

# Randomized Linear Programming Solves the Discounted Markov Decision Problem In Nearly-Linear (Sometimes Sublinear) Run Time

Mengdi Wang

Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ  
email: mengdiw@princeton.edu

September 1, 2017

## Abstract

We propose a novel randomized linear programming algorithm for approximating the optimal policy of the discounted Markov decision problem. By leveraging the value-policy duality and binary-tree data structures, the algorithm adaptively samples state-action-state transitions and makes exponentiated primal-dual updates. We show that it finds an  $\epsilon$ -optimal policy using nearly-linear run time in the worst case. When the Markov decision process is ergodic and specified in some special data formats, the algorithm finds an  $\epsilon$ -optimal policy using run time linear in the total number of state-action pairs, which is sublinear in the input size. These results provide a new venue and complexity benchmarks for solving stochastic dynamic programs.

**Keywords:** Markov decision process, randomized algorithm, linear programming, duality, primal-dual method, run-time complexity, stochastic approximation

## 1 Introduction

Markov decision process (MDP) is a fundamental model for sequential decision-making problems in dynamic and random environments. It models a stochastic control process in which a planner aims to make a sequence of decisions as the state of the process evolves. MDP serves as the basic mathematical framework for dynamic programming, stochastic control and reinforcement learning. It is widely applied in engineering systems, artificial intelligence, e-commerce and finance.

We focus on the Discounted Markov Decision Problem (DMDP) in which one aims to make an infinite sequence of decisions and optimize some cumulative sum of discounted rewards. An instance of the DMDP can be described by a tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$ , where  $\mathcal{S}$  is a finite state space of size  $|\mathcal{S}|$ ,  $\mathcal{A}$  is a finite action space of size  $|\mathcal{A}|$ ,  $\gamma \in (0, 1)$  is a discount factor,  $\mathcal{P}$  is the collection of state-to-state transition probabilities  $\mathcal{P} = \{p_{ij}(a) \mid i, j \in \mathcal{S}, a \in \mathcal{A}\}$ ,  $\mathbf{r}$  is the collection of state transitional rewards  $\mathbf{r} = \{r_{ij}(a) \mid i, j \in \mathcal{S}, a \in \mathcal{A}\}$  where  $r_{ij}(a) \in [0, 1]$ . We also denote by  $\mathbf{r}_a \in \mathbb{R}^{|\mathcal{S}|}$  the vector of expected state-transition reward under action  $a$ , where  $\mathbf{r}_{a,i} = \sum_{j \in \mathcal{S}} p_{ij}(a)r_{ij}(a)$ . Suppose that the decision process is in state  $i$ , if action  $a$  is selected, the process moves to a next state  $j$  with probability  $p_{ij}(a)$  and generates a reward  $r_{ij}(a)$ . The input size of the DMDP tuple  $\mathcal{M}$  is  $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ .

Our goal is to find the best sequence of actions to choose at all possible states in order to maximize the expected cumulative reward. More precisely, we want to find a (stationary) policy that specifies which action to choose at each state. A stationary and randomized policy can be represented by a collection of probability distributions  $\pi = \{\pi_i\}_{i \in \mathcal{S}}$ , where  $\pi_i : \mathcal{A} \mapsto [0, 1]$  is a vector of probability distribution over actions at state  $i$ . We denote by  $P^\pi$  the transition probability matrix of the DMDP under a fixed policy  $\pi$ , where  $P_{ij}^\pi = \sum_{a \in \mathcal{A}} \pi_i(a)p_{ij}(a)$  for all  $i, j \in \mathcal{S}$ . The objective of the DMDP is to find an optimal policy  $\pi^*$  such that the infinite-horizon sum of discounted rewards is maximized regardless of the initial state  $i_0$ :

$$\max_{\pi} \mathbf{E}^{\pi} \left[ \sum_{t=1}^{\infty} \gamma^t r_{i_t i_{t+1}}(a_t) \mid i_0 \right],$$

where  $\{i_0, a_0, i_1, a_1, \dots, i_t, a_t, \dots\}$  are state-action transitions generated by the Markov decision process under the fixed policy  $\pi$ , and the expectation  $\mathbf{E}^\pi[\cdot]$  is taken over the entire trajectory. In total there are  $|\mathcal{A}|^{|\mathcal{S}|}$  distinct deterministic policies.

