On the Complexity of Value Iteration

Nikhil Balaji
University of Oxford, UK
nikhil.balaji@cs.ox.ac.uk

Stefan Kiefer
University of Oxford, UK
stekie@cs.ox.ac.uk

Petr Novotný
Masaryk University, Brno, Czech Republic
petr.novotny@fi.muni.cz

Guillermo A. Pérez
University of Antwerp, Belgium
guillermoalberto.perez@uantwerpen.be

Mahsa Shirmohammadi
CNRS, Paris, France
IRIF, Paris, France
mahsa.shirmohammadi@irif.fr

Abstract

Value iteration is a fundamental algorithm for solving Markov Decision Processes (MDPs). It computes the maximal n-step payoff by iterating n times a recurrence equation which is naturally associated to the MDP. At the same time, value iteration provides a policy for the MDP that is optimal on a given finite horizon n. In this paper, we settle the computational complexity of value iteration. We show that, given a horizon n in binary and an MDP, computing an optimal policy is **EXPTIME**-complete, thus resolving an open problem that goes back to the seminal 1987 paper on the complexity of MDPs by Papadimitriou and Tsitsiklis. To obtain this main result, we develop several stepping stones that yield results of an independent interest. For instance, we show that it is **EXPTIME**-complete to compute the n-fold iteration (with n in binary) of a function given by a straight-line program over the integers with max and + as operators. We also provide new complexity results for the bounded halting problem in linear-update counter machines.

2012 ACM Subject Classification Theory of computation → Probabilistic computation; Theory of computation → Logic and verification; Theory of computation → Markov decision processes

Keywords and phrases Markov decision processes, Value iteration, Formal verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.102

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of the paper is available at `https://arxiv.org/abs/1807.04920`.

Funding *Nikhil Balaji*: Part of this work was done was when the author was a Postdoctoral fellow at Ulm university supported by DFG grant TH 472/4.
Stefan Kiefer: is supported by a Royal Society University Research Fellowship.
Petr Novotný: is supported by the Czech Science Foundation, grant no. GJ19-15134Y "Verification and Analysis of Probabilistic Programs."
Mahsa Shirmohammadi: is supported by the "AAPS" PEPS JCJC grant.

Acknowledgements We thank James Worrell for helpful comments on early version of this work.

1 Introduction

Markov decision processes (MDP) are a fundamental formalism of decision making under probabilistic uncertainty [29, 9]. As such, they play a prominent role in numerous domains, including artificial intelligence and machine learning [34, 33], control theory [10, 1], operations research and finance [11, 31], as well as formal verification [12, 5], to name a few. Informally, an MDP represents a system which is, at every time step, in one of the *states* from a finite set S. The system evolves in steps: in each step, we can perform an *action* (or *decision*) from a finite set A. When using an action $a \in A$ in state $s \in S$, we collect an immediate *reward* $R(s, a)$ and then transition stochastically to a successor state according to a rational-valued distribution $P(s, a)$, which is given as a part of the MDP. This interaction with an MDP proceeds over either a *finite* or *infinite* horizon. In the finite-horizon case, we are given a bound $H \in \mathbb{N}$ (a *horizon*) such that the interaction stops after H steps; in the infinite horizon case the process goes on forever. To *solve* an MDP means to find an optimal *policy*; that is, a blueprint for selecting actions that maximizes the expected reward accumulated over a finite or infinite horizon. The accumulated rewards are typically *discounted* by some factor $0 < \gamma \leq 1$; for infinite horizon, we need $\gamma < 1$ to ensure that the infinite sum is well defined.

Value iteration. Given the importance of MDPs, it is hardly surprising that they have attracted significant interest in the theory community. Past research on MDPs included the study of complexity issues [27] as well as the design and analysis of algorithms for solving MDPs [22, 24, 38, 39]. In this paper, we provide a fresh look on one of the most familiar algorithms for MDPs: *value iteration (VI).* Introduced by Bellman in the 1950s [6], VI makes use of the optimality principle: the maximal n-step reward achievable from a state s, which we denote by $\vec{v}_n(s)$, satisfies the recurrence

$$\vec{v}_n(s) = \max_{a \in A} \left\{ R(s, a) + \gamma \cdot \sum_{s' \in S} P(s, a)(s') \cdot \vec{v}_{n-1}(s') \right\} , \tag{1}$$

with $\vec{v}_0(s) = 0$. Consequently, a finite-horizon policy is optimal if and only if it chooses, in a situation when the current state is s and n steps are remaining, an action maximizing the right-hand side (RHS) of (1). Thus, to solve an MDP with a finite horizon H, the VI algorithm computes the values $\vec{v}_n(s)$ for all $0 \leq n \leq H$ and all states s, by iterating the recurrence (1). Using these values, VI then outputs (using some tie-breaking rule) some policy satisfying the aforementioned optimality characterization. VI can be deployed also for infinite-horizon MDPs: one can effectively compute a horizon H such that action a is optimal in state s for an infinite horizon[1] if it maximizes the RHS of (1) for $n = H$ [8]. This H has a bit-size which is polynomial in the size of the original MDP, but the magnitude of H can be exponential in the size of the MDP if the discount factor is given in binary [22].

VI is one of the most popular MDP-solving algorithms due to its versatility (as shown above, it can be used for several MDP-related problems) and conceptual simplicity, which makes it easy to implement within different programming paradigms [30, 37], including implementation via neural nets [35]. Several variants of VI with improved performance were developed [36, 14]. For instance, the recent paper by Sidford et al. [32] presented a new class of randomized VI techniques with the best theoretical runtime bounds (for certain values

[1] In infinite-horizon MDPs, there is always an optimal *stationary* policy, which makes decisions based only on the current state. [29]

of parameters) among all known MDP solvers. The paper also expresses hope that their techniques "*will be useful in the development of even faster MDP algorithms.*" To get insight into the underlying structure of VI, which might enable or limit further such accelerations, we take a complexity-theoretic vantage point and study the theoretical complexity of computing an *outcome* of a VI execution. That is, we consider the following decision problem ValIt: given an MDP with a finite horizon H (encoded as a binary number), does a given action a maximize the RHS of (1) for $n = H$? This problem is inspired by the paper of Fearnley and Savani [16], where they show **PSPACE**-hardness (and thus also completeness) for the problem of determining an outcome of *policy iteration*, another well-known algorithm for MDP solving. To the best of our knowledge, VI has not yet been explicitly subjected to this type of analysis. However, questions about the complexity of ValIt were implicitly raised by previous work on the complexity of finite-horizon MDPs, as discussed in the next paragraph.

