Complexity of Probabilistic Planning under Average Rewards

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

Abstract

A general and expressive model of sequential decision making under uncertainty is provided by the Markov decision processes (MDPs) framework. Complex applications with very large state spaces are best modelled implicitly (instead of explicitly by enumerating the state space), for example as precondition-effect operators, the representation used in AI planning. This kind of representations are very powerful, and they make the construction of policies/plans computationally very complex. In many applications, average rewards over unit time is the relevant rationality criterion, as opposed to the more widely used discounted reward criterion, and for providing a solid basis for the development of efficient planning algorithms, the computational complexity of the decision problems related to average rewards has to be analyzed. We investigate the complexity of the policy/plan existence problem for MDPs under the average reward criterion, with MDPs represented in terms of conditional probabilistic precondition-effect operators. We consider policies with and without memory, and with different degrees of sensing/observability. The unrestricted policy existence problem for the partially observable cases was earlier known to be undecidable. The results place the remaining computational problems to the complexity classes EXP and NEXP (deterministic and nondeterministic exponential time.)

1 Introduction

Markov decision processes (MDPs) formalize decision making in controlling a nondeterministic transition system so that given utility criteria are satisfied. An MDP consists of a set of states, a set of actions, transition probabilities between the states for every action, and rewards/costs associated with the states and actions. A policy determines for every state which action is to be taken. Policies are valued according to the rewards obtained or costs incurred.

Applications for the kind of planning problems addressed by this work include agent-based systems, including Internet agents and autonomous robots, that have to repeatedly perform actions over an extended period of time in the presence of uncertainty about the environment, and the actions have to – in order to produce the desired results – follow a high-level strategy, expressed as a plan.

Classical deterministic AI planning is the problem of finding a path between the initial state and a goal state. For explicit representations of state spaces as graphs this problem is solvable in polynomial time, and for implicit representations of state spaces in terms of state variables and precondition-effect operators, which sometimes allows an exponentially more concise representation of the problem instances, the path existence problem is PSPACE-complete [Bylander, 1994]. This result is closely related to the PSPACE-completeness of the existence of paths in graphs represented as circuits [Papadimitriou and Yannakakis, 1986; Lozano and Balcázar, 1990]. Similarly, the complexity of most other graph problems increases when a compact graph representation is used [Galperin and Wigderson, 1983; Papadimitriou and Yannakakis, 1986; Lozano and Balcázar, 1990; Balcázar, 1996; Feigenbaum *et al.*, 1999].

MDPs and POMDPs can be viewed as an extension of the graph-based deterministic planning framework with probabilities: an action determines a successor state only with a certain probability. The objective is to visit valuable states with a high probability. A policy (a plan) determines which actions are chosen given the current state (or a set of possible current states, possibly together with some information on the possible predecessor states.) For explicitly represented MDPs, policy evaluation under average rewards reduces to the solution of sets of linear equations. Sets of linear equations can be solved in polynomial time. Similarly, policies for many types of explicitly represented MDPs can be constructed in polynomial time by linear programming. Papadimitriou and Tsitsiklis [1987] have shown that policy existence for explicitly represented MDPs is P-complete. Madani et al. [1999] have shown the undecidability of policy existence for UMDPs and POMDPs with all main rationality criteria.

Like in classical AI planning, MDPs/POMDPs can be concisely represented in terms of state variables and precondition-effect operators. The important question in this setting is, what is the impact of concise representations on the complexity of these problems. In related work [Mundhenk *et al.*, 2000; Littman, 1997; Littman *et al.*, 1998], this question has been investigated in the context of finite horizons. Not surprisingly, there is in general an exponential increase in problem complexity, for example from deterministic polynomial time to deterministic exponential time. The undecidability results for explicitly represented UMDPs and POMDPs directly implies the undecidability of the respective decision problems with concise representations.

In the present work we investigate the complexity of the policy existence problems for MDPs and POMDPs under expected average rewards over an infinite horizon. For many practically interesting problems from AI – for example autonomous robots, Internet agents, and so on – the number of actions taken is high over a period time and lengths of sequences of actions are unbounded. Therefore there is typically no reasonable interpretation for discounts nor reasonable upper bounds on the horizon length, and average reward is the most relevant criterion. A main reason for the restriction to bounded horizons and discounted rewards in earlier work is that the structure of the algorithms in these cases is considerably simpler, because considerations on MDP structural properties, like recurrence and periodicity, can be avoided. Also, for many applications of MDPs that represent phenomena over extended periods of times (years and decades), for example in economics, the discounts simply represent the unimportance of events in the distant future, for example transcending the lifetimes of the decision makers. Boutilier and Puterman [1995] have advocated the use of average-reward criteria in AI.

