Complexity of Planning with Partial Observability

Jussi Rintanen
Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, 79110 Freiburg im Breisgau
Germany

Abstract

We show that for conditional planning with partial observability the problem of testing existence of plans with success probability 1 is 2-EXP-complete. This result completes the complexity picture for non-probabilistic propositional planning. We also give new proofs for the EXP-hardness of conditional planning with full observability and the EXPSPACE-hardness of conditional planning without observability. The proofs demonstrate how lack of full observability allows the encoding of exponential space Turing machines in the planning problem, and how the necessity to have branching in plans corresponds to the move to a complexity class defined in terms of alternation from the corresponding deterministic complexity class. Lack of full observability necessitates the use of beliefs states, the number of which is exponential in the number of states, and alternation corresponds to the choices a branching plan can make.

Introduction

The computational complexity of planning is characterized by the associated decision problem of deciding whether for a given problem instance a plan exists. Sometimes further constraints on the plan are imposed, for example related to success probability, resource consumption or other similar properties of plans. The plan existence problem for classical planning, that is planning with deterministic actions and only one initial state, is PSPACE-complete (Bylander 1994).

The classical planning problem can be generalized for example by having nondeterministic actions. In this case definition of plans as sequences of actions does not in general suffice. Instead, plans have to be defined as mappings from states to actions, or alternatively as simple programs with execution conditional on the nondeterministic outcomes of actions. This form of planning is called *conditional planning*. Assuming that the current state of the world can be exactly observed during plan execution (full observability), the plan existence problem is EXP-complete (Littman 1997). However, if no observations are possible, and either actions are nondeterministic or there are several initial states so that the current state cannot be known unambiguously, plans are again simply sequences of actions, but the plan existence problem is more difficult than either of the above problems: it is EXPSPACE-complete (Haslum & Jonsson 2000). This nondeterministic planning problem is sometimes called *conformant planning*.

In both of the above nondeterministic planning problems it is required that a plan reaches a goal state with certainty. There are also probabilistic variants of these problems in which the alternative events associated with nondeterministic actions are assigned probabilities. In planning without observability, determining the existence of a plan with success probability $\geq c$ is undecidable (Madani, Hanks, & Condon 2003). Both full observability and restriction to exponentially long plan executions make the problem decidable and bring it down to EXPSPACE and below (Littman, Goldsmith, & Mundhenk 1998; Mundhenk *et al.* 2000). The undecidability of the most general of these problems, probabilistic conditional planning with partial observability, in which some observations are possible but do not in general allow to determine the current state unambiguously, directly follows from the undecidability of plan existence in the corresponding unobservable case.

In this paper, we address an important problem that generalizes the two non-probabilistic planning problems mentioned above, and is a special case of the probabilistic conditional planning problem with partial observability. This is the conditional planning problem with partial observability and the requirement that goals are reached with probability 1. The computational complexity of this problem had remained unknown until now. Unlike the corresponding probabilistic problem, this problem is decidable because for reaching the goals with probability 1 the exact probabilities of nondeterministic events do not matter, and the uncertainty about the current state can be represented in terms of a set of possible current states. This is in strong contrast to the corresponding probabilistic problem, in which the possible current states – the belief states – correspond to probability distributions over the set of all states. The number of these probability distributions for a finite state space is infinite, while the number of non-probabilistic belief states, interpreted as sets of states, is finite.

The conditional planning problem with partial observability and success probability 1 is important for many applications in which low-probability plan failures do not have to be considered explicitly, or the application itself can be modified so that success probability 1 can be guaranteed. This is

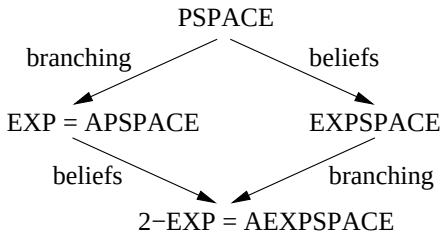

Figure 1: Effect of branching and partial observability on complexity of planning: classical planning is PSPACE-complete, beliefs (partial observability) add space requirements exponentially, and branching (nondeterminism) adds alternation.

so especially in many kinds of manufacturing and engineering applications, in contrast to many optimizations problems typically solved in the MDP/POMDP framework, for which existence of solutions is obvious, and the problem is to find a solution that is optimal or close to optimal.

We show that the plan existence problem for non-probabilistic conditional planning with partial observability is 2-EXP-complete. We outline new proofs of the EXP-hardness of conditional planning with full observability and EXPSPACE-hardness of planning without observability, and obtain the 2-EXP-hardness proof as a generalization of both of these two new proofs. The proofs intuitively explain the problem complexities in terms of the types of Turing machines simulated.

The results complete the complexity picture of non-probabilistic propositional planning in its most general forms, as summarized in Figure 1. Transition from the state space to the belief space leads to exactly an exponential increase in space complexity. From classical planning to conformant planning this is from PSPACE to EXPSPACE, and from nondeterministic full-information planning to nondeterministic planning with partial observability this is from APSPACE = EXP to AEXPSPACE = 2-EXP. Similarly, transition from the deterministic to the corresponding nondeterministic planning problem[1] means a transition from a deterministic complexity class to the corresponding alternating complexity class; this corresponds to the introduction of branches into the plans. From classical planning to nondeterministic planning with full observability this is from PSPACE to APSPACE = EXP, and from conformant planning to general partially observable planning this is from EXPSPACE to AEXPSPACE = 2-EXP.

The structure of the paper is as follows. First we define alternating Turing machines and explain the relations between deterministic complexity classes and their alternating counterparts, followed by a definition of the planning problems we address. In the rest of the paper we analyze the computational complexity of the fully observable, unobservable and partially observable planning problems, in first two cases giving a new more direct hardness proof, and in the third case we establish the complexity for the first time. Before concluding the paper we discuss related work.

[1]We can view conformant planning as deterministic planning in the belief space, because the successor belief state uniquely determined by the action and the preceding belief state.

Preliminaries: Complexity Classes

In this section we define alternating Turing machines and the complexity classes used in the paper.