Finite-horizon MDPs. The complexity of finite-horizon MDPs is a long-standing open problem. Since "finding an optimal policy" is a function problem, we can instead consider the decision variant: "In a given finite-horizon MDP, is it optimal to use a given action in the first step?" As discussed above, this is exactly the ValIt problem in disguise.

In the seminal 1987 paper on the complexity of MDPs [27], Papadimitriou and Tsitsiklis showed **P**-completeness of a special case of finite-horizon optimization where the horizon H has magnitude polynomial in the size of the MDP. At the same time, they noted that in the general case of binary-encoded H, VI can be executed on an **EXPTIME**-bounded Turing machine (since H is represented using $\log(H)$ bits, the number of iterations is exponential in the size of the input). Hence ValIt is in **EXPTIME**. However, the exact complexity of the general finite-horizon optimization remained open ever since, with the best lower bound being the **P**-hardness inherited from the "polynomial H" sub-problem. Tseng [36] presented a more efficient (though still exponential) algorithm for finite-horizon MDPs satisfying a certain stability condition; in the same paper, he comments that *"in view of the stability assumptions needed to obtain an exact solution and the absence of negative results, we are still far from a complete complexity theory for this problem."*

In this paper, we address this issue, provide the missing negative results, and *provide tight bounds on the computational complexity* of ValIt and finite-horizon MDP optimization.

Our Results

The main result of the paper is that ValIt is **EXPTIME**-complete (Theorem 1). In the rest of this section, we first explain some challenges we needed to overcome to obtain the result. Then we sketch our main techniques and conclude with discussing the significance of our results, which extends beyond MDPs to several areas of independent interest.

Challenges

Bitsize of numbers. One might be tempted to believe that ValIt is in **PSPACE**, since the algorithm needs to store only polynomially many values at a time. However, the bitsize of these values may become exponentially large during the computation (e.g., the quantity $\vec{v}_n(s)$ may halve in every step). Hence, the algorithm *cannot* be directly implemented by a polynomial-space Turing machine (TM). One could try to adapt the method of Allender et al. [20, 2] based on an intricate use of the Chinese remainder representation (CRR) of integers. However, there is no known way of computing the max operation directly and efficiently on numbers in CRR.

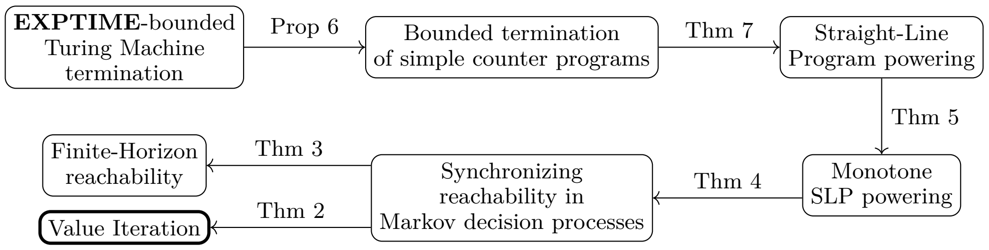

EXPTIME-bounded Turing Machine termination
Prop 6
Bounded termination of simple counter programs
Thm 7
Straight-Line Program powering
Thm 5
Monotone SLP powering
Thm 4
Synchronizing reachability in Markov decision processes
Thm 3
Finite-Horizon reachability
Thm 2
Value Iteration

Figure 1 Chain of reductions.

Complex optimal policies. Another hope for **PSPACE** membership would be a possibly special structure of optimal policies. Fixing any concrete policy turns an MDP into a Markov chain, whose H-step behavior can be evaluated in polynomial space (using, e.g., the aforementioned CRR technique of Allender et al.). If we could prove that (A) an optimal policy can be represented in polynomial space and (B) that the Markov chain induced by such a policy is polynomially large in the size of the MDP, we would get the following **PSPACE** algorithm: cycle through all policies that satisfy (A) and (B), evaluate each of them, and keep track of the best one found so far. Tseng [36] commented that optimal policies in finite-horizon MDPs are "poorly understood". Hence, there was still hope that optimal Markovian deterministic policies may have a shape that satisfies both (A) and (B). Unless **PSPACE** = **EXPTIME**, our results put such hopes to rest.

No hardness by succinctness. One might try to prove **EXPTIME**-hardness using a *succinctness* argument. The results of [27] show that ValIt is **P**-hard when the horizon is written in unary, and many optimization problems over discrete structures incur an exponential blow-up in complexity when the discrete structure is encoded succinctly, e.g., by a circuit [28]. Giving a horizon H in binary amounts to a succinct encoding of an exponentially large MDP obtained by "unfolding" the original MDP into a DAG-like MDP of depth H. This unfolded MDP is "narrow" in the sense that it consists of many polynomial-sized layers, while standard **EXPTIME**-hardness-by-succinctness proofs, use succinct structures of an exponential "width" and "depth", accommodating the tape contents of an **EXPTIME**-bounded TM. Hence, straightforward succinctness proofs do not apply here; e.g., there does not seem to be a direct reduction from the succinct circuit value problem.

Our Techniques

To obtain **EXPTIME**-hardness of ValIt, we proceed by a sequence of non-trivial reductions. Below we outline these reductions in the order in which they appear in the sequence, see Figure 1. In the main text, we present the reductions in a different order (indicated by the numbering of propositions and theorems), so that we start with MDPs and gradually introduce more technical notions.

We start from a canonical **EXPTIME**-complete problem: the halting problem for an exponential-time TM. We then present a reduction to a halting problem for a class of counter programs (CPs; simple imperative programs with integer variables) that allow for linear variable updates. In this way, we encode the tape contents into numerical values (6). The crucial feature of this reduction is that the produced CP possesses a special *simplicity* property, which imposes certain restrictions on the use of tests during the computation.

