The Complexity of Deterministically Observable Finite-Horizon Markov Decision Processes

Judy Goldsmith * Chris Lusena * Martin Mundhenk †
University of Kentucky‡

December 13, 1996

Abstract

We consider the complexity of the decision problem for different types of partially-observable Markov decision processes (MDPs): given an MDP, does there exist a policy with performance > 0? Lower and upper bounds on the complexity of the decision problems are shown in terms of completeness for NL, P, NP, PSPACE, EXP, NEXP or EXPSPACE, dependent on the type of the Markov decision process. For several NP-complete types, we show that they are not even polynomial time ε-approximable for any fixed ε, unless P = NP. These results also reveal interesting trade-offs between the power of policies, observations, and rewards.

Topics: computational and structural complexity, computational issues in A.I.

1 Introduction

A Markov decision process (MDP) is a model of a decision maker or agent interacting synchronously with a probabilistic system. The agent is able to observe the state of the system and to influence the behaviour of the system as it evolves through time. It does the latter by making decisions or choosing actions which evolve costs or gain rewards. Its policy of choices rely on his observations of the system's state and on his goal to cause the system to perform optimally with respect to the expected sum of rewards. Since the system is ongoing, its state prior to tomorrow's observation depends on today's action (in a probabilistic manner). Consequently, actions must not be chosen myopically, but must anticipate the opportunities and rewards (which may be both positive and negative) associated with future system's states. We restrict our attention to finite discrete-time, finite-state, stochastic dynamical systems where the next state (at any fixed time) depends only on the current state, i.e. Markov chains.

Reaching out from operations research roots in the 1950's, MDP models have gained recognition in such diverse fields as operations research, A.I., learning theory, economics, ecology and communications engineering. Their popularity relies on the fact that an optimal solution can be computed in polynomial time via Dynamic Programming or Linear Programming techniques, if the system has a finite number of states which are fully observable by the agent. (For a detailed discussion see e.g. [1, 8].)

The MDP model allows a variety of assumptions about the capability of the agent, the rewards and the time span allowed for the agent to collect the rewards. Rewards can be restricted to be nonnegative. The action time of the agent – the *horizon* – can be infinite or finite. We will consider finite horizons which are bounded by the number of states (or by its logarithm) of the MDP under consideration.

The agents capability is determined by observation and memory restrictions. With regard to observations, two extremes are *fully observable MDPs*, in which the agent knows exactly what state it is in (at each time), and *unobservable MDPs*, in which the agent receives *no* information about the system's state during

*Supported in part by NSF grant CCR-9315354

†Supported in part by the Office of the Vice Chancellor for Research and Graduate Studies at the University of Kentucky, and by the Deutsche Forschungsgemeinschaft (DFG), grant Mu 1226/2-1

‡mundhenk@ti.uni-trier.de, goldsmit@cs.engr.uky.edu

execution. Both are special cases of *partially observable MDPs*, in which the agent may receive incomplete (or noisy) information about the systems state.[1] This generalization of fully-observable MDPs yields greater expressive power (i.e. more complex situations can be modeled), but – naturally – does not make the search for the optimal solution easier. With regard to memory, the agent may be memoryless, or it may be able to remember the number of actions taken, or it may remember all previous observations. In the first case, the decisions of the agent depend only on his observation of the current state of the process and its type of policy is called *stationary*. In the latter cases the types of policy are called *time-dependent* respectively *history-dependent*. Stationary policies are special cases of time-dependent policies which again generalize to history-dependent policies. It is straightforward to see that for a given MDP an optimal stationary policy is the hardest to find with respect to this hierarchy.

To make MDPs applicable in a broader way one can make use of structures in the state space to provide small descriptions of very big systems [2]. Whereas those systems are not tractable by classical methods, there is some hope – expressed in different algorithmic approaches – that many special cases of these *structured* MDPs can be solved more efficiently. Nevertheless, in [2] it is conjectured that in general structured MDPs are intractable.

In this paper, we systematically investigate a variety of types of MDPs:

- fully-observable, unobservable, and partially-observable,
- performance under stationary, time-dependent an history-dependent policies,
- horizon bounded by the size of the state space $|\mathcal{S}|$, or by its logarithm $\lceil \log |\mathcal{S}| \rceil$,
- rewards restricted to be nonnegative, or unrestricted,
- unstructured and structured (called succinct) representations.

For each of these types, we consider the complexity of the problem, given an MDP, whether a policy with performance > 0 exists for it. Dependent on the type, we show completeness results for a broad variety of complexity classes from nondeterministic logarithmic space to exponential space. We prove the above conjecture by [2] to be true by showing that in many cases the complexity of our decision problems increases exponentially if structured MDPs are considered instead of MDPs. We also consider the problem of computing an optimal policy. For the decision problems shown to be NP-complete, we consider the question whether there exist polynomial time algorithms which approximate the performance of the optimal policy. We show that this is not possible unless P = NP, improving results from Burago et al.[3]

Papadimitriou and Tsitsiklis [6] considered MDPs with nonnegative rewards and investigated the complexity of the decision problem: given an MDP, does there exist a time-dependent or history-dependent policy with expected sum of rewards *equal to* 0. For finite horizon $|\mathcal{S}|$ they showed that this problem is P-complete for fully-observable MDPs under any policy, PSPACE-complete for partially-observable MDPs under history-dependent policies, and NP-complete for unobservable MDPs. In [3] it is shown that the problem of constructing even a very weak approximation to an optimal policy is NP-hard. Note that the decision problem by Papadimitriou and Tsitsiklis is a minimization problem. Also it cannot be used in a binary search to determine the minimal performance of an MDP.

The decision problem considered in this paper can be used in a binary search to compute the maximal performance of an MDP. Because maximization and minimization problems may have very different complexities (look at the max-flow and min-flow problems) our results differ greatly from the results in [6]. For example, our decision problem for partially-observable MDPs with nonnegative rewards under history-dependent policy is NL-complete (cf. Theorem 3.12, compare to PSPACE-complete in the previous paragraph).

All the results are summarized in the following table.[2] A † denotes that the result is obtained by straightforward modifications from proofs in [6]. Formal definitions follow in Section 2. In Section 3 we consider the complexity of MDPs with nonnegative rewards, in Section 4 that of MDPs with unrestricted rewards, and in Section 5 we consider non-approximability of optimal policies.

[1] We consider this information as deterministic. We do not consider the case where it is randomized.

[2] Note that all hardness results also hold for MDPs with randomized observation function.

Completeness and hardness results

policy	horizon	observation	succinct	reward	complexity	theorem
s/t/h	$\lvert\mathcal{S}\rvert$	full/null	–	+	NL-complete	3.2, 3.3
t/h	$\lvert\mathcal{S}\rvert$	partial	–	+	NL-complete	3.12
s	$\lvert\mathcal{S}\rvert$	partial	–	+	NP-complete	3.10
s	$\lvert\mathcal{S}\rvert$	null	–	$-/+$	in P and NL-hard	4.1
t/h	$\lvert\mathcal{S}\rvert$	null	–	$-/+$	NP-complete †	4.3
s/t/h	$\lvert\mathcal{S}\rvert$	full	–	$-/+$	P-complete †	4.6
s/t	$\lvert\mathcal{S}\rvert$	partial	–	$-/+$	NP-complete	4.10, 4.11
h	$\lvert\mathcal{S}\rvert$	partial	–	$-/+$	PSPACE-complete †	4.12
s	$\lvert\mathcal{S}\rvert$	null	n	+	in PSPACE and NP-hard	3.6
t/h	$\lvert\mathcal{S}\rvert$	null	n	+	PSPACE-complete	3.5
s/t/h	$\lvert\mathcal{S}\rvert$	full	n	+	PSPACE-complete	3.4
s	$\lvert\mathcal{S}\rvert$	partial	n	+	NEXP-complete	3.11
t/h	$\lvert\mathcal{S}\rvert$	partial	n	+	PSPACE-complete	3.14
s/t/h	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	full/null/part.	n	+	NP-complete	3.7,3.8, 3.9,3.13
s	$\lvert\mathcal{S}\rvert$	null	n	$-/+$	NP-hard † and in EXP	4.2
t/h	$\lvert\mathcal{S}\rvert$	null	n	$-/+$	NEXP-complete	4.4
s/t/h	$\lvert\mathcal{S}\rvert$	full	n	$-/+$	EXP-complete	4.7
s/t	$\lvert\mathcal{S}\rvert$	partial	n	$-/+$	NEXP-complete	4.13, 4.14
h	$\lvert\mathcal{S}\rvert$	partial	n	$-/+$	EXPSPACE-complete †	4.15
s/t/h	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	null	$\log n$	$-/+$	in PSPACE and NP-hard	4.5
s/t	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	full	n	$-/+$	in EXP and PSPACE-hard	4.8
h	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	full	$\log n$	$-/+$	PSPACE-complete	4.9
s	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	partial	n	$-/+$	NEXP-complete	4.16
t	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	partial	n	$-/+$	PSPACE-hard	4.17
h	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	partial	$\log n$	$-/+$	PSPACE-complete	4.18