Definition 1 *An* alternating Turing machine *(ATM) is a tuple* $\langle \Sigma, Q, \delta, q_0, g \rangle$ *where*

- Q *is a finite set of states (the internal states of the ATM),*
- Σ *is a finite alphabet (the contents of tape cells),*
- δ *is a transition function* $\delta : Q \times \Sigma \cup \{|, \square\} \rightarrow 2^{\Sigma \cup \{|\} \times Q \times \{L,N,R\}}$,
- q_0 *is the initial state, and*
- $g : Q \rightarrow \{\forall, \exists, \textit{accept}, \textit{reject}\}$ *is a labeling of the states.*

The symbols | and □ are the left-end-of-tape and the blank symbol, respectively. We require that $s = |$ and $m = R$ for all $\langle s, q', m \rangle \in \delta(q, |)$ and any $q \in Q$, that is, at the beginning of the tape the movement is to the right and | may not be overwritten. For $\langle s', q', m \rangle \in \delta(q, s)$ such that $s \in \Sigma$, we require $s' \in \Sigma$.

A *configuration* of a TM, consisting of the internal state q and the tape contents, is *final* if $g(q) \in \{\text{accept,reject}\}$.

The acceptance of an input string by an ATM is defined inductively starting from final configurations that are accepting. A final configuration is *accepting* if $g(q) = \text{accept}$. Non-final configurations are accepting if the state is universal ($\forall$) and all the successor configurations are accepting or if the state is existential ($\exists$) and at least one of the successor configurations is accepting. Finally, an ATM accepts a given input string if the initial configuration with initial state q_0 and the input string on the work tape is accepting.

A nondeterministic Turing machine (NDTM) is an ATM without universal states. A deterministic Turing machine is an NDTM with $|\delta(q, s)| = 1$ for all $q \in Q$ and $s \in \Sigma \cup \{|\}$.

PSPACE is the class of decision problems solvable by deterministic Turing machines that use a number of tape cells bounded by a polynomial on the input length n. Formally,

$$\text{PSPACE} = \bigcup_{k \geq 0} \text{DSPACE}(n^k).$$

Similarly other complexity classes are defined in terms of time consumption (DTIME($f(n)$), or time and space consumption on alternating Turing machines (ATIME($f(n)$) and ASPACE($f(n)$)) (Balcázar, Díaz, & Gabarró 1988; 1990).

$$\begin{aligned}
\text{EXP} &= \bigcup_{k \geq 0} \text{DTIME}(2^{n^k}) \\
\text{EXPSPACE} &= \bigcup_{k \geq 0} \text{DSPACE}(2^{n^k}) \\
\text{2-EXP} &= \bigcup_{k \geq 0} \text{DTIME}(2^{2^{n^k}}) \\
\text{APSPACE} &= \bigcup_{k \geq 0} \text{ASPACE}(n^k) \\
\text{AEXPSPACE} &= \bigcup_{k \geq 0} \text{ASPACE}(2^{n^k})
\end{aligned}$$

There are many useful connections between these classes (Chandra, Kozen, & Stockmeyer 1981), for example

$$\begin{aligned}
\text{EXP} &= \text{APSPACE} \\
\text{2-EXP} &= \text{AEXPSPACE}.
\end{aligned}$$

Preliminaries: Planning

We formally define the conditional planning problem.

Definition 2 *A* problem instance *in planning is* $\langle A, I, O, G, V\rangle$ *where* A *is a set of Boolean state variables,* I *and* G *are Boolean formulae on* A *respectively describing the sets of initial and goal states,* O *is a set of operators* $\langle c, e\rangle$ *where* c *is a formula on* A *describing the precondition and* e *is an effect, and* $V \subseteq A$ *is the set of* observable state variables. *Effects are recursively defined as follows.*

1. a *and* $\neg a$ *for* $a \in A$ *are effects.*
2. $e_1 \wedge \cdots \wedge e_n$ *is an effect if* $e_1, \ldots, e_n$ *are effects (the special case with* $n = 0$ *is the empty conjunction* $\top$*.)*
3. $c \rhd e$ *is an effect if* c *is a formula on* A *and* e *is an effect.*
4. $e_1 | \cdots | e_n$ *is an effect if* $e_1, \ldots, e_n$ *for* $n \geq 2$ *are effects.*

Above $e_1 \wedge \cdots \wedge e_n$ means that all the effects e_i simultaneously take place. The notation $c \rhd e$ is for conditionality, that is, effect e takes place if c is true in the current state. Nondeterministic effects $e_1 | \cdots | e_n$ mean randomly choosing one of the effects e_i.

Definition 3 (Operator application) *Let* $\langle c, e\rangle$ *be an operator over* A*. Let* s *be a state, that is an assignment of truth values to* A*. The operator is* applicable in s *if* $s \models c$*.*

Recursively assign effects e *a set* $[e]_s$ *of sets of literals.*

1. $[a]_s = \{\{a\}\}$ *and* $[\neg a]_s = \{\{\neg a\}\}$ *for* $a \in A$*.*
2. $[e_1 \wedge \cdots \wedge e_n]_s = \{\bigcup_{i=1}^{n} f_i | f_1 \in [e_1]_s, \ldots, f_n \in [e_n]_s\}$*.*
3. $[c' \rhd e]_s = [e]_s$ *if* $s \models c'$ *and* $[c' \rhd e]_s = \{\emptyset\}$ *otherwise.*
4. $[e_1 | \cdots | e_n]_s = [e_1]_s \cup \cdots \cup [e_n]_s$

The successor states $img_o(s)$ *of a state* s *under an operator* o *are obtained from* s *by making the literals in some* $f \in [e]_s$ *true and retaining the truth-values of state variables not occurring in* f*. Because for nondeterministic* e *there may be several sets* $f \in [e]_s$*, there may also be several successor states for a given state and operator.*

Compound observations and observations dependent on the last applied operator can be reduced in polynomial time to the basic model defined above in which the same set V of state variables is observable all the time.