Next, we introduce *straight-line programs (SLPs)* with max, +, and − operations. SLPs are a standard model of arithmetic computation [3] and they can be equivalently viewed as arithmetic circuits consisting (in our case) of max, +, and − gates. We also consider a sub-class of SLPs with only max, + operations, so called *monotone SLPs.* We define the following *powering problem:* given a function $f\colon \mathbb{Q}^n \to \mathbb{Q}^n$ represented as an SLP, a horizon H, an initial argument $\vec{x} \in \{0,1\}^n$, and two indices $1 \le i, j \le n$, is it true that the i-component of $f^H(\vec{x})$, i.e. the image of $\vec{x}$ with respect to the H-fold composition of f, is greater than the j-component of $f^H(\vec{x})$? Although VI in MDPs does not necessarily involve integers, the powering problem for monotone SLPs captures the complexity inherent in iterating the recurrence (1). To obtain a reduction from CPs to SLP powering, we construct SLP gadgets with max, + and − (minus) operations to simulate the tests in CPs; the simplicity of the input CP is crucial for this reduction to work (Theorem 7). To get rid of the minus operation, we adapt a technique by Allender et al. [4], which introduces a new "offset" counter and models subtraction by increasing the value of the offset (Theorem 5).

A final step is to show a reduction from monotone SLP powering to ValIt. The reduction proceeds via an intermediate problem of *synchronizing reachability in MDPs* (maximize the probability of being in a target set of states T *after exactly* H steps [15]). This divides a rather technical reduction into more comprehensible parts. We present novel reductions from monotone SLP powering to synchronizing reachability (Theorem 4), and from the latter problem to ValIt (Theorem 2). As a by-product, we present a reduction proving **EXPTIME**-hardness of finite-horizon reachability in MDPs, arguably the conceptually simplest objective in probabilistic decision-making (Theorem 3).

Significance

As our main result, we characterize the complexity of computing an outcome of VI, one of the fundamental algorithms for solving both finite- and infinite-horizon MDPs. As a consequence, we resolve a long-standing complexity issue [27] of solving finite-horizon MDPs.

On our way to proving this result, we encounter non-trivial stepping stones which are of an independent interest. First, we shed light on the complexity of succinctly represented arithmetic circuits, showing that comparing two output wires of a given (max, +)-circuit incurs an exponential blow-up in complexity already when employing a very rudimental form of succinctness: composing a single (max, +)-circuit with H copies of itself, yielding a circuit of exponential "height" but only polynomial "width." Second, we obtain new hardness results for the bounded reachability problem in linear-update counter programs. CPs are related to several classical abstractions of computational machines, such as Minsky machines and Petri nets [25], see [13] for a recent breakthrough in this area. Our work establishes a novel connection between counter programs and MDPs.

Further Related Work

Our work is also related to a series of papers on finite-horizon planning [21, 17, 18, 23]. The survey paper [26] provides a comprehensive overview of these results. These papers consider either MDPs with a polynomially large horizon, or *succinctly* represented MDPs of possibly exponential "width" (the succinctness was achieved by circuit-encoding). The aforementioned hardness-by-succinctness proofs are often used here. The arbitrary horizon problem for standard MDPs, which we study, is left open in these papers, and our work employs substantially different techniques. The complexity of finite-horizon *decentralized* MDPs was studied in [7].

2 Markov Decision Processes and Finite-Horizon Problems

We start with some preliminaries. A *probability distribution* $d : S \to [0,1]$ over a finite set S is a function such that $\sum_{s \in S} d(s) = 1$. We denote by $\mathcal{D}(S)$ the set of all (rational) probability distributions over S. The *Dirac* distribution on $s \in S$ assigns probability 1 to s.

A Markov decision process (MDP) $\mathcal{M} = (S, A, P, R, \gamma)$ consists of a finite set S of *states*, a finite set A of *actions*, a *transition function* $P \colon S \times A \to \mathcal{D}(S)$, a *reward function* $R : S \times A \to \mathbb{Q}$, and a *discount factor* $\gamma \in (0,1]$. The transition function P assigns to each state s and action a a distribution over the successor states, while the reward function assigns to s and a a rational reward.

A *path* ϱ is an alternating sequence $s_0 a_1 s_1 \cdots a_n s_n$ of visited states and played actions in $\mathcal{M}$ (that starts and ends in a state); write $|\varrho| = n$ for the length of ϱ. We may use $s_0 \xrightarrow{\varrho} s_n$ to denote that path ϱ goes from s_0 to s_n. We extend the reward function R from single state-action pairs to paths by $R(\varrho) = \sum_{1 \le i \le n} R(s_{i-1}, a_i)\gamma^{i-1}$.

A *policy* for the controller is a function σ that assigns to each path a distribution over actions. Let $\mathbb{P}_{\mathcal{M},s,\sigma}(\varrho)$ denote the probability of a path ϱ starting in s when the controller follows the policy σ. This probability is defined inductively by setting $\mathbb{P}_{\mathcal{M},s,\sigma}(s_0) = 1$ if $s = s_0$, and $\mathbb{P}_{\mathcal{M},s,\sigma}(s_0) = 0$ otherwise. For a path $\varrho = s_0 a_1 s_1 \cdots s_{n-1} a_n s_n$, we set

$$\mathbb{P}_{\mathcal{M},s,\sigma}(\varrho) = \mathbb{P}_{\mathcal{M},s,\sigma}(s_0 \cdots s_{n-1}) \cdot \sigma(s_0 \cdots s_{n-1})(a_n) \cdot P(s_{n-1}, a_n)(s_n) \enspace .$$

We omit the subscripts from $\mathbb{P}_{\mathcal{M},s,\sigma}(\cdot)$ if they are clear from the context. Additionally, we extend $\mathbb{P}_{\mathcal{M},s,\sigma}(\cdot)$ to sets of paths of the same length by summing the probabilities of all the paths in the set.

In this paper, we focus on a special class of policies: A *(deterministic) Markov* policy is a function $\sigma : \mathbb{N} \times S \to A$. Intuitively, a controller following a Markov policy plays $\sigma(n, s)$ from s if it is the n-th visited state, irrespective of the other states in the path. Markov policies suffice for the problems we consider.