Nonapproximability results

s	$\lvert\mathcal{S}\rvert$	partial	–	+	nonapproximable	5.1
s	$\lvert\mathcal{S}\rvert$	partial	–	$-/+$	nonapproximable	5.2
s/t/h	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	full/null/part.	$\log n$	+	nonapproximable	5.3
s/t/h	$\lceil\log\lvert\mathcal{S}\rvert\rceil$	full/null/part.	$\log n$	$-/+$	nonapproximable	5.4

2 Definitions and Preliminaries

For definitions of complexity classes, reductions, and standard results from complexity theory we refer to [5].

2.1 Markov Decision Processes

A Markov decision process (MDP) describes a world by its states and the consequences of actions on the world. It is denoted as a tuple $M = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$, where

- $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{O}$ are finite sets of *states*, *actions* and *observations*,
- $s_0 \in \mathcal{S}$ is the *initial state*,
- $t : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the *state transition function*, where $t(s, a, s')$ is the probability that state s' is reached from state s on action a (where $\Sigma_{s' \in \mathcal{S}} t(s, a, s') \in \{0, 1\}$ for every $s \in \mathcal{S}, a \in \mathcal{A}$),
- $o : \mathcal{S} \to \mathcal{O}$ is the *observation function*, where $o(s)$ is the observation made in state s,

- $r : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$ is the *reward function*, where $r(s, a)$ is the reward gained by taking action a in state s; $\mathbf{R}$ is the set of real numbers.

If states and observations are identical, i.e. $\mathcal{O} = \mathcal{S}$ and o is the identity function (or a bijection), then the MDP is called *fully-observable*. Otherwise it is called *(deterministically) partially-observable*.[3] Another special case is *unobservable MDPs*, where the set of observations contains only one element, i.e. in every state the same observation is made and therefore the observation function is constant. For simplicity of description we omit $\mathcal{O}$ and o if possible, and describe fully-observable or unobservable MDPs as tuples $(\mathcal{S}, s_0, \mathcal{A}, t, r)$.

2.2 Policies and Performances

A policy describes how the agent acts depending on his observations. Because each action may change the state of the world, we need a notion of a sequence of states through which the world develops.

Let $M = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$ be an MDP. A *trajectory θ for M* is a finite sequence of states $\theta = \sigma_1, \sigma_2, \ldots, \sigma_m$ ($m \geq 0$, $\sigma_i \in \mathcal{S}$) with $\sigma_1 = s_0$. Let $\theta[i]$ denote the prefix $\sigma_1, \sigma_2, \ldots, \sigma_i$ of θ, and $|\theta| = m - 1$, i.e. the number of transitions in the trajectory. The set of all trajectories of length k is denoted $\mathcal{S}^k$, and $\mathcal{S}^* = \bigcup_k \mathcal{S}^k$. $\mathcal{O}^*$ is defined similarly. In the following let $\theta = \sigma_1, \sigma_2, \ldots, \sigma_m$.

A *stationary policy π_s* (for M) is a function $\pi_s : \mathcal{O} \to \mathcal{A}$, mapping each observation ω to an action $\pi(\omega)$. The trajectory θ is *consistent with a stationary policy π_s*, if $t(\sigma_i, \pi_s(o(\sigma_i)), \sigma_{i+1}) > 0$ for every i, $1 \leq i \leq |\theta|$.

A *time-dependent policy π_t* is a function $\pi_t : \mathcal{O} \times \mathbf{N} \to \mathcal{A}$, mapping each pair $\langle$observation, time$\rangle$ to an action. Trajectory θ is *consistent with a time-dependent policy π_t* if $t(\sigma_i, \pi_t(o(\sigma_i), i), \sigma_{i+1}) > 0$ for every i, $1 \leq i \leq |\theta|$.

A *history-dependent policy π_h* is a function $\pi_h : \mathcal{O}^* \to \mathcal{A}$, mapping each finite sequence of observations to an action. The trajectory θ is *consistent with a history-dependent policy π_h* if $t(\sigma_i, \pi_h(o(\sigma_1), \ldots, o(\sigma_i)), \sigma_{i+1}) > 0$ for every i, $1 \leq i \leq |\theta|$.

Note that every policy can be defined as a function from $\mathcal{O}^*$ to $\mathcal{A}$.

The "quality" of a policy for a Markov decision process is measured by the expected rewards which are gained by an agent following the policy. The *value* of a trajectory $\theta = \sigma_1, \sigma_2, \ldots, \sigma_m$ for a policy π (for an MDP $M = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$) is $V_\pi(\theta) = \Sigma_{i=1}^{|\theta|} r(\sigma_i, a_i)$, where a_i is the action chosen by π on the observations on $\theta[i]$ (depending on what kind of policy π is).

The *performance of a policy* π for finite-horizon k is the expected sum of rewards received on the next k steps by following the policy π, i.e. $E_{\theta \in \Theta(\pi,k)}[V_\pi(\theta)]$ where $\Theta(\pi, k)$ denotes the set of all trajectories of length k which are consistent with π.

2.3 Decision problems

Because we are interested in the maximal performance of any policy, the decision problem we are considering asks whether there exists a policy with performance > 0 for a given finite horizon. Using binary search, one can compute the maximal or minimal performance using this decision problem as a subroutine.

Given an MDP, a policy, and a horizon k, the performance of the policy can be computed in time polynomial in the number of states of the MDP, the size of the policy, and k. For stationary and time-dependent policies this yields a time bound that is polynomial in the size of the MDP for the computation of the performance, if the horizon is bounded by a polynomial in the number of the states of the MDP.

If the horizon is exponential in the number of states, no sub-exponential algorithm is known to compute the performance of a policy, but hardness results for exponential time classes do not seem to be achievable because of the restricted expressive power of MDPs with a relatively small number of states. Using succinctly described MDPs, we can fill this gap.

Because a history-dependent policy for an MDP with state set $\mathcal{S}$ with a horizon N may have a description of length $O(|\mathcal{S}|^N)$, one gets an upper time bound which is exponential in the size of the MDP for the performance.

For simplicity, we assume that the size of an MDP is determined by the size $|\mathcal{S}|$ of its state space. We assume that there are no more actions than states, and that each state transition probability is given as

[3] We don't consider MDPs having a probabilistic observation function for most of this paper.

binary fraction with $|\mathcal{S}|$ bits and each reward is an integer of $|\mathcal{S}|$ bits. This is no real restriction, since adding unreachable "dummy" states allows one to use more bits for transition probabilities and rewards. Also, it is straightforward to transform an MDP M with non-integer rewards to M' with integer rewards such that the performance of M is c times the performance of M', for a constant c and every policy.

To encode an MDP one can use the "natural" encoding of functions by tables. Thus, the description of an MDP with n states and n actions requires $O(n^4)$ bits. For MDPs with sufficient structure, we can use the concept of *succinct representations* introduced by Papadimitriou and Yannakakis [7]. In this case, the transition table of an MDP with n states and actions is represented by a Boolean circuit C with $4\lceil\log n\rceil$ input bits such that $C(s,a,s',l)$ outputs the l-th bit of $t(s,a,s')$. Encodings of those circuits are no larger than "natural" encodings, and may be much smaller, namely of size $O(\log n)$. A further special case of "small" MDPs are those with n states where the transition probabilities and rewards need only $\log n$ bits to be stored. They can be represented by circuits as above which have only $3\lceil\log n\rceil + \lceil\log\log n\rceil$ input bits.

To encode an MDP one can use the standard encoding of functions by tables. We assume that each entry in such a table has n bits, where n is the number of states of the MDP. To take advantage of the structure of an MDP in order to store it using less space, we use the concept of *succinct representations* introduced by Papadimitriou and Yannakakis [7]. When

We are now ready to define formally our decision problems.