Definition 4 *Let* A *be a set of state variables. A problem instance* $\langle A, I, O, G, V\rangle$ *is*

1. fully observable *if* $V = A$*,*
2. unobservable *if* $V = \emptyset$*, and*
3. partially observable *if there are no restrictions on* V*.*

Plans are directed graphs with nodes of degree 1 labeled with operators and edges from branch nodes labeled with formulae.

Definition 5 *Let* $\langle A, I, O, G, V\rangle$ *be a problem instance in planning. A plan is a triple* $\langle N, b, l\rangle$ *where*

- N *is a finite set of nodes,*
- $b \in N$ *is the initial node,*
- $l : N \to (O \times N) \cup 2^{\mathcal{L} \times N}$ *is a function that assigns each node an operator and a successor node* $\langle o, n\rangle \in O \times N$ *or a set of formulae and successor nodes* $\langle \phi, n\rangle$*.*
  Only the observable state variables V *may occur in the branch labels* ϕ*. Nodes with* $l(n) = \emptyset$ *are* terminal*.*

An execution of a plan starts from the initial node b. Executing an operator node with label $\langle o, n\rangle$ is by executing o and continuing execution from the node n. Executing a branch node is by evaluating the branch label ϕ for each $\langle \phi, n\rangle$ in the node label, and if the branch label ϕ is currently true then continue execution from node n. Exactly one branch label must be true for this to be well-defined.

Definition 6 (The execution graph of a plan) *Let* $\langle A, I, O, G, V\rangle$ *be a problem instance and* $\pi = \langle N, b, l\rangle$ *be a plan. Then we define* the execution graph *of* π *as a pair* $\langle G, E\rangle$ *where*

1. $G = S \times N$ *where* S *is the set of states (Boolean valuations of* A*),*
2. E *has an edge from* $\langle s, n\rangle$ *to* $\langle s', n'\rangle$ *if and only if*
   (a) $n \in N$ *is an operator node with* $l(n) = \langle o, n'\rangle$ *and* $s' \in img_o(s)$*, or*
   (b) $n \in N$ *is a branch node with* $\langle \phi, n'\rangle \in l(n)$ *and* $s' = s$ *and* $s \models \phi$*.*

Definition 7 *A plan* $\pi = \langle N, b, l\rangle$ solves a problem instance $\langle A, I, O, G, V\rangle$ *if its execution graph fulfills the following.*

For all states s *such that* $s \models I$*, for every* (s', n) *to which there is a path from* (s, b) *that does not visit* G*, there is also a path from* (s', n) *of length* ≥ 0 *to some* (s'', n') *such that* $s'' \models G$ *and* n' *is a terminal node in the plan.*

The definition says that a plan solves a problem instance if none of its executions reaches a state from which goals are not reachable.

The definition can be interpreted probabilistically. For every nondeterministic choice in an operator we have to assume that each of the alternatives has a non-zero probability. A plan with unbounded looping is simply a plan that has no finite upper bound on the length of its executions, but that with probability 1 eventually reaches a goal state. A non-looping plan also reaches a goal state with probability 1, but there is a finite upper bound on the execution length.

The general decision problem addressed in this paper for non-probabilistic conditional planning with partial observability can be defined as follows. Definitions for the various special cases restrict the sets of operators O and sets of observable state variables V.

Definition 8 *The* plan existence problem *is the decision problem of determining whether for a problem instance* $\langle A, I, O, G, V\rangle$ *there is a plan that solves it.*

Planning with Full Observability

The simplest planning problem without any uncertainty about the current states is the one with one initial state and

deterministic operators. In this problem a sequence of operators always leads to a uniquely defined state. Computationally this problem is simpler than the ones with nondeterminism or several initial states.

Theorem 9 (Bylander 1994) *The plan existence problem for problem instances with only one initial state and deterministic operators is PSPACE-complete.*

With nondeterminism (with or without probabilities) and full observability the problem of existence of plans that reach the goal states with probability $\geq c$ is EXP-complete (Littman 1997). Of course, without probabilities only $c = 1$ is meaningful. Littman showed EXP-hardness by reduction from the game G_4 (Stockmeyer & Chandra 1979). The reduction uses $c = 1$, and hence shows hardness for both the probabilistic and the non-probabilistic problem.

Here we sketch a new proof based on the equality EXP = APSPACE. In Theorem 15 we generalize this proof and our EXPSPACE-hardness proof from Theorem 13 to a 2-EXP-hardness proof for the general partially observable problem.

Theorem 10 *The plan existence problem for problem instances with full observability, restricted to acyclic plans, is EXP-hard.*

Proof: We just give a proof sketch. The proof is a Turing machine simulation like the PSPACE-hardness proof of classical planning (Bylander 1994), and, more closely, a simulation of alternating Turing machines in our 2-EXP-hardness proof for planning with partial observability in Theorem 15 but without the watched tape cell construction for handling exponentially long working tapes. We simulate the class of alternating Turing machines with a polynomial space bound, yielding hardness for EXP = APSPACE. Because only a polynomial amount of tape is needed, the tape contents can be explicitly represented in the state variables just like in the PSPACE-hardness proof by Bylander (1994).

The difference to Bylander's proof is caused by the $\forall$ and $\exists$ states of the ATM. For $\forall$ states all the successor configurations have to be accepting, and this is represented as one nondeterministic operator that simulates the nondeterministic transition to one of the successor configurations. For $\exists$ states at least one of the successor configurations has to be accepting, and as the successor configuration can be chosen by the plan, there are several deterministic operators, each choosing one successor configuration. After the nondeterministic $\forall$ transitions the conditional plan branches and chooses the appropriate way to proceed, yielding an exact correspondence with the AND-OR tree evaluation in the definition of alternating Turing machines: the ATM accepts if and only if there is a plan that reaches the goal states under all nondeterministic choices. For more details see the proof of Theorem 15. □

Testing plan existence, and also finding a plan, can be done in exponential time.

