sum_{s \in \mathcal{S}} b(s)\beta_a(s) \big\} \quad (5)$$

where each β_a is an α-vector with the action a associated to it and for all $s \in \mathcal{S}$:

$$\beta_a(s) = c(s,a) + \sum_{z \in \mathcal{Z}} \arg\min_{\beta_{z,\alpha} : \alpha \in \Gamma} \sum_{s' \in \mathcal{S}} b(s')\beta_{z,\alpha}(s')$$

where $\beta_{z,\alpha}(s) = \sum_{s' \in \mathcal{S}} o(s,a,s',z)t(s,a,s')\alpha(s')$. The backup operation is central to all point-based POMDP solvers. The belief states to be backed up are selected by traversing the *reachable belief space* from the initial belief state. The reachable belief space consists of those belief states that can be obtained from the initial belief state by multiple applications of (1) without having zero in the divisor. If the belief states are selected appropriately $V_\Gamma(b_0)$ converges toward $V^*(b_0)$ as the number of backups goes to infinity (Smith and Simmons 2004).

For our purposes we use a factored representation of the POMDP model (Boutilier and Poole 1996), which is both more information-rich and allows for a more compact model representation. The state space is then represented by a set of *state variables* $\mathbf{X} = \{X_1, \ldots, X_n\}$ and a state is an assignment to those variables. Similarly, the observation space is represented by a set of *observation variables* $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$. The transition and observation functions are represented separately for each action by a Bayesian network containing $\mathbf{X}$, $\mathbf{Y}$, and another set of *post-state variables* $\mathbf{X}' = \{X'_1, \ldots, X'_n\}$ that represent the state after an action is performed, the *post-state*. The structure of each network is such that the variables in $\mathbf{X}$ have no parents, the variables in $\mathbf{X}'$ may only have parents in $\mathbf{X}$ and $\mathbf{X}'$, and the variables in $\mathbf{Y}$ may have parents among all variables.

We use Ω_V and $\Omega_\mathbf{V}$ to denote the outcome spaces of a variable V and a set of variables $\mathbf{V}$ respectively. We use $X_i(s)$, $X'_i(s')$, and $Y_i(z)$ to denote the values of X_i, X'_i, and Y_i given the state s, the post-state s' and the observation z respectively. We use $b(X = x)$ to denote the combined probability of all $s \in \mathcal{S}$ where $X(s) = x$. We may condition a probability with a state, a post-state, an observation and/or an action, e.g. $P(V = v | s, s', z, a)$ is the probability that the variable V has the value v in the network for action a given that $\mathbf{X} = \mathbf{X}(s)$, $\mathbf{X}' = \mathbf{X}'(s')$, and $\mathbf{Y} = \mathbf{Y}(z)$. When it is clear from context that $v \in \Omega_V$, we may substitute $V = v$ with v.

Using the factored representation the transition and observation functions can be expressed as follows:

$$t(s,a,s') = \prod_{i=1}^{|\mathbf{X}|} P(X'_i(s') | X'_1(s'), ..., X'_{i-1}(s'), s, a) \quad (6)$$

$$o(s,a,s',z) = \prod_{i=1}^{|\mathbf{Y}|} P(Y_i(z) | Y_1(z), ..., Y_{i-1}(z), s, s', a) \quad (7)$$

where the variables in $\mathbf{X}'$ and $\mathbf{Y}$ are ordered so that parent variables within the same set have lower indexes.

Exploiting Deterministic Structures

We want to reduce the size of the reachable belief space by abstracting away actions that deterministically affect fully observable variables. First we need to identify the deterministic structures that we intend to exploit. These are variables that are fully observable, actions that deterministically affect the fully observable variables, and actions whose effects given the state of the fully observable variables are similar as defined by a special similarity measure. Then we show how several actions can be combined into macro actions such that a goal POMDP can be solved considering only actions that affect partially observable variables.

Definition 1 (Fully Observable Variable). The state variable $X_i \in \mathbf{X}$ is *fully observable* if for every reachable belief state b, $b(x) = 1$ for exactly one value $x \in \Omega_{X_i}$.

Variables that are not fully observable are said to be *partially observable*. Let $\mathbf{X}_{fo} \subseteq \mathbf{X}$ be the set of all fully observable variables and $\mathbf{X}_{po} = \mathbf{X} \setminus \mathbf{X}_{fo}$ be the set of all partially observable variables. We do a similar separation of the post-state variables: $\mathbf{X}' = \mathbf{X}'_{fo} \cup \mathbf{X}'_{po}$. For a belief state b, we use $\mathbf{x}_b$ to denote the values of the fully observable variables in b. This is referred to as the *fully observable partial state* of b. It is possible to identify some fully observable variables directly from the factored model using the following theorem:

Theorem 1 (Fully Observable Variable). The variable X_i belongs to $\mathbf{X}_{fo}$ if $b_0(x) = 1$ for exactly one value $x \in \Omega_{X_i}$ and any of the following sufficient conditions are true:

1. The value of X_i can always be determined from the observation gained by any action: For all actions $a \in \mathcal{A}$ and all values $x, x' \in \Omega_{X'_i}$ where $x \neq x'$, $\mathcal{Z}_{x,a} \cap \mathcal{Z}_{x',a} = \emptyset$ where $\mathcal{Z}_{x,a} = \{z \in \mathcal{Z} : P(z|x,a) > 0\}$ and $\mathcal{Z}_{x',a} = \{z \in \mathcal{Z} : P(z|x',a) > 0\}$.
2. The value of X_i always changes deterministically: For all actions $a \in \mathcal{A}$ and all states $s \in \mathcal{S}$, $P(x|s,a) = 1$ for exactly one value $x \in \Omega_{X'_i}$, and in the networks for all actions $a \in \mathcal{A}$ the ancestors of X'_i are also fully observable variables.