2.1 Finite-Horizon Problems

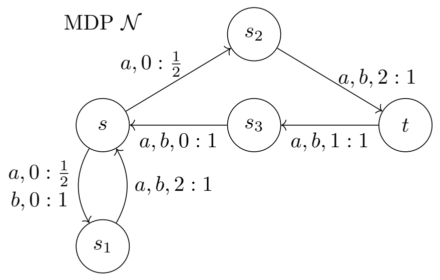
Given an MDP $\mathcal{M}$, the core problem of MDPs is computing the values of states with respect to the *maximum expected reward*. Let $\vec{v}_n \in \mathbb{Q}^S$ denote the vector of n-step maximum expected rewards obtainable from each state of the MDP. That is, for all $s \in S$ we have that

$$\vec{v}_n(s) = \max_{\sigma} \left(\sum_{|\varrho| = n} \mathbb{P}_{s,\sigma}(\varrho) \cdot R(\varrho) \right) \enspace .$$

Note that $\vec{v}_0 = \vec{0}$ by this definition. The vector $\vec{v}_n$ can be computed by *value iteration*, i.e. by iterating the recurrence stated in Equation (1). From that recurrence, for each $n \in \mathbb{N}$ and state s_0, one can extract an (optimal) Markov policy σ that achieves the maximum value $\vec{v}_n(s_0)$ after n steps: for each $s \in S$ and for $1 \le i \le n$ we have

$$\sigma(i-1, s) = \operatorname*{argmax}_{a \in A} \left\{ R(s,a) + \gamma \cdot \sum_{s' \in S} P(s,a)(s') \cdot \vec{v}_{n-i}(s') \right\} \enspace .$$

Papadimitriou and Tsitsiklis posed the **finite-horizon reward problem** which asks to compute such an optimal policy for the controller [27]. Formally, given an MDP $\mathcal{M}$, an initial state $s_0 \in S$, a distinguished action $a \in A$, and a horizon $H \in \mathbb{N}$ encoded in binary, the finite-horizon reward problem asks whether there exists a policy achieving $\vec{v}_H(s_0)$ by



$$\begin{aligned}
\vec{v}_n(s) &= \max\left(\frac{1}{4}\vec{v}_{n-1}(s_1) + \frac{1}{4}\vec{v}_{n-1}(s_2), \frac{1}{2}\vec{v}_{n-1}(s_1)\right)\\
\vec{v}_n(s_1) &= 2 + \frac{1}{2}\vec{v}_{n-1}(s)\\
\vec{v}_n(s_2) &= 2 + \frac{1}{2}\vec{v}_{n-1}(t)\\
\vec{v}_n(t) &= 1 + \frac{1}{2}\vec{v}_{n-1}(s_3)\\
\vec{v}_n(s_3) &= \frac{1}{2}\vec{v}_{n-1}(s)
\end{aligned}$$

Figure 2 The transitions are labelled with actions, rewards and their probabilities. For example, the reward of the transition from s to s_1 on action a is 0, and its probability is $\frac{1}{2}$.

choosing a as the first action from s_0. Note that this problem is equivalent to the ValIt problem defined in the introduction.

Consider the MDP $\mathcal{N}$ depicted in Figure 2 with $\gamma = \frac{1}{2}$. By iterating the indicated recurrence, we have that $\vec{v}_5(s) = \max(\frac{1}{4}\vec{v}_4(s_1) + \frac{1}{4}\vec{v}_4(s_2), \frac{1}{2}\vec{v}_4(s_1)) = \frac{41}{32}$. The value of $\vec{v}_5(s)$ is due to the second argument of max (corresponding to action b), hence a policy to maximize $\vec{v}_5(s)$ starts with b in s.

The finite-horizon reward problem can be decided by value iteration in exponential time by unfolding recurrence (1) for H steps [29], while the best known lower bound is **P**-hardness [27]. Our main result closes this long-standing complexity gap:

▶ **Theorem 1.** *The finite-horizon reward problem (and thus also the* ValIt *problem) is* **EXPTIME***-complete.*

To prove **EXPTIME**-completeness of the finite-horizon reward problem, we introduce a variant of reachability, which we call *synchronized reachability* [15]. Let $t \in S$ be a target state. For reachability, the objective is to maximize the probability of taking a path from s to t, whereas in synchronized reachability only a subset of such paths with the same length are considered.

Let $\mathcal{M}$ be an MDP, s_0 an initial state, and a an action. Define $\vec{p}_{\leq n} \in \mathbb{Q}^S$ as the vector of maximum probabilities of taking a path to t within n steps. Similarly, define $\vec{p}_{=n} \in \mathbb{Q}^S$ to be the vector of maximum probabilities of taking such a path with length exactly n. Formally, for all $s \in S$ we have that

$$\vec{p}_{\leq n}(s) = \max_{\sigma}\left(\mathbb{P}_{s,\sigma}(\{s \xrightarrow{\varrho} t : |\varrho| \leq n\})\right) \text{ and } \vec{p}_{=n}(s) = \max_{\sigma}\left(\mathbb{P}_{s,\sigma}(\{s \xrightarrow{\varrho} t : |\varrho| = n\})\right) .$$

Given a horizon H, encoded in binary, the **finite-horizon reachability problem** asks whether an optimal policy achieving $\vec{p}_{\leq H}(s_0)$ chooses action a as the first action from s_0; the **finite-horizon synchronized-reachability problem** asks whether an optimal policy achieving $\vec{p}_{=H}(s_0)$ chooses action a as the first action from s_0.

2.2 Connections Among Finite-Horizon Problems

We now prove the following theorem.