The structure of the paper is as follows. Section 2 describes the planning problems addressed by the paper, and Section 3 introduces the required complexity-theoretic concepts. In Section 4 we present the results on the complexity of testing the existence of policies for MDPs under average reward criteria, and Section 5 concludes the paper.

2 Probabilistic Planning with Average Rewards

The computational problem we consider is the existence of policies for MDPs (fully observable), UMDPs (unobservable) and POMDPs (partially observable, generalizing both MDPs and UMDPs) that are represented concisely; that is, states are represented as valuations on state variables and transition probabilities are given as operators that affect the state variables. The policies we consider may have an arbitrary size, but we also briefly discuss complexity reduction obtained by restricting to polynomial size policies.

As pointed out in Example 2.1, the average reward of a policy sometimes cannot unambiguously be stated as a single real number. The computational problem that we consider is the following. Is the expected average reward greater than (or equal to) some constant c. This amounts to identifying the recurrent classes determined by the policy, and then taking a weighted average of the rewards according to the probabilities with which the classes are reached.

2.1 Definition of MDPs

MDPs can be represented explicitly as a set of states and a transition relation that assigns a probability to transitions between states under all actions. We restrict to finite MDPs and formally define them as follows.

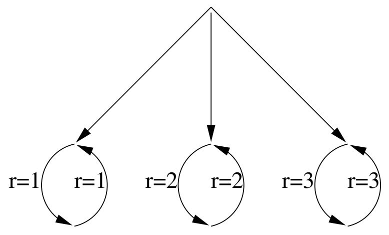

Figure 1: A multichain MDP

Definition 1 *A (partially observable) Markov decision process is a tuple $\langle S, A, T, I, R, B\rangle$ where S is a set of states, A is a set of actions, $T : A \times S \times S \to \mathcal{R}$ gives the transition probability between states (the transition probabilities from a given state must sum to 1.0) for every action, $I \in S$ is the initial state, $R : S \times A \to \mathcal{R}$ associates a reward for applying an action in a given state, and $B \subseteq 2^S$ is a partition of S to classes of states that cannot be distinguished.*

Policies map the current and past observations to actions.

Definition 2 *A policy $P : (2^S)+ \to A$ is a mapping from a sequence of observations to an action. A stationary policy $P : 2^S \to A$ is a mapping from the current observation to an action.*

For UMDPs the observation is always the same (S), for MDPs the observations are singleton sets of states (they determine the current state uniquely), and for POMDPs the observations are members of a partition of S to sets of states that are indistinguishable from each other (the limiting cases are S and singletons $\{s_i\}$ for $s_i \in S$: POMDPs are a generalization of both UMDPs and MDPs.)

The expected average reward of a policy is the limit

$$\lim_{N\to\infty} \frac{1}{N} \sum_{t\geq 0}^{N} \sum_{a\in A, s\in S} r_{a,s} p_{a,s,t}$$

where $r_{a,s}$ is the reward of taking action a in state s and $p_{a,s,t}$ is the probability of taking action a at time point t in state s. There are policies for which the limit does not exist [Puterman, 1994, Example 8.1.1], but when the policy execution has only a finite number of internal states (like stationary policies have), the limit always exists.

The recurrent classes of a POMDP under a given policy are sets of states that will always stay reachable from each other with probability 1.

Example 2.1 Consider a policy that induces the structure shown in Figure 1 on a POMDP. The three recurrent classes each consist of two states. The initial state does not belong to any of the recurrent classes. The state reached by the first transition determines the average reward, which will be 1, 2 or 3, depending on the recurrent class. ■

2.2 Concise Representation of MDPs

An exponentially more concise representation of MDPs is based on state variables. Each state is an assignment of truth-values to the state variables, and transitions between states are expressed as changes in the values of the state variables.

In AI planning, problems are represented by so-called STRIPS operators that are pairs of sets of literals, the *precondition* and the *effects*. For probabilistic planning, this can be extended to probabilistic STRIPS operators (PSOs) (see [Boutilier *et al.*, 1999] for references and a discussion of PSOs and other concise representations of transition systems with probabilities.) In this paper, we further extend PSOs to what we call extended PSOs (EPSOs). An EPSO can represent an exponential number of PSOs, and we use them because they are closely related to operators with conditional effects commonly used in AI planning. Apart from generating the state space of a POMDP, the operators can conveniently be taken to be the actions of the POMDP.