Proof. By induction over the reachable belief states. The condition in Definition 1 holds for b_0. Assume that the condition holds for an arbitrary reachable belief state b. Then we must show for arbitrary $a \in \mathcal{A}$ and $z \in \mathcal{Z}$ such that $\tau(b,a,z) > 0$ that it holds for the next belief state $b' = f(b,a,z)$. Assume the first condition of Theorem 1 is true. If $z \in \mathcal{Z}_{x,a}$ then $b'(s') = 0$ for all $s' \in \mathcal{S}$ where $X_i(s') = x' \neq x$ because $P(z|x',a) = 0$ and thereby $o(s,a,s',z) = 0$ for all $s \in \mathcal{S}$. Assume the second condition of Theorem 1 is true. Then all ancestors $\mathbf{X}_i$ of X'_i are fully observable variables. For every $s_j, s_k \in \mathcal{S}$ where $b(s_j) > 0$ and $b(s_k) > 0$ we have $\mathbf{X}_i(s_j) = \mathbf{X}_i(s_k)$ because $\mathbf{X}_i$ are fully observable variables. Therefore, for every $x \in \Omega_{X'_i}$, $P(x|s_j,a) = P(x|s_k,a)$. Let $x \in \Omega_{X'_i}$ be the value for which $P(x|s_j,a) = 1$, then $b'(s') = 0$ for all $s' \in \mathcal{S}$ where $X_i(s') = x' \neq x$ because $P(x'|s,a) = 0$ for all $s \in \mathcal{S}$ and thereby $t(s',a,s) = 0$. □

The first condition of Theorem 1 can be checked by evaluating $P(z|x,a)$ for all $a \in \mathcal{A}$, $z \in \mathcal{Z}$, $X'_i \in \mathbf{X}'$, $x \in \Omega_{X'}$. The second condition can be checked by iterating through all $a \in \mathcal{A}$, $s \in \mathcal{S}$, $X'_i \in \mathbf{X}'$, $x \in \Omega_{X'}$. Unless $P(x|s,a) = 1$ for exactly one value $x \in \Omega_{X'_i}$, the variable X_i and all its descendants that are not already fully observable variables by the first condition are flagged as partially observable. The remaining variables are then fully observable.

An action with non-zero cost that does not affect the belief state cannot be assigned to a reachable belief state in a policy with finite expected cost, and is considered inapplicable.

Definition 2 (Applicable Action). An action a is said to be *applicable* in a belief state b iff its expected cost $\bar{c}(b,a) = 0$ or there is $z \in \mathcal{Z}$ such that $\tau(b,a,z) > 0$ and $f(b,a,z) \neq b$.

An action precondition is a boolean function of the state variables that must evaluate to true for the action to be applicable. We will define special preconditions, *FO-preconditions*, as a function of only the fully observable variables such that an action's behavior and cost are similar with regard to the values of the fully observable variables in all belief states satisfying the FO-precondition.

Definition 3 (FO-Precondition). A *FO-precondition* of an action a is a boolean function of the fully observable variables $p_a : \Omega_{\mathbf{X}_{fo}} \mapsto \{\bot, \top\}$ such that:

- for all $b \in \mathcal{B}$, if $p_a(\mathbf{x}) = \bot$, then a is not applicable in b,
- for all $s_i, s_j \in \mathcal{S}$ such that $p_a(\mathbf{X}_{fo}(s_i)) = p_a(\mathbf{X}_{fo}(s_j)) = \top$ and $\mathbf{X}_{po}(s_i) = \mathbf{X}_{po}(s_j)$:

$$c(a,s_i) = 0 \text{ iff } c(a,s_j) = 0, \tag{8}$$

and for all $v \in \Omega_V$ where $V \in \mathbf{X}' \cup \mathbf{Y}$:

$$P(v|s_i,a) \begin{cases} = 0 & \text{if } P(v|s_j,a) = 0, \\ \in (0,1) & \text{if } P(v|s_j,a) \in (0,1), \\ = 1 & \text{if } P(v|s_j,a) = 1. \end{cases} \tag{9}$$

The conditions (8) and (9) describe how similarly an action must behave in different states where its FO-preconditions are satisfied. The definition of similar actions in Definition 3 is very generous. However, the theoretical results still hold with a more narrow definition by specifying threshold parameters $p_1, \ldots, p_n$ and substituting (9) with:

$$P(v|s_i,a) \begin{cases} = 0 & \text{if } P(v|s_j,a) = 0, \\ \in (0,p_1) & \text{if } P(v|s_j,a) \in (0,p_1), \\ \in [p_{k-1},p_k) & \text{if } P(v|s_j,a) \in [p_{k-1},p_k) \\ & \text{for } k = 2,\ldots,n, \\ \in [p_n,1) & \text{if } P(v|s_j,a) \in [p_n,1), \\ = 1 & \text{if } P(v|s_j,a) = 1. \end{cases} \tag{10}$$

In general, not every action in $\mathcal{A}$ has a FO-precondition. However, an action can be partitioned into multiple actions that are applicable for non-overlapping subsets of the state space where the original action is applicable and that do have FO-preconditions. Let $\mathcal{A}'$ be some set of actions with FO-preconditions created from $\mathcal{A}$ this way. Given a belief state there is a one-to-one correspondence between applicable actions in $\mathcal{A}$ and $\mathcal{A}'$. Thus for every policy of applicable actions $\pi : \mathcal{B} \mapsto \mathcal{A}$ there is an equivalent policy of applicable actions $\pi' : \mathcal{B} \mapsto \mathcal{A}'$ such that $V_\pi = V_{\pi'}$ and vice versa.

Actions that deterministically manipulate the fully observable variables we call *support actions* because they can be used to form a deterministic plan to satisfy the FO-preconditions of another action.

Definition 4 (Support Action). An action $a \in \mathcal{A}'$ is a *support action* if for every fully observable variable $X'_i \in \mathbf{X}'_{fo}$ there is a value $x \in \Omega_{X'_i}$ such that for all states $s \in \mathcal{S}$ where $b(s) > 0$, $P(X'_i = x|s,a) = 1$.

We say that two belief states b_i and b_j are *similar* if they have non-zero probability in the same partially observable states, i.e. for all $s_k, s_l \in \mathcal{S}$ such that $\mathbf{X}_{po}(s_k) = \mathbf{X}_{po}(s_l)$, $b_i(s_k) > 0$ iff $b_j(s_l) > 0$. In a belief state b, an action a will be considered *relevant* for a policy if, when applied in some belief state similar to b where it is applicable, it can gain information about the state or affect the state in some way a support action cannot. When making a backup (5) in b we can ignore all support actions and consider instead the actions that are relevant in b. However these actions need not be applicable in b. The solution is to create *macro actions* where a non-applicable relevant action is made applicable by placing it after a sequence of support actions that satisfy its FO-preconditions. The solver can then focus on relevant actions in a more shallow search space.