The $\alpha\beta\gamma\mathcal{M}$ *Markov decision process problem* $(\alpha\beta\gamma\mathcal{M})$ is the set of all MDPs M with performance > 0, where

- α is the type of policy in { *stationary, time-dependent, history-dependent* },
- β is the length of the horizon in { $|\mathcal{S}|$*-horizon,* $\log|\mathcal{S}|$*-horizon* },
- γ is the type of observation in { *fully-observable, partially-observable, unobservable* },
- for $\mathcal{M}$ we use MDPP if the MDP is in standard encoding, sMDPP for succinctly encoded instances, and $s_{\log}$MDPP for succinctly encoded instances where each transition probability and reward takes $\log|\mathcal{S}|$ many bits.

3 MDPs with nonnegative rewards

For MDPs with nonnegative rewards (i.e. the reward function r maps states to real numbers ≥ 0) the MDP problem simplifies greatly. Because negative rewards do not exist, the expected reward for a policy π is > 0 if and only if at least one trajectory $\sigma_0, \ldots, \sigma_m$ consistent with π exists for which (1) the action chosen by π for σ_m yields a reward > 0, and (2) no state appears twice in $\sigma_0, \ldots, \sigma_m$. The latter bounds the length of the trajectories to be considered by the number of the states of the MDP.

3.1 Fully-observable and unobservable MDPs with nonnegative rewards

Theorem 3.1 *The stationary $|\mathcal{S}|$-horizon fully-observable* MDPP *with nonnegative rewards is* NL*-complete.*

Proof Following the above observations, a policy for the MDP can be guessed "on-line" (this means, an action is guessed when needed) and doesn't need to be stored. This yields the following algorithm, showing that the problem is in NL.

```
input M, M = (S, s_0, A, t, r)
s := s_0
for i := 1 to |S| do
  guess a (* determines the policy for the next step *)
  if r(s,a) > 0 then accept end
  guess s' ∈ S
  if t(s,a,s') = 0 then reject else s := s' end
end
reject
```

To sketch the correctness of the algorithm, assume that there exists a policy π with performance > 0 for M. Then there exists a trajectory $\theta = \sigma_0, \ldots, \sigma_i, \ldots, \sigma_m$ for π where $r(\sigma_m, \pi(\sigma_m)) > 0$. Obtain θ' by cutting all cycles from θ. Then $\theta' = \sigma_0, \ldots, \sigma_{i'}, \ldots, \sigma_m$ is a trajectory of length $\leq |\mathcal{S}|$ for π which also has value > 0. Because no state appears twice in θ', it follows that there is a computation of the above algorithm which in the i-th repetition of the for-loop guesses $a = \pi(\sigma_{i-1})$ and $s' = \sigma_i$ and therefore accepts in the $|\theta'|$-th repetition.

On the other hand, let M be an MDP which is accepted in the m-th repetition of the loop, where a_i and s'_i are the values of the variables a and s' guessed in the i-repetition of the loop. Then π with $\pi(s) = a_{j(s)}$ for $j(s) = \max_i[s = \sigma_i]$ is a policy with positive performance.

To store the values of the variables i, a, s, and s' takes logarithmic space.

To show NL-hardness, we show that the NL-complete problem REACHABILITY logspace reduces to the MDPP. The problem REACHABILITY consists of directed graphs G with set of nodes $\{1, 2, \ldots, k\}$ (for any k) which contain a path from node 1 to node k.

Let $G = (V, E)$ be a graph with set of nodes $V = \{1, 2, \ldots, k\}$ and set of edges E. For every $u \in V$, let $d(u)$ be its outdegree: $d(u) = |\{v \in V \mid (u, v) \in E\}|$. We construct an MDP $M(G) = (V, 1, \{a\}, t, r)$ with

$$t(u, a, v) = \begin{cases} \frac{1}{d(u)}, & \text{if } (u, v) \in E \\ 0, & \text{otherwise} \end{cases}, \qquad r(u, a) = \begin{cases} 1, & \text{if } (u, k) \in E \\ 0, & \text{otherwise} \end{cases}$$

From the above description it is clear that $M(G)$ can be computed from G in logarithmic space.

We now show the correctness of the reduction. If $u_1, u_2, \ldots, u_m$ is a path from 1 to k in G, then $u_1, u_2, \ldots, u_m$ is a trajectory for the only possible policy $\pi = a$ for $M(G)$, and this trajectory has value 1. On the other hand, if $u_1, u_2, \ldots, u_m$ is a trajectory for π with value > 0, then for some $i < m$, $(u_i, k) \in E$ and therefore $u_1, u_2, \ldots, u_i, k$ is a path from 1 to k in G. ■

Note that the reduction in the above proof maps graphs to MDPs which have only one possible action. Therefore, there exists only one "constant" policy $\pi = a$ for this MDP, and thus, the hardness proof also holds for time-dependent or history-dependent policies as well as for unobservable MDPPs. Also, the decision algorithm remains the same for time-dependent or history-dependent policies. This yields the following theorem.

Theorem 3.2 *The $|\mathcal{S}|$-horizon fully-observable* MDPP *with nonnegative rewards is* NL*-complete, for stationary, time-dependent, or history-dependent policies.*

The algorithm given in the proof of Theorem 3.1 for the stationary fully-observable MDPP can be easily transformed to an algorithm for the stationary unobservable MDPP, by once guessing the action which determines the full policy at the beginning. This yields the same complexity results for the unobservable case.

Theorem 3.3 *The $|\mathcal{S}|$-horizon unobservable* MDPP *with nonnegative rewards is* NL*-complete, for stationary, time-dependent, or history-dependent policies.*

The *s*MDPPs – the decision problems for succinctly represented MDPs – can be shown to be in PSPACE by similar arguments as above. Note that in order to check whether $r(s, a) > 0$ or $t(s, a, s') > 0$, it suffices to check whether at least one bit of these values is 1. There is no need to write down the whole numbers, which would exceed the allowed space usage.

To show their PSPACE-hardness, one could try to translate the logspace reduction in the proof of Theorem 3.1 from REACHABILITY to the MDPP to a polynomial-time reduction from the reachability problem for succinctly represented graphs to the *s*MDPP. But because the degree of a node in a succinctly represented graph is not computable in polynomial time, this will not work. Fortunately, a slight change in $M(G)$ solves the problem.

Theorem 3.4 *The $|\mathcal{S}|$-horizon fully-observable s*MDPP *with nonnegative rewards is* PSPACE*-complete, for stationary, time-dependent, or history-dependent policies.*

Proof Containment in PSPACE follows using the same algorithm as in the proof of Theorem 3.1. But here, to store the variables may need memory of the size of the input. Therefore the algorithm runs nondeterministically in polynomial space, which shows that the decision problem is in NPSPACE (= PSPACE). To show hardness, we sketch the reduction from the succinct representation of REACHABILITY. Define $M'(G) = (V, 1, V, t, r)$ to be an MDP with

$$t(u, a, v) = \begin{cases} 1, & \text{if } v = a, (u, v) \in E \\ 0, & \text{otherwise} \end{cases}$$

$$r(u, a) = \begin{cases} 1, & \text{if } a = k, (u, k) \in E \\ 0, & \text{otherwise} \end{cases}$$

Note that now the action determines the next node on the path. A similar argument as in the proof of Theorem 3.1 shows the correctness of the reduction. Also, by the description of $M'(G)$ it follows that its succinct description can be computed in polynomial time from the succinct description of G. ■

With a similar argument we can prove the same complexity for the unobservable case, if the policy is not stationary.

Theorem 3.5 *The time-dependent or history-dependent $|\mathcal{S}|$-horizon unobservable sMDPP with nonnegative rewards is* PSPACE*-complete.*

For stationary policies we can only prove an upper bound by similar arguments as above. A lower bound better than NP (from Theorem 3.10) is not known. (Compare this to Theorem 4.1.) Because the number of states may be exponential in the size of the succinct description of the MDP, we cannot show that the problem is contained in NP.

Theorem 3.6 *The stationary $|\mathcal{S}|$-horizon unobservable sMDPP with nonnegative rewards is* NP*-hard and in* PSPACE*.*

The complexity of succinct MDPPs with logarithmic horizon lies between the complexity of MDPPs and succinct MDPPs with horizon $|\mathcal{S}|$.