Despite its strong power of modeling, MDP is generally considered as a difficult problem due to the *curse of dimensionality*, especially for problems with large state and action spaces. There have been tremendous efforts in analyzing the complexity of MDP and its solution methods. Most existing studies focus on deterministic methods that find the exact optimal policy. Due to the curse of dimensionality, finding the exact optimal policy is often prohibitively difficult, especially in large-scale applications such as computer games and robotics. We ask the following question:

*Is there a way to trade the precision of the exact optimal policy for a better time complexity?*

Motivated by this question, we are interested in developing randomized algorithms that can approximate the optimal policy efficiently. In particular, we are interested in reducing the complexity's dependence on  $|\mathcal{S}|$  and  $|\mathcal{A}|$  - sizes of the state and action spaces. Throughout this paper, we measure the run-time complexity of an algorithm in terms of the total number of arithmetic operations, which include query to a specific entry of the input, addition, subtraction, multiplication and exponentiation. We use  $\mathcal{O}(1)$  to denote some absolute constant number, and we use  $\tilde{\mathcal{O}}(1)$  to hide  $\text{polylog}\left(|\mathcal{S}|, |\mathcal{A}|, \frac{1}{\epsilon}, \frac{1}{1-\gamma}\right)$  factors.

## 1.1 Our Approach and Technical Novelties

In this paper, we develop a randomized linear programming method for solving the DMDP. It can be viewed as a special stochastic primal-dual method that takes advantages of three features: (1) adaptive action sampling according to the current randomized policy; (2) multiplicative policy updates using information projection onto a specifically constructed constraint set; and (3) using binary-tree data structures to simulate state transitions and make policy updates in nearly constant time. Let us outline the development of our method and its analysis.

1. Our starting point is to formulate the nonlinear Bellman equation for the DMDP into a stochastic saddle point problem (see Section 3). The primal and dual variables correspond to the value and the policy, respectively. Our saddle point formulation involves specially chosen constraints and a weight vector, which are crafted to incorporate structural information and prior knowledge (if any) about the DMDP, such as the discount factor, magnitudes of reward, and range of ergodic distributions. In particular, the dual constraint can be viewed as an information set that contains all possible randomized policies and facilitates  $\tilde{\mathcal{O}}(1)$ -time projection with respect to the some variant of the relative entropy.
2. To aid the algorithm design, we develop two programming techniques: (1) We show that by processing the input transition probabilities into binary trees (using  $\tilde{\mathcal{O}}(|\mathcal{S}|^2|\mathcal{A}|)$  time), one can sample a single state transition of the Markov decision process using  $\tilde{\mathcal{O}}(1)$  arithmetic operations; (2) We show that a randomized policy can be represented using tree data structures, such that each coordinate update can be made in  $\tilde{\mathcal{O}}(1)$  time and each random action can be sampled in  $\tilde{\mathcal{O}}(1)$  time. These two sampling techniques enable us to develop randomized algorithms that simulate state transitions and make appropriate dual update in  $\tilde{\mathcal{O}}(1)$  time.
3. Our randomized algorithm has two main components. First, it uses adaptive importance sampling of state-action-state triplets. In other words, it simulates the Markov decision process using the current dual variable as the control policy to balance the exploration-exploitation tradeoff in estimating the optimal policy. Second, our method iteratively makes exponentiated and re-weighted updates in the dual variable and projects it onto an information set with respect to a specific divergence function. Here the update rule and the divergence are designed jointly in a way such that each iteration takes  $\tilde{\mathcal{O}}(1)$  arithmetic operations.
4. Analyzing the convergence is complicated by the use of adaptive action sampling and weighted exponentiated updates, which leads to substantial noises with unbounded second moments. As a result, the classical primal-dual analysis by [24, 17] no longer works. To tackle this difficulty, we develop an independent convergence proof by analyzing the stochastic improvement of a particular relative

entropy. We obtain a finite-time duality gap bound that characterizes how much the complementarity condition of the Bellman linear program is violated by the output dual variable.

5. Another critical piece of our analysis is to study the relation between the duality gap of the iterate and the efficiency loss of the output randomized policy. We show that when the Markov decision process is sufficiently ergodic, the duality gap provides a sharp estimate for the value loss of the output approximate policy.
6. Finally, we provide a meta algorithm that performs multiple independent trials of the randomized primal-dual iteration to get a good policy with high probability. To achieve this goal, we develop a subroutine for policy evaluation and show that it computes an  $\epsilon$ -accurate value in  $\tilde{\mathcal{O}}(\frac{1}{\epsilon^2(1-\gamma)^2})$  run time. We prove that the meta algorithm is able to select the best policy out of many candidates with probability arbitrarily close to 1.