Definition 5 (Relevant Action). An action $a \in \mathcal{A}'$ is *relevant* for a belief state b if there exists a belief state b_a similar to b where a is applicable and at least one of the following conditions is true:

1. The action may result in an observation that changes our belief in s_i: There is an observation variable $Y \in \mathbf{Y}$, a value $y \in \Omega_Y$, and states $s_i, s_j \in \mathcal{S}$ such that $b_a(s_i) > 0$, $b_a(s_j) > 0$ and $P(y|s_i, a) \neq P(y|s_j, a)$.
2. The action may change the value of a partially observable state variable: There is a partially observable state variable $X_i \in \mathbf{X}_{po}$ and a state $s \in \mathcal{S}$ such that $b_a(s) > 0$ and $P(X'_i = X_i(s)|s, a) < 1$.
3. The action may change the value of a fully observable state variable but fails to qualify as a support action: There is a fully observable state variable $X_i \in \mathbf{X}_{fo}$ and a state $s \in \mathcal{S}$ such that $b_a(s) > 0$ and $P(X'_i = X_i(s)|s, a) < 1$ and a is not a support action in b_a.
4. the action is a zero-cost action that lets us remain among the goal states: The belief state b_a is a goal belief state, the cost $\bar{c}(a, b_a) = 0$, and for all $z \in \mathcal{Z}$, the next belief state $f(b_a, a, z)$ is also a goal belief state.

Let $\mathcal{A}_{rel}(b) \subseteq \mathcal{A}'$ be the set of all actions that are relevant for the belief state b. An action $a \in \mathcal{A}'$ can be tested for membership in $\mathcal{A}_{rel}(b)$ by evaluating the conditions of Definition 5 in an arbitrary belief state b_a that is selected so that a has its FO-preconditions satisfied in b_a and b_a has the same probability distribution over the partially observable variables as b. A support action may be relevant in certain belief states. However, we do not want to consider those support actions that are relevant for the current belief state b, because these can be found in $\mathcal{A}_{rel}(b)$ if they are needed for a solution. Let $\mathcal{A}_{supp}(b) \subseteq \mathcal{A}' \setminus \mathcal{A}_{rel}(b)$ be the set of all support actions that are not relevant in b. Since every action in $\mathcal{A}_{supp}(b)$ is irrelevant in b, the next belief state after executing such an action will have the same probability distribution over the partially observable variables, and thereby also have the same set of relevant actions. The relevant actions will be the same for all belief states that are similar.

Lemma 1. Let b_i, b_j be two similar belief states. Then $\mathcal{A}_{rel}(b_i) = \mathcal{A}_{rel}(b_j)$.

Proof. For all actions with valid FO-preconditions, each condition in Definition 5 evaluates to the same when b_i and b_j are similar. □

If we apply an action to a pair of similar belief states the next belief states will be similar to each other.

Lemma 2. Let b_i, b_j be two similar belief states where the action $a \in \mathcal{A}'$ has its FO-precondition satisfied. Then for arbitrary $z \in \mathcal{Z}$ $f(b_i, a, z)$, $f(b_j, a, z)$ are similar belief states, $\bar{c}(a, b_i) = 0$ iff $\bar{c}(a, b_j) = 0$, and the observation probabilities:

$$\tau(b_i, a, z) \begin{cases} = 0 & \text{if } \tau(b_j, a, z) = 0, \\ \in (0, 1) & \text{if } \tau(b_j, a, z) \in (0, 1), \\ = 1 & \text{if } \tau(b_j, a, z) = 1. \end{cases}$$

Proof. The result follows from Definition 3, (3), and (1). □

Definition 6 (Macro Action). Given a belief state b and an action $a \in \mathcal{A}_{rel}(b)$, a *macro action* of a in b, $\bar{a}_{b,a}$, is a sequence of actions $(a_1, \ldots, a_n, a)$ where $a_1, \ldots, a_n \in \mathcal{A}_{supp}(b)$ such that if the actions $a_1, \ldots, a_n$ are performed in order starting from belief state b, a belief state is reached where the FO-precondition of action a is satisfied. The expected cost of the macro action is

$$\bar{c}(\bar{a}_{b,a}, b) = \bar{c}(a, b_{n+1}) + \sum_{i=1}^{n} \bar{c}(a_i, b_i)$$

where $b_1, \ldots, b_{n+1}$ are belief states such that $b_1 = b$ and $b_{i+1}(s') = \sum_{s \in \mathcal{S}} t(s, a_i, s') b_i(s)$ for all states $s' \in \mathcal{S}$. An action that is not a macro action is a primitive action.

To efficiently generate macro actions, we use two directed multigraphs $\mathcal{G}_{fo}$ and $\mathcal{G}_{supp,b}$ to represent how the fully observable partial states are connected by actions in $\mathcal{A}'$ and $\mathcal{A}_{supp}(b)$ respectively. In $\mathcal{G}_{fo} = (\Omega_{\mathbf{X}_{fo}}, \mathcal{E}_{fo})$ the nodes are labeled with fully observable partial states and each edge is associated with an action and an observation. There is an edge from node $\mathbf{x}$ to node $\mathbf{x}'$ associated with the action $a \in \mathcal{A}'$ and the observation $z \in \mathcal{Z}$ if $p_a(\mathbf{x}) = \top$ and a changes $\mathbf{x}$ to $\mathbf{x}'$ when z is observed, i.e. $P(\mathbf{X}'_{fo} = \mathbf{x}'|\mathbf{X}_{fo} = \mathbf{x}, z, a) = 1$. The graph $\mathcal{G}_{supp,b} = (\Omega_{\mathbf{X}_{fo}}, \mathcal{E}_{supp,b})$ has the same set of nodes as $\mathcal{G}_{fo}$, but the edges are only those of $\mathcal{G}_{fo}$ which are associated with a support action in $\mathcal{A}_{supp}(b)$ which can easily be filtered out. Since none of the actions in $\mathcal{A}_{supp}(b)$ are relevant, every edge from a node x associated with an $a \in \mathcal{A}_{supp}(b)$ will be directed to the same node.