Theorem 3.7 *The stationary $\lceil \log |\mathcal{S}| \rceil$-horizon fully-observable sMDPP with nonnegative rewards is* NP*-complete.*

Proof The following algorithm shows that the problem is in NP. It guesses a policy for $\lceil \log |\mathcal{S}| \rceil$ observations, then guesses step by step a trajectory consistent with that policy and checks if the trajectory has value > 0.

```
input M, M = (S, s_0, A, t, r)
Π := ∅ (* used to store the guessed policy *)
for i := 1 to ⌈log |S|⌉ do
  guess (s, a) where s ∈ S − {s' | (s', b) ∈ Π for some b}, a ∈ A
  Π := Π ∪ {(s, a)}
end
s := s_0
for i := 1 to ⌈log |S|⌉ do
  if (s, a) ∉ Π for some a ∈ A
    then reject
    else if r(s, a) > 0 then accept end
      guess s' ∈ S
      if t(s, a, s') = 0 then reject else s := s' end
  end
end
reject
```

The correctness of the algorithm is not hard to see. Because the size of the input is at least $\log|\mathcal{S}|$, it follows that the considered problem is in NP.

To show NP-completeness, we use a polynomial time reduction from HAMILTONIAN-CIRCUIT.

Let $G = (V, E)$ be a graph, where $V = \{1, 2, \ldots, k\}$. We define an unobservable MDP $M(G) = (\mathcal{S}, s_0, \mathcal{A}, t, r)$ which simulates the guess and check method to decide an NP-problem: the states represent sequences of nodes, and a reward is gained if this sequence is a cycle through all nodes.

$$
\begin{aligned}
\mathcal{S} &= \{\langle u_1, \ldots, u_m\rangle \mid u_1, \ldots, u_m \in V, 1 \leq m \leq k+1\}\\
s_0 &= \langle 1\rangle\\
\mathcal{A} &= \{a\}\\
t(\langle u_1, \ldots, u_m\rangle, a, \langle v_1, \ldots, v_{m'}\rangle) & \\
&= \begin{cases} \frac{1}{|V|}, & \text{if } \langle u_1, \ldots, u_m\rangle = \langle v_1, \ldots, v_m\rangle, m' = m+1\\ 0, & \text{otherwise}\end{cases}\\
r(\langle u_1, \ldots, u_m\rangle, a) &= \begin{cases} 1, & \text{if } \{u_1, \ldots, u_m\} = V, (u_i, u_{i+1}) \in E, 1 \leq i < m, (u_m, u_1) \in E\\ 0, & \text{otherwise.}\end{cases}
\end{aligned}
$$

Note that $|\mathcal{S}|$ is exponential in $|V|$. If G has a Hamiltonian circuit $u_1, \ldots, u_k$ then $\langle u_1\rangle, \ldots, \langle u_1, \ldots, u_k\rangle$ is a trajectory of length $\leq \lceil\log|\mathcal{S}|\rceil$ for $\pi = a$ with value 1, and therefore π has performance > 0. On the other hand, if $\langle u_1\rangle, \ldots, \langle u_1, \ldots, u_k\rangle, \langle u_1, \ldots, u_k, u_{k+1}\rangle$ is a trajectory with performance 1 for $\pi = a$, then it follows from the definition of t and r that $u_1, \ldots, u_k, u_1$ is a Hamiltonian circuit for G. ■

The above reduction maps to MDPs having only one action. Therefore the reduction also works for time-dependent and history-dependent policies, and for the unobservable MDPP.

Theorem 3.8 *The $\lceil\log|\mathcal{S}|\rceil$-horizon fully-observable s*MDPP *with nonnegative rewards is* NP*-complete, for stationary, time-dependent, or history-dependent policies.*

Proof Hardness for NP can be shown using the same reduction function as in the proof of Theorem 3.7. Because a trajectory of length $\leq \lceil\log|\mathcal{S}|\rceil$ with positive reward can be guessed and checked in polynomial time (remember that the input has length $\geq \log|\mathcal{S}|$), the problem is in NP. ■

Finally, similar arguments yield the same complexity for unobserved succinctly represented MDPs.

Theorem 3.9 *The $\lceil\log|\mathcal{S}|\rceil$-horizon unobservable s*MDPP *with nonnegative rewards is* NP*-complete, for stationary, time-dependent, or history-dependent policies.*

3.2 Partially-observable MDPs with nonnegative rewards

The decision problem for partially-observable MDPs is harder than for fully-observable MDPs. Informally, the reason is that observations can be used to store information. Whenever the same observation is made, the stationary policy must take the same action.

Theorem 3.10 *The stationary $|\mathcal{S}|$-horizon partially-observable* MDPP *with nonnegative rewards is* NP*-complete.*

Proof To show that the problem is in NP, guess a policy and a trajectory with value > 0, and then check whether the trajectory is consistent to π. Since the same observation can be made in different states which can appear in the trajectory, this computation cannot be performed in logarithmic space as in the unobservable and fully-observable cases, unless NP = NL. The following algorithm performs that strategy.

input M, $M = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$
$\Pi := \emptyset$ (* used to store the guessed policy *)
for all $b \in \mathcal{O}$ **do**
  guess $a \in \mathcal{A}$
  $\Pi := \Pi \cup \{(b, a)\}$
end
$s := s_0$

for $i := 1$ **to** $|\mathcal{S}|$ **do**
  find a such that $(o(s), a) \in \Pi$
  if $r(s,a) > 0$ **then** *accept* **end**
  guess $s' \in \mathcal{S}$
  if $t(s,a,s') = 0$ **then** *reject* **else** $s := s'$ **end**
end
reject

It is not hard to see that this nondeterministic algorithm accepts an input M if and only if there exists a trajectory with positive value and that is consistent with some stationary policy. It is also clear that this algorithm runs in time polynomial in the size of the input. Thus, the considered MDPP is shown to be in NP.

To show NP-hardness, we polynomial time reduce the NP-complete satisfiability problem 3SAT to it. Let $\phi(x_1, \ldots, x_n)$ be such a formula with variables $x_1, \ldots, x_n$, clauses $C_1, \ldots, C_m$, where clause $C_j = (l_{v(1,j)} \vee l_{v(2,j)} \vee l_{v(3,j)})$ for $l_i \in \{x_i, \neg x_i\}$. We say that variable x_i *appears in* C_j *with signum* 0 *(resp.* 1*)*, if $\neg x_i$ (resp. x_i) is contained in C_j. W.l.o.g. we assume that every variable appears at most once in each clause.

From ϕ, we construct a partially-observable MDP $M(\phi) = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$ with

$$\begin{array}{rcl}
\mathcal{S} & = & \{(i,j) \mid 1 \le i \le n, 1 \le j \le m\} \cup \{F, T\} \\
s_0 & = & (v(1,1),1), \qquad \mathcal{A} = \{0,1\}, \qquad \mathcal{O} = \{x_1, \ldots, x_n, F, T\}
\end{array}$$

$$t(s,a,s') = \begin{cases}
1, & \text{if } s = (v(i,j),j), s' = (1,j+1), j < m, 1 \le i \le 3, \\
 & \qquad \text{and } x_{v(i,j)} \text{ appears in } C_j \text{ with signum } a \\
1, & \text{if } s = (v(i,m),m), s' = T, 1 \le i \le 3, \\
 & \qquad \text{and } x_{v(i,m)} \text{ appears in } C_m \text{ with signum } a \\
1, & \text{if } s = (v(i,j),j), s' = (v(i+1,j),j), 1 \le i < 3, \\
 & \qquad \text{and } x_{v(i,j)} \text{ appears in } C_j \text{ with signum } 1-a \\
1, & \text{if } s = (v(3,j),j), s' = F, \\
 & \qquad \text{and } x_{v(3,j)} \text{ appears in } C_j \text{ with signum } 1-a \\
1, & \text{if } s = s' = F \text{ or } s = s' = T \\
0, & \text{otherwise}
\end{cases}$$

$$r(s,a) = \begin{cases} 1, & \text{if } t(s,a,T) = 1, s \neq T \\ 0, & \text{otherwise} \end{cases}, \qquad o(s) = \begin{cases} x_i, & \text{if } s = (i,j) \\ T, & \text{if } s = T \\ F, & \text{if } s = F \end{cases}.$$

Note that every trajectory has value 0 or 1.