## 1.2 Main Results

We analyze the run-time complexity of the proposed randomized algorithm for obtaining an  $\epsilon$ -optimal policy, i.e., a policy that achieves  $\epsilon$ -optimal cumulative reward (to be specified in more details later). The run-time complexity of an algorithm is measured by the number of arithmetic operations. Our main results are summarized as follows:

1. We show that the proposed algorithm finds an  $\epsilon$ -optimal policy  $\hat{\pi}$  with probability at least  $1 - \delta$  in run time

$$\tilde{\mathcal{O}}\left(\frac{|\mathcal{S}|^3|\mathcal{A}|}{(1-\gamma)^6\epsilon^2}\log\left(\frac{1}{\delta}\right)\right).$$

We recall that the input size of the DMDP is  $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ . This result establishes a *nearly-linear run time*. This is the first computational complexity result for using randomized linear programming to solve DMDP.

2. Here comes the more interesting result: In the case where the decision process is ergodic under any stationary policy, we show that the algorithm finds an  $\epsilon$ -optimal policy with probability at least  $1 - \delta$  in run time

$$\tilde{\mathcal{O}}\left(|\mathcal{S}|^2|\mathcal{A}| + \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4\epsilon^2}\log\left(\frac{1}{\delta}\right)\right).$$

The first term  $\tilde{\mathcal{O}}(|\mathcal{S}|^2|\mathcal{A}|)$  is due to an initialization step that preprocesses the input data (consisting mainly of arrays of transition probabilities) into a tree-based sampler. This run time is *linear with respect to the input size*. Although requiring an additional ergodicity assumption, it has better dependence on  $|\mathcal{S}|, |\mathcal{A}|$  than the best known simplex method (also the policy iteration method) [32],[27] and the value iteration method [22].

3. In addition, the preprocessing step can be omitted or expedited if the input data are given in a suitable data structure that can be directly used as a sampler (e.g., binary trees or arrays of cumulative sums). In these cases, the complexity upper bound reduces to

$$\tilde{\mathcal{O}}\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4\epsilon^2}\log\left(\frac{1}{\delta}\right)\right) \ll |\mathcal{S}|^2|\mathcal{A}|,$$

where the inequality holds when  $|\mathcal{S}|$  is sufficiently large. This is a surprising *sublinear run-time complexity*. In other words, it is possible to compute a near-optimal policy without even reading all entries of the input data. To the author's best knowledge, this is the first sublinear run-time result for DMDP.

We compare the above complexity upper bounds with recent results on the computational complexity lower bound of DMDP [9]. It shows that any algorithm needs at least  $\Omega(|\mathcal{S}|^2|\mathcal{A}|)$  run time to get an  $\epsilon$ -approximate policy with high probability in general. It also shows that the lower bound reduces to  $\Omega(\frac{|\mathcal{S}||\mathcal{A}|}{\epsilon})$  when the input data are in the format of binary trees or cumulative sums (for which the preprocessing step can be skipped), making sublinear-time algorithms possible. Comparing our main results with the lower bounds, we observe a counter-intuitive phenomenon: *The computational complexity of DMDP depends on the input data structure.*

**Notations** All vectors are considered as column vectors. For a vector  $\mathbf{x} \in \mathbb{R}^n$ , we denote by  $x_i$  or  $x(i)$  its  $i$ -th component, denote by  $\mathbf{x}^\top$  its transpose, and denote by  $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$  its Euclidean norm. We denote by  $\mathbf{e} = (1, \dots, 1)^\top$  the vector with all entries equaling 1, and we denote by  $\mathbf{e}_i$  the vector with its  $i$ -th entry equaling 1 and other entries equaling 0. For a positive number  $x$ , we denote by  $\log x$  the natural logarithm of  $x$ . For two probability distributions  $p, q$  over a finite set  $X$ , we denote by  $D_{KL}(p||q)$  their Kullback-Leibler divergence, i.e.,  $D_{KL}(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$ .

## 2 Related Literatures

Three major approaches for solving MDP are the value iteration method, the policy iteration method, and linear programming methods. See the textbooks [3, 5, 26, 4] and references therein for more detailed surveys on MDP and its solution methods.

