Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Planning and Learning with Stochastic Action Sets

Craig Boutilier, Alon Cohen, Avinatan Hassidim, Yishay Mansour,
Ofer Meshi, Martin Mladenov and **Dale Schuurmans**
Google Research
{cboutilier,aloncohen,avinatan,mansour,meshi,schuurmans}@google.com

Abstract

In many practical uses of reinforcement learning (RL) the set of actions available at a given state is a random variable, with realizations governed by an exogenous stochastic process. Somewhat surprisingly, the foundations for such sequential decision processes have been unaddressed. In this work, we formalize and investigate *MDPs with stochastic action sets (SAS-MDPs)* to provide these foundations. We show that optimal policies and value functions in this model have a structure that admits a compact representation. From an RL perspective, we show that Q-learning with sampled action sets is sound. In model-based settings, we consider two important special cases: when individual actions are available with independent probabilities, and a sampling-based model for unknown distributions. We develop polynomial-time value and policy iteration methods for both cases, and provide a polynomial-time linear programming solution for the first case.

1 Introduction

Markov decision processes (MDPs) are the standard model for sequential decision making under uncertainty, and provide the foundations for reinforcement learning (RL). With the recent emergence of RL as a practical AI technology in combination with deep learning [Mnih *et al.*, 2013; 2015], new use cases are arising that challenge basic MDP modeling assumptions. One such challenge is that many practical MDP and RL problems have *stochastic sets of feasible actions*; that is, the set A_s of feasible actions at state s *varies stochastically* with each visit to s. For instance, in online advertising, the set of available ads differs at distinct occurrences of the same state (e.g., same query, user, contextual features), due to exogenous factors like campaign expiration or budget throttling. In recommender systems with large item spaces, often a set of *candidate* recommendations is first generated, from which top scoring items are chosen; exogenous factors often induce non-trivial changes in the candidate set. With the recent application of MDP and RL models in ad serving and recommendation [Charikar *et al.*, 1999; Li *et al.*, 2009; Archak *et al.*, 2010; 2012; Amin *et al.*, 2012; Silver *et al.*, 2013; Theocharous *et al.*, 2015; Mladenov *et al.*, 2017], understanding how to capture the stochastic nature of available action sets is critical.

Somewhat surprisingly, this problem seems to have been largely unaddressed in the literature. Standard MDP formulations [Puterman, 1994] allow each state s to have its own feasible action set A_s, and it is not uncommon to allow the set A_s to be non-stationary or time-dependent. However, they do not support the treatment of A_s as a stochastic random variable. In this work, we: (a) introduce the *stochastic action set MDP (SAS-MDP)* and provide its theoretical foundations; (b) describe how to account for stochastic action sets in model-free RL (e.g., Q-learning); and (c) develop tractable algorithms for solving SAS-MDPs in important special cases.

An obvious way to treat this problem is to embed the set of available actions into the state itself. This provides a useful analytical tool, but it does not immediately provide tractable algorithms for learning and optimization, since each state is augmented with all possible *subsets* of actions, incurring an exponential blow up in state space size. To address this issue, we show that SAS-MDPs possess an important property: the Q-value of an available action a at a state s is independent of the availability of other actions. This allows us to prove that optimal policies can be represented compactly using (state-specific) decision lists (or orderings) over the action set.

This special structure allows one to solve the SAS RL problem effectively using, for example, Q-learning. We also devise model-based algorithms that exploit this policy structure. We develop value and policy iteration schemes, showing they converge in a polynomial number of iterations (w.r.t. the size of the underlying "base" MDP). We also show that per-iteration complexity is polynomial time for two important special forms of action availability distribution: (a) when action availabilities are independent, both methods are exact; (b) when the distribution over sets A_s is sampleable, we obtain approximation algorithms with polynomial sample complexity. In fact, policy iteration is strongly polynomial under additional assumptions (for a fixed discount factor). We show that a linear program for SAS-MDPs can be solved in polynomial time as well. Finally, we offer a simple empirical demonstration of the importance of accounting for stochastic action availability when computing an MDP policy.

Additional discussion and full proofs of all results can be found in a longer version of this paper [Boutilier *et al.*, 2018].

2 MDPs with Stochastic Action Sets

We first introduce SAS-MDPs and provide a simple example illustrating how action availability impacts optimal decisions. See [Puterman, 1994] for more background on MDPs.

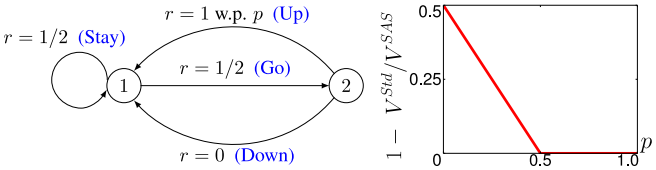
2.1 The SAS-MDP Model

Our formulation of *MDPs with Stochastic Action Sets (SAS-MDPs)* derives from a standard, finite-state, finite-action MDP (the *base MDP*) $\mathcal{M}$, with n states S, *base* actions B_s for $s \in S$, and transition and reward functions, $P : S \times B \to \Delta(S)$ and $r : S \times B \to \mathbb{R}$. We use $p^k_{s,s'}$ and r^k_s to denote the probability of transition to s' and the accrued reward, respectively, when action k is taken at state s. For notational ease, we assume that feasible action sets for each $s \in S$ are identical, so $B_s = B$ (allowing distinct base sets at different states has no impact on what follows). Let $|B| = m$ and $M = |S \times B| = nm$. We assume an infinite-horizon, discounted objective with fixed discount rate γ, $0 \leq \gamma < 1$.

In a SAS-MDP, the set of actions available at state s at any stage t is a random subset $A^{(t)}_s \subseteq B$. We assume a family of *action availability distributions* $P_s \in \Delta(2^B)$ defined over the powerset of B. These can depend on $s \in S$ but are otherwise history-independent, hence $\Pr(A^{(t)}_s | s^{(1)}, \ldots, s^{(t)}) = \Pr(A^{(t)}_s | s^{(t)})$. Only actions $k \in A^{(t)}_s$ in the realized available action set can be executed at stage t. Apart from this, the dynamics of the MDP is unchanged: when an (available) action is taken, state transitions and rewards are prescribed as in the base MDP. In what follows, we assume that some action is always available, i.e., $\Pr(A^{(t)}_s = \emptyset) = 0$ for all s, t.[1] Note that a SAS-MDP does not conform to the usual definition of an MDP.