Claim 1 *If $\phi(b_1, \ldots, b_n)$ is true for an assignment $b_1 \cdots b_n \in \{0,1\}$, then there exists a trajectory of length $\le |\mathcal{S}|$ for the policy π with $\pi(x_i) = b_i$ for $M(\phi)$ with value* 1.

Let i_j be the smallest i such that $x_{v(i,j)}$ appears in C_j with signum b_i, for $j = 1, 2, \ldots, m$. Since $\phi(b_1, \ldots, b_n)$ is true, such an i_j exists for every C_j. Then,

$$(v(1,1),1), \ldots, (v(i_1,1),1), (v(1,2),2), \ldots, (v(i_2,2),2), \cdots, (v(1,m),m), \ldots, (v(i_m,m),m), T$$

is a trajectory for π as claimed. Because $t((v(i_m,m),m), \pi(x_m), T) = 1$, this trajectory has value 1. □

Claim 2 *If a trajectory for a policy π for $M(\phi)$ has value* 1*, then $\phi(\pi(x_1), \ldots, \pi(x_n))$ is true.*

Let θ be a trajectory for π for $M(\phi)$ with value 1. Then by the definition of $M(\phi)$, θ has the form

$$(v(1,1),1), \ldots, (v(i_1,1),1), \cdots, (v(1,m),m), \ldots, (v(i_m,m),m), T, \ldots, T$$

Therefore, $x_{v(i_j,j)}$ appears in C_j with signum $\pi(x_{i_j})$ for every $j = 1, 2, \ldots, m$. This means that every clause C_j contains a literal that is satisfied by the assignment $\pi(x_1) \cdots \pi(x_n)$, and therefore $\pi(x_1) \cdots \pi(x_n)$ is a satisfying assignment for ϕ. □

From the above claims follows the correctness of the reduction. By the description of the reduction it also follows that $M(\phi)$ is computable from ϕ in polynomial time, which completes the proof. ■

Because succinctly represented 3SAT is NEXP-complete [7], the above proof translates to NEXP.

Theorem 3.11 *The stationary $|\mathcal{S}|$-horizon partially-observable s*MDPP *with nonnegative rewards is* NEXP*-complete.*

A time-dependent or history-dependent policy may choose different actions for different appearances of the same state in a trajectory. Therefore the complexity of the respective decision problems is smaller.

Theorem 3.12 *The time-dependent or history-dependent $|\mathcal{S}|$-horizon partially-observable* MDPP *with nonnegative rewards is* NL*-complete.*

Proof Containment in NL can be shown using the same algorithm as in the proof of Theorem 3.1. NL-hardness follows from Theorem 3.3, because unobservable MDPs are partially observable. ■

Theorem 3.13 *The stationary, time-dependent or history-dependent $\lceil \log |\mathcal{S}| \rceil$-horizon partially-observable s*MDPP *with nonnegative rewards is* NP*-complete.*

Proof Containment in NP can be shown using a similar algorithm as in the proof of Theorem 3.10, where instead of guessing a policy Π for every observation, a policy for only $\lceil \log |\mathcal{S}| \rceil$ many observations is guessed. (More observations cannot be made on a trajectory of that length.) NP-hardness follows from the hardness of the fully-observable *s*MDPP (shown in Theorem 3.8), which is a special case of the partially-observable case. ■

Theorem 3.14 *The time- or history-dependent $|\mathcal{S}|$-horizon partially-observable s*MDPP *with nonnegative rewards is* PSPACE*-complete.*

Proof The same algorithm as in the proof of Theorem 3.1 decides the problem. But because the input is succinctly represented, the algorithm needs polynomial space instead of logarithmic space as in the proof of Theorem 3.1. Hardness for PSPACE follows from Theorem 3.5. ■

4 MDPs with unrestricted rewards

To decide the performance of a policy for an MDP with positive and negative rewards, it is not sufficient to check only one trajectory for that policy as in the case of MDPs with nonnegative rewards. Instead, a full "tree" of trajectories has to be evaulated. It seems that this increases the complexity of the respective decision problems.

4.1 Unobservable MDPs

Unfortunately, we cannot prove completeness of the stationary finite-horizon unobservable MDPP. (Compare to MDPs with nonnegative rewards, Theorem 3.6.) One reason for this may be that the performance of a given policy is computable in a parallel manner. Because there are few possible policies to check in the unobservable case, the whole process can be parallelized. Therefore we conjecture that this problem is in NC_2.

Theorem 4.1 *The stationary $|\mathcal{S}|$-horizon unobservable* MDPP *is in* P *and is* NL*-hard.*

Proof Compute the performance of every policy and accept if and only if one of these is > 0. Because there are only as many policies as actions in the MDP and each performance can be computed in time polynomial in the size of the MDP, this takes time polynomial in the size of the MDP. Hardness follows from Theorem 3.3. ■

With a similar argument as above, the following is shown.

Theorem 4.2 *The stationary $|\mathcal{S}|$-horizon unobservable s*MDPP *is* NP*-hard and is in* EXP.

Proof NP-hardness follows from Theorem 3.6. Note that we now have to solve an MDP which may have exponentially many states in the size of its description. Therefore, an algorithm as in the proof of Theorem 4.1 takes exponential time. ■

For the time-dependent case we can prove NP-completeness.

Theorem 4.3 *The time-dependent or history-dependent $|\mathcal{S}|$-horizon unobservable* MDPP *is* NP*-complete.*

Proof Because history-dependent policies are also time-dependent for unobservable MDPs, we only need to consider the time-dependent case. That it is in NP follows from the fact that a policy with performance > 0 can be guessed and checked in polynomial time. NP-hardness follows from the following reduction from 3SAT. In the proof of Theorem 3.10, we constructed from a formula an MDP which searches through the literals of every clause, one clause after another. Here, we can search in parallel through every clause, independent of whether the variable appears in that clause. At the first step, a clause is chosen randomly. At step $i+1$, the assignment of variable i is determined. If a clause was satisfied by this assignment, it will gain reward 1, if not, the reward will be $-m$, where m is the number of clauses of the formula. Therefore, if all clauses are satisfied, the expected value will be positive, otherwise negative.

We formally define the reduction. Let ϕ be a formula with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Define the unobservable MDP $M(\phi) = (\mathcal{S}, s_0, \mathcal{A}, t, r)$ where

$$
\begin{aligned}
\mathcal{S} &= \{(i,j) \mid 1 \le i \le n, 1 \le j \le m\} \cup \{s_0, T, F\} \\
\mathcal{A} &= \{0,1\} \\
t(s,a,s') &= \begin{cases} \frac{1}{m}, & \text{if } s = s_0, a = 0, s' = (1,j), 1 \le j \le m \\ 1, & \text{if } s = (i,j), s' = T,\ x_i \text{ appears in } C_j \text{ with signum } a \\ 1, & \text{if } s = (i,j), s' = (i+1,j), i < n,\ x_i \text{ doesn't appear in } C_j \text{ with signum } a \\ 1, & \text{if } s = (n,j), s' = F,\ x_n \text{ doesn't appear in } C_j \text{ with signum } a \\ 1, & \text{if } s = s' = F \text{ or } s = s' = T, a = 0 \\ 0, & \text{otherwise} \end{cases} \\
r(s,a) &= \begin{cases} 1, & \text{if } t(s,a,T) > 0 \text{ and } s \ne T \\ -m, & \text{if } t(s,a,F) > 0 \text{ and } s \ne F \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
$$

By this description it is clear that $M(\phi)$ can be computed from ϕ in time polynomial in $|\phi|$.

Note that a time-dependent policy for an unobservable MDP is a function mapping natural numbers to actions.