Theorem 11 *The plan existence problem for problem instances with full observability is in EXP.*

Proof: We give a brief proof sketch. Taking a problem instance represented succinctly in terms of operators and producing a corresponding exponential size non-succinct explicit representation of its state space takes only exponential time in the size of the problem instance. There are algorithms for finding conditional plans (with loops) in polynomial time in the size of the state space, see for example (Cimatti *et al.* 2003). Hence determining the existence of a plan can be done in exponential time. □

Planning without Observability

The plan existence problem in probabilistic planning without observability is undecidable. Madani et al. (2003) prove this by using the close connection of the problem to the emptiness problem of probabilistic finite automata (Paz 1971; Condon & Lipton 1989). Without observability, plans are sequences of actions. The emptiness problem is about the existence of a word with an acceptance probability higher than c. This equals the planning problem when plans are identified with words and goal states are identified with accepting states. The acceptance probability may be increased by increasing the length of the word, and in general from a given c no finite upper bound can be derived and the problem is therefore undecidable.

When $c = 1$ the situation is completely different. Probabilities strictly between 0 and 1 can be identified, which leads to a finite discrete belief space. For any problem instance there is a finite upper bound on plan length, if a plan exists, and repetitive strategies for increasing success probability to 1 do not have to be used and they do not help in reaching 1. Notice that for every other fixed $c \in]0, 1[$ undecidability holds as the general problem can be reduced to the fixed-c problem for any $c \in]0, 1[$ by adding a first action that scales every success probability to the fixed c.

The unobservable planning problem is easily seen to be in EXPSPACE. Haslum and Jonsson (2000) point out this fact and outline the proof which is virtually identical to the PSPACE membership proof of plan existence for classical planning (Bylander 1994) except that it works at the level of belief states instead of states. For a problem represented in terms of n state variables, there may be 2^n states, but 2^{2^n} belief states, which is why plans may be much longer and the complexity of the unobservable problem is much higher.

Theorem 12 *The plan existence problem for problem instances without observability is in EXPSPACE.*

Haslum and Jonsson (2000) also show that an unobservable planning problem similar to ours is EXPSPACE-hard. Their proof is a reduction from the EXPSPACE-hard universality problem of regular expressions with exponentiation (Hopcroft & Ullman 1979).

Our EXPSPACE-hardness proof simulates deterministic exponential-space Turing machines by planning. The main problem to be solved is the simulation of exponentially long tapes. In the PSPACE-hardness proof of classical planning each tape cell can be represented by one state variable, but with an exponentially long tape this kind of simulation is

not possible. Instead, we use a randomization technique that forces the tape contents to be faithfully represented in the plan.

Theorem 13 *The plan existence problem for problem instance with unobservability is EXPSPACE-hard. This holds also for problem instances with deterministic operators.*

Proof: Let $\langle \Sigma, Q, \delta, q_0, g \rangle$ be a deterministic Turing machine with an exponential space bound $e(x)$, and σ an input string of length n. We denote the ith symbol of σ by σ_i.

For encoding numbers from 0 to $e(n)+1$ we need $m = \lceil \log_2(e(n)+2) \rceil$ Boolean state variables.

We construct a problem instance without observability for simulating the Turing machine. The size of the instance is polynomial in the size of the TM and the input string.

It turns out that when not everything is observable, instead of encoding all tape cells in the planning problem, it is sufficient to keep track of only one tape cell (which we call the *watched tape cell*) which is randomly chosen in the beginning of every plan execution.

The set A of state variables consists of

1. $q \in Q$ for the internal states of the TM,
2. w_i for $i \in \{0, \ldots, m-1\}$ for the watched tape cell,
3. $s \in \Sigma \cup \{|, \square\}$ for contents of the watched tape cell, and
4. h_i, $i \in \{0, \ldots, m-1\}$ for the position of the R/W head.

Notice that we use the symbols q and s for states and symbols as well as for corresponding state variables.

The initial state formula describes several initial states corresponding to the different choices of the watched tape cell (the watched tape cell encoded by state variables w does not change later.) Otherwise the formula encodes the initial configuration of the TM, and it is the conjunction of the following formulae.

1. q_0
2. $\neg q$ for all $q \in Q \backslash \{q_0\}$
3. Contents of the watched tape cell:

$$\begin{array}{rcl} | & \leftrightarrow & (w = 0) \\ \square & \leftrightarrow & (w > n) \\ s & \leftrightarrow & \bigvee_{i \in \{1,\ldots,n\}, \sigma_i = s} (w = i) \text{ for all } s \in \Sigma \end{array}$$

4. $h = 1$ for the initial position of the R/W head.

So the initial state formula allows any values for state variables w_i and the values of the state variables $s \in \Sigma$ are determined by the values of w_i. The expressions $w = i$ and $w > i$ denote the obvious formulae for integer equality and inequality of the numbers encoded by $w_0, w_1, \ldots$. We also use effects $h := h+1$ and $h := h-1$ for incrementing and decrementing the number encoded by variables h_i.

The goal is the following formula.

$$G = \bigvee \{q \in Q | g(q) = \text{accept}\}$$

To define the operators, we first define effects corresponding to all possible transitions.

For all $\langle s, q \rangle \in (\Sigma \cup \{|, \square\}) \times Q$ and $\langle s', q', m \rangle \in (\Sigma \cup \{|\}) \times Q \times \{L, N, R\}$ define the effect $\tau_{s,q}(s', q', m)$ as $\alpha \wedge \kappa \wedge \theta$ where the effects α, κ and θ are defined as follows.

The effect α describes the change to the current tape symbol. If $s = s'$ then $\alpha = \top$ as nothing on the tape changes. Otherwise, $\alpha = ((h = w) \rhd (\neg s \wedge s'))$ to denote that the new symbol in the watched tape cell is s' and not s.

The effect κ changes the internal state of the TM. If there is no R/W head movement or it is to the left, $\kappa = \neg q \wedge q'$ if $q \neq q'$ and $\kappa = \top$ otherwise. If R/W head movement is to the right then $\kappa = \neg q \wedge ((h < e(n)) \rhd q')$ if $q \neq q'$ and $\kappa = (h = e(n)) \rhd \neg q$ otherwise. This prevents reaching an accepting state if the space bound is violated: no further operators are applicable.