When exploring the reachable belief state space or performing a backup of the α-vectors we want to know which actions are available in a given belief state b. Then we generate a macro action for every relevant action $a \in \mathcal{A}_{rel}(b)$ where there exists a directed path in $\mathcal{G}_{supp,b}$ from the unique node $\mathbf{x}_b$ corresponding to the fully observable part of b to any other node $\mathbf{x}$ such that $p_a(\mathbf{x}) = \top$. The actions on this path, if any, will be inserted before a in the macro action that is generated. Several such paths may exist, and which one is chosen will depend on the selection strategy used (e.g. we use the A* algorithm (Hart, Nilsson, and Raphael 1968) for path selection). Let $\mathcal{A}_T(b)$ be the set of macro actions that are generated in the belief state b with strategy T.

In order to safely use macro actions, we need to know that if a goal-satisfying solution can be found without macro actions, i.e. $V^*(b_0) < \infty$, then we can also find a goal-satisfying solution with macro actions. This is guaranteed if we can replace each relevant action and its leading irrelevant support actions in an optimal policy with a macro action. To do this we must be able to find a path to satisfy the FO-precondition of the next relevant action of the optimal policy, regardless where in $\mathcal{G}_{fo}$ we may end up after performing the previous macro actions corresponding to the previous relevant actions of the optimal policy. To show this we will need the following constructs. Let $\pi : \mathcal{B} \mapsto \mathcal{A}'$ be a policy of FO-preconditioned primitive actions corresponding to an optimal policy π^*. Let $\mathbf{z} = (z_0, z_1, \ldots,)$ be an arbitrary sequence of observations with infinite length such that the subsequence of the first $n+1$ elements, $\mathbf{z}_n$, are observations that could potentially result from applying π from the initial belief state $n+1$ times. Let $\mathbf{b} = (b_0, b_1, \ldots)$ and $\mathbf{b}_n = (b_0, \ldots, b_n)$ be the corresponding sequence of belief states, where $b_{i+1} = f(\pi(b_i), z_i, b_i)$. Now we are interested in those actions in the sequence $\mathbf{a}_n = (a_0, \ldots, a_n) = (\pi(b_0), \ldots, \pi(b_n))$ that are relevant when executed. Let $\mathbf{r}_n = (r_1, r_2, \ldots)$ be a sequence of indexes such that $r_i = j$ iff a_j is the ith action in $\mathbf{a}_n$ such that $a_j \in \mathcal{A}_{rel}(b_j)$. Let $\mathcal{N}_1, \ldots, \mathcal{N}_{|\mathbf{r}_n|}$ be sets of fully observable partial states where there exists a traversable path in $\mathcal{G}_{fo}$ passing through $\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathbf{r}_n|}$ in that order for every possible selection of $\mathbf{x}_i \in \mathcal{N}_i$. These paths can be seen as every possible path that we can take in $\mathcal{G}_{fo}$ when we replace the relevant actions of π with macro actions. Formally, let $\mathcal{N}_1 = \{\mathbf{x} \in \Omega_{\mathbf{X}_{fo}} : p_{a_{r_1}}(\mathbf{x}) = \top\}$, and for $i > 1$ let $\mathcal{N}_i$ be the largest set such that for all $\mathbf{x}_{i-1} \in \mathcal{N}_{i-1}$, $\mathbf{x}_i \in \mathcal{N}_i$: $p_{a_{r_i}}(\mathbf{x}_i) = \top$, and there is a traversable path in $\mathcal{G}_{fo}$ from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ where the first edge is associated with $a_{r_{i-1}}$ and $z_{r_{i-1}}$.

Theorem 2 (Completeness). If $V^*(b_0) < \infty$ and

$$\mathcal{N}_{|\mathbf{r}_n|} \neq \emptyset \text{ for all } n > 0, \tag{11}$$

then there is a policy consisting of macro actions, $\pi' : \mathcal{B} \mapsto \bigcup_{b \in \mathcal{B}} \mathcal{A}_T(b)$ such that $V_{\pi'}(b_0) < \infty$.

Proof. We know that $V^*(b_0) < \infty$, so the policy π of FO-preconditioned primitive actions corresponding to an optimal policy also has finite cost: $V_\pi(b_0) < \infty$. Then there exists $g < \infty$ such that for $i \geq g$, b_i in $\mathbf{b}$ is a goal belief state. We now show that we can generate a policy of macro actions π' from π such that $V_{\pi'}(b_0) < \infty$. Let $\bar{a}_1, \ldots, \bar{a}_{|\mathbf{r}_n|}$ be actions such that each $\bar{a}_i$ is a macro action of a_{r_i} in a belief state $\bar{b}_i$ that is similar to b_{r_i}. Then we can show that there exists $\pi' : \mathcal{B} \mapsto \bigcup_{b \in \mathcal{B}} \mathcal{A}_T(b)$ such that $V_{\pi'}(b_0) < \infty$ by showing that every composite action used in π', $\bar{a}_1, \ldots, \bar{a}_{|\mathbf{r}_n|}$, can be generated using an arbitrary path selection strategy T and that every $\bar{b}_i$ is a goal belief state iff b_{r_i} is a goal belief state. We do this by induction.

The belief state $\bar{b}_1$ is the initial state b_0 and it must be similar to b_{r_1} which results from the actions $a_0, \ldots, a_{r_1 - 1}$ that are in $\mathcal{A}_{supp}(b_0)$ and preserve similarity. By Lemma 1 we get that $\mathcal{A}_{rel}(\bar{b}_1) = \mathcal{A}_{rel}(b_{r_1})$. Now assume that b_{r_i} and $\bar{b}_i$ are similar. Then $\mathcal{A}_{rel}(\bar{b}_i) = \mathcal{A}_{rel}(b_{r_i})$ and by Lemma 2 we get that b_{r_i+1} and $\bar{b}_{i+1}$ are similar because the action a_{r_i} is the only relevant primitive action in the macro action $\bar{a}_i$. The actions $a_{r_i+1}, \ldots, a_{r_{i+1}-1}$ are in $\mathcal{A}_{supp}(b_{r_i+1})$ and therefore b_{r_i+1} is similar to $b_{r_{i+1}}$. Because $\mathcal{N}_i \neq \emptyset$ the macro action $\bar{a}_i$ can be generated. We have now showed that for all $i = 1, \ldots, n$ b_i is similar to $\bar{b}_i$.