2.2 Example

The following simple MDP shows the importance of accounting for stochastic action availability when making decisions. The MDP below has two states. Assume the agent starts at state s_1, where two actions (indicated by directed edges for their transitions) are always available: one ($Stay$) stays at s_1, and the other (Go) transitions to state s_2, both with reward $1/2$. At s_2, the action $Down$ returns to s_1, is always available and has reward 0. A second action Up also returns to s_1, but is available with only probability p and has reward 1.



A naive solution that ignores action availability is as follows: we first compute the optimal Q-function assuming all actions are available (this can be derived from the optimal value function, computed using standard techniques). Then at each stage, we use the best action available at the current state where actions are ranked by Q-value. Unfortunately, this leads to a suboptimal policy when the Up action has low availability, specifically if $p < 0.5$.

The best naive policy always chooses to move to s_2 from s_1; at s_2, it picks the best action available. This yields a reward of $1/2$ at even stages, and an expected reward of p at odd stages. However, by anticipating the possibility that action Up is unavailable at s_2, the optimal (SAS) policy always stays at s_1, obtaining reward $1/2$ at all stages. For $p < 1/2$, the latter policy dominates the former: the plot on the right shows the fraction of the optimal (SAS) value *lost* by the naive policy (Std) as a function of the availability probability p. This example also illustrates that as action availability probabilities approach 1, the optimal policy for the base MDP is also optimal for the SAS-MDP.

2.3 Related Work

While a general formulation of MDPs with stochastic action availability does not appear in the literature, there are two strands of closely related work. In the bandits literature, *sleeping bandits* are bandit problems in which the arms available at each stage are determined randomly or adversarially (sleeping experts are similar, with complete feedback than bandit feedback) [Kleinberg *et al.*, 2010; Kanade *et al.*, 2009]. Best action orderings (analogous to our decision list policies for SAS-MDPs) are often used to define regret in these models. The goal is to develop exploration policies to minimize regret. Since these models have no state, if the action reward distributions are known, the optimal policy is trivial: always take the best *available* action. By contrast, a SAS-MDP, even a known model, induces a difficult optimization problem, since the quality of an action depends not just on its immediate reward, but also on the availability of actions at reachable (future) states. This is our focus.

The second closely related branch of research comes from the field of stochastic routing. The "Canadian Traveller Problem"—the problem of minimizing travel time in a graph with unavailable edges—was introduced by Papadimitriou and Yannakakis [1991], who give intractability results (under weaker assumptions about edge availability, e.g. adversarial). Poliyhondrous and Tsitsiklis [1996] consider a stochastic version of the problem, where edge availabilities are random but static (and any edge unavailable remains so throughout the scenario). Most similar to our setting is the work of Nikolova and Karger [2008], who discuss the case of resampling edge costs at each node visit; however, the proposed solution is well-defined only when the edge costs are finite and does not easily extend to unavailable actions/infinite edge costs. Due to the specificity of their modeling assumptions, none of the solutions found in this line of research can be adapted in a straightforward way to SAS-MDPs.

3 Two Reformulations of SAS-MDPs

The randomness of feasible actions means that SAS-MDPs do not conform to the usual definition of an MDP. In this section, we develop two reformulations of SAS-MDPs that transform them into MDPs. We discuss the relative advantages of each, outline key properties and relationships between these models, and describe important special cases of the SAS-MDP model itself.

[1] Models that trigger process termination when $A^{(t)}_s = \emptyset$ are well-defined, but we set aside this model variant here.

3.1 The Embedded MDP

We first consider a reformulation of the SAS-MDP in which we embed the (realized) available action set into the state space itself. This is a straightforward way to recover a standard MDP. The *embedded MDP* $\mathcal{M}_e$ for a SAS-MDP has state space $S_e = \{s \circ A : s \in S, A \subseteq B\}$, with $s \circ A$ having feasible action set A.[2] The history independence of P_s allows transitions to be defined as:

$$p^k_{s\circ A, s'\circ A'} = P(s' \circ A' | s \circ A, k) = p^k_{s,s'} P_{s'}(A'), \ \ \forall k \in A.$$

Rewards are defined similarly: $r^k(s \circ A) = r^k(s)$ for $k \in A$.

In our earlier example, the embedded MDP has three states: $s_1 \circ \{Stay, Go\}, s_2 \circ \{Up, Down\}, s_2 \circ \{Down\}$ (other action subsets have probability 0 hence their corresponding embedded states are unreachable). The feasible actions at each state are given by the embedded action set, and the only stochastic transition occurs when Go is taken at s_1: it moves to $s_2 \circ \{Up, Down\}$ with probability p and $s_2 \circ \{Down\}$ with probability $1 - p$.

Clearly, the induced reward process and dynamics are Markovian, hence $\mathcal{M}_e$ is in fact an MDP under the usual definition. Given the natural translation afforded by the embedded MDP, we view this as providing the basic "semantic" underpinnings of the SAS-MDP model. This translation affords the use of standard MDP analytical tools and methods.

A (stationary, deterministic, Markovian) policy $\pi : S_e \to B$ for $\mathcal{M}_e$ is restricted so that $\pi(s \circ A) \in A$. The policy backup operator T^π_e and Bellman operator T^*_e for $\mathcal{M}_e$ decompose naturally as follows:

$$T^\pi_e V_e(s \circ A_s) = r^{\pi(s\circ A_s)}_s + \gamma \sum_{s'} p^{\pi(s\circ A_s)}_{s,s'} \sum_{A_{s'} \subseteq B} P_{s'}(A_{s'}) V_e(s' \circ A_{s'}), \quad (1)$$

$$T^*_e V_e(s \circ A_s) = \max_{k \in A_s} r^k_s + \gamma \sum_{s'} p^k_{s,s'} \sum_{A_{s'} \subseteq B} P_{s'}(A_{s'}) V_e(s' \circ A_{s'}) \quad (2)$$

Their fixed points, V^π_e and V^*_e respectively, can be expressed similarly.

Obtaining an MDP from an SAS-MDP via action-set embedding comes at the expense of a (generally) exponential blow-up in the size of the state space, which can increase by a factor of $2^{|B|}$.