Definition 3 (Extended probabilistic STRIPS operators) *An* extended probabilistic STRIPS operator *is a pair $\langle C, E\rangle$, where C is a Boolean circuit and E consists of pairs $\langle c, f\rangle$, where c is a Boolean circuit and f is a set of pairs $\langle p, e\rangle$, where $p \in]0..1]$ is a real number and e is a set of literals such that for every f the sum of the probabilities p is 1.0.*

For all $\langle c_1, f_1\rangle \in E$ and $\langle c_2, f_2\rangle \in E$, if e_1 contradicts e_2 for some $\langle p_1, e_1\rangle \in f_1$ and $\langle p_2, e_2\rangle \in f_2$, then c_1 must contradict c_2.

This definition generalizes PSOs by not requiring that the cs of members $\langle c, F\rangle$ of E are logically disjoint and their disjunction is a tautology. Hence in EPSOs the effects may take place independently of each other. Some of the hardness proofs given later would be more complicated – assuming that they are possible – if we had to restrict to PSOs.

The application of an EPSO is defined iff the precondition C is true in the current state. Then the following takes place for every $\langle c, f\rangle \in E$. If c is true, one of the $\langle p, e\rangle \in f$ is chosen, each with probability p, and literals e are changed to true.

Example 2.2 Let

$$\begin{aligned} o = \langle \top, \{ \quad & \langle p_1, \{\langle 1.0, \{\neg p_1\}\rangle\}\rangle, \\ & \langle \neg p_1, \{\langle 1.0, \{p_1\}\rangle\}\rangle, \\ & \ldots, \\ & \langle p_n, \{\langle 1.0, \{\neg p_n\}\rangle\}\rangle, \\ & \langle \neg p_n, \{\langle 1.0, \{p_n\}\rangle\}\rangle\}\rangle. \end{aligned}$$

Now o is an EPSO but not a PSO because the antecedents $p_1, \neg p_1, p_2, \neg p_2, \ldots$ are not logically disjoint. A set of PSOs corresponding to o has cardinality exponential on n. ■

Rewards are associated with actions and states. When an action is taken in an appropriate state, a reward is obtained. For every action, the set of states that yields a given reward is represented by a Boolean circuit.

Definition 4 (Concise POMDP) A concise POMDP *over a set P of state variables is a tuple $\langle I, O, r, B\rangle$ where I is an initial state (assignment $P \rightarrow \{\top, \bot\}$), O is a set of EPSOs representing the actions, and $r : O \rightarrow C \times \mathcal{R}$ associates a Boolean circuit and a real-valued reward with every action, and $B \subseteq P$ is the set of observable state variables.*

Having a set of variables observable – instead of arbitrary circuits/formulae – is not a restriction. Assume that the values of a circuit are observable (but the individual input gates are not.) We could make every EPSO evaluate the value of this circuit and set the value of an observable variable accordingly.

Definition 5 (Concise MDP) A concise MDP *is a concise POMDP with $B = P$.*

Definition 6 (Concise UMDP) A concise UMDP *is a concise POMDP with $B = \emptyset$.*

2.3 Concise Representation of Policies

We consider history/time-dependent and stationary policies, and do not make a distinction between history and time-dependent ones. Traditionally explicit (or flat) representations of policies have been considered in research on MDPs/POMDPs: each state or belief state is explicitly associated with an action. In our setting, in which the number of states can be very high, also policies have to be represented concisely. Like with concise representations of POMDPs, there is no direct connection between the size of a concisely represented policy and the number of states of the POMDP.

A concise policy could, in the most general case, be a program in a Turing-equivalent programming language. This would, however, make many questions concerning policies undecidable. Therefore less powerful representations of policies have to be used. A concise policy determines the current action based on the current observation and the past history. We divide this to two subtasks: keeping track of the history (maintaining the internal state of the execution of the policy), and mapping the current observation and the internal state of the execution of the policy to an action. The computation needed in applying one operator is essentially a state transition of a concisely represented finite automaton.

A sensible restriction would be that computation of the action to be taken and the new internal state of the policy execution is polynomial time. An obvious choice is the use of Boolean circuits, because the circuit value problem is P-complete (one of the hardest problems in P.) Work on algorithms for concise POMDPs and AI planning have not used this general a policy representation, but for our purposes this seems like a well-founded choice. Related definitions of policies as finite-state controllers have been proposed earlier [Hansen, 1998; Meuleau *et al.*, 1999; Lusena *et al.*, 1999].