If b_{r_i} is a goal belief state, then a_{r_i} is a relevant action by Condition 4 in Definition 5 which means that it has zero cost and b_{r_i+1} is also a goal belief state. The belief states b_{r_i} and b_{r_i+1} will be similar to $\bar{b}_i$ and $\bar{b}_{i+1}$ respectively. The last action in $\bar{a}_i$ is a_{r_i} so $\bar{b}_{i+1}$ is a goal belief state. Thus there exists $g < \infty$ such that for $i \geq g$, $\bar{b}_i$ is a goal belief state. □

To verify whether (11) is true, we can analyze the strongly connected components of $\mathcal{G}_{fo}$. A strongly connected component of a directed graph $\mathcal{G}$ is a largest subgraph $\mathcal{G}'$ such that from every node in $\mathcal{G}'$ there is a directed path to every other node in $\mathcal{G}'$. The strongly connected components can be extracted in time linear in the graph size (Tarjan 1971). In the trivial case that we have only one strongly connected component, (11) is true because for every node in $\mathcal{G}_{fo}$ we can always find a path to every other node where the FO-precondition of the next action is true. When there are many strongly connected components we may risk dead ends by reaching a belief state $\bar{b}_i$ similar to b_{r_i} for which no reachable belief state b exists such that b is similar to $b_{r_{i+1}}$ and the FO-precondition of the next relevant action $a_{r_{i+1}}$ is satisfied. This problem cannot occur if no action has its FO-precondition satisfied in more than one strongly connected component and transitions between strongly connected components are the same regardless which fully observable partial state satisfies the FO-precondition of the action.

Corollary 1. Let $\mathcal{C}_1, \ldots, \mathcal{C}_{scc}$ be the strongly connected components of $\mathcal{G}_{fo}$. Let $\mathcal{A}'_1, \ldots, \mathcal{A}'_{scc} \subseteq \mathcal{A}'$ such that $a \in \mathcal{A}'_i$ if there exists $\mathbf{x} \in \mathcal{C}$ such that $p_a(\mathbf{x}) = \top$. Then the condition (11) of Theorem 2 is true if all of the following is true:

- The sets $\mathcal{A}'_1, \ldots, \mathcal{A}'_{scc}$ are disjoint.
- For all $a \in \mathcal{A}'_i$, $\mathbf{x}_j, \mathbf{x}_k \in \mathcal{C}$, and $z \in \mathcal{Z}$, there exists an edge in $\mathcal{G}_{fo}$ from $\mathbf{x}_j$ to $\mathbf{x}'_j$ associated with a and z iff there also exists an edge in $\mathcal{G}_{fo}$ associated with a and z directed from $\mathbf{x}_k$ to $\mathbf{x}'_k$, and both $\mathbf{x}'_j$ and $\mathbf{x}'_k$ belong to the same strongly connected component.

Proof. By induction. In the base case $\mathcal{N}_1 \neq \emptyset$ because it consists of all $\mathbf{x} \in \Omega_{\mathbf{X}_{fo}}$ where $p_{a_{r_1}}(\mathbf{x}) = \top$ and all nodes in $\mathcal{N}_1$ belong to the same strongly connected component because $\mathcal{A}'_1, \ldots, \mathcal{A}'_{scc}$ are disjoint. Assume that $\mathcal{N}_i \neq \emptyset$ and that $\mathcal{N}_i$ is a subset of the strongly connected component $\mathcal{C}_j$. Let $\mathcal{C}_k$ be the strongly connected component that the fully observable partial state of b_{r_i} belongs to, and let $\mathcal{C}_l$ be the strongly connected component that is reached by a_{r_i} and z_{r_i}. Since $\mathcal{N}_i \subseteq \mathcal{C}_j$ there exists a path in $\mathcal{G}_{fo}$ from every node in $\mathcal{N}_i$ to every node in $\mathcal{C}_k$, thus the macro action $\bar{a}_i$ can be generated and, with the observation z_{r_i} it will reach a node in $\mathcal{C}_l$. Thereby $\mathcal{N}_{i+1} \neq \emptyset$ and $\mathcal{N}_{i+1} \subseteq \mathcal{C}_l$. □

After identifying the strongly connected components, we can check if a POMDP fulfills the conditions of Corollary 1 in a single iteration over all actions. A model that does not satisfy the conditions of Corollary 1 can be altered by adding to the model a fully observable variable X_{scc} indicating the index of the strongly connected component that the system is in and an observation variable Y_{scc} that detects all transitions such that $P(Y_{scc} = i | X'_{scc} = i, a) = 1$. If we split actions on distinct values of X_{scc} the first part of Corollary 1 is satisfied. The additional observation variable causes all actions that do not satisfy the second condition of Corollary 1 to have illegal FO-preconditions.

Experimental Results

We have implemented algorithms for identifying the fully observable variables satisfying Theorem 1, creating preconditioned actions following Definition 3, and generating macro actions on top of *POMDPSolver*, a Java implementation of several point-based algorithms made available by Shani (2012). The selection strategy for generating macro actions is A* (Hart, Nilsson, and Raphael 1968) and the threshold values for splitting similar actions as in (10) are 0.2, 0.4, 0.6, and 0.8.

We will demonstrate the method on instances of two toy examples and a third real-world example where a mechanic troubleshoots and repairs a hydraulic braking system on a truck, described in (Pernestål, Warnquist, and Nyberg 2009; 2012). This last example is chosen to show industrially relevant problems where the method is useful.

Avalanche. The first example is a new domain where an unmanned aerial vehicle (UAV) is searching for avalanche victims in an open terrain represented by a two-dimensional grid. Prior to the search, a radar scan capable of detecting debris under the snow has narrowed down the possible locations where victims may be buried to a subset of the cells. The UAV is equipped with a long range sensor that can detect the signal from avalanche transceivers worn by the victims. This sensor is noisy and has a larger detection probability if the UAV is oriented in the direction of the victim ± 45 degrees. At a suspected location it may use another short range sensor capable of verifying the existence of the victim. The goal is achieved when the locations of all victims are verified. The starting position and orientation of the UAV is known. The UAV moves deterministically by either moving forward or turning 45 degrees in any direction in the horizontal plane. An instance *Av-n-v-l* of the problem consists of an *n* by *n* grid with *v* victims distributed uniformly among *l* locations.