3.2 The Compressed MDP

The embedded MDP provides a natural semantics for SAS-MDPs, but is problematic from an algorithmic and learning perspective given the state space blow-up. Fortunately, the history independence of the availability distributions gives rise to an effective, compressed representation. The *compressed MDP* $\mathcal{M}_c$ recasts the embedded MDP in terms of the original state space, using expectations to express value functions, policies, and backups over S rather than over the (exponentially larger) S_e. As we will see below, the compressed MDP induces a blow-up in action space rather than state space, but offers significant computational benefits.

[2] Embedded states whose embedded action subsets have zero probability are unreachable and can be ignored.

Formally, the state space for $\mathcal{M}_c$ is S. To capture action availability, the feasible action set for $s \in S$ is the set of *state policies*, or mappings $\mu_s : 2^B \to B$ satisfying $\mu_s(A_s) \in A_s$. In other words, once we reach s, μ_s dictates what action to take for any realized action set A_s. A policy for $\mathcal{M}_c$ is a family $\mu_c = \{\mu_s : s \in S\}$ of such state policies. Transitions and rewards use expectations over A_s:

$$p^{\mu_s}_{s,s'} = \sum_{A_s \subseteq B} P_s(A_s) p^{\mu_s(A_s)}_{s,s'} \text{ and } r^{\mu_s}_s = \sum_{A_s \subseteq B} P_s(A_s) r^{\mu_s(A_s)}_s .$$

In our earlier example, the compressed MDP has only two states, s_1 and s_2. Focusing on s_2, its "actions" in the compressed MDP are the set of state policies, or mappings from the realizable available sets $\{\{Up, Down\}, \{Down\}\}$ into action choices (as above, we ignore unrealizable action subsets). In this case, there are two such state policies: the first selects Up for $\{Up, Down\}$ and (obviously) $Down$ for $\{Down\}$; the second selects $Down$ for $\{Up, Down\}$ and $Down$ for $\{Down\}$.

It is not hard to show that the dynamics and reward process defined above over this compressed state space and expanded action set (i.e., the set of state policies) are Markovian. Hence we can define policies, value functions, optimality conditions, and policy and Bellman backup operators in the usual fashion. For instance, the Bellman and policy backup operators, $T^\star_c$ and T^c_μ, on compressed value functions are:

$$T^*_c V_c(s) = \mathop{\mathbb{E}}_{A_s \subseteq B} \max_{k \in A_s} r^k_s + \gamma \sum_{s'} p^k_{s,s'} V_c(s'), \quad (3)$$

$$T^\mu_c V_c(s) = \mathop{\mathbb{E}}_{A_s \subseteq B} r^{\mu_s(A_s)}_s + \gamma \sum_{s'} p^{\mu_s(A_s)}_{s,s'} V_c(s'). \quad (4)$$

It is easy to see that any state policy μ induces a Markov chain over base states, hence we can define a standard $n \times n$ transition matrix P^μ for such a policy in the compressed MDP, where $p^\mu_{s,s'} = \mathbb{E}_{A \subseteq B} p^{\mu(s)(A)}_{s,s'}$. When additional independence assumptions hold, this expectation over subsets can be computed efficiently (see Section 3.4).

Critically, we can show that there is a direct "equivalence" between policies and their value functions (including optimal policies and values) in $\mathcal{M}_c$ and $\mathcal{M}_e$. Define the action-expectation operator $E : \mathbb{R}^{n2^m} \to \mathbb{R}^n$ to be a mapping that compresses a value function V_e for $\mathcal{M}_e$ into a value function V^e_c for $\mathcal{M}_c$:

$$V^e_c(s) = EV_e(s) = \mathop{\mathbb{E}}_{A_s \subseteq B} V_e(s \circ A_s) = \sum_{A_s \subseteq B} P_s(A_s) V_e(s \circ A_s).$$

We emphasize that E transforms an (arbitrary) value function V_e in embedded space into a new value function V^e_c defined in compressed space (hence, V^e_c is *not* defined w.r.t. $\mathcal{M}_c$).

Lemma 1 *$ET^*_e V_e = T^*_c EV_e$. Hence, T^*_c has a unique fixed point $V^*_c = EV^*_e$.*

Proof:

$$\begin{aligned} ET^e V_e(s) &= \mathop{\mathbb{E}}_{A \subseteq B} T^e V_e(s \circ A) \\ &= \mathop{\mathbb{E}}_{A \subseteq B} \max_{k \in A} r_s^k + \gamma \sum_{s' \circ A'} p^k_{s \circ A, s' \circ A'} V_e(s' \circ A') \\ &= \mathop{\mathbb{E}}_{A \subseteq B} \max_{k \in A} r_s^k + \gamma \sum_{s'} p^k_{s,s'} \mathop{\mathbb{E}}_{A' \subseteq B} V_e(s' \circ A') \\ &= \mathop{\mathbb{E}}_{A \subseteq B} \max_{k \in A} r_s^k + \gamma \sum_{s'} p^k_{s,s'} EV^e(s') \\ &= T^c EV^e(s'). \end{aligned}$$

■

Lemma 2 *Given the optimal value function V_c^* for $\mathcal{M}_c$, the optimal policy π_e^* for $\mathcal{M}_e$ can be constructed directly. Specifically, for any $s \circ A$, the optimal policy $\pi_e^*(s \circ A)$ and optimal value $V_e^*(s \circ A)$ at that embedded state can be computed in polynomial time.*

Proof Sketch: Given $s \circ A$, the expected value of each action in $k \in A$ can be computed using a one-step backup of V_c^*. Then $\pi_e^*(s \circ A)$ is the action with maximum value, and $V_e^*(s \circ A)$ is its backed-up expected value. ■

Therefore, it suffices to work directly with the compressed MDP, which allows one to use value functions (and Q-functions) over the original state space. The price is that one needs to use state policies, since the best action at s depends on the available set A_s. In other words, while the embedded MDP causes an exponential blow-up in state space, the compressed MDP causes an exponential blow-up in action space. We now turn to assumptions that allow us to effectively manage this action space blow-up.

3.3 Decision List Policies

The embedded and compressed MDPs do not, *prima facie*, offer much computational or representational advantage, since they rely on an exponential increase in the size of the state space (embedded MDP) or decision space (compressed MDP). Fortunately, SAS-MDPs have optimal policies with a useful, concise form. We first focus on the policy representation itself, then describe the considerable computational leverage it provides.