The effect θ describes the movement of the R/W head.

$$\theta \quad = \quad \begin{cases} h := h-1 & \text{if } m = L \\ \top & \text{if } m = N \\ h := h+1 & \text{if } m = R \end{cases}$$

Now, these effects $\tau_{s,q}(s', q', m)$ which simulate the DTM transitions are used in the operators. Let $\langle s, q \rangle \in (\Sigma \cup \{|, \square\}) \times Q$ and $\delta(s, q) = \{\langle s', q', m \rangle\}$.

If $g(q) = \exists$ then define the operator

$$o_{s,q} = \langle ((h \neq w) \vee s) \wedge q, \tau_{s,q}(s', q', m) \rangle.$$

The simulation is faithful assuming that on all executions of operator $o_{s,q}$ the current tape symbol is indeed s. So assume that some $o_{s,q}$ is the first operator that misrepresents the tape contents. Let $h = c$ for some c be the location of the R/W head at this point of the plan. Now there is an execution (initial state) of the plan so that $w = c$. On this execution the precondition of $o_{s,q}$ is not satisfied, and the plan is not executable. Hence a valid plan cannot contain operators that misrepresent the tape contents. □

Planning with Partial Observability

Showing that the plan existence problem for planning with partial observability is in 2-EXP is straightforward. The easiest way to see this is to view the partially observable planning problem as a nondeterministic fully observable planning problem with belief states viewed as states. An operator maps a belief state to another belief state nondeterministically: compute the image of a belief state with respect to an operator, and choose the subset of its states that correspond to one of the possible observations. Like pointed out in the proof of Theorem 11, the algorithms for fully observable planning run in polynomial time in the size of the state space. The state space with the belief states as the states has a doubly exponential size in the size of the problem instance, and hence the algorithm runs in doubly exponential time in the size of the problem instance. This gives us the membership in 2-EXP.

Theorem 14 *The plan existence problem for problem instances with partial observability is in 2-EXP.*

The main result of the paper, the 2-EXP-hardness of partially observable planning, is obtained as a generalization of

the proofs of Theorems 10 and 13. From the EXP-hardness proof we take the simulation of alternation by nondeterministic operators, and from the EXPSPACE-hardness proof the simulation of exponentially long working tapes. These yield a simulation of alternating Turing machines with an exponential space bound, and thereby proof of AEXPSPACE-hardness.

Theorem 15 *The plan existence problem for problem instance with partial observability, restricted to acyclic plans, is 2-EXP-hard.*

Proof: Let $\langle \Sigma, Q, \delta, q_0, g \rangle$ be any alternating Turing machine with an exponential space bound $e(x)$. Let σ be an input string of length n. We denote the ith symbol of σ by σ_i.

We construct a problem instance in nondeterministic planning with partial observability for simulating the Turing machine. The problem instance has a size that is polynomial in the size of the description of the Turing machine and the input string.

Here we just list the differences to the proof of Theorem 13 needed for handling alternation.

The set of state variables is extended with

1. s^* for $s \in \Sigma \cup \{|\}$ for the symbol last written, and
2. L, R and N for the last movement of the R/W head.

The observable state variables are L, N and R, $q \in Q$, and s^* for all $s \in \Sigma$. These are needed by the plan to decide how to proceed execution after a nondeterministic $\forall$-transition.

The initial state formula is conjoined with $\neg s^*$ for all $s \in \Sigma \cup \{|\}$ and with $\neg L \wedge \neg N \wedge \neg R$. The goal formula remains unchanged.

Next we define the operators. All the transitions may be nondeterministic, and the important thing is whether the transition is for a $\forall$ state or an $\exists$ state. For a given input symbol and a $\forall$ state, the transition corresponds to one nondeterministic operator, whereas for a given input symbol and an $\exists$ state the transitions corresponds to a set of deterministic operators.

Effects $\tau_{s,q}(s', q', m) = \alpha \wedge \kappa \wedge \theta$ are like in the proof of Theorem 13 except for the following modifications. The effect α is modified to store the written symbols to the state variables s^*. If $s = s'$ then $\alpha = \top$ as nothing on the tape changes. Otherwise, $\alpha = [(h = w) \rhd (\neg s \wedge s')] \wedge s'^* \wedge \bigwedge_{s'' \in \Sigma \setminus \{s'\}} \neg s''^*$. The effect θ is similarly extended to store the tape movement to L, N and R.

$$\theta \quad = \quad \begin{cases} (h := h-1) \wedge L \wedge \neg N \wedge \neg R & \text{if } m = L \\ N \wedge \neg L \wedge \neg R & \text{if } m = N \\ (h := h+1) \wedge R \wedge \neg L \wedge \neg N & \text{if } m = R \end{cases}$$

Now, these effects $\tau_{s,q}(s', q', m)$ which represent possible transitions are used in the operators that simulate the ATM. Operators for existential states $q, g(q) = \exists$ and for universal states $q, g(q) = \forall$ differ. Let $\langle s, q \rangle \in (\Sigma \cup \{|, \Box\}) \times Q$ and $\delta(s, q) = \{\langle s_1, q_1, m_1 \rangle, \ldots, \langle s_k, q_k, m_k \rangle\}$.

If $g(q) = \exists$, then define k deterministic operators

$$\begin{aligned} o_{s,q,1} &= \langle (((h \neq w) \vee s) \wedge q, \tau_{s,q}(s_1, q_1, m_1) \rangle \\ o_{s,q,2} &= \langle (((h \neq w) \vee s) \wedge q, \tau_{s,q}(s_2, q_2, m_2) \rangle \\ &\vdots \\ o_{s,q,k} &= \langle (((h \neq w) \vee s) \wedge q, \tau_{s,q}(s_k, q_k, m_k) \rangle. \end{aligned}$$

That is, the plan determines which transition is chosen. If $g(q) = \forall$, then define one nondeterministic operator

$$\begin{aligned} o_{s,q} = \langle (((h \neq w) \vee s) \wedge q, \quad & (\tau_{s,q}(s_1, q_1, m_1) | \\ & \vdots \\ & \tau_{s,q}(s_k, q_k, m_k)) \rangle. \end{aligned}$$

That is, the transition is chosen nondeterministically.