Definition 7 (Concise policy) A concise policy *for a concise POMDP $M = \langle I, O, r, B\rangle$ is a tuple $\langle T, C, v\rangle$ where T is a Boolean circuit with $|B| + p$ input gates and p output gates, C is a Boolean circuit with $|B| + p$ input gates and $\lceil \log_2 |O| \rceil$ output gates, and v is a mapping from $\{1, \ldots, p\}$ to $\{\bot, \top\}$.*

The circuit T encodes the change in execution state in terms of the preceding state and the observable state variables

	stationary	history-dependent
UMDP	PSPACE-hard, in EXP (L8,9)	undecidable
MDP	EXP (T11)	EXP (C12)
POMDP	NEXP (T13)	undecidable

Table 1: Complexity of policy existence, with references to the lemmata, theorems, and corollaries.

B. The circuit C encodes the action to be taken, and v gives the initial state of the execution. The integer p is the number of bits for the representation of the internal state of the execution. When $p = 0$ we have a stationary policy.

The complexity results do not deeply rely on the exact formal definition of policies. An important property of the definition is that one step of policy execution can be performed in polynomial time.

3 Complexity Classes

The complexity class P consists of decision problems that are solvable in polynomial time by a deterministic Turing machine. NP is the class of decision problems that are solvable in polynomial time by a nondeterministic Turing machine. $C_1^{C_2}$ denotes the class of problems that is defined like the class C_1 except that Turing machines with an oracle for a problem in C_2 are used instead of ordinary Turing machines. Turing machines with an oracle for a problem B may perform tests for membership in B for free. A problem L is *C-hard* if all problems in the class C are polynomial time *many-one reducible* to it; that is, for all problems $L' \in C$ there is a function $f_{L'}$ computable in polynomial time on the size of its input and $f_{L'}(x) \in L$ if and only if $x \in L'$. A problem is *C-complete* if it belongs to the class C and is C-hard.

PSPACE is the class of decision problems solvable in deterministic polynomial space. EXP is the class of decision problems solvable in deterministic exponential time ($O(2^{p(n)})$ where $p(n)$ is a polynomial.) NEXP is the class of decision problems solvable in nondeterministic exponential time. A more detailed description of the complexity classes can be found in standard textbooks on complexity theory, for example by Balcazár et al. [1988].

4 Complexity Results

Table 1 summarizes the complexity of determining the existence of stationary and history-dependent policies for UMDPs, MDPs and POMDPs. In the average rewards case the existence of history-dependent and stationary policies for MDPs coincide. The undecidability of UMDP and POMDP policy existence with history-dependent policies of unrestricted size was shown by Madani et al. [1999]. The result is based on the emptiness problem of probabilistic finite automata [Paz, 1971; Condon and Lipton, 1989] that is closely related to the unobservable plan existence problem.

The results do not completely determine the complexity of the UMDP stationary policy existence problem, but as the stationary UMDP policies repeatedly apply one single operator, the problem does not seem to have the power of EXP. It is also not trivial to show membership in PSPACE.

The rest of the paper formally states the results summarized in Table 1 and gives their proof outlines.

Lemma 8 *Existence of a policy with average reward $r \geq c$ for UMDPs, MDPs and POMDPs with only one action is PSPACE-hard.*

Proof: It is straightforward to reduce any decision problem in PSPACE to the problem. This is by constructing a concise UMDP/MDP/POMDP with only one action that simulates a polynomial-space deterministic Turing machine for the problem in question.

There are state variables for representing the input, the working tape, and the state of the Turing machine. The EPSO that represents the only action is constructed to follow the state transitions of the Turing machine. The size of the EPSO is polynomial on the size of the input. When the machine accepts, it is restarted. A reward $r \geq c$ is obtained as long as the machine has not rejected. If the machine rejects, all future rewards will be 0. Therefore, if the Turing machine accepts the average reward is r, and otherwise it is 0. □

There are two straightforward complexity upper bounds respectively for polynomial size and stationary policies. Polynomial size policies can maintain at most an exponential number of different representations of the past history, and hence an explicit representation of the product of the POMDP and the possible histories has only exponential size, just like the POMDP state space alone. Stationary policies, on the other hand, do not maintain a history at all, and they therefore encode at most an exponential number of different decision situations, one for each (observable) state of a (PO)MDP. For the unrestricted size partially observable non-stationary case there is no similar exponential upper bound, and the problem is not decidable.