A *decision list (DL) policy* μ is a type of policy for $\mathcal{M}_e$ that can be expressed compactly using $O(nm \log m)$ space and executed efficiently. Let Σ_B be the set of permutations over base action set B. A DL policy $\mu : S \to \Sigma_B$ associates a permutation $\mu(s) \in \Sigma_B$ with each state, and is executed at embedded state $s \circ A$ by executing $\min\{i \in \{1, \ldots, m\} : \mu(s)(i) \in A\}$. In other words, whenever base state s is encountered and A is the available set, the first action $k \in A$ in the order dictated by DL $\mu(s)$ is executed. Equivalently, we can view $\mu(s)$ as a state policy μ_s for s in $\mathcal{M}_c$. In our earlier example, one DL $\mu(s_2)$ is $[Up, Down]$, which requires taking (base) action Up if it is available, otherwise taking $Down$.

For any SAS-MDP, we have optimal DL policies:

Theorem 1 *$\mathcal{M}_e$ has an optimal policy that can be represented using a decision list. The same policy is optimal for the corresponding $\mathcal{M}_c$.*

Proof Sketch: Let V^* be the (unique) optimal value function for $\mathcal{M}_e$ and Q^* its corresponding Q-function (see Sec. 5.1 for a definition). A simple inductive argument shows that no DL policy is optimal only if there is some state s, action sets $A \neq A'$, and (base) actions $j \neq k$, s.t. (i) $j, k \in A, A'$; (ii) for some optimal policy $\pi^*(s \circ A) = j$ and $\pi^*(s \circ A') = k$; and (iii) either $Q^*(s \circ A, j) > Q^*(s \circ A, k)$ or or $Q^*(s \circ A', k) > Q^*(s \circ A', j)$. However, the fact that the optimal Q-value of any action $k \in A$ at state $s \circ A$ is independent of the other actions in A (i.e., it depends only on the base state) implies that these conditions are mutually contradictory. ■

3.4 The Product Distribution Assumption

The DL form ensures that optimal policies and value functions for SAS-MDPs can be expressed polynomially in the size of the base MDP $\mathcal{M}$. However, their computation still requires the computation of expectations over action subsets, e.g., in Bellman or policy backups (Eqs. 3, 4). This will generally be infeasible without some assumptions on the form the action availability distributions P_s.

One natural assumption is the *product distribution assumption (PDA)*. PDA holds when $P_s(A)$ is a product distribution where each action $k \in B$ is available with probability ρ_s^k, and subset $A \subseteq B$ has probability $\rho_s^A = \prod_{k \in A} \rho_s^k \prod_{k \in B \setminus A} (1 - \rho_s^k)$. This assumption is a reasonable approximation in the settings discussed above, where state-independent exogenous processes determine the availability of actions (e.g., the probability that one advertiser's campaign has budget remaining is roughly independent of another advertiser's). For ease of notation, we assume that ρ_s^k is identical for all states s (allowing different availability probabilities across states has no impact on what follows). To ensure the MDP is well-founded, we assume some default action (e.g., no-op) is always available.[3] Our earlier running example trivially satisifes PDA: at s_2, Up's availability probability (p) is independent of the availability of $Down$ (1).

When the PDA holds, the DL form of policies allows the expectations in policy and Bellman backups to be computed efficiently without enumeration of subsets $A \subseteq B$. For example, given a fixed DL policy μ, we have

$$T_c^\mu V_c(s) = \sum_{i=1}^{m} \left[\prod_{j=1}^{i-1} (1 - \rho_s^{\mu(s)(j)}) \right] \rho_s^{\mu(s)(i)} \left(r_s^{\mu(s)(i)} + \gamma \sum_{s'} p^{\mu(s)(i)}_{s,s'} V_c(s') \right). \quad (5)$$

The Bellman operator has a similar form. We exploit this below to develop tractable value iteration and policy iteration algorithms, as well as a practical LP formulation.

3.5 Arbitrary Distributions with Sampling (ADS)

We can also handle the case where, at each state, the availability distribution is unknown, but is sampleable. Using sam-

[3] We omit the default action from analysis for ease of exposition.

ples to approximate expectations w.r.t. available action subsets provides a means to estimate values and approximate optimal policies. Critically, the required sample size is polynomial in $|B|$, and not in the size of the *support* of the distribution (see below). Of course, this approach does not allow us to compute the optimal policy exactly. However, it has important implications for the sample complexity of learning algorithms like Q-learning.

We note that the ability to sample available action subsets is quite natural in many domains. For instance, in ad domains, it may not be possible to model the process by which eligible ads are generated (e.g., specific and evolving advertiser targeting criteria, budgets, frequency capping, etc.). But the eligible subset of ads considered for each impression opportunity is an action subset sampled from this process.

Under ADS, we compute approximate backup operators as follows. Let $\mathcal{A}_s = \{A_s^{(1)}, \ldots, A_s^{(T)}\}$ be an i.i.d. sample of size T of action subsets in state s. For a subset of actions A, an index i and a decision list μ, define $I_{[i,A,\mu]}$ to be 1 if $\mu(i) \in A$ and for each $j < i$ we have $\mu(j) \notin A$, or 0 otherwise. Similar to Eq. (5), we define:

$$T_c^{\mu} V_c(s) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{m} I_{\left[i, A_s^{(t)}, \mu(s)\right]} \left(r_s^{\mu(s)(i)} + \gamma \sum_{s'} p_{s,s'}^{\mu(s)(i)} V_c(s') \right).$$

We now consider the quality of of the policies generated using this approximation (see the longer paper [Boutilier *et al.*, 2018] for proofs and more details). The following lemma is a direct application of Hoeffding's concentration inequality along with a union bound (for any $0 < \delta < 1$ and error tolerance ϵ):

Lemma 3 *Given samples $A_s^{(1)}, \ldots, A_s^{(T)}$ for each $s \in S$, if*

$$T = \Omega\left(\frac{m + \log(n/\delta)}{(1-\gamma)^2 \epsilon^2} \right),$$

then with probability $1 - \delta$, for each DL policy μ we have:

$$\left| \mathop{\mathbb{E}}_{A \subseteq B} Q^{\mu}(s, \mu_s(A)) - \frac{1}{T} \sum_{t=1}^{T} Q^{\mu}(s, \mu_s(A_s^{(t)})) \right| \leq \frac{\epsilon(1-\gamma)}{2\gamma}.$$

The action-set samples induce an *approximate SAS-MDP*, defined using the empirical distributions above. Under the conditions stated, Lemma 3 leads to bounds on the quality of the optimal policy in the approximate SAS-MDP:

Theorem 2 *Let $\hat{\mu}$ be the optimal policy for the approximate SAS-MDP and Q^* the optimal Q-function for the true SAS-MDP. With probability $1 - \delta$ we have, for each $s \in S, k \in B$, $Q^{\hat{\mu}}(s,k) \geq Q^{\star}(s,k) - \epsilon$.*

In the sequel, we focus largely on PDA; in most cases equivalent results can be derived in the ADS model.