Claim 3 *If $\phi(b_1, \ldots, b_n)$ is true for $b_1, \ldots, b_n \in \{0,1\}$, then every trajectory of length $\le n+1$ for the policy π_t for $M(\phi)$ with $\pi_t(0) = 0$ and $\pi_t(i) = b_i$ has value 1.*

Every trajectory for π_t searches through the literals of one clause until it finds one which is satisfied by the action chosen by the policy. Because the policy is determined by a satisfying assignment, such a literal will be found. Therefore, a transition to the state T will be made, which yields reward 1. □

Claim 4 *Let π_t be a policy for $M(\phi)$. If every trajectory of length $\le n+1$ for π_t has value 1, then $\pi_t(1), \ldots, \pi_t(n)$ is a satisfying assignment for ϕ.*

Let θ be any trajectory for π_t for $M(\phi)$ with value 1. Then by the definition of $M(\phi)$, θ has the form

$$s_0, (1,j), \ldots, (i,j), T, \ldots, T$$

Therefore, x_i appears in C_j with signum $\pi_t(i)$. Since for every $j = 1, 2, \ldots, m$ there exists such an i (otherwise θ couldn't have value 1), every clause C_j contains a literal that is satisfied by the assignment $\pi_t(1) \cdots \pi_t(n)$. □

There are m different trajectories of length $|\mathcal{S}|$ for any policy π_t for $M(\phi)$. By the above claims it follows that for some policy the expected value of every trajectory is 1, if and only if ϕ is satisfiable. If ϕ is not satisfiable, then at least one of the trajectories has value $-m$, and therefore the expected value is at most $\frac{(m-1)-m}{m}$, which is negative. ■

The proof of Theorem 4.3 can be translated for succinctly represented MDPs.

Theorem 4.4 *The time-dependent or history-dependent $|\mathcal{S}|$-horizon unobservable s*MDPP *is* NEXP*-complete.*

For the "intermediate" horizon we again are not able to prove completeness, even though we restrict the rewards and transition probabilities to be represented by $\log|\mathcal{S}|$ many bits. Note that this restriction is essential to show that the problem is in PSPACE.

Theorem 4.5 *The stationary, time-dependent or history-dependent $\lceil\log|\mathcal{S}|\rceil$-horizon unobservable $s_{\log}$*MDPP *is in* PSPACE *and is* NP*-hard.*

4.2 Fully-observable MDPs

How to compute optimal policies for MDPs has been a very central and well-solved optimization problem. The maximal performance of any stationary policy for a fully-observable Markov decision process can be solved by linear or dynamic programming techniques in polynomial time.[4] Furthermore, it is known that for these MDPs the maximal performance of all history-dependent or time-dependent policies is also yielded by a stationary policy (see e.g. [8] for an overview). The related MDPPs can be shown to be complete for P. Because the proof is a straightforward modification of a proof in [6], we omit it here (the interested reader can find it in the Appendix).

Theorem 4.6 *The $|\mathcal{S}|$-horizon fully-observable* MDPP *is* P*-complete for stationary, time-dependent or history-dependent policies.*

The proof also translates to succinctly represented circuits.

Theorem 4.7 *The $|\mathcal{S}|$-horizon fully-observable s*MDPP *is* EXP*-complete for stationary, time-dependent or history-dependent policies.*

Again, we get an intermediate complexity for intermediate sMDPPs.

Theorem 4.8 *The stationary or time-dependent $\lceil\log|\mathcal{S}|\rceil$-horizon fully-observable s*MDPP *is* PSPACE*-hard and in* EXP.

Proof We consider the case for stationary policies at first. Containment in EXP follows from Theorem 4.7. To prove hardness, we show a polynomial time reduction from QBF, the validity problem for quantified Boolean formulae.

Informally, from a formula Φ with n variables we construct an MDP with $2^{n+1}-1$ states, where every state represents an assignment of Boolean values to the first i variables ($0 \le i \le n$) of Φ. Transitions from state s can reach the two states representing the same assignments as s and an assignment to the next unassigned variable. If this variable is bound by an existential quantifier, then the action taken in s assigns a value to that variable; otherwise the transition is random and independent on the action. Reward 1 is gained for every action after a state representing a satisfying assignment for the formula is reached. If a state representing an unsatisfying assignment is reached, reward $-(2^n)$ is gained.

Formally, let $\Phi = Q_1x_1Q_2x_2\cdots Q_nx_n\phi(x_1,\ldots,x_n)$ be a quantified Boolean formula with quantifier-free matrix $\phi(x_1,\ldots,x_n)$. We construct an instance $M(\Phi)$ of the fully-observable sMDPP where $M(\Phi) = (\mathcal{S}, s_0, \mathcal{A}, t, r)$ with

$$\begin{array}{rcl}
\mathcal{S} & = & \{\{0,1\}^i \mid 0 \le i \le n\} \\
s_0 & = & \varepsilon \\
\mathcal{A} & = & \{0,1\} \\
t(s,a,s') & = & \begin{cases} 1, & \text{if } |s| = i, Q_{i+1} = \exists, sa = s' \\ \frac{1}{2}, & \text{if } |s| = i, Q_{i+1} = \forall, a = 0, s0 = s' \\ \frac{1}{2}, & \text{if } |s| = i, Q_{i+1} = \forall, a = 0, s1 = s' \\ 1, & \text{if } |s| = n, a = 0, s = s' \\ 0, & \text{otherwise} \end{cases} \\
r(s,a) & = & \begin{cases} 1, & \text{if } |s| = n \text{ and } \phi(s) \text{ is true} \\ -(2^n), & \text{if } |s| = n \text{ and } \phi(s) \text{ is false} \\ 0, & \text{otherwise} \end{cases}
\end{array}$$

[4] Note that this holds also for performances for infinite horizon, which are not considered in this paper.

From this description of $M(\Phi)$ it follows that a succinct description of it can be constructed in time polynomial in $|\Phi|$.

We prove the correctness of the reduction, i.e. we show that Φ is true iff some stationary policy for $M(\Phi)$ has performance > 0 with finite horizon $n+1$ $(= \lceil \log |\mathcal{S}| \rceil)$. Consider some Φ with n quantifiers, and let $M(\Phi) = (\mathcal{S}, s_0, \mathcal{A}, t, r)$ be obtained from Φ as described above. Note that every trajectory of length $n+1$ for any policy for $M(\Phi)$ has the form

$$\varepsilon, b_1, b_1 b_2, \ldots, b_1 b_2 \cdots b_{n-1}, b_1 b_2 \cdots b_n, b_1 b_2 \cdots b_n$$

(for $b_i \in \{0,1\}$). A reward is gained only from the last action in the trajectory. The trajectory has value 1 if $\phi(b_1 \cdots b_n)$ is true, and it has value $-(2^n)$ otherwise.

Claim 5 *If Φ is true, then there exists a policy π such that every trajectory of length $n+1$ for π has value 1.*

To prove the claim we proceed by induction on the number of quantifiers in a true formula Φ. If Φ has no quantifier, then ε, ε is the only trajectory of length 1 for any policy. Because Φ is true, $\phi(\varepsilon)$ is true, and therefore this trajectory has value 1.

For the induction step, consider $\Phi = Q_1 x_1 Q_2 x_2 \cdots Q_{i+1} x_{i+1} \phi(x_1, x_2, \ldots, x_{i+1})$ with $i+1$ quantifiers. If $Q_1 = \exists$, then for some $a \in \{0,1\}$ the formula $\Phi_a = Q_2 x_2 \cdots Q_{i+1} x_{i+1} \phi(a, x_2, \ldots, x_{i+1})$ is true. By the induction hypothesis, there exists a policy π_a for $M(\Phi_a)$ such that every trajectory $\theta = \sigma_1, \ldots, \sigma_{i+1}$ for π_a has value 1. Define a policy π for $M(\Phi)$ by $\pi(\varepsilon) = a$ and $\pi(as) = \pi_a(s)$. Then for every length $i+2$ trajectory θ for π there exists a trajectory $\theta' = \sigma_1, \ldots, \sigma_{i+1}$ for π_a such that $\theta = \varepsilon, a\sigma_1, \ldots, a\sigma_{i+1} (=: a\theta')$. Thus every trajectory for π has value 1.

If $Q_1 = \forall$, then by the induction hypothesis there exist policies π_0 for $M(\Phi_0)$ and π_1 for $M(\Phi_1)$ which fulfill the claim. Define π as $\pi(\varepsilon) = 0$, $\pi(0s) = \pi_0(s)$, and $\pi(1s) = \pi_1(s)$. Since for every length $i+2$ trajectory θ for π there exists either a trajectory θ_0 for π_0 or a trajectory θ_1 for π_1 such that $\theta = 0\theta_0$ or $\theta = 1\theta_1$, the claim follows. □

In a very similar way we can prove

Claim 6 *If Φ is false, then for every policy π there exists a trajectory of length $n+1$ with value $-(2^n)$.*

For every policy for $M(\Phi)$ there are at most 2^n trajectories of length $n+1$, and each such trajectory has either value 1 or $-(2^n)$. If Φ is true, then by Claim 5 there exists a policy with performance 1. If Φ is false, then by Claim 6 every policy has performance $\leq \frac{(2^n-1)-2^n}{2^n}$. Because the numerator is negative, the performance is also negative.