Lemma 9 *Let c be a real number. Testing the existence of a poly-size MDP/UMDP/POMDP policy with average reward $r \geq c$ is in EXP.*

Proof: This computation has complexity $\text{NP}^{\textit{EXP}} = \textit{EXP}$, that corresponds to guessing a polynomial size policy (NP) followed by the evaluation of the policy by an EXP oracle. Policy evaluation proceeds as follows. Produce the explicit representation of the product of the POMDP and the state space of the policy. They respectively have sizes $2^{p_1(x)}$ and $2^{p_2(x)}$ for some polynomials $p_1(x)$ and $p_2(x)$. The product, which is a Markov chain and represents the states the POMDP and the policy execution can be in, is of exponential size $2^{p_1(x)+p_2(x)}$.

From the explicit representation of the state space one can identify the recurrent classes in polynomial time, for example by Tarjan's algorithm for strongly connected components. The probabilities of reaching the recurrent classes can be computed in polynomial time on the size of the explicit representation of the state space. The steady state probabilities associated with the states in the recurrent classes can be determined in polynomial time by solving a set of linear equations [Nelson, 1995]. The average rewards can be obtained in polynomial time by summing the products of the probability and reward associated with each state. Hence all the computation is polynomial time on the explicit representation of the problem, and therefore exponential time on the size of the concise POMDP representation, and the problem is in EXP. □

Lemma 10 *Let c be a real number. Testing the existence of a stationary MDP/UMDP/POMDP policy with average reward $r \geq c$ is in NEXP. The policy evaluation problem in this case is in EXP.*

Proof: First a stationary policy (potentially of exponential size as every state may be assigned a different action) is guessed, which is NEXP computation.

The rest of the proof is like in Lemma 9: the number of states that have to be considered is exponential, and evaluating the value of the policy is EXP computation. Hence the whole computation is in NEXP. □

Theorem 11 *Let c be a real number. Testing the existence of an arbitrary stationary policy with average reward $r \geq c$ for a MDP is EXP-complete.*

Proof: EXP-hardness is by reduction from testing the existence of winning strategies of the perfect-information (fully observable) game G_4 [Stockmeyer and Chandra, 1979]. This game was used by Littman [1997] for showing that finite-horizon planning with sequential effect trees is EXP-hard.

G_4 is a game in which two players take turns in changing the truth-values of variables occurring in a DNF formula. Each player can change his own variables only. Who first makes the formula true has won. For $2n$ variables the game is formalized by n EPSOs, each of which reverses the truth-value of one variable (if it is the turn of player 1) or reverses the truth-value of a randomly chosen variable (if it is the turn of player 2.) Reward 1 is normally obtained, but if the DNF formula evaluates to true after player 2 has made his move, all subsequent rewards will be 0. This will eventually take place if the policy does not represent a winning strategy for player 1, and the average reward will hence be 0. Therefore, the existence of a winning strategy for player 1 coincides with the existence of a policy with average reward 1.

EXP membership is by producing the explicit exponential size representation of the MDP, and then using standard solution techniques based on linear programming [Puterman, 1994]. Linear programming is polynomial time. □

Corollary 12 *Let c be a real number. Testing the existence of an arbitrary history-dependent policy with average reward $r \geq c$ for a MDP is EXP-complete.*

Proof: For fully observable MDPs and policies of unrestricted size, the existence of arbitrary policies with a certain value coincides with the existence of stationary policies with the same value. □

Theorem 13 *Let c be a real number. Testing the existence of an arbitrary stationary policy with average reward $r \geq c$ for a POMDP is NEXP-complete.*

Proof: Membership in NEXP is by Lemma 10. For NEXP-hardness we reduce the NEXP-complete succinct 3SAT [Papadimitriou and Yannakakis, 1986] to concise POMDPs. The reduction is similar to the reduction from the NP-complete 3SAT in [Mundhenk *et al.*, 2000, Theorem 4.13]. Their Theorem 4.25 claims a reduction of succinct 3SAT to stationary policies of POMDPs represented as circuits.