Bellman [1] developed the value iteration as a successive approximation method to solve the nonlinear fixed-point Bellman equation. Its convergence and complexity have been thoroughly analyzed; see e.g. [28, 22]. The best known complexity result for value iteration is  $\mathcal{O}(|\mathcal{S}|^2 |\mathcal{A}| L \frac{\log(1/(1-\gamma))}{1-\gamma})$ , where  $L$  is the number of bits to represent the input ( $L \geq |\mathcal{S}|^2 |\mathcal{A}|$ ). Value iteration can be also used to find an approximate solution in  $\mathcal{O}(|\mathcal{S}|^2 |\mathcal{A}| \frac{\log(1/\epsilon(1-\gamma))}{1-\gamma})$ . Later it was shown by [15] that the value iteration method is not strongly polynomial for DMDP. Policy iteration was developed by Howard [16], and its complexity has also been analyzed extensively; see e.g. [23, 32, 27]. Ye [32] showed that policy iteration (which is a variant of the general simplex method for linear programming) is strongly polynomial and terminates in  $\mathcal{O}(\frac{S^2 A}{1-\gamma} \log(\frac{|\mathcal{S}|}{1-\gamma}))$  number of iterations. Later [27] improved the result by removing the  $\log |\mathcal{S}|$  factor. Not long after the development of value and policy iterations, [14] and [13] discovered that the Bellman equation can be formulated into an equivalent linear program. It followed that one can apply linear programming method such as the simplex method by Dantzig [11] to solve MDP *exactly*. Later [31] designed a combinatorial interior-point algorithm (CIPA) that solves the DMDP in strongly polynomial time. Recent developments [20, 21] showed that linear programs can be solved in  $\tilde{\mathcal{O}}(\sqrt{\text{rank}(A)})$  number of linear system solves, which, applied to DMDP, leads to a run time of  $\tilde{\mathcal{O}}(|\mathcal{S}|^{2.5} |\mathcal{A}| L)$ . We also note that there have been many methods for approximate linear programming. However, they do not apply to DMDP directly because an  $\epsilon$  error in the linear programming formation of the DMDP might lead to arbitrarily large policy error.

Value Iteration	$ \mathcal{S} ^2  \mathcal{A}  L \frac{\log(1/(1-\gamma))}{1-\gamma}$ and $ \mathcal{S} ^2  \mathcal{A}  \frac{\log(1/(1-\gamma)\epsilon)}{1-\gamma}$	[28, 22]
Policy Iteration (Block Simplex)	$\frac{ \mathcal{S} ^4  \mathcal{A} ^2}{1-\gamma} \log(\frac{1}{1-\gamma})$	[32], [27]
LP Algorithm	$\tilde{\mathcal{O}}( \mathcal{S} ^{2.5}  \mathcal{A}  L)$	[20]
Combinatorial Interior Point Algorithm	$ \mathcal{S} ^4  \mathcal{A} ^4 \log \frac{ \mathcal{S} }{1-\gamma}$	[31]
Phased Q-Learning	$\tilde{\mathcal{O}}( \mathcal{S} ^2  \mathcal{A}  +  \mathcal{S} ^2  \mathcal{A}  + \frac{ \mathcal{S}   \mathcal{A} }{(1-\gamma)^6 \epsilon^2})$	[18] and This Work
Randomized Primal-Dual Method	$\tilde{\mathcal{O}}(\frac{ \mathcal{S} ^3  \mathcal{A} }{(1-\gamma)^6 \epsilon^2})$	Main Result 1
Randomized Primal-Dual Method	$\tilde{\mathcal{O}}( \mathcal{S} ^2  \mathcal{A}  + \frac{ \mathcal{S}   \mathcal{A} }{(1-\gamma)^4 \epsilon^2})$ under ergodicity assumption	Main Result 2
Randomized Primal-Dual Method	$\tilde{\mathcal{O}}(\frac{ \mathcal{S}   \mathcal{A} }{(1-\gamma)^4 \epsilon^2})$ under ergodicity assumption and special input format	Main Result 3
Lower Bound	$\Omega( \mathcal{S} ^2  \mathcal{A} )$	[9]
Lower Bound	$\Omega(\frac{ \mathcal{S}   \mathcal{A} }{\epsilon})$ under special input format	[9]

Table 1: Run-Time Complexity for solving DMDP, where  $|\mathcal{S}|$  is the number of states,  $|\mathcal{A}|$  is the number of actions per state,  $\gamma \in (0, 1)$  is the discount factor, and  $L$  is the total bit size to present the DMDP input.