With the same construction we can prove also the same lower bound for the other types of policies, because every state excepted the last one appears at most once in every (consistent) trajectory. ■

The history-dependent $\log n$-succinct case can additionally be shown to be in PSPACE.

Theorem 4.9 *The history-dependent $\lceil \log |\mathcal{S}| \rceil$-horizon fully-observable $s_{\log}$*MDPP *is* PSPACE*-complete.*

Proof PSPACE-hardness was shown in the proof of Theorem 4.8. To show that the problem is in PSPACE, consider all possible trajectories of length $\lceil \log |\mathcal{S}| \rceil$ as a tree, where each node represents a state, and the sequence of nodes from the root to a node is the history of that node. Because every node has a unique history, every choice of actions yields a history-dependent policy. Thus we only need to guess an action for every node and to evaluate the respective subtree. This can be done in space at most square of the size of the input. Since PSPACE = NPSPACE, the theorem follows. ■

4.3 Partially-observable MDPs

Surprisingly, the complexity of the stationary partially-observable MDPP does not depend on whether the rewards are nonnegative or unrestricted. Partially-observable MDPs seem to obtain their expressive power by time- or history-dependent policies.

Theorem 4.10 *The stationary $|\mathcal{S}|$-horizon partially-observable* MDPP *is* NP*-complete.*

Proof That it is in NP follows from "guess a policy and check it." NP-hardness follows from the NP-hardness of the stationary $|\mathcal{S}|$-horizon partially-observable MDPP with nonnegative rewards (Theorem 3.10). ■

Theorem 4.11 *The time-dependent $|\mathcal{S}|$-horizon partially-observable* MDPP *is* NP*-complete.*

Proof Containment in NP follows from the standard guess-and-check argument. The NP-hardness of the unobservable case (Theorem 4.3) completes the proof. ■

Theorem 4.12 *The history-dependent $|\mathcal{S}|$-horizon partially-observable* MDPP *is* PSPACE*-complete.*

Proof We use a straightforward modification of the proof by Papadimitriou and Tsitsiklis [6, Theorem 6]. ■

All these proofs translate for succinct representations.

Theorem 4.13 *The stationary $|\mathcal{S}|$-horizon partially-observable s*MDPP *is* NEXP*-complete.*

Theorem 4.14 *The time-dependent $|\mathcal{S}|$-horizon partially-observable s*MDPP *is* NEXP*-complete.*

Theorem 4.15 *The history-dependent $|\mathcal{S}|$-horizon partially-observable s*MDPP *is* EXPSPACE*-complete.*

The intermediate horizon turns out to be more interesting. In fact, in the stationary case the same completeness as in the $|\mathcal{S}|$-horizon case holds for $\lceil \log |\mathcal{S}| \rceil$-horizon.

Theorem 4.16 *The stationary $\lceil \log |\mathcal{S}| \rceil$-horizon partially-observable s*MDPP *is* NEXP*-complete.*

Proof To show NEXP-hardness of the problem, we reduce the NEXP-complete problem of succinctly represented 3SAT to it. We change the technique from the proof of Theorem 3.10 by introducing parallel checking of the clauses (as in the proof of Theorem 4.3). Therefore, all trajectories will get their reward after at most four actions. In general, this is less than the $\lceil \log |\mathcal{S}| \rceil$-horizon allowed in the statement of this theorem.

Formally, from ϕ with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, we construct a partially-observable MDP $M(\phi) = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$ with

$$
\begin{aligned}
\mathcal{S} &= \{(i,j) \mid 1 \le i \le n, 1 \le j \le m\} \\
s_0 &= (v(1,1),1) \\
\mathcal{A} &= \{0,1\} \\
\mathcal{O} &= \{x_1, \ldots, x_n\} \\
t(s,a,s') &= \begin{cases} 1, & \text{if } s = (v(i,j),j), s' = T, 1 \le i \le 3, \\ & \quad \text{and } x_{v(i,j)} \text{ appears in } C_j \text{ with signum } a \\ 1, & \text{if } s = (v(i,j),j), s' = (v(i+1,j),j), 1 \le i < 3, \\ & \quad \text{and } x_{v(i,j)} \text{ appears in } C_j \text{ with signum } 1-a \\ 1, & \text{if } s = (v(3,j),j), s' = F, \\ & \quad \text{and } x_{v(3,j)} \text{ appears in } C_j \text{ with signum } 1-a \\ 1, & \text{if } s = s' = F \text{ or } s = s' = T, a = 0 \\ 0, & \text{otherwise} \end{cases} \\
o(s) &= \begin{cases} x_i, & \text{if } s = (i,j) \\ T, & \text{if } s = T \\ F, & \text{if } s = F \end{cases} \\
r(s,a) &= \begin{cases} 1, & \text{if } t(s,a,T) = 1, s \ne T \\ -m, & \text{if } t(s,a,F) = 1, s \ne F \\ 0, & \text{otherwise .} \end{cases}
\end{aligned}
$$

The correctness of the reduction can be seen using the arguments from the proofs of Theorem 3.10 and Theorem 4.3. ■

Theorem 4.17 *The time-dependent $\lceil \log |\mathcal{S}| \rceil$-horizon partially-observable* sMDPP *is* PSPACE*-hard and is in* NEXP.

Proof Hardness follows from Theorem 4.8. Containment in NEXP follows from the standard guess-and-check approach. ■

Theorem 4.18 *The history-dependent $\lceil \log |\mathcal{S}| \rceil$-horizon partially-observable* $s_{\log}$MDPP *is* PSPACE*-complete.*

Proof Hardness follows from Theorem 4.9. Containment in PSPACE follows by a similar argument as in the proof of Theorem 4.9. ■

5 Nonapproximability

How hard is it to find the policy with maximal performance for a given MDP? Given a policy and a MDP, one can compute the performance of the policy in polynomial time. Therefore, computing an optimal policy is at least as hard as deciding the MDPP, whenever the MDPP is in a class containing P. Instead of asking for an optimal policy, one can also ask for a nearly optimal policy. A polynomial time algorithm computing such a sub-optimal policy is called an ε-approximation (for $0 < \varepsilon < 1$), where ε indicates the quality of the approximation in the following way. Let A be a polynomial time algorithm which for any MDP M computes a policy π_α^M. The algorithm A is called an ε-approximation for some type of MDP, if for any MDP M of that type,

the performance of π_α^M on M $\quad > \quad$ $(1-\varepsilon)\cdot$ performance of the optimal policy of type α on M.

(See [5] for more detailed definitions.) Approximability distinguishes NP-complete problems: there are problems which are ε-approximable for all ε, for certain ε, or for no ε (unless P = NP).

We consider the question whether the optimal stationary policy can be ε-approximated for partially-observable MDPs with nonnegative rewards. Remember that the related decision problem is NP-complete (Theorem 3.10).

Theorem 5.1 *The optimal stationary policy for partially-observable MDPs with nonnegative rewards can be ε-approximated for some $\varepsilon < 1$ if and only if* P = NP.

Proof If P = NP, then one can compute exactly the maximal performance of the given MDP in polynomial time.

To derive P = NP, we use a reduction from 3SAT similar to that in the proof of Theorem 3.10, but with different rewards. We add a counter for the number of clauses satisfied by the policy. If this counter finally reached the number of clauses m in ϕ, then a reward of m^2 is gained. Otherwise the final reward is the number of satisfied clauses. One can show that every ε-approximation computes m^2 on input ϕ iff $\phi \in$ 3SAT. This yields that P = NP.