We claim that the problem instance has a plan if and only if the Turing machine accepts without violating the space bound. If the Turing machine violates the space bound, then $h > e(n)$ and an accepting state cannot be reached because no further operator will be applicable.

From an accepting computation tree of an ATM we can construct a plan, and vice versa. Accepting final configurations are mapped to terminal nodes of plans, $\exists$-configurations are mapped to operator nodes in which an operator corresponding to the transition to an accepting successor configuration is applied, and $\forall$-configurations are mapped to operator nodes corresponding to the matching nondeterministic operators followed by a branch node that selects the plan nodes corresponding to the successors of the $\forall$ configuration. The successors of $\forall$ and $\exists$ configurations are recursively mapped to plans.

Construction of computation trees from plans is similar, but involves small technicalities. A plan with DAG form can be turned into a tree by having several copies of the shared subplans. Branches not directly following the nondeterministic operator causing the uncertainty can be moved earlier so that every nondeterministic operator is directly followed by a branch that chooses a successor node for every possible new state, written symbol and last tape movement. With these transformations there is an exact match between plans and computation trees of the ATM, and mapping from plans to ATMs is straightforward like in the opposite direction.

Because alternating Turing machines with an exponential space bound are polynomial time reducible to the nondeterministic planning problem with partial observability, the plan existence problem is AEXPSPACE=2-EXP-hard. □

Plans with Loops

The complexities of the fully and partially observable problems hold also when loops are allowed in the plans. Loops are needed for finitely representing repetitive strategies for solving nondeterministic problem instances without an upper bound on execution length. A typical problem would be to toss a die until its value is six. Assuming that the probability of the die falling on every side is non-zero, eventually getting six has probability 1, although the zero-probability event of having to unsuccessfully throw the die infinitely many times is still possible.

In this section we extend Theorems 10 and 15 to general plans with loops. The problem looping plans cause in the proofs of these theorems is that a Turing machine computation of infinite length is not accepting but the corresponding infinite length zero-probability plan execution is allowed to be a part of plan and would incorrectly count as an accepting Turing machine computation.

To eliminate infinite plan executions we have to modify the Turing machine simulations. This is by counting the length of the plan executions and failing when at least one state or belief state must have been visited more than once. This modification makes infinite loops ineffective, and any plan containing a loop can be translated to a finite non-looping plan by unfolding the loop. In the absence of loops the simulation of alternating Turing machines is faithful.

Theorem 16 *The plan existence problem for problem instances with full observability is EXP-hard.*

Proof: This is an easy extension of the proof of Theorem 10. If there are n state variables, an acyclic plan exists if and only if a plan with execution length at most 2^n exists, because visiting any state more than once is unnecessary. Plans that rely on loops can be invalidated by counting the number of actions taken and failing when this exceeds 2^n. This counting can be done by having $n+1$ auxiliary state variables $c_0, \ldots, c_n$ that are initialized to false. Every operator $\langle p, e\rangle$ is extended to $\langle p, e \wedge t\rangle$ where t is an effect that increments the binary number encoded by $c_0, \ldots, c_n$ by one until the most significant bit c_n becomes one. The goal G is replaced by $G \wedge \neg c_n$.

Then a plan exists if and only if an acyclic plan exists if and only if the alternating Turing machine accepts. □

For the fully observable case counting the execution length does not pose a problem because we only have to count an exponential number of execution steps, which can be represented by a polynomial number of state variables, but in the partially observable case we need to count a doubly exponential number of execution steps, as the number of belief states to be visited may be doubly exponential. A binary representation of these numbers requires an exponential number of bits, and we cannot use an exponential number of state variables for the purpose, because the reduction to planning would not be polynomial time. However, partial observability together with only a polynomial number of auxiliary state variables can be used to force the plans to count doubly exponentially far.

Theorem 17 *The plan existence problem for problem instances with partial observability is 2-EXP-hard.*

Proof: We extend the proof of Theorem 15 by a counting scheme that makes cyclic plans ineffective. We show how counting the execution length can be achieved within a problem instance obtained from the alternating Turing machine and the input string in polynomial time.

Instead of representing the exponential number of bits explicitly as state variables, we use a randomizing technique for forcing the plans to count the number of Turing machine transitions. The technique has resemblance to the idea in simulating exponentially long tapes in the proofs of Theorems 13 and 15.

For a problem instance with n state variables (representing the Turing machine configurations) executions that visit each belief state at most once may have length 2^{2^n}. Representing numbers from 0 to $2^{2^n}-1$ requires 2^n binary digits. We introduce $n+1$ new unobservable state variables $d_0, \ldots, d_n$ for representing the index of one of the digits and v_d for the value of that digit, and new state variables $c_0, \ldots, c_n$ through which the plan indicates changes in the counter of Turing machine transitions. There is a set of operators by means of which the plan sets the values of these variables before every transition of the Turing machine is made.

The idea of the construction is the following. Whenever the counter of TM transitions is incremented, one of the 2^n digits in the counter changes from 0 to 1 and all of the less significant digits change from 1 to 0. The plan is forced to communicate the index of the digit that changes from 0 to 1 by the state variables $c_0, \ldots, c_n$. The unobservable state variables $d_0, \ldots, d_n, v_d$ store the index and value of one of the digits (chosen randomly in the beginning of the plan execution), that we call *the watched digit*, and they are used for checking that the reporting of $c_0, \ldots, c_n$ by the plan is truthful. The test for truthful reporting is randomized, but this suffices to invalidate plans that incorrectly report the increments, as a valid plan has to reach the goals on every possible execution. The plan is invalid if reporting is false or when the count can exceed 2^{2^n}. For this reason a plan for the problem instance exists if and only if an acyclic plan exists if and only if the Turing machine accepts the input string.

Next we exactly define how the problem instances defined in the proof of Theorem 15 are extended with a counter to prevent unbounded looping.