4 Q-Learning in the Compressed MDP

Suppose we are faced with learning the optimal value function or policy for an SAS-MDP from a collection of trajectories. The (implicit) learning of the transition dynamics and rewards can proceed as usual; the novel aspect of the SAS model is that the action availability distribution must also be considered. Remarkably, Q-learning can be readily augmented to incorporate stochastic action sets: we require only that our training trajectories are augmented with the set of actions that were available at each state,

$$\ldots s^{(t)}, A^{(t)}, k^{(t)}, r^{(t)}, s^{(t+1)}, A^{(t+1)}, k^{(t+1)}, r^{(t+1)}, \ldots,$$

where: $s^{(t)}$ is the realized state at time t (drawn from distribution $P(\cdot | s^{(t-1)}, k^{(t-1)})$); $A^{(t)}$ is the realized available set at time t, drawn from $P_{s^{(t)}}$; $k^{(t)} \in A^{(t)}$ is the action taken; and $r^{(t)}$ is the realized reward. Such augmented trajectory data is typically available. In particular, the required sampling of available action sets is usually feasible (e.g., in ad serving as discussed above).

SAS-Q-learning can be applied directly to the compressed MDP $\mathcal{M}_c$, requiring only a minor modification of the standard Q-learning update for the base MDP. We simply require that each Q-update maximize over the *realized available actions* $A^{(t+1)}$:

$$Q^{new}(s^{(t)}, k^{(t)}) \leftarrow (1 - \alpha_t) Q^{old}(s^{(t)}, k^{(t)}) + \alpha_t [r^{(t)} + \gamma \max_{k \in A^{(t+1)}} Q^{old}(s^{(t+1)}, k)].$$

Here Q^{old} is the previous Q-function estimate and Q^{new} is the updated estimate, thus it encompasses both online and batch Q-learning, experience replay, etc.; and $0 \leq \alpha_t < 1$ is our (adaptive) learning rate.

It is straightforward to show that, under the usual exploration conditions, SAS-Q-learning will converge to the optimal Q-function for the compressed MDP, since the expected maximum over sampled action sets at any particular state will converge to the expected maximum at that state.

Theorem 3 *The SAS-Q-learning algorithm will converge w.p. 1 to the optimal Q-function for the (discounted, infinite-horizon) compressed MDP $\mathcal{M}_c$ if the usual stochastic approximation requirements are satisfied. That is, if (a) rewards are bounded and (b) the subsequence of learning rates $\alpha_{t(s,k)}$ applied to (s,k) satisfies $\sum \alpha_{t(s,k)} = \infty$ and $\sum \alpha_{t(s,k)}^2 < \infty$ for all state-action pairs (s,k) (see, e.g., [Watkins and Dayan, 1992]).*

Moreover, function approximation techniques, such as DQN [Mnih *et al.*, 2015], can be directly applied with the same action set-sample maximization. Implementing an optimal policy is also straightforward: given a state s and the realization A_s of the available actions, one simply executes $\arg\max_{k \in A_s} Q(s,k)$.

We note that extracting the optimal value function $V_c(s)$ for the compressed MDP from the learned Q-function is not viable without some information about the action availability distribution. Fortunately, one need not know the expected value at a state to implement the optimal policy.[4]

[4] It is, of course, straightforward to learn an optimal value function if desired.

5 Value Iteration in the Compressed MDP

Computing a value function for $\mathcal{M}_c$, with its "small" state space S, suffices to execute an optimal policy. We develop an efficient *value iteration (VI)* method to do this.

5.1 Value Iteration

Solving an SAS-MDP using VI is challenging in general due to the required expectations over action sets. However, under PDA, we can derive an efficient VI algorithm whose complexity depends only polynomially on the base set size $|B|$.

Assume a current iterate V^t, where $V^t(s) = \mathbb{E}_{A_s}[\max_{k \in A_s} Q^t(s,k)]$. We compute V^{t+1} as follows:

- For each $s \in S, k \in B$, compute its $(t+1)$-stage-to-go Q-value: $Q^{t+1}(s,k) = r_s^k + \gamma \sum_{s'} p_{s,s'}^k V^t(s')$.
- Sort these Q-values in descending order. For convenience, we re-index each action by its Q-value rank (i.e., $k_{(1)}$ is the action with largest Q-value, and $\rho_{(1)}$ is its probability, $k_{(2)}$ the second-largest, etc.).
- For each $s \in S$, compute its $(t+1)$-stage-to-go value:

$$V^{t+1}(s) = \mathbb{E}_{A_s}\left[\max_{k \in A_s} Q^{t+1}(s,k)\right] = \sum_{i=1}^{m-1}\left(\prod_{j=1}^{i-1}(1-\rho_{(j)})\right)\rho_{(i)} Q^{t+1}(s,k_{(i)}).$$

Under ADS, we use the approximate Bellman operator:

$$\widehat{V}^{t+1}(s) = \mathbb{E}_{A_s}\left[\max_{k \in A_s} \widehat{Q}^{t+1}(s,k)\right] = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{m} I_{\left[i, A_s^{(t)}, \mu(s)\right]} \widehat{Q}^{t+1}(s, \mu(s)(i)) \ ,$$

where $\mu(s)$ is the DL resulting from sorting $\widehat{Q}^{t+1}$-values.

The Bellman operator under PDA is tractable:

Observation 1 *The compressed Bellman operator T_c^* can be computed in $O(nm \log m)$ time.*

Therefore the per-iteration time complexity of VI for $\mathcal{M}_c$ compares favorably to the $O(nm)$ time of VI in the base MDP. The added complexity arises from the need to sort Q-values.[5] Conveniently, this sorting process immediately provides the desired DL state policy for s.

Using standard arguments, we obtain the following results, which immediately yield a polytime approximation method.