Formally, from ϕ, we construct a partially-observable MDP $M(\phi) = (\mathcal{S}, s_0, \mathcal{A}, \mathcal{O}, t, o, r)$ with

$$\mathcal{S} = \{(i,j,q) \mid 1 \le i \le n, 1 \le j \le m, 0 \le q \le m\} \cup \{F, T\}$$

$$s_0 = (v(1,1),1,0), \qquad \mathcal{A} = \{0,1\}, \qquad \mathcal{O} = \{x_1, \ldots, x_n, F, T\}$$

$$
t(s,a,s') = \begin{cases} 1, & \text{if } s=(v(i,j),j,q), s'=(1,j+1,q+1), j<m, 1\le i\le 3, \\ & \quad \text{and } x_{v(i,j)} \text{ appears in } C_j \text{ with signum } a \\ 1, & \text{if } s=(v(i,m),m,m-1), s'=T, 1\le i\le 3, \\ & \quad \text{and } x_{v(i,m)} \text{ appears in } C_m \text{ with signum } a \\ 1, & \text{if } s=(v(i,j),j,q), s'=(v(i+1,j),j,q), 1\le i<3, \\ & \quad \text{and } x_{v(i,j)} \text{ appears in } C_j \text{ with signum } 1-a \\ 1, & \text{if } s=(v(3,j),j,q), s'=(1,j+1,q), j<m, \\ & \quad \text{and } x_{v(3,j)} \text{ appears in } C_j \text{ with signum } 1-a \\ 1, & \text{if } s=(v(3,m),m,q), s'=F, q<m, \\ & \quad \text{and } x_{v(3,m)} \text{ appears in } C_j \text{ with signum } 1-a \\ 1, & \text{if } s=s'=F \text{ or } s=s'=T \\ 0, & \text{otherwise} \end{cases}
$$

$$
r(s,a) = \begin{cases} m^2, & \text{if } s=(i,j,m-1) \text{ and } t(s,a,T)=1 \\ q, & \text{if } s=(i,j,q), q<m, \text{ and } t(s,a,F)=1 \\ 0, & \text{otherwise} \end{cases}
$$

$$
o(s) = \begin{cases} x_i, & \text{if } s=(i,j,q) \\ T, & \text{if } s=T \\ F, & \text{if } s=F \quad . \end{cases}
$$

Note that every trajectory has value $0,1,2,\ldots,m-1$ or m^2. Using an argument similar as in the proof of Theorem 3.10, we can show that there exists a stationary policy for $M(\phi)$ with performance m^2 iff $\phi \in 3\text{S}\small{\text{AT}}$. Let A compute an ε-approximation $\pi(M)$ for the optimal stationary policy for each partially-observable MDP M. Fix some $\phi \in 3\text{S}\small{\text{AT}}$, and assume that $\pi(M(\phi))$ is not the optimal policy for $M(\phi)$. Then we can estimate the quality of A by the above formula yielding

$$
\frac{m^2-q}{m^2} > \frac{m^2-m}{m^2} = \frac{m-1}{m} > \varepsilon
$$

for every $\varepsilon < 1$ and almost every m, contradicting the assumption. Because the optimal policy for $M(\phi)$ can be used to compute a satisfying assignment for ϕ in polynomial time, we get that P = NP. ■

Corollary 5.2 *The optimal stationary policy for $|\mathcal{S}|$-horizon partially-observable MDPs can be ε-approximated for some $\varepsilon < 1$ if and only if* P = NP.

A similar counting technique can be used to show the nonapproximability of optimal policies for MDPs shown to be NP-complete in Theorems 3.7, 3.8, 3.9, and 3.13.

Theorem 5.3 *The optimal policy for $\lceil \log |\mathcal{S}| \rceil$-horizon succinctly represented MDPs with nonnegative rewards can be ε-approximated for some $\varepsilon < 1$ if and only if* P = NP*, where the policy and the observability of the MDP can be of any type.*

Corollary 5.4 *Theorem 5.3 also holds for MDPs with unrestricted rewards.*

Acknowledgements. We thank Anne Condon, Matt Levy, and Antoni Lozano for helpful comments.

References

[1] D.P. Bertsekas. *Dynamic programming and optimal control.* Athena Scientific, Belmont, Massachusetts, 1995. Volume 1 and 2.

[2] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *14th International Conference on AI*, 1995.

[3] D. Burago, M. de Rougemont, and A. Slissenko. On the complexity of partially observed Markov decision processes. *Theoretical Computer Science*, 157(2):161–183, 1996.

[4] L.M. Goldschlager. The monotone and planar circuit value problems are complete for P. *SIGACT News*, 9:25–29, 1977.

[5] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[6] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[7] C.H. Papadimitriou and M. Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71:181–185, 1986.

[8] M.L. Puterman. *Markov decision processes.* John Wiley & Sons, New York, 1994.

Appendix

We give the modified proofs of Papadimitriou and Tsitsiklis [6].

Theorem 4.6 *The stationary $|\mathcal{S}|$-horizon fully-observable* MDPP *is* P*-complete.*

Proof This MDPP can be solved by the known techniques in polynomial time, i.e. it is in P.

We continue by showing that the problem is hard for P. The monotone Boolean circuit value problem CVP is the set of all of monotone Boolean circuits with fan-in 2 which evaluate to 1. Its P-completeness is shown in [4]. We prove that CVP reduces to our MDPP.

Let C be a monotone Boolean circuit with gates $1, 2, \ldots, k$. Every gate in that circuit is either an input gate with value 0 or 1, an $\wedge$-gate or an $\vee$-gate. Then C can be represented by a list $g_1, \ldots, g_k$ of tuples $g_i = (t_i, g_i^0, g_i^1)$, where g_i stands for gate i of type t_i getting its inputs from gate g_i^0 and g_i^1.

From C we construct the Markov decision process $M(C)$ as follows.

$$
\begin{array}{rcl}
\mathcal{S} & = & \{0, 1, 2, \ldots, k\} \\
s_0 & = & k \\
\mathcal{A} & = & \{0, 1\} \\
t(s, a, s') & = & \begin{cases} 1, & \text{if } s \text{ is an input gate, } s' = 0, a = 0 \\ 1, & \text{if } s \text{ is an } \vee\text{-gate and } g_s^a = s' \\ \frac{1}{2}, & \text{if } s \text{ is an } \wedge\text{-gate, } s' \text{ is an input for } s,\ a = 0 \\ 1, & \text{if } s = s' = 0 \\ 0, & \text{otherwise} \end{cases} \\
r(s, a) & = & \begin{cases} 1, & \text{if } s \text{ is an input gate with value 1} \\ -(2^k), & \text{if } s \text{ is an input gate with value 0} \\ 0, & \text{otherwise} \end{cases}
\end{array}
$$

We show that a policy with performance > 0 exists for $M(C)$ if and only if C has value 1. Let C_i be the subcircuit of C with output gate i. Note that $C_k = C$.

Claim 7 *For $i = 1, 2, \ldots, k$, if C_i has value 1, then there exists a policy for $M(C)$ such that every trajectory of length $\leq i$ starting with i has value ≥ 1.*

We prove the claim by induction on i. If C_1 has value 1, then C_1 consists of an input gate with constant value 1. Because every trajectory starting with 1 has the form $1, 0$ having value ≥ 1, the base case is proven. If gate $i+1$ is an input gate, the same argument as above holds. If $i+1$ is an $\vee$-gate, then for one of its inputs $g_{i+1}^a < i+1$ there exists a policy π_a having the properties of the statement of the claim. Define a policy π with $\pi(i+1) = a$ and $\pi(j) = \pi_a(j)$ for $j < i+1$. Then every trajectory θ for π starting with $i+1$ consists of $i+1, \theta_a$ for a trajectory θ_a starting with g_{i+1}^a for π_a. By the induction hypothesis, θ has value ≥ 1.

If $i+1$ is an $\wedge$-gate, then for both of its inputs $g_{i+1}^0, g_{i+1}^1 < i+1$ there exists a policy π_0 resp. π_1 having the properties of the statement of the claim. Define a policy π with $\pi(i+1) = a$ and $\pi(j) = \min_{l \in \{0,1\}} \pi_l(j)$ for $j < i+1$. Then every trajectory θ for π starting with $i+1$ consists of $i+1, \theta_l$ for a trajectory θ_l starting with g_{i+1}^l for π_l. By the induction hypothesis, θ has value ≥ 1. $\square$

Using a similar technique as above we can also show

Claim 8 *For $i = 1, 2, \ldots, k$, if C_i has value 0, then for every policy for $M(C)$ there exists a trajectory of length $\leq i$ starting with i and having value $-(2^k)$.*

For every policy for $M(C)$ there exist at most 2^k trajectories of length k, and each such trajectory has value either 1 or $-(2^k)$. If C has value 1, then by Claim 7 there exists a policy with performance 1. If C has value 0, then by Claim 8 every policy has performance $\leq \frac{(2^k-1)-2^k}{2^k}$. Because the numerator is negative, the performance is also negative. ■