A most popular method in reinforcement learning is the Q-learning method. It refers to a class of sampling-based variants of the value iteration. The work [18] proposed the phased Q-learning method and proved that it takes  $\tilde{\mathcal{O}}(\frac{|\mathcal{S}| |\mathcal{A}|}{\epsilon^2})$  sample transitions to compute an  $\epsilon$ -optimal policy, where the dependence on  $\gamma$  is left unspecified. If we carry out a more careful analysis (which is not available in [18]), we can see that the sample complexity of phased Q-learning is actually  $\tilde{\mathcal{O}}(\frac{|\mathcal{S}| |\mathcal{A}|}{(1-\gamma)^6 \epsilon^2})$ . No run-time analysis is given explicitly for phased Q-learning in [18]. Using the techniques developed in this work, we can apply the binary-tree

sampling techniques described in Section 4.1 and apply Prop. 1 to the phased Q-learning. Then we can implement the method using appropriate preprocessing and  $\tilde{O}(1)$  run time per sample/update, leading to a total run time  $\tilde{O}(|\mathcal{S}|^2|\mathcal{A}| + \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^6\epsilon^2})$ . Such run-time analysis is not available in any existing literature. Unlike Q-learning, most reinforcement learning algorithms need far more than  $\tilde{O}(1)$  arithmetic operations to process a single sample transition, so they are not efficient solvers for DMDP.

Table 1 summarizes the best-known run-time complexity, in terms of the total number of arithmetic operations, of solution methods for DMDP. Our second result is the sharpest among existing methods, which however requires an additional assumption on the ergodicity of the Markov chains (which we believe to be mild). The upper bound result  $\tilde{O}(|\mathcal{S}|^2|\mathcal{A}| + \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4\epsilon^2})$  nearly matches the lower bound  $\Omega(|\mathcal{S}|^2|\mathcal{A}|)$  by [9], except for the extra ergodicity assumption. Our third main result reveals a surprising phenomenon: In the case where the input data are given in preprocessed data format that enables sampling, the complexity reduces to  $\tilde{O}(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4\epsilon^2})$ , which is a sublinear run-time result and nearly match the lower bound  $\Omega(\frac{|\mathcal{S}||\mathcal{A}|}{\epsilon})$  for the same case by [9].

Our new method and analysis is motivated by the line of developments on stochastic first-order optimization methods. They originate from the stochastic approximation method for the root finding problem; see [19, 2, 6]. They find wide applications in stochastic convex optimization, especially problems arising from machine learning applications. Several studies have been conducted to analyze the sample complexity of stochastic first-order methods, started by the seminal work [24] and followed by many others. In particular, our proposed method is closely related to stochastic first-order method for saddle point problems. The earliest work of this type is by [25], which studied a stochastic approximation method for convex-concave saddle point problem and gives explicit convergent rate estimates. Later the work [17] studied a class of stochastic mirror-prox methods and their rate of convergence for solving stochastic monotone variational inequalities, which contains the convex-concave saddle point as a special case. Another related work is by [10], which developed a class of sublinear algorithms for minimax problems from machine learning. It showed that it is possible to find a pair of primal-dual solutions achieving  $\epsilon$  duality gap in run time  $\tilde{O}(\frac{1}{\epsilon^2})$ .

Unfortunately, none of existing results on the general stochastic saddle point problem directly applies to the DMDP. There are several gaps to be filled. The first gap lies in the saddle point formulation of the Bellman equation. Prior to this work, it was not clear how to appropriately formulate the DMDP into a saddle point problem with appropriate constraints in order to maintain complexity guarantee and enable constant-time projection at the same time. The second gap lies in the implementation and run-time efficiency of algorithms. Earlier works on stochastic mirror-prox methods mainly focused on the iteration/sample complexity. In this work, we care about the overall run-time complexity for solving DMDP. To achieve the best run-time efficiency, we will provide an integrated algorithmic design that combines the mathematics together with programming techniques. The third gap lies in the proof of convergence. Our algorithm uses importance adaptive sampling of actions, which creates unbounded noises and disables the analysis used in [25, 17]. As a result, we have to come up with an independent primal-dual convergence analysis. The fourth gap lies between the duality gap and the performance of the output policy. A small duality gap does not necessarily imply a small policy error. In this work, we aim to close all these gaps and develop efficient randomized algorithms with run-time guarantees.