Lemma 4 *T_c^* is a contraction with modulus γ i.e., $||T_c^* v_c - T_c^* v_c'|| \leq \gamma ||v_c - v_c'||$.*

Corollary 1 *For any precision $\varepsilon < 1$, the compressed value iteration algorithm converges to an ε-approximation of the optimal value function in $O(\log(L/\varepsilon))$ iterations, where $L \leq [\max_{s,k} r_s^k]/(1-\gamma)$ is an upper bound on $||V_e^*||$.*

We provide an even stronger result next: VI, in fact, converges to an *optimal* solution in polynomial time.

[5] The products of the action availability probabilities can be computed in linear time via caching.

5.2 The Complexity of Value Iteration

Given its polytime per-iteration complexity, to ensure VI is polytime, we must show that it converges to a value function that induces an optimal policy in polynomially many iterations. To do so, we exploit the compressed representation and adapt the technique of [Tseng, 1990].

Assume, w.r.t. the base MDP $\mathcal{M}$, that the discount factor γ, rewards r_s^k, and transition probabilities $p_{s,s'}^k$, are rational numbers represented with a precision of $1/\delta$ (δ is an integer). Tseng shows that VI for a standard MDP is strongly polynomial, assuming constant γ and δ, by proving that: (a) if the t'th value function produced by VI satisfies

$$||V^t - V^*|| < 1/(2\delta^{2n+2}n^n),$$

then the policy induced by V^t is optimal; and (b) VI achieves this bound in polynomially many iterations.

We derive a similar bound on the number of VI iterations needed for convergence in an SAS-MDP, using the same input parameters as in the base MDP, and applying the same precision δ to the action availability probabilities. We apply Tseng's result by exploiting the fact that: (a) the optimal policy for the embedded MDP $\mathcal{M}_e$ can be represented as a DL; (b) the transition function for any DL policy can be expressed using an $n \times n$ matrix (we simply take expectations, see above); and (c) the corresponding linear system can be expressed over the *compressed* rather than the embedded state space to determine V_c^* (rather than V_e^*).

Tseng's argument requires some adaptation to apply to the compressed VI algorithm. We extend his precision assumption to account for our action availability probabilities as well, ensuring ρ_s^k is also represented up to precision of $1/\delta$.

Since $\mathcal{M}_c$ is an MDP, Tseng's result applies; but notice that each entry of the transition matrix for any state's DL μ, which serves as an action in $\mathcal{M}_c$, is a product of $m+1$ probabilities, each with precision $1/\delta$. We have that $p_{s,s'}^{\mu}$ has precision of $1/\delta^{m+1}$. Thus the required precision parameter for our MDP is at most δ^{m+1}. Plugging this into Tseng's bound, VI applied to $\mathcal{M}_c$ must induce an optimal policy at the t'th iteration if

$$||V^t - v^*|| < 1/(2(\delta^{(m+1)})^{2n}n^n) = 1/(2\delta^{(m+1)2n}n^n) \ .$$

This in turn gives us a bound on the number of iterations of VI needed to reach an optimal policy:

Theorem 4 *VI applied to $\mathcal{M}_c$ converges to a value function whose greedy policy is optimal in t^* iterations, where*

$$t^* \leq \log(2\delta^{2n(m+1)}n^n M)/\log(1/\gamma)$$

Combined with Obs. 1, we have:

Corollary 2 *VI yields an optimal policy for the SAS-MDP corresponding to $\mathcal{M}_c$ in polynomial time.*

Under ADS, VI merely approximates the optimal policy. In fact, one cannot compute an exact optimal policy without observing the entire support of the availability distributions (requiring exponential sample size).

6 Policy Iteration in the Compressed MDP

We now outline a policy iteration (PI) algorithm.

6.1 Policy Iteration

The concise DL form of optimal policies can be exploited in PI as well. Indeed, *the greedy policy π^V with respect to any value function V in the compressed space* is representable as a DL. Thus the policy improvement step of PI can be executed using the same independent evaluation of action Q-values and sorting as used in VI above:

$$Q^V(s,k) = r(s,k) + \gamma \sum_{s'} p^k_{s,s'} V(s'),$$

$$Q^V(s,A_s) = \max_{k \in A_s} Q^V(s,k) \text{ , and } \pi^V(s,A_s) = \arg\max_{k \in A_s} Q^V(s,k).$$

The DL policy form can also be exploited in the policy evaluation phase of PI. The tractability of policy evaluation requires a tractable representation of the action availability probabilities, which PDA provides, leading to the following PI method that exploits PDA:

1. Initialize an arbitrary policy π in decision list form.
2. Evaluate π by solving the following linear system over variables $V^\pi(s), \forall s \in S$: (Note: We use $Q^\pi(s,k)$ to represent the relevant linear expression over V^π.)

$$V^\pi(s) = \sum_{i=1}^{n} [\prod_{j=1}^{i-1} (1 - \rho_{(j)})] \, \rho_{(i)} Q^\pi(s, k_{(i)})$$

3. Let π' denote the greedy policy w.r.t. V^π, which can be expressed in DL form for each s by sorting Q-values $Q^\pi(s,k)$ as above (with standard tie-breaking rules). If $\pi'(s) = \pi(s)$, terminate; otherwise replace π with π' and repeat (Steps 2 and 3).

Under ADS, PI can use the approximate Bellman operator, giving an approximately optimal policy.

6.2 The Complexity of Policy Iteration

The per-iteration complexity of PI in $\mathcal{M}_c$ is polynomial: as in standard PI, policy evaluation solves an $n \times n$ linear system (naively, $O(n^3)$) plus the additional overhead (linear in M) to compute the compounded availability probabilities; and policy improvement requires $O(mn^2)$ computation of action Q-values, plus $O(nm \log m)$ overhead for sorting Q-values (to produce improving DLs for all states).

An optimal policy is reached in a number of iterations no greater than that required by VI, since: (a) the sequence of value functions for the policies generated by PI contracts at least as quickly as the value functions generated by VI (see, e.g., [Meister and Holzbaur, 1986; Hansen *et al.*, 2013]); (b) our precision argument for VI ensures that the greedy policy extracted at that point will be optimal; and (c) once PI finds an optimal policy, it will terminate (with one extra iteration). Hence, PI is polytime (assuming a fixed discount $\gamma < 1$).