The initial state description is extended with the conjunct $\neg d_v$ to signify that the watched digit is initially 0 (all the digits in the counter implicitly represented in the belief state are 0.) The state variables $d_0, \ldots, d_n$ may have any values which means that the watched digit is chosen randomly. The state variables $d_v, d_0, \ldots, d_n$ are all unobservable so that the plan does not know the watched digit (may not depend on it).

There is also a failure flag f that is initially set to false by having $\neg f$ in the initial states formula.

The goal is extended by $\neg f \wedge ((d_0 \wedge \cdots \wedge d_n) \rightarrow \neg d_v)$ to prevent executions that lead to setting f true or that have length $2^{2^{n+1}-1}$ or more. The conjunct $(d_0 \wedge \cdots \wedge d_n) \rightarrow \neg d_v$ is false if the index of the watched digit is $2^{n+1}-1$ and the digit is true, indicating an execution of length $\geq 2^{2^{n+1}-1}$.

Then we extend the operators simulating the Turing machine transitions, as well as introduce new operators for indicating which digit changes from 0 to 1.

The operators for indicating the changing digit are

$$\begin{array}{ll} \langle \top, c_i\rangle & \text{for all } i \in \{0, \ldots, n\} \\ \langle \top, \neg c_i\rangle & \text{for all } i \in \{0, \ldots, n\} \end{array}$$

The operators for Turing machine transitions are extended with the randomized test that the digit the plan claims to

change from 0 to 1 is indeed the one: every operator $\langle p, e\rangle$ defined in the proof of Theorem 15 is replaced by $\langle p, e \wedge t\rangle$ where the test t is the conjunction of the following effects.

$$\begin{array}{rcl} ((c = d) \wedge d_v) & \rhd & f \\ (c = d) & \rhd & d_v \\ ((c > d) \wedge \neg d_v) & \rhd & f \\ (c > d) & \rhd & \neg d_v \end{array}$$

Here $c = d$ denotes $(c_0 \leftrightarrow d_0) \wedge \cdots \wedge (c_n \leftrightarrow d_n)$ and $c > d$ encodes the greater-than test for the binary numbers encoded by $c_0, \ldots, c_n$ and $d_0, \ldots, d_n$.

The above effects do the following.

1. When the plan claims that the watched digit changes from 0 to 1 and the value of d_v is 1, fail.
2. When the plan claims that the watched digit changes from 0 to 1, change d_v to 1.
3. When the plan claims that a more significant digit changes from 0 to 1 and the value of d_v is 0, fail.
4. When the plan claims that a more significant digit changes from 0 to 1, set the value of d_v to 0.

That these effects guarantee the invalidity of a plan that relies on unbounded looping is because the failure flag f will be set if the plan lies about the count, or the most significant bit with index $2^{n+1} - 1$ will be set if the count reaches $2^{2^{n+1}-1}$. Attempts of unfair counting are recognized and consequently f is set to true because of the following.

Assume that the binary digit at index i changes from 0 to 1 (and therefore all less significant digits change from 1 to 0) and the plan incorrectly claims that it is the digit j that changes, and this is the first time on that execution that the plan lies (hence the value of d_v is the true value of the watched digit.)

If $j > i$, then i could be the watched digit (and hence $c > d$), and for j to change from 0 to 1 the less significant bit i should be 1, but we would know that it is not because d_v is false. Consequently on this plan execution the failure flag f would be set.

If $j < i$, then j could be the watched digit (and hence $c = d$), and the value of d_v would indicate that the current value of digit j is 1, not 0. Consequently on this plan execution the failure flag f would be set.

So, if the plan does not correctly report the digit that changes from 0 to 1, then the plan is not valid. Hence any valid plan correctly counts the execution length which cannot exceed $2^{2^{n+1}-1}$. □

Impact of Determinism on Complexity

Our proofs for the EXP-hardness and 2-EXP-hardness of plan existence for fully and partially observable planning use nondeterminism. Also the EXP-hardness proof of Littman (1997) uses nondeterminism. The question arises whether complexity is lower when all operators are deterministic.

Theorem 18 *The plan existence problem for problem instances with full observability and deterministic operators is in PSPACE.*

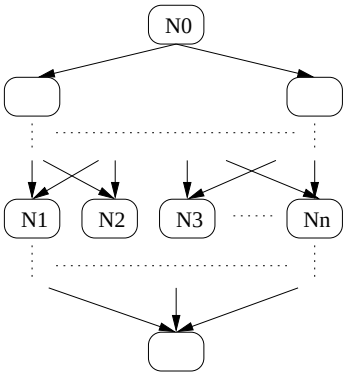

Figure 2: In a deterministic problem instance, the sum of the cardinalities of the possible sets of states at plan nodes $N_1, \ldots, N_n$ cannot exceed the cardinality of the possible set of states at the root node of the plan N_0. Here $N_1, \ldots, N_n$ are plan nodes that intersect the plan in the sense that no node is an ancestor of another.

Proof: This is because iteration over an exponential number of initial states needs only polynomial space, and testing goal reachability for each initial state needs only polynomial space like in the PSPACE membership proof of classical planning (Bylander 1994). □

Under unobservability determinism does not reduce complexity: proof of Theorem 13 uses deterministic operators only. But for the general partially observable problem determinism does reduce complexity. EXPSPACE-hardness is inherited from the deterministic unobservable problem, and membership in EXPSPACE is as follows.

Theorem 19 *The plan existence problem for problem instances with partial observability and deterministic operators is in EXPSPACE.*

Proof: The idea is similar to the EXPSPACE membership proof of planning without observability: go through all possible intermediate stages of a plan by binary search. Determinism yields an exponential upper bound on the sum of the cardinalities of the belief states that are possible after branching and taking a given number of actions, (see Figure 2), and determinism also makes it unnecessary to visit any belief state more than once. Hence plan executions have doubly exponential length and binary search needs only exponential recursion depth.