▶ **Theorem 2.** *The finite-horizon synchronized-reachability problem reduces, in polynomial time, to the finite-horizon reward problem.*

Consider an MDP $\mathcal{M}$, an initial state s_0, an action a and a target state t. The following recurrence can be used to compute $\vec{p}_{=n}(s)$:

$$\vec{p}_{=n}(s) = \max_{a \in A}\left\{\sum_{s' \in S} P(s,a)(s') \cdot \vec{p}_{=n-1}(s')\right\} , \tag{2}$$

where $\vec{p}_0(t) = 1$ and $\vec{p}_0(s) = 0$ for all $s \neq t$. We construct a new MDP $\mathcal{N}$ obtained from $\mathcal{M}$ by replacing all transitions by two consecutive transitions. The construction is such that the probability of going from s to t with a path of length n in $\mathcal{M}$ is equal to the probability of going from s to t with a path of length $2n$ in $\mathcal{N}$. More formally, for all s, s' and a with $P(s,a)(s') = p$, the transition $\textcircled{s} \xrightarrow{-\,:\,p} \textcircled{s'}$ is replaced with $\textcircled{s} \xrightarrow{1\,:\,p} \bigcirc \xrightarrow{0\,:\,1} \textcircled{s'}$ if $s = t$ and with $\textcircled{s} \xrightarrow{0\,:\,p} \bigcirc \xrightarrow{\frac{1}{\gamma}\,:\,1} \textcircled{s'}$ otherwise; where $0 < \gamma \leq 1$ is an arbitrary chosen discount factor for $\mathcal{N}$, and the intermediate state in both cases is a new state. The MDP $\mathcal{N}$ in Figure 2 is the result of applying the construction to $\mathcal{M}$ in Figure 3 with $\gamma = \frac{1}{2}$.

For the constructed MDP $\mathcal{N}$, one can show that for all states s, an action is optimal to maximize $\vec{p}_{=2H}(s)$ if and only if it is optimal to maximize $\vec{v}_{2H+1}(s)$. Consider the MDPs from Figure 2 as an example. We have previously argued that a policy maximizing $\vec{v}_5(s)$ in $\mathcal{N}$ starts with action b. Observe that the optimal first choice to maximize $\vec{p}_4(s)$ is also b. This implies that an optimal policy of $\mathcal{M}$ for synchronized-reachability with $H = 2$ starts with b, too. By the above argument, the finite-horizon synchronized-reachability problem reduces to the finite-horizon reward problem.

Hence, to obtain Theorem 1, it remains to determine the complexity of the finite-horizon synchronized-reachability problem. To this aim, we show a close connection between MDPs and a class of piecewise-affine functions represented by *straight line programs* (SLPs). Section 3 provides the details.

Finite-horizon reachability. We also show the finite-horizon synchronized-reachability problem reduces to the finite-horizon reachability problem. We remark that the natural probability-1 variants of these problems have different complexities: specifically, the problem of reaching t from s within H steps with probability 1 is in **P**; however, the analogous problem of reaching t from s in exactly H steps with probability 1 is **PSPACE**-complete [15].

▶ **Theorem 3.** *The finite-horizon synchronized reachability problem reduces, in polynomial time, to the finite-horizon reachability problem.*

3 Straight-Line Programs and The Powering Problem

We now establish the connection between MDPs and SLP powering. We start with preliminaries.

For all $n \in \mathbb{N}$, define the set $\mathit{var}_n := \{x_1, \ldots, x_n\}$ of variables and the collection of terms

$$\mathcal{T}_n := \{a_1 x_{j_1} + \cdots + a_n x_{j_n} + b \mid a_i, b \in \{-1, 0, 1\} \text{ and } 1 \leq j_i \leq n, \text{ for all } 1 \leq i \leq n\}.$$

A *straight-line program* (SLP) of *order* n is a sequence $c_1, \ldots, c_m$ of *commands* of the form $x \leftarrow \max(T)$, where $x \in \mathit{var}_n$ and $T \subseteq \mathcal{T}_n$ is non-empty. We refer to commands $x \leftarrow b$ as *initializations*. Recall that $\min(x,y) = -\max(-x,-y)$.

For complexity analyses we shall assume that T, for every command, is given explicitly as a list of terms. Each term is also assumed to be explicitly represented as a constant, a list of coefficients a_i, and a list of indices j_i, both lists having length n (i.e. the number of variables). The size of T, and also that of the command, corresponds to the length of its list of terms; the size of the SLP, the sum of the sizes of its commands.

A *valuation* ν is a vector in $\mathbb{Z}^n$, where the i-th coordinate gives the value of x_i. The semantics of a command c is a function $[\![c]\!] : \mathbb{Z}^n \to \mathbb{Z}^n$, transforming a valuation into another. An SLP $S = c_1, \ldots, c_m$ defines the function $[\![S]\!] : \mathbb{Z}^n \to \mathbb{Z}^n$ obtained by composing the constituent commands: $[\![S]\!] = [\![c_m]\!] \circ \cdots \circ [\![c_1]\!]$. Clearly this is a piecewise-affine function. Given a function $f : \mathbb{Z}^n \to \mathbb{Z}^n$, we define its m-th power as $f^m : \mathbb{Z}^n \to \mathbb{Z}^n$ where

$$f^m = \underbrace{f \circ \cdots \circ f}_{m \text{ times}}$$

is the m-fold composition of f.

We denote by $\mathcal{T}_n^+$ the set of terms $a_1 x_{j_1} + \cdots + a_n x_{j_n} + b$ where the coefficients $a_1, \cdots, a_n, b$ are in $\{0, 1\}$. An SLP that only uses terms in $\mathcal{T}_n^+$ is called *monotone*. Note that monotone SLPs induce monotone functions from $\mathbb{Z}^n$ to $\mathbb{Z}^n$ (subtraction and min are not allowed).

3.1 The Powering Problem

For an SLP S of order n, a valuation $\nu \in \mathbb{N}^n$ and $m \in \mathbb{N}$ (encoded in binary), let $\nu' = [\![S]\!]^m(\nu)$. Given two variables $x, y \in var_n$ of the SLP, the **powering problem** asks whether $\nu'(x) \geq \nu'(y)$. Since the initial valuations ν are always non-negative, all valuations obtained by powering monotone SLPs are non-negative. The above problem is **P**-complete if the exponent m is written in unary [19].

Observe that all numbers generated by powering an SLP can be represented using exponentially-many bits in the bitsize of the exponent. It follows that the powered SLP can be explicitly evaluated in exponential time. We provide a matching lower bound in Section 4. Before that, we show the connection of SLP powering to MDPs.

3.2 Synchronized Reachability and SLP Powering

The connection is stated in the following Theorem.