Theorem 5 *PI yields an optimal policy for the SAS-MDP corresponding to $\mathcal{M}_c$ in polynomial time.*

In the longer version of the paper [Boutilier *et al.*, 2018], we adapt more direct proof techniques [Ye, 2011; Hansen *et al.*, 2013] to derive polynomial-time convergence of PI for SAS-MDPs under additional assumptions. Concretely, for a policy μ and actions k_1, k_2, let $\eta_\mu(s, k_1, k_2)$ be the probability, over action sets, that at state s, the optimal $\mu^\star$ selects k_1 and μ selects k_2. Let $q > 0$ be such that $\eta_\mu(s, k_1, k_2) \geq q$ whenever $\eta_\mu(s, k_1, k_2) > 0$. We show:

Theorem 6 *The number of iterations it takes policy iteration to converge is no more than*

$$O\left(\frac{nm^2}{1-\gamma} \log \frac{m}{1-\gamma} \log \frac{e}{q}\right).$$

Under PDA, the theorem implies *strongly-polynomial* convergence of PI if each action is available with constant probability. In this case, for any μ, k_i, k_j, and s, we have $\eta_\mu(s, k_i, k_j) \geq \rho_s^{k_i} \cdot \rho_s^{k_j} = \Omega(1)$, which in turn implies that we can take $q = \Omega(1)$ in the bound above.

7 Linear Programming in the Compressed MDP

An alternative model-based approach is linear programming (LP). The primal formulation for the embedded MDP $\mathcal{M}_e$ is straightforward (since it is a standard MDP), but requires exponentially many variables (one per embedded state) and constraints (one per embedded state, base action pair).

A (nonlinear) primal formulation for the compressed MDP $\mathcal{M}_c$ reduces the number of variables to $|S|$:

$$\min_{\mathbf{v}} \sum\nolimits_{s \in S} \alpha_s v_s, \quad \text{s.t. } v_s \geq \mathbb{E}_{A_s} \max_{k \in A_s} Q(s,k) \quad \forall s. \tag{6}$$

Here α is an arbitrary, positive state-weighting, over the embedded states corresponding to each base state and

$$Q(s,k) = r^k_s + \sum_{s' \in S} p^k_{s,s'} v_{s'}$$

abbreviates the linear expression of the action-value backup at the state and action in question w.r.t. the value variables v_s. This program is valid given the definition of $\mathcal{M}_c$ and the fact that a weighting over embedded states corresponds to a weighting over base states by taking expectations. Unfortunately, this formulation is non-linear, due to the max term in each constraint. And while it has only $|S|$ variables, it has factorially many constraints; moreover, the constraints themselves are not compact due to the presence of the expectation in each constraint.

PDA can be used to render this formulation tractable. Let σ denote an arbitrary (inverse) permutation of the action set (so $\sigma(i) = j$ means that action j is ranked in position i). As above, the optimal policy at base state s w.r.t. a Q-function is expressible as a DL (with actions sorted by Q-values) and its expected value given by the expression derived below. Specifically, if σ reflects the relative ranking of the (optimal) Q-values of the actions at some fixed state s, then $V(s) = Q(s, \sigma(1))$ with probability $\rho_{\sigma(1)}$, i.e., the probability that $\sigma(1)$ occurs in A_s. Similarly, $V(s) = Q(s, \sigma(2))$ with probability $(1 - \rho_{\sigma(1)})\rho_{\sigma(2)}$, and so on. We define the Q-value of a DL σ as follows:

$$Q^V_s(\sigma) = \sum_{i=1}^{n} [\prod_{j=1}^{i-1} (1 - \rho_{\sigma(j)})] \, \rho_{\sigma(i)} Q^V(s, \sigma(i)). \tag{7}$$

Thus, for any fixed action permutation σ, the constraint that v_s at least matches the expectation of the maximum action's Q-value is linear. Hence, the program can be recast as an LP by enumerating action permutations for each base state, replacing the constraints in Eq. (6) as follows:

$$v_s \geq Q_s^V(\sigma) \quad \forall s \in S, \forall \sigma \in \Sigma. \tag{8}$$

The constraints in this LP are now each compactly represented, but it still has factorially many constraints. Despite this, it can be solved in polynomial time. First, we observe that the LP is well-suited to constraint generation. Given a relaxed LP with a subset of constraints, a greedy algorithm that simply sorts actions by Q-value to form a permutation can be used to find the maximally violated constraint at any state. Thus we have a practical constraint generation algorithm for this LP since (maximally) violated constraints can be found in polynomial time.

More importantly from a theoretical standpoint, the constraint generation algorithm can be used as a separation oracle within an ellipsoid method for this LP. This directly yields an exact, (weakly) polynomial time algorithm for this LP [Grötschel *et al.*, 1988].

8 Empirical Illustration

We now provide a somewhat more substantial empirical demonstration of the effects of stochastic action availability. Consider an MDP that corresponds to a routing problem on a real-world road network (Fig. 1) in the San Francisco Bay Area. The shortest path between the source and destination locations is sought. The dashed edge in Fig. 1 represents a bridge, available only with probability p, while all other edges correspond to action choices available with probability 0.5. At each node, a no-op action (waiting) is available at constant cost; otherwise the edge costs are the geodesic lengths of the corresponding roads on the map. The optimal policies for different choices $p = 0.1, 0.2$ and 0.4 are depicted in Fig. 1, where line thickness and color indicate traversal probabilities under the corresponding optimal policies. We see that lower values of p lead to policies with more redundancy (i.e., more alternate routes).

Fig. 2 shows the effect of solving the routing problem when ignoring stochastic action availability (i.e., assuming actions are always available). The SAS-optimal policy allows graceful scaling of the expected travel time from source to destination as bridge availability decreases. The effects of violating the PDA assumption are also investigated in the longer version of this paper [Boutilier *et al.*, 2018].