The reduction works as follows. The POMDP randomly chooses one of the clauses and makes the proposition of its first literal observable (the state variables representing the proposition together with two auxiliary variables are the only observable state variables). The stationary policy observes the proposition and assigns it a truth-value. If the literal became true, evaluation proceeds with another clause, and otherwise with the next literal in the clause. Because the policy is stationary, the same truth-value will be selected for the variable irrespective of the polarity of the literal and the clause. If none of the literals in the clause is true, the reward which had been 1 so far will on all subsequent time points be 0.

The succinct 3SAT problem is represented as circuits $\mathcal{C}$ that map a clause number and a literal location (0, 1, 2) to the literal occurring in the clause in the given position. The POMDP uses the following EPSOs the application order of which has been forced to the given order by means of auxiliary state variables. The first EPSO selects a clause by assigning truth-values to state variables representing the clause number. The second EPSO copies the number of the proposition in the current literal (first, second or third literal of the clause) to observable variables, The third and fourth EPSO respectively select the truth-value true and false (this is the only place where the policy has a choice.) The fifth EPSO checks whether the truth-value matches, and if it does not and the literal was the last one, the reward is turned to 0. If it does, the execution continues from the first EPSO, and otherwise, the literal was not the last one and execution continues from the second EPSO and the next literal. □

5 Conclusions

We have analyzed the complexity of probabilistic planning with average rewards, and placed the most important decidable decision problems in the complexity classes EXP and NEXP. Earlier it had been shown that without full observability the most general policy existence problems are not decidable. These results are not very surprising because the problems generalize computational problems that were already known to be very complex (PSPACE-hard), like plan existence in classical deterministic AI planning. Also, these problems are closely related to several finite-horizon problems that were earlier shown EXP-complete and NEXP-complete [Mundhenk *et al.*, 2000]. The results are helpful in devising algorithms for average-reward planning as well as in identifying further restrictions that allow more efficient planning. As shown by Lemma 9, polynomial policy size brings the complexity down to EXP, also in the otherwise undecidable cases. There are likely to be useful structural restrictions on POMDPs that could bring down the complexity further. Restricted but useful problems in PSPACE would be of high interest.

References

[Balcázar *et al.*, 1988] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*. Springer-Verlag, Berlin, 1988.

[Balcázar, 1996] José L. Balcázar. The complexity of searching implicit graphs. *Artificial Intelligence*, 86(1):171–188, 1996.

[Boutilier and Puterman, 1995] Craig Boutilier and Martin L. Puterman. Process-oriented planning and average-reward optimality. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1096–1103. Morgan Kaufmann Publishers, 1995.

[Boutilier *et al.*, 1999] Craig Boutilier, Thomas Dean, and Steve Hanks. Planning under uncertainty: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

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

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

[Feigenbaum *et al.*, 1999] Joan Feigenbaum, Sampath Kannan, Moshe Y. Vardi, and Mahesh Viswanathan. Complexity of problems on graphs represented as OBDDs. *Chicago Journal of Theoretical Computer Science*, 5(5), 1999.

[Galperin and Wigderson, 1983] Hana Galperin and Avi Wigderson. Succinct representations of graphs. *Information and Control*, 56:183–198, 1983. See [**?**] for a correction.

[Hansen, 1998] Eric A. Hansen. Solving POMDPs by searching in policy space. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 1998 Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 211–219. Morgan Kaufmann Publishers, 1998.

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

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

[Lozano and Balcázar, 1990] Antonio Lozano and José L. Balcázar. The complexity of graph problems for succinctly represented graphs. In Manfred Nagl, editor, *Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG'89*, number 411 in Lecture Notes in Computer Science, pages 277–286. Springer-Verlag, 1990.

[Lusena *et al.*, 1999] Christopher Lusena, Tong Li, Shelia Sittinger, Chris Wells, and Judy Goldsmith. My brain is full: When more memory helps. In Kathryn B. Laskey and Henri Prade, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Fifteenth Conference (UAI-99)*, pages 374–381. Morgan Kaufmann Publishers, 1999.

[Madani *et al.*, 1999] Omid Madani, Steve Hanks, and Anne Condon. On the decidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99) and the 11th Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 541–548. AAAI Press, 1999.

[Meuleau *et al.*, 1999] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In Kathryn B. Laskey and Henri Prade, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Fifteenth Conference (UAI-99)*, pages 417–426. Morgan Kaufmann Publishers, 1999.

[Mundhenk *et al.*, 2000] Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM*, 47(4):681–720, 2000.

[Nelson, 1995] Randolph Nelson. *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer-Verlag, 1995.

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

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

[Paz, 1971] Azaria Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

[Puterman, 1994] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

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