▶ **Theorem 4.** *The powering problem for monotone SLPs reduces, in polynomial time, to the finite-horizon synchronized reachability problem in MDPs.*

To illustrate this reduction, let us consider the SLP S of order 2:

$$x_1 \leftarrow \max(x_1 + x_2, x_2 + x_2); \qquad x_2 \leftarrow \max(x_1 + x_1, x_1 + x_1).$$

This SLP is *normalized*, that is to say all its max commands have exactly two arguments $t_1, t_2 \in \mathcal{T}_n^+$ and furthermore t_1, t_2 have exactly two summands. (Note that focusing on normalized SLPs is no loss of generality.) We are interested in the 2-nd power of S with initial valuation $\nu(x_1) = 0$ and $\nu(x_2) = 1$. In Figure 3, two copies of S are shown on the right to visualize the concept of powering it. To obtain an MDP, we consider a set of actions $A = \{a, b\}$ and have each variable x_i become a state. In the example, s and t are the corresponding states for x_1 and x_2. The t_1, t_2 arguments of max commands determine the successors of actions a, b, respectively, where each successor has probability $\frac{1}{2}$. The command $x_1 \leftarrow \max(x_1 + x_2, x_2 + x_2)$ translates to $P(s, a)(s) = P(s, a)(t) = \frac{1}{2}$ and $P(s, b)(s) = 1$, as shown in the MDP in Figure 3. Since $\nu(x_2) = 1$, we make t a target state. Now the i-th iteration of value iteration of (2) (corresponding to the i-th step *before* the horizon) is tightly connected to the i-th power of the SLP. Indeed, letting $\nu_i = [\![S]\!]^i(\nu)$, one can prove that $\vec{p}_{=i}(s) = \frac{1}{2^i}\nu_i(s)$ and $\vec{p}_{=i}(t) = \frac{1}{2^i}\nu_i(t)$.

SLP vs. monotone SLP powering. It thus remains to provide a lower bound for the Monotone SLP powering problem. The crucial step, which we cover in Section 4, is providing lower bounds for the non-monotone variant. The remaining step from non-monotone to monotone powering can be made by adapting the techniques of Allender et al. [4].

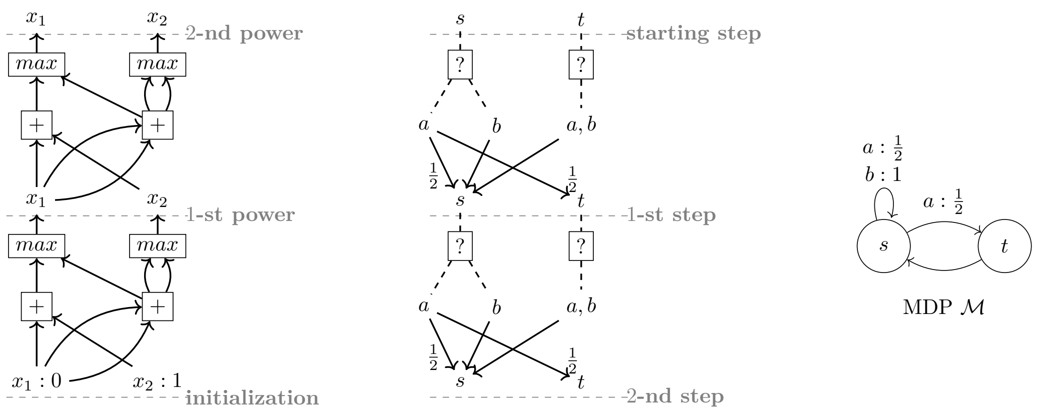


Figure 3 An example for the translation from SLPs to MDPs.

▶ **Theorem 5.** *The powering problem for arbitrary SLPs reduces, in polynomial time, to the powering problem for monotone SLPs.*

4 Main Reductions

To show **EXPTIME**-hardness of all the problems introduced so far, we introduce a class of counter programs that allow linear updates on counters and show that a (time-)bounded version of the termination problem for these programs is **EXPTIME**-complete. Finally, we reduce this bounded termination problem to the powering problem.

A *deterministic linear-update counter program* (CP) consists of n *counters* $\{c_i \mid 1 \leq i \leq n\}$, ranging over $\mathbb{Z}$, and a sequence of m instructions. We consider instructions of the form

$$p : c_1 \leftarrow c_2 + c_3 \qquad p : \texttt{if } \; c_1 \geq c_2 \; \texttt{goto } \; t \qquad p : c_1 \leftarrow c_2 - c_3$$

where $1 \leq p < m$ and $1 \leq t \leq m$, and the final instruction is always m : `halt`. More precisely, the instructions allow

(i) adding or subtracting two counters, assigning the result to a third one, and continuing to the next instruction;
(ii) testing two counters against each other, and jumping to some given instruction if the result of the test is positive, continuing to the next instruction otherwise.

The `halt` instruction only loops to itself.

A *configuration* of a CP is a tuple $(p, v_1, \ldots, v_n) \in \{1, \ldots, m\} \times \mathbb{Z}^n$ consisting of an instruction p and values of the counters (e.g., v_1 is the value for the counter c_1). We equip CPs with a fixed initial configuration lying in $\{1\} \times \mathbb{N}^n$. Given a CP, the **termination problem** asks whether the `halt` instruction is reached. The **bounded termination problem** additionally takes as input an integer $N \in \mathbb{N}$, encoded in binary, and asks whether the `halt` instruction is reached within N steps.

The bounded termination problem is in **EXPTIME**: in a computation with N steps, the magnitude of the counters is bounded by 2^N, so each step can be simulated in time exponential in the bitsize of N. We will now show that the problem is **EXPTIME**-hard already for a certain subclass of CPs which facilitates the reductions to the powering problem.

Simple counter programs. A CP is *simple* if it satisfies the following conditions. First, all values in all reachable configurations $(p, v_1, \ldots, v_n)$ are non-negative: $v_i \in \mathbb{N}$ for all $1 \leq i \leq n$ (one may "guard" subtractions by test instructions to achieve this). Second, all test instructions q : `if` $c_i \geq c_j$ `goto` r use counters c_1 and c_2 exclusively. Moreover, for each such instruction q, there are counters $c_{\bar{q}_1}, c_{\bar{q}_2}$ such that in all reachable configurations $(q, v_1, \ldots, v_n)$ we have that

1. $v_1 = a_1 v_{\bar{q}_1}$ and $v_2 = a_2 v_{\bar{q}_2}$ with $a_1, a_2 \in \{64, 64 \cdot 10, 64 \cdot 12\}$. That is, the values of tested counters are "scaled-up" versions of the values of other counters.
2. Additionally, the absolute difference of the values of the tested counters is larger than the values of all other counters, in symbols $|v_1 - v_2| \geq \max\{v_k \mid 3 \leq k \leq n\}$.