9 Concluding Remarks

We have developed a new MDP model, *SAS-MDPs*, that extends the usual finite-action MDP model by allowing the set of available actions to vary stochastically. This captures an important use case that arises in many practical applications (e.g., online advertising, recommender systems). We have shown that embedding action sets in the state gives a standard MDP, supporting tractable analysis at the cost of an exponential blow-up in state space size. Despite this, we demonstrated that (optimal and greedy) policies have a useful decision list

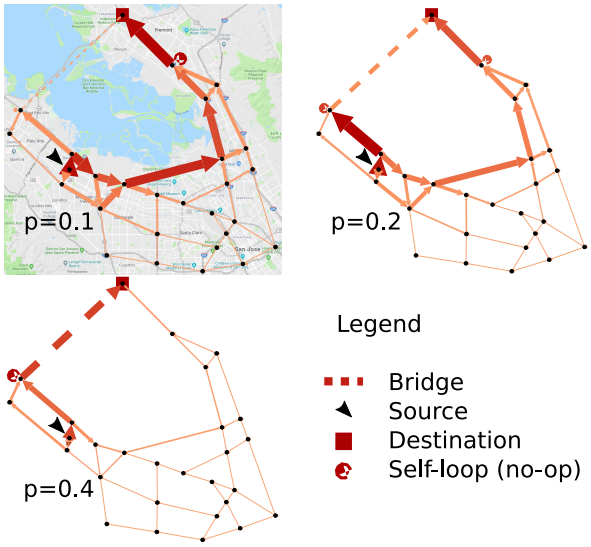


Figure 1: Stochastic action MDPs applied to routing.

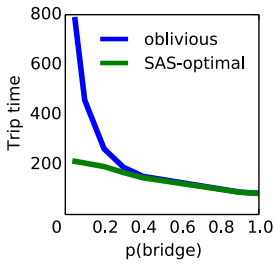


Figure 2: Expected trip time from source to destination under the SAS-optimal policy vs. under the oblivious optimal policy (the MDP solved as if actions are fully available) as a function of bridge availability.

structure. We showed how this DL format can be exploited to construct tractable Q-learning, value and policy iteration, and linear programming algorithms.

While our work offers firm foundations for stochastic action sets, most practical applications will not use the algorithms described here explicitly. For example, in RL, we generally use function approximators for generalization and scalability in large state/action problems. We have successfully applied Q-learning using DNN function approximators (i.e., DQN) using sampled/logged available actions in ads and recommendations domains as described in Sec. 4. This has allowed us to apply SAS-Q-learning to problems of significant, commercially viable scale. Model-based methods such as VI, PI, and LP also require suitable (e.g., factored) representations of MDPs and structured implementations of our algorithms that exploit these representations. For instance, extensions of approximate linear programming or structured dynamic programming to incorporate stochastic action sets would be extremely valuable.

Other important questions include developing a polynomial-*sized* direct LP formulation; and deriving sample-complexity results for RL algorithms like Q-learning is also of particular interest, especially as it pertains to the

sampling of the action distribution. Finally, we are quite interested in relaxing the strong assumptions embodied in the PDA model—of particular interest is the extension of our algorithms to less extreme forms of action availability independence, for example, as represented using concise graphical models (e.g., Bayes nets).

Acknowledgments

Thanks to the reviewers for their helpful suggestions.

References

[Amin *et al.*, 2012] K. Amin, M. Kearns, P. Key, and A. Schwaighofer. Budget optimization for sponsored search: Censored learning in MDPs. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pp.543–553, Catalina, CA, 2012.

[Archak *et al.*, 2010] N. Archak, V. S. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. In *Proceedings of the 19th Intl. World Wide Web Conference (WWW 2010)*, pp.31–40, Raleigh, NC, 2010.

[Archak *et al.*, 2012] N. Archak, V. Mirrokni, and S. Muthukrishnan. Budget optimization for online campaigns with positive carryover effects. In *Proceedings of the 8th Intl. Workshop on Internet and Network Economics (WINE-12)*, pp.86–99, Liverpool, 2012.

[Boutilier *et al.*, 2018] C. Boutilier, A. Cohen, A. Hassidim, Y. Mansour, O. Meshi, M. Mladenov, and D. Schuurmans. Planning and learning in Markov decision processes with stochastic action sets. Technical Report arXiv:1805.02363 [cs.AI], ArXiV, May 2018.

[Charikar *et al.*, 1999] M. Charikar, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. On targeting Markov segments. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC-99)*, pp.99–108, Atlanta, 1999.

[Grötschel *et al.*, 1988] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, Vol. 2 of *Algorithms and Combinatorics*. Springer, 1988.

[Hansen *et al.*, 2013] T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM)*, 60(1), 2013. Article 1, 16pp.

[Kanade *et al.*, 2009] V. Kanade, H. B. McMahan, and B. Bryan. Sleeping experts and bandits with stochastic action availability and adversarial rewards. In *12th Intl. Conference on Artificial Intelligence and Statistics (AIStats-09)*, pp.272–279, Clearwater Beach, FL, 2009.

[Kleinberg *et al.*, 2010] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2–3):245–272, 2010.

[Li *et al.*, 2009] T. Li, N. Liu, J. Yan, G. Wang, F. Bai, and Z. Chen. A Markov chain model for integrating behavioral targeting into contextual advertising. In *Proceedings of the 3rd Intl. Workshop on Data Mining and Audience Intelligence for Advertising*, pp.1–9, Paris, 2009.

[Meister and Holzbaur, 1986] U. Meister and U. Holzbaur. A polynomial time bound for Howard's policy improvement algorithm. *OR Spektrum*, 8:37–40, 1986.

[Mladenov *et al.*, 2017] M. Mladenov, C. Boutilier, D. Schuurmans, O. Meshi, G. Elidan, and T. Lu. Logistic Markov decision processes. In *Proceedings of the 26th Intl. Joint Conference on Artificial Intelligence (IJCAI-17)*, pp.2486–2493, Melbourne, 2017.

[Mnih *et al.*, 2013] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Mnih *et al.*, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A Rusu, J. Veness, Marc G Bellemare, A. Graves, M. Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Nikolova and Karger, 2008] E. Nikolova and D. R. Karger. Route planning under uncertainty: The Canadian traveller problem. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pp.969–974, 2008.

[Papadimitriou and Yannakakis, 1991] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127 – 150, 1991.

[Polychronopoulos and Tsitsiklis, 1996] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, 1996.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

[Silver *et al.*, 2013] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall. Concurrent reinforcement learning from customer interactions. In *Proceedings of the 30th Intl. Conference on Machine Learning (ICML-13)*, pp.924–932, Atlanta, 2013.

[Theocharous *et al.*, 2015] G. Theocharous, P. S. Thomas, and M. Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. In *Proceedings of the 24th Intl. Joint Conference on Artificial Intelligence (IJCAI-15)*, pp.1806–1812, Buenos Aires, 2015.

[Tseng, 1990] P. Tseng. Solving h-horizon, stationary Markov decision problems in time proportional to log(h). *Operations Research Letters*, 9(5):287–297, 1990.

[Watkins and Dayan, 1992] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[Ye, 2011] Y. 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.