Let us also mention that there are two prior attempts (by the author of this paper) to use primal-dual iteration for online policy estimation of MDP. The work [29] and its journal version [8] considered a basic stochastic primal-dual iteration that uses Euclidean projection and uniform sampling of state-action pairs to solve DMDP and established a sample complexity upper bound  $\mathcal{O}(|\mathcal{S}|^{4.5}|\mathcal{A}|\epsilon^{-2})$ . No run-time complexity analysis is available.

### 3 Bellman Equation, Linear Programs, and Stochastic Saddle Point Problem

Consider a DMDP tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$ . For a fixed policy  $\pi$ , the value vector  $\mathbf{v}^\pi \in \mathfrak{R}^{|\mathcal{S}|}$  is defined as

$$v_i^\pi = \mathbf{E}^\pi \left[ \sum_{t=1}^{\infty} \gamma^t r_{i_t i_{t+1}}(a_t) \mid i_1 = i \right], \quad \forall i \in \mathcal{S},$$

where  $\mathbf{E}^\pi[\cdot]$  is taken over the state-action trajectory  $\{i_1, a_1, i_2, a_2, \dots\}$  generated by the Markov decision process under policy  $\pi$ . The optimal value vector  $\mathbf{v}^* \in \mathfrak{R}^{|\mathcal{S}|}$  is defined as

$$v_i^* = \max_{\pi} \mathbf{E}^\pi \left[ \sum_{t=1}^{\infty} \gamma^t r_{i_t i_{t+1}}(a_t) \mid i_1 = i \right] = \mathbf{E}^{\pi^*} \left[ \sum_{t=1}^{\infty} \gamma^t r_{i_t i_{t+1}}(a_t) \mid i_1 = i \right], \quad \forall i \in \mathcal{S}.$$

According to the theory of dynamic programming [26, 3], a vector  $\mathbf{v}^*$  is the optimal value function to the DMDP if and only if it satisfies the following  $|\mathcal{S}| \times |\mathcal{S}|$  system of equations, known as the *Bellman equation*, given by

$$v_i^* = \max_{a \in \mathcal{A}} \left\{ \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* + \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right\}, \quad \forall i \in \mathcal{S},$$

When  $\gamma \in (0, 1)$ , the Bellman equation has a unique fixed point solution  $\mathbf{v}^*$ , and it equals to the optimal value vector of the DMDP. Moreover, a policy  $\pi^*$  is an optimal policy of the DMDP if and only if it attains the elementwise maximization in the Bellman equation. For finite-state DMDP, there always exists at least one optimal policy  $\pi^*$ . If the optimal policy is unique, it is also a deterministic policy. If there are multiple optimal policies, there exist infinitely many optimal randomized policies.

The nonlinear Bellman equation is equivalent to the following  $|\mathcal{S}| \times (|\mathcal{S}||\mathcal{A}|)$  linear programming problem (see [26] Section 6.9 and the paper [12]):

$$\begin{aligned} & \text{minimize } (1 - \gamma) \mathbf{q}^\top \mathbf{v} \\ & \text{subject to } (I - \gamma P_a) \mathbf{v} - \mathbf{r}_a \geq 0, \quad \forall a \in \mathcal{A}, \end{aligned} \quad (1)$$

where  $\mathbf{q} \in \mathfrak{R}^{|\mathcal{S}|}$  is an *arbitrary* vector of probability distribution satisfying  $\mathbf{e}^\top \mathbf{q} = 1$  and  $\mathbf{q} > 0$ ,  $P_a \in \mathfrak{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  is matrix whose  $(i, j)$ -th entry equals to  $p_{ij}(a)$ ,  $I$  is the identity matrix with dimension  $|\mathcal{S}| \times |\mathcal{S}|$  and  $\mathbf{r}_a \in \mathfrak{R}^{|\mathcal{S}|}$  is the expected state-transition reward vector under action  $a$ , i.e.,  $r_a(i) = \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a)$  for all  $i \in \mathcal{S}$ . The dual linear program of (1) is

$$\begin{aligned} & \text{maximize } \sum_{a \in \mathcal{A}} \mu_a^\top \mathbf{r}_a \\ & \text{subject to } \sum_{a \in \mathcal{A}} (I - \gamma P_a^\top) \mu_a = (1 - \gamma) \mathbf{q}, \quad \mu_a \geq 0, \quad \forall a \in \mathcal{A}. \end{aligned} \quad (2)$$

It is well known that each deterministic policy corresponds to a basic feasible solution to the dual linear program (2). A randomized policy is a mixture of deterministic policies, so it corresponds to a feasible solution of program (2). We denote by  $\mu^* = (\mu_a^*)_{a \in \mathcal{A}}$  the optimal solution to the dual linear program (2). If there is a unique dual solution, it must be a basic feasible solution. In this case, it is well known that the basis of  $\mu^*$  corresponds to an optimal deterministic policy.