Let $\langle C_1, \ldots, C_k\rangle$ be the classes of observationally indistinguishable states. Let S be the set of states. For a belief state B and a set L of belief states with $\sum_{B' \in L} |B'| \leq |S|$, test reachability of L from B with plans of maximum execution length 2^i by the following procedure.

procedure reach(B,L,i)
if $i = 0$ **then**
  begin
    for each $j \in \{1, \ldots, k\}$
      if $\text{img}_o(B) \cap C_j \not\subseteq B'$ for all $o \in O$ and $B' \in L$
      and $B \cap C_j \not\subseteq B'$ for all $B' \in L$

	deterministic	non-deterministic
full observability	PSPACE	EXP
no observability	EXPSPACE	EXPSPACE
partial observability	EXPSPACE	2-EXP

Table 1: Summary of complexities of planning with multiple initial states, deterministic or non-deterministic operators, and different degrees of observability.

	deterministic	non-deterministic
full observability	T9, T18	T16, T11
no observability	T13, T12	T13, T12
partial observability	T13, T19	T14, T17

Table 2: Theorems establishing hardness and membership.

```
        then return false
      end
      return true;
    end
else
    for each L' ⊆ 2^S such that Σ_{B'∈L'} |B'| ≤ |B| and
        for every B' ∈ L', B' ⊆ C_j for some j ∈ {1,...,k}
      if reach(B,L',i − 1) then
        begin
          flag := true;
          for each B' ∈ L'
            if reach(B',L,i − 1) = false then flag := false;
          if flag = true then return true
        end
    end
return false
```

The base case $i = 0$ tests whether for every observation from B a belief state in L is reached by one operator or directly. The inductive case $i \geq 1$ iterates over sets L' of intermediate belief states and recursively tests reachability from B to L' and further from L' to L.

We can now test plan existence by calling reach(B,L,$|S|$) for every $B = I \cap C_i, i \in \{1, \ldots, k\}$ and $L = \{G \cap C_i | i \in \{1, \ldots, k\}\}$. The algorithm always terminates, and a plan exists if and only if answer *true* is obtained in all cases.

The space consumption is (only) exponential because the recursion depth is exponential and the sets $L' \subseteq 2^S$ with $\sum_{B' \in L'} |B'| \leq |B|$ have size $\leq |S|$. Sets L' this small suffice because all operators are deterministic, and after any number of actions, independently of how the plan has branched, the sum of the cardinalities of the possible belief states is not higher than the number of initial states. □

Summary

The complexities of the problems discussed in this paper are summarized in Table 1 and the corresponding theorems showing hardness and membership are listed in Table 2. Restriction to only one initial state affects the deterministic unobservable and partially observable planning problems only: they both come down to PSPACE from EXPSPACE.

Related Work and Conclusions

In this paper we have proved that the plan existence problem of propositional non-probabilistic planning with partial observability is 2-EXP-complete and the special case with deterministic operators is EXPSPACE-complete. Complexity of classical planning and non-probabilistic propositional planning with observability restrictions (full observability, no observability) was already known (Bylander 1994; Littman 1997; Haslum & Jonsson 2000).

We also gave more direct proofs of EXPSPACE-hardness of planning without observability and EXP-hardness of planning with full observability, shedding more light to results first proved by Haslum and Jonsson and by Littman, and discussed the impact of determinism on complexity.

Our results do not address plan optimality with respect to a cost measure, like plan size or plan depth, but it seems that for many cost measures there is no increase in complexity and that this is in many cases easy to prove. On the other hand, even when it is required that goals are reached with certainty, probabilistic optimality measures like expected costs can easily be seen to make the plan existence problem undecidable.

The complexity of conditional planning with different combinations of restrictions on plan sizes and execution lengths (polynomial plan size, polynomial execution length, constant execution length) have been investigated in earlier work (Rintanen 1999; Baral, Kreinovich, & Trejo 2000; Turner 2002).

The complexity of constructing and evaluating policies for MDPs with the restriction to finite-horizon performance has been thoroughly analyzed by Mundhenk et al. (2000). They evaluate the computational complexity of policy evaluation and policy existence under different restrictions. While in the fully observable, unobservable and in the general case we have problems complete for EXP, EXPSPACE and 2-EXP, Mundhenk et al. have EXP, NEXP and EXPSPACE for history-dependent POMDPs with problem instances represented as circuits, exponentially long horizons and same observability restrictions. The latter complexities are this low because of the exponential horizon length.

References

Balcázar, J. L.; Díaz, J.; and Gabarró, J. 1988. *Structural Complexity I*. Berlin: Springer-Verlag.

Balcázar, J. L.; Díaz, J.; and Gabarró, J. 1990. *Structural Complexity II*. Berlin: Springer-Verlag.

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122(1):241–267.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Chandra, A.; Kozen, D.; and Stockmeyer, L. 1981. Alternation. *Journal of the ACM* 28(1):114–133.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via sym-

bolic model checking. *Artificial Intelligence* 147(1–2):35–84.

Condon, A., and Lipton, R. J. 1989. On the complexity of space bounded interactive proofs (extended abstract). In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, 462–467. IEEE.

Haslum, P., and Jonsson, P. 2000. Some results on the complexity of planning with incomplete information. In Biundo, S., and Fox, M., eds., *Recent Advances in AI Planning. Fifth European Conference on Planning (ECP'99)*, number 1809 in Lecture Notes in Artificial Intelligence, 308–318. Springer-Verlag.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.

Littman, M. L.; Goldsmith, J.; and Mundhenk, M. 1998. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research* 9:1–36.

Littman, M. L. 1997. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, 748–754. Menlo Park: AAAI Press.

Madani, O.; Hanks, S.; and Condon, A. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence* 147(1–2):5–34.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM* 47(4):681–720.

Paz, A. 1971. *Introduction to Probabilistic Automata*. Academic Press.

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research* 10:323–352.

Stockmeyer, L. J., and Chandra, A. K. 1979. Provably difficult combinatorial games. *SIAM Journal on Computing* 8(2):151–174.

Turner, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *Logics in Artificial Intelligence, European Conference, JELIA 2002*, number 2424 in Lecture Notes in Computer Science, 111–124. Springer-Verlag.