GoalRockSample. In the RockSample benchmark problem a robot is navigating a grid in search of rock samples to collect that can be either good or bad (Smith and Simmons 2004). The robot has a sensor that can sense whether a specific rock is good or bad. The sensor noise depends on the distance to the rock. Because the original problem is stated for discounted POMDPs we will use an adaption of this problem for goal POMDPs instead. The adaption is done by increasing all action costs by one and removing the positive reward for collecting good samples and replacing it with a negative reward of equal size that is gained if the good sample is uncollected when the robot leaves the map. Leaving the map causes the robot to reach a goal state.

Results. We used the algorithms HSVI2 (Smith and Simmons 2005) and FSVI (Shani, Brafman, and Shimony 2007) with and without macro actions on two instances of the avalanche rescue problem, *Av-6-2-4* and *Av-9-2-4*, and two instances of the GoalRockSample problem, *GRS-5-5*, a 5 by 5 grid with 5 rocks, and *GRS-7-8*, a 7 by 7 grid with 8 rocks. A summary of the instances is shown in Table 1 where we have listed the size of the state space $|\mathcal{S}|$, the number of fully observable partial states $|\Omega_{\mathbf{X}_{fo}}|$, the number of actions $|\mathcal{A}|$, the number of support actions $|\mathcal{A}_{supp}| = |\bigcap_{b\in\mathcal{B}} \mathcal{A}_{supp}(b)|$,

Instance	$\lvert\mathcal{S}\rvert$	$\lvert\Omega_{\mathbf{X}_{fo}}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{A}_{supp}\rvert$	$\lvert\mathcal{A}_{rel}\rvert$
Av-6-2-4	3168	288	10	128	19
Av-9-2-4	7128	648	10	296	19
GRS-5-5	960	30	10	17	11
GRS-7-8	14336	56	13	25	17

Table 1: Summary of the Avalanche and GoalRockSample.

Instance		HSVI2		FSVI	
Av-6-2-4	$V^*(b_0)$	58.0*	58.0*	58.0	59.8
4 s	10 %	10 s	96 s	5 s	241 s
2 s	1 %	30 s	373 s	78 s	10000 s
	0.1 %	115 s	1783 s	6310 s	10000 s
Av-9-2-4	$V^*(b_0)$	64.1*	64.0*	64.4	69.3
8 s	10 %	11 s	134 s	4 s	113 s
3 s	1 %	54 s	1454	57 s	2368 s
	0.1 %	440 s	4255 s	3568 s	3237 s
GRS-5-5	$V^*(b_0)$	17.3*	17.0	17.3	17.1
<1 s	10 %	<1 s	2 s	<1 s	5 s
<1 s	1 %	5 s	31 s	7 s	16 s
	0.1 %	25 s	1361 s	7 s	27 s
GRS-7-8	$V^*(b_0)$	31.2	30.6	31.2	30.6
4 s	10 %	32 s	251 s	121 s	187 s
8 s	1 %	5425 s	4192 s	319 s	2996 s
	0.1 %	6143 s	7934 s	5171 s	6152 s

Table 2: Results for Avalanche and GoalRockSample. The value with macro actions is listed to the left for each algorithm. The precompilation times are listed in the instance column with macro actions first.

and the number of relevant actions $|\mathcal{A}_{rel}| = |\bigcup_{b\in\mathcal{B}} \mathcal{A}_{rel}(b)|$. We see that when we identify FO-preconditions the total number of actions increases. However, most are support actions which are abstracted away into the macro actions.

For the experiments we measured the expected cost of the currently best policy over time. We let all algorithms improve the policy until 10000 seconds had passed. In order to study the convergence rate of the algorithms, we sampled the current expected cost during execution. The time needed to compute the expected cost was not credited to the execution time. The results are presented in Table 2 where for each problem and algorithm we report the expected cost of the policy at 10000 seconds, $V_\pi(b_0)$, and the time in seconds when a policy was found with an expected cost within 10 %, 1 %, and 0.1 % of that of the best policy. These percentage levels are chosen to illustrate convergence. HSVI2 also provides upper and lower bounds of the optimal policy. Problems for which HSVI2 managed to converge within the time limit so that the difference between these bounds was smaller than 0.1 cost units are indicated with an asterisk at the expected cost. In Table 2 we have also listed for each problem instance the precompilation time necessary for parsing the model, computing macro actions if used, and initialize the value functions.

The results show an overall improvement with macro actions, in many cases by more than a factor of ten. The improvement is especially prominent for convergence up to the 10 % and 1 % levels. For *Av-9-2-4* and *GRS-5-5*, the expected cost after 10000 seconds is higher when macro actions are used despite that the value is fully converged. This means that the optimal policy is missed because of the macro actions. For example, in the avalanche rescue problems, actions using the long range sensor will be executed in the closest cell where it is applicable, but with regard to the next action it may be better to execute it in another cell. However, the expected cost of the policies with macro actions at 10000 seconds is overall close to the expected cost without macro actions. Sometimes the expected cost is even better because the policy without macro actions still was not close to the optimal policy at 10000 seconds. In *GRS-7-8*, HSVI2 appears to perform worse with macro actions than without. This can be explained by that it converges toward a suboptimal policy for which after 10000 seconds it is actually more converged even though the expected cost is higher.

The macro actions cause an increase in the branching factor, but this is compensated by smaller number of steps to reach the goal. For example in the problem *GRS-7-8*, the final polices of HSVI2 and FSVI both need in average 26 actions to reach the goal without macro actions, but with macro actions the average distance to the goal is 12 macro actions for HSVI2 and 13 for FSVI. The average of the branching factor without macro actions taken over all belief states visited by the algorithms is 7.6 for HSVI2 and 7.9 for FSVI. With macro actions the average branching factor is 17.0 for HSVI2 and 12.0 for FSVI.

Truck Braking System. In the second example a mechanic must troubleshoot and repair a malfunctioning hydraulic braking system of a truck. A POMDP-controller aids the mechanic by recommending actions so that the expected cost of repair is minimized. The actions are replacements of components, inspections, tests, and the removing and fitting of parts on the vehicle. The braking system consists of 20 components that can be faulty in one or more ways. The health state of the components can only be observed indirectly by inspecting or performing tests. To gain access to an area of the system where an action is applicable it may be necessary to remove or fit parts on the vehicle. Replacing components and removing and fitting parts is modeled with deterministic actions. It may be costly to become absolutely certain that a possible fault, which currently is not manifesting itself, is not present. Therefore, instead of replacing a component, it is possible to choose to ignore it against a penalty cost proportional to the fault probability. This penalty embodies the cost and badwill incurred by having to revisit the workshop.