We formulate the linear programs (1)-(2) into an equivalent minimax problem, given by

$$\min_{\mathbf{v} \in \mathcal{V}} \max_{\mu \in \mathcal{U}_{\theta, \mathbf{q}}} (1 - \gamma) \mathbf{q}^\top \mathbf{v} + \sum_{a \in \mathcal{A}} \mu_a^\top ((\gamma P_a - I) \mathbf{v} + \mathbf{r}_a). \quad (3)$$

We construct  $\mathcal{V}$  and  $\mathcal{U}_{\theta, \mathbf{q}}$  to be the search spaces for the value and the policy, respectively, given by

$$\mathcal{V} = \left\{ \|\mathbf{v}\|_\infty \leq \frac{1}{1 - \gamma}, \mathbf{v} \geq 0 \right\}, \quad \mathcal{U}_{\theta, \mathbf{q}} = \left\{ \mathbf{e}^\top \mu = 1, \mu \geq 0, \sum_{a \in \mathcal{A}} \mu_a \geq \theta \mathbf{q} \right\},$$

where  $\theta$  is a small value to be specified. Let us verify that  $\mathbf{v}^* \in \mathcal{V}$  and  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ . Since all rewards  $r_{ij}(a)$  belong to  $[0, 1]$ , we can easily verify that  $0 \leq v_i^* \leq \frac{1}{1 - \gamma}$  for all  $i$ , therefore  $\mathbf{v}^* \in \mathcal{V}$ . By multiplying both sides of the dual constraint  $\sum_{a \in \mathcal{A}} (I - \gamma P_a^\top) \mu_a^* = (1 - \gamma) \mathbf{q}$  with  $\mathbf{e}^\top$ , we also verify that  $\mathbf{e}^\top \mu^* = 1$  because  $\mathbf{q}$  is a probability distribution. In subsequent analysis, we will specify choices of  $\theta, \mathbf{q}$  so that  $\sum_{a \in \mathcal{A}} \mu_a^* \geq \theta \mathbf{q}$  and  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ . As long as  $\mathbf{v}^* \in \mathcal{V}$  and  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ , we will be able to tackle the DMDP by approximately solving the saddle point problem (3).

There are many ways to formulate the minimax problem (3) into a stochastic saddle point problem. A key observation is that the probability transition matrix  $P_a$  and the reward vector  $\mathbf{r}_a$  are naturally expectations of some random variables. For example, we can rewrite (3) as follows

$$\min_{\mathbf{v} \in \mathcal{V}} \max_{\mu \in \mathcal{U}_{\theta, \mathbf{q}}} (1 - \gamma) \mathbf{q}^\top \mathbf{v} + \sum_{a \in \mathcal{A}} \mu_a^\top \left( \left( \gamma \sum_{i \in \mathcal{S}} \mathbf{E}_{j|i, a} [\mathbf{e}_i \mathbf{e}_j^\top] - I \right) \mathbf{v} + \sum_{i \in \mathcal{S}} \mathbf{E}_{j|i, a} [r_{ij}(a) \mathbf{e}_i] \right),$$

where the expectation  $\mathbf{E}_{j|i, a} [\cdot]$  is taken over  $j \sim p_{ij}(a)$  where  $i$  and  $a$  are fixed. The preceding formulation suggests that one can simulate transitions  $j | i, a$  of the Markov decision process in order to draw samples of the Lagrangian function and its partial derivatives. In addition, we will show later that simulation of the Markov decision process is almost *free* in the sense that each transition can be simulated in  $\tilde{\mathcal{O}}(1)$  arithmetic operations. This motivates the use of a stochastic primal-dual iteration for solving the DMDP.

## 4 Algorithms

We develop our main algorithms in this section. We first develop a few programming techniques that may be useful to all simulation-based methods for DMDP. Then we propose the randomized primal-dual algorithms and analyze their run-time complexity per iteration.

### 4.1 Programming Techniques for Markov Decision Processes

Randomized algorithms for DMDP inevitably involve simulating the Markov decision process and making policy updates. Our first step is to develop implementation techniques for the two operations so that they take as little as  $\tilde{\mathcal{O}}(1)$  run time.