Note that the class of simple CPs is a semantically defined subclass of all CPs. Further observe that for every test instruction we necessarily have that $\bar{q}_1, \bar{q}_2 \geq 3$.

The following proposition kick-starts our sequence of reductions.

▶ **Proposition 6.** *The bounded termination problem for simple CPs is* **EXPTIME***-complete.*

To prove the proposition, we follow the classical recipe of first simulating a Turing machine using a machine with two stacks, and then simulating the two-stack machine by a CP. We note two key differences between our construction and the classical reduction: (1) We use the expressiveness of linear updates in CPs to simulate pushing and popping on the stack in a linear number of steps of the CP. (2) We instrument the two-stack machine to ensure that the height of the two stacks differs by at most 1 along any computation. This is crucial to allow us to simulate the two-stack machine by a *simple* linear-update counter program.

4.1 From the Termination Problem to the Powering Problem

We now sketch the main ideas behind the last (and most technically involved) missing link in our sequence of reductions.

▶ **Theorem 7.** *The bounded termination problem for simple CPs reduces, in polynomial time, to the powering problem for SLPs.*

The encoding. Given a CP $\mathcal{C}$ we construct an SLP S of order $\geq 2n$ with variables including $\{x_1, \ldots, x_{2n}\}$. Let us denote x_{n+i} by Q_i for $1 \leq i \leq n$. The reduction is such that a configuration $(p, v_1, \ldots, v_n)$ of $\mathcal{C}$ is encoded as a valuation $\nu : var_{2n} \to \mathbb{Z}$ of the SLP with the property that $\nu(x_i) = v_i$ and $\nu(Q_i) = p\nu(x_i) = pv_i$ for all $1 \leq i \leq n$. In this way, the instruction p of the CP is encoded in the variables of the SLP (recall that SLPs are stateless).

Given this encoding, the main challenge is to realize the transition function of the CP as a function computed by an SLP. Once this is accomplished, for every $m \in \mathbb{N}$, the m-th power of the SLP S represents the m-step transition function of the CP.

Conditional commands. Intuitively, to encode the transition function we would like to equip the SLP with *conditional commands*, whose execution depends on a conditional. Specifically, we want to implement the following two kinds of conditional updates

$$(y \leftarrow y \pm x_k \text{ if } Q_k = px_k) \quad \text{and} \quad (Q_k \leftarrow p \cdot x_k \text{ if } x_i \geq x_j)$$

in terms of primitive commands of an SLP. In both commands, if the condition is not satisfied, the command is not executed, and the value of y or Q_k remains unchanged. For example, one can simulate the first type of conditional commands by executing $y \leftarrow y \pm \max(0, x_k + t)$, where t is an expression that is 0 if the test is passed and less than $-x_k$ otherwise. Intuitively, we think of t as "masking" the assignment if the test fails.

For the following result, which formalizes how we implement conditional commands, we call a valuation ν *valid* if there exists $q \in \{1, \ldots, m\}$ with $\nu(x_i) \geq 0$ and $\nu(Q_i) = q\nu(x_i)$ for all $1 \leq i \leq n$.

▶ **Lemma 8.** *Let $p \in \{1, \ldots, m\}$ and $i, j, k \in \{1, \ldots, n\}$ be distinct. The following equation holds for all valid valuations ν:*

$$\max(0, \nu(x_k) + \min(\nu(Q_k) - p\nu(x_k), p\nu(x_k) - \nu(Q_k))) = \begin{cases} \nu(x_k) & \text{if } \nu(Q_k) = p\nu(x_k) \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Moreover, if $|\nu(x_i) - \nu(x_j)| \geq \nu(x_k)$, then the following holds:

$$\max(0, \nu(x_k) + \min(0, \nu(x_i) - \nu(x_j))) = \begin{cases} \nu(x_k) & \text{if } \nu(x_i) \geq \nu(x_j) \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Proof. The equations follow directly from the assumption that ν is valid, since if $\nu(Q_j) \neq p\nu(x_j)$ then we also have $|\nu(Q_j) - p\nu(x_j)| \geq \nu(x_j)$. In addition, if $|\nu(x_i) - \nu(x_j)| \geq \nu(x_k)$ and $\nu(x_i) < \nu(x_j)$, we will have $\nu(x_k) + \nu(x_i) - \nu(x_j) \leq 0$. ◀

Using the property that the simulated program is simple, Equation (3) can be used to simulate the conditional update ($y \leftarrow y \pm x_k$ if $Q_k = px_k$) where $t = \min(Q_k - px_k, px_k - Q_k)$ masks the update. Likewise, Equation (4) can be used to simulate the second type of conditional update ($Q_k \leftarrow p{\cdot}x_k$ if $x_i \geq x_j$) where the masking expression is $t = \min(0, x_i - x_j)$. Finally, the multiplication-by-a-constant required for the second type of the conditional update is achieved via repeated addition.

Encoding the instructions. We recall that we encode being at the instruction p of the CP by a valuation ν such that $\nu(Q_i) = p\nu(x_i)$ for all $1 \leq i \leq n$.

Using the aforementioned conditional commands, we can construct the SLP S as the composition of m smaller SLPs. Each sub-SLP π_p simulates an instruction p from the given CP $\mathcal{C}$. Hence S, when applied upon a valid valuation ν (i.e., a properly-encoded configuration of $\mathcal{C}$), simulates all of its instructions at once. By using conditional commands, we make sure that only one sub-SLP results in a non-zero update: executing π_p has no effect on the valuation unless $p\nu(x_i) = \nu(Q_i)$ for all $1 \leq i \leq n$.

In this way, powering S allows us to simulate consecutive steps of $\mathcal{C}$. In particular, for all $N \in \mathbb{N}$ we have that $[\![S]\!]^N(\nu)(Q_1) \geq m \cdot [\![S]\!]^N(\nu)(x_1)$, where m is the *halt* instruction, holds if and only if $\mathcal{C}$ halts after at most N steps.