Initially it is known that all parts are fitted and that certain fault codes have triggered. A fault code is an alarm generated when suspicious behavior is detected. This system has ten fault codes. For each combination of the triggered fault codes, it is possible to derive from the model an initial probability distribution of which components are faulty[1]. We have restricted the initial distribution so that at most two faults may be present at the same time.

Results. After analyzing the model we get that the number of reachable fully observable partial states $|\mathcal{G}_{fo}| = 73$. The model has 69 actions, all of which can be given FO-

[1]A Bayesian network model for this system is specified in full detail in (Warnquist 2011)

Instance		HSVI2		FSVI		$\lvert\mathcal{S}\rvert$
1	$V^*(b_0)$	347.7*	347.7*	349.6	350.4	16374
	10 %	44 s	610 s	46 s	795 s	20 s
	1 %	44 s	610 s	656 s	5158 s	13 s
	0.1 %	124 s	1058 s	656 s	5158 s	
2	$V^*(b_0)$	292.6*	292.6*	293.7	294.2	14622
	10 %	10 s	78 s	<1 s	2 s	17 s
	1 %	19 s	146 s	58 s	1746 s	10 s
	0.1 %	99 s	527 s	1036 s	1746 s	
3	$V^*(b_0)$	1579.2	1590.5	1548.3	1548.3	14184
	10 %	29 s	44 s	<1 s	15 s	16 s
	1 %	4444 s	44 s	339 s	3892 s	10 s
	0.1 %	7153 s	4600 s	492 s	5452 s	
4	$V^*(b_0)$	299.7*	299.7*	299.7	299.7	10860
	10 %	13 s	327 s	<1	7 s	13 s
	1 %	13 s	327 s	<1	7 s	7 s
	0.1 %	177 s	327 s	7 s	7 s	
5	$V^*(b_0)$	798.9*	798.9*	798.9	798.9	6317
	10 %	4 s	19 s	<1 s	2 s	7 s
	1 %	25 s	303 s	329 s	4332 s	4 s
	0.1 %	31 s	305 s	725 s	4332 s	
6	$V^*(b_0)$	1734.4*	1741.5	1742.0	1741.5	24027
	10 %	96 s	569 s	7 s	25 s	28 s
	1 %	957 s	618 s	49 s	1875 s	20 s
	0.1 %	990 s	3576 s	180 s	1875 s	
7	$V^*(b_0)$	1797.2*	1797.2	1799.3	1799.3	17766
	10 %	20 s	14 s	<1 s	<1 s	21 s
	1 %	20 s	14 s	<1 s	<1 s	14 s
	0.1 %	77 s	3305 s	279 s	16 s	
8	$V^*(b_0)$	884.7*	884.7*	884.7	885.6	11281
	10 %	8 s	7 s	<1 s	2 s	13 s
	1 %	8 s	7 s	<1 s	2 s	7 s
	0.1 %	22 s	232 s	3322 s	2 s	
9	$V^*(b_0)$	352.0*	352.0*	352.4	352.6	16374
	10 %	19 s	130 s	<1 s	15 s	19 s
	1 %	37 s	301 s	62 s	1479 s	12 s
	0.1 %	37 s	301 s	489 s	1479 s	
10	$V^*(b_0)$	326.9*	326.9*	327.0	328.4	14622
	10 %	16 s	74 s	<1 s	3 s	17 s
	1 %	170 s	1078 s	120 s	254 s	10 s
	0.1 %	284 s	2655 s	2781 s	254 s	

Table 3: Results for the truck braking system. The value with macro actions is listed to the left for each algorithm. The precompilation times in the rightmost column are listed with macro actions first.

preconditions without the need for splitting actions. Of these actions, 26 are support actions and the maximum number of relevant actions for any belief state is 47.

To explore the behavior of the proposed method on this model, we have randomly drawn 10 out of the 1024 possible initial observations from which we derive 10 different initial belief states. The setup for the experiment is the same as before. In Table 3 we report the expected cost of the best policy $V(b_0)$ achieved after 10000 seconds of computing and the time in seconds needed to achieve policies with an expected cost that is within 10 %, 1%, and 0.1 % of the best policy for each problem instance and each algorithm. Also listed is the size of the reachable state space $|\mathcal{S}|$ for each instance as well as precompilation times.

The results again show an overall improvement in the convergence for both HSVI2 and FSVI when using macro actions. For the instances where HSVI2 managed to converge to a difference between the upper and lower bounds that is smaller than 0.1 cost units, the expected cost is the same with and without macro actions. This indicates that the macro actions do not result in suboptimal solutions for these problems. For instance 3 with HSVI2 and instances 8 and 10 with FSVI, the convergence times are worse with macro actions. However, for these instances the expected cost is lower with macro actions which means that the policy actually has converged further toward an optimal policy.

Related Work

In contrast to other methods that also exploit deterministic actions (Bonet 2009; Besse and Chaib-draa 2009), the proposed method does not require that *all* actions in the POMDP affect the states deterministically. Instead it targets models where *some* actions affect the fully observable partial state deterministically.

Ong et al. (2010) showed that the convergence of point-based algorithms can be sped up for MOMDPs by reducing the number of α-vectors to consider when backing up a single belief state to be only those that share the same fully observable partial state. This is not in conflict with the proposed method and instead macro actions could be used as a complement since this reduces the number of belief states needed to be backed up after each exploration phase (which is proportional to the search depth).

There are various methods for speeding up POMDP-solving by decomposing the planning problem into a hierarchy of subproblems where the solutions to the subproblems can be incorporated into solving the larger POMDP using macro actions. Many of these methods require that a hierarchy is given together with the POMDP model, see e.g. (Pineau, Roy, and Thrun 2001; Theocharous, Mahadevan, and Kaelbling 2005; Hansen and Zhou 2003). Charlin, Poupart, and Shioda (2007) discover hierarchies represented as finite state controllers that can be used by solvers based on policy iteration (Hansen 1997). The method is applicable only for discounted POMDPs and is slower in comparison with HSVI2. He, Brunskill, and Roy (2010) create macro actions by generating a short sequential plan to a randomly sampled state with good properties. The method does not create a policy. Instead it depends on a planning algorithm that interleaves planning with execution and is therefore not comparable with the proposed method.