5 Conclusion

By the virtue of our chain of reductions (see Figure 1), we get the following theorem.

▶ **Theorem 9.** *All the following problems are* **EXPTIME***-complete:*

- *The finite-horizon reward problem for MDPs, and thus also the* ValIt *problem.*
- *The finite-horizon reachability and synchronized reachability problems for MDPs.*
- *The powering problem for SLPs and for monotone SLPs.*
- *The bounded termination problem for simple counter programs.*

The exact complexity of the following variant of the problem remains open: given an MDP and a horizon encoded in binary, determine whether there exists a policy achieving some given expected-reward threshold (with no restriction on the actions used to do so).

References

1 Pieter Abbeel and Andrew Y. Ng. Learning first-order Markov models for control. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1–8. MIT Press, 2005. URL: `http://papers.nips.cc/paper/2569-learning-first-order-markov-models-for-control.pdf`.

2 Eric Allender, Nikhil Balaji, and Samir Datta. Low-Depth Uniform Threshold Circuits and the Bit-Complexity of Straight Line Programs. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014. `doi:10.1007/978-3-662-44465-8_2`.

3 Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.

4 Eric Allender, Andreas Krebs, and Pierre McKenzie. Better complexity bounds for cost register automata. *Theory of Computing Systems*, pages 1–19, 2017.

5 Christel Baier and Katoen Joost-Pieter, editors. *Principles of Model Checking*. MIT Press, 2008.

6 Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

7 Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

8 Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

9 Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena scientific Belmont, MA, 2005.

10 Vincent D Blondel and John N Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

11 Nicole Bäuerle and Ulrich Rieder. *Markov Decision Processes with Applications to Finance*. Springer-Verlag Berlin Heidelberg, 2011.

12 Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Bloem Roderick, editors. *Handbook of Model Checking*. Springer International Publishing, 2018.

13 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The Reachability Problem for Petri Nets is Not Elementary (Extended Abstract). *CoRR*, abs/1809.07115, 2018. `arXiv:1809.07115`.

14 Peng Dai, Mausam, Daniel S. Weld, and Judy Goldsmith. Topological Value Iteration Algorithms. *J. Artif. Intell. Res.*, 42:181–209, 2011. URL: `http://jair.org/papers/paper3390.html`.

15 Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Limit Synchronization in Markov Decision Processes. In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8412 of *Lecture Notes in Computer Science*, pages 58–72. Springer, 2014. `doi:10.1007/978-3-642-54830-7_4`.

16 John Fearnley and Rahul Savani. The Complexity of the Simplex Method. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 201–208, New York, NY, USA, 2015. ACM. `doi:10.1145/2746539.2746558`.

17 Judy Goldsmith, Michael L Littman, and Martin Mundhenk. The complexity of plan existence and evaluation in probabilistic domains. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence (UAI'97)*, pages 182–189. Morgan Kaufmann Publishers Inc., 1997.

18 Judy Goldsmith and Martin Mundhenk. Complexity Issues in Markov Decision Processes. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity, Buffalo,*

New York, USA, June 15-18, 1998, pages 272–280. IEEE Computer Society, 1998. doi:10.1109/CCC.1998.694621.

19 R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory.* Oxford University Press, 1995. URL: https://books.google.fr/books?id=YZHnCwAAQBAJ.

20 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.

21 Michael L Littman. Probabilistic propositional planning: Representations and complexity. In *AAAI'97*, pages 748–754, 1997.

22 Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the Complexity of Solving Markov Decision Problems. In Philippe Besnard and Steve Hanks, editors, *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 394–402. Morgan Kaufmann, 1995. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=457&proceeding_id=11.

23 Michael L Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

24 Yishay Mansour and Satinder P. Singh. On the Complexity of Policy Iteration. In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 401–408. Morgan Kaufmann, 1999. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=192&proceeding_id=15.

25 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the thirteenth annual ACM Symposium on Theory of computing (STOC'81)*, pages 238–246. ACM, 1981.

26 Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of Finite-horizon Markov Decision Process Problems. *J. ACM*, 47(4):681–720, July 2000. doi:10.1145/347476.347480.

27 Christos H. Papadimitriou and John N. Tsitsiklis. The Complexity of Markov Decision Processes. *Math. Oper. Res.*, 12(3):441–450, 1987. doi:10.1287/moor.12.3.441.

28 Christos H. Papadimitriou and Mihalis Yannakakis. A Note on Succinct Representations of Graphs. *Information and Control*, 71(3):181–185, 1986. doi:10.1016/S0019-9958(86)80009-2.

29 Martin L. Puterman. *Markov Decision Processes.* Wiley-Interscience, 2005.

30 Tim Quatmann and Joost-Pieter Katoen. Sound Value Iteration. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification*, pages 643–661. Springer International Publishing, 2018.

31 Manfred Schäl. Markov decision processes in finance and dynamic options. In *Handbook of Markov Decision Processes*, International Series in Operations Research & Management Science, pages 461–487. Springer, 2002.

32 Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. *Variance Reduced Value Iteration and Faster Algorithms for Solving Markov Decision Processes*, pages 770–787. SIAM, 2018. doi:10.1137/1.9781611975031.50.

33 Olivier Sigaud and Olivier Buffet. *Markov Decision Processes in Artificial Intelligence.* John Wiley & Sons, 2013.

34 R.S. Sutton and A.G Barto. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning. MIT Press, 2018.

35 Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value Iteration Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2154–2162. Curran Associates, Inc., 2016. URL: http://papers.nips.cc/paper/6046-value-iteration-networks.pdf.

36 Paul Tseng. Solving H-horizon, Stationary Markov Decision Problems in Time Proportional to log(H). *Operations Research Letters*, 9(5):287–297, 1990.

37 Zhimin Wu, Ernst Moritz Hahn, Akin Günay, Lijun Zhang, and Yang Liu. GPU-accelerated value iteration for the computation of reachability probabilities in mdps. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, pages 1726–1727. IOS Press, 2016. `doi:10.3233/978-1-61499-672-9-1726`.

38 Yinyu Ye. A New Complexity Result on Solving the Markov Decision Problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

39 Yinyu Ye. The Simplex and Policy-Iteration Methods are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.