Another approach using automatically generated macro-actions is Milestone Guided Sampling (Kurniawati et al. 2011). This approach targets motion-planning problems with long planning horizons modeled with discounted POMDPs. A set of *milestones* are sampled from the state space which are connected by sequences of actions forming macro actions. The assumption from robotics is that states that are close in the reachability graph also have similar properties. It is a point-based algorithm similar to SARSOP (Kurniawati, Hsu, and Lee 2008) where the macro actions are used to select belief states to back up, but the primitive actions are used during the backup. Because Milestone Guided Sampling is only applicable for discounted POMDPs it cannot be compared with the proposed method.

Discussion

The method we propose can be applied on any undiscounted POMDP with non-positive rewards and identifiable goal states. However, for the method to be efficient, it must be possible to identify a number of fully observable state variables and support actions, otherwise the set of relevant possible macro actions will be the same as the applicable primitive actions. (Ong et al. 2010) showed that it is possible to treat a POMDP that has some *almost* fully observable variables as a MOMDP and then use the MOMDP solution as an approximate solution to the original POMDP. This result is also applicable for the method presented here.

We saw in the examples that using macro actions caused the branching factor to increase. This is because the splitting of actions necessary to give all actions valid FO-preconditions increases the total number of available actions. On the other hand, actions that previously could not be performed in a belief state now become reachable in one step because of the macro actions. When the increased branching factor becomes too large relative to the decreased goal depth, the benefit of being able to reach goal states in fewer steps diminishes and the performance may actually worsen. However, when many support actions only affect the fully observable variables and the non-deterministic actions behave similarly when they are applicable, the branching factor may instead decrease because the support actions can be abstracted away.

Conclusion

We have presented a novel method for identifying and exploiting deterministic structures in goal POMDPs. Actions that deterministically affect the fully observable component of the world state can be abstracted away and combined into macro actions that reach further in the search space. We have shown how the model can be analyzed to identify the necessary structures and we have provided theoretical results showing that the use of macro actions preserves solvability. The method here is used together with the POMDP solvers HSVI2 and FSVI, but there are no limits for using it together with other solvers applicable for goal POMDPs. With examples we have demonstrated that the convergence of these algorithms can be significantly improved with the method.

For future work we believe it is possible to make the method applicable for a broader range of problems by extending the method to discounted POMDPs. Other future work is to explore the possibility of also treating actions with non-deterministic effects on the fully observable variables as support actions.

Acknowledgments

This work is supported in part by Scania CV AB and grants from the Vinnova program FFI, the Excellence Center at Linköping and Lund in Information Technology (ELLIIT), the EU FP7 project SHERPA, grant agreement 600958, the Swedish Foundation for Strategic Research CUAS Project, and the Research Council (VR) Linnaeus Center CADICS.

References

Araya-Lòpez, M.; Thomas, V.; Buffet, O.; and Charpillet, F. 2010. A Closer Look at MOMDPs. In *Proceedings of the Twenty-Second IEEE International Conference on Tools with Artificial Intelligence*.

Besse, C., and Chaib-draa, B. 2009. Quasi-Deterministic Partially Observable Markov Decision Processes. In *Neural Information Processing*, volume 5863, 237–246.

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*.

Bonet, B. 2009. Deterministic pomdps revisited. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

Boutilier, C., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Cassandra, A.; Kaelbling, L.; and Littman, M. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101(1-2):99–134.

Charlin, L.; Poupart, P.; and Shioda, R. 2007. Automated hierarchy discovery for planning in partially observable environments. *Advances in Neural Information Processing Systems* 19:225.

Hansen, E., and Zhou, R. 2003. Synthesis of hierarchical finite-state controllers for POMDPs. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling*.

Hansen, E. A. 1997. An Improved Policy Iteration Algorithm for Partially Observable MDPs. In *Advances in Neural Information Processing Systems 10*, 1015–1021. MIT Press.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

He, R.; Brunskill, E.; and Roy, N. 2010. PUMA: Planning under uncertainty with macro-actions. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Kurniawati, H.; Du, Y.; Hsu, D.; and Lee, W. 2011. Motion planning under uncertainty for robotic tasks with long time horizons. *International Journal of Robotics Research* 30(3):308–323.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Proceedings of the 2008 Robotics: Science and Systems Conference*.

Ong, S.; Png, S.; Hsu, D.; and Lee, W. 2010. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research* 29(8):1053–1068.

Pernestål, A.; Warnquist, H.; and Nyberg, M. 2009. Modeling and Troubleshooting with Interventions Applied to an Auxiliary Truck Braking System. In *Proceedings of 2nd IFAC workshop on Dependable Control of Discrete Systems*.

Pernestål, A.; Warnquist, H.; and Nyberg, M. 2012. Modeling and inference for troubleshooting with interventions applied to a heavy truck auxiliary braking system. *Engineering Applications of Artificial Intelligence* 25(4):705 – 719.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*.

Pineau, J.; Roy, N.; and Thrun, S. 2001. A hierarchical approach to POMDP planning and execution. In *Proceedinngs of the ICML Workshop on Hierarchy and Memory in Reinforcement Learning*.

Shani, G.; Brafman, R. I.; and Shimony, S. E. 2007. Forward search value iteration for POMDPs. In *In Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Shani, G. 2012. POMDPSolver - an Java implementaion arranged as an Eclipse package, of most of the point-based algorithms for solving POMDPs. Retrieved from http://www.bgu.ac.il//~shanigu.

Smith, T., and Simmons, R. G. 2004. Heuristic Search Value Iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*.

Smith, T., and Simmons, R. G. 2005. Point-Based POMDP Algorithms: Improved Analysis and Implementation. In *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence*.

Sondik, E. 1971. *The optimal control of partially observable Markov processes.* Ph.D. Dissertation, Stanford.

Spaan, M. T. J., and Vlassis, N. A. 2005. Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research* 24:195–220.

Tarjan, R. 1971. Depth-first search and linear graph algorithms. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory*, 114–121.

Theocharous, G.; Mahadevan, S.; and Kaelbling, L. 2005. Spatial and temporal abstractions in POMDPs applied to robot navigation. Technical report, DTIC Document.

Warnquist, H. 2011. Computer-assisted troubleshooting for efficient off-board diagnosis. Licentiate Thesis, Linköping University.