**Proposition 1.** *Suppose that we are given arrays of transition probabilities  $\mathcal{P} = (P_a)_{a \in \mathcal{A}}$  and a randomized policy  $\pi = \{\pi_{i, a}\}_{i \in \mathcal{S}, a \in \mathcal{A}}$ . We are also given a stream of updates to the weight vectors, each update taking the form  $\pi_{i, a} \leftarrow \tilde{\pi}_{i, a}, \pi_i \leftarrow \pi_i / \|\pi_i\|_1$  for some  $i \in \mathcal{S}, a \in \mathcal{A}$ . There exists an algorithm that preprocesses  $\mathcal{P}$  in  $\tilde{\mathcal{O}}(|\mathcal{S}|^2 |\mathcal{A}|)$  time, makes each update in  $\mathcal{O}(\log |\mathcal{A}|)$  time, and, for any initial state and at any time, generates a single state transition of the Markov decision process under the current policy in  $\mathcal{O}(\log |\mathcal{A}|) + \mathcal{O}(\log |\mathcal{S}|)$  time.*

*Proof.* Let us apply the binary-tree scheme [30] to the computation of Markov decision process. We are given the transition probabilities  $(P_a)_{a \in \mathcal{A}}$  as arrays. For each state-action pair  $(i, a)$ , we preprocess each row vector of transition probabilities  $P_a(i, \cdot)$  into a binary tree, where each leaf stores the value  $P_a(i, j)$  for some  $j$  and each node stores the sum of its two children. We need to construct  $|\mathcal{S}| |\mathcal{A}|$  trees for all transition probabilities, and the preprocessing time is  $\tilde{\mathcal{O}}(|\mathcal{S}|^2 |\mathcal{A}|)$ . Then for a given state-action pair  $(i, a)$ , it is possible to generate a random coordinate  $j$  with probability  $p_{ij}(a)$  by drawing a random variable uniformly from  $[0, 1]$  and search for the leaf that corresponds to an interval that contains the random variable. This procedure takes  $\mathcal{O}(\log |\mathcal{S}|)$  arithmetic operations.

We represent a randomized policy  $\pi$  using a collection of  $|\mathcal{S}|$ -dimensional vectors of nonnegative weights  $\{\mathbf{w}_i\}_{i \in \mathcal{S}}$  such that  $\pi_{i, a} = \frac{w_{i, a}}{\sum_{a \in \mathcal{A}} w_{i, a}}$ . Similar to  $P_a(i, \cdot)$ 's, the weight vectors  $\{\mathbf{w}_i\}_{i \in \mathcal{S}}$  can be represented as  $|\mathcal{S}|$  binary trees, one for each  $i \in \mathcal{S}$ . For any given  $i \in \mathcal{S}$ , one can generate a random coordinate  $a$  with probability  $\pi_{i, a} = \frac{w_{i, a}}{\|\mathbf{w}_i\|_1}$  using  $\mathcal{O}(\log |\mathcal{S}|)$  arithmetic operations. Now suppose that we want to make the policy update  $\pi_{i, a} \leftarrow \tilde{\pi}_{i, a}, \pi_i \leftarrow \pi_i / \|\pi_i\|_1$ . Then we update the corresponding leaf from  $w_{i, a}$  to  $\|\mathbf{w}_i\|_1 \cdot \tilde{\pi}_{i, a}$ , where  $\|\mathbf{w}_i\|_1$  is simply the value of the root. We also need to update the values of all nodes on the path from the root to the leaf  $w_{i, a}$ , so that each node remains the sum of its two children. This update takes  $\mathcal{O}(\log |\mathcal{A}|)$  operations.

Finally, suppose that we want to simulate the decision process and generate a state transition according to the current policy. For a given state  $i \in \mathcal{S}$ , we first sample an action  $a$  according to the policy and then sample a coordinate  $j$  according to  $P_a(i, \cdot)$ , which takes  $\mathcal{O}(\log |\mathcal{A}|) + \mathcal{O}(\log |\mathcal{S}|) = \tilde{\mathcal{O}}(1)$  arithmetic operations in total.  $\blacksquare$

Proposition 1 implies that simulation-based methods for DMDP can be potentially very efficient, because the sampling and updating operations are computationally cheap. This result suggests an intriguing connection between the sample complexity for estimating the optimal policy and the run-time complexity for approximating the optimal policy.

---

**Algorithm 1** Randomized Primal-Dual Method for DMDPs

---

- 1: **Input:** DMDP tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$ ,  $\mathbf{q} = \frac{1}{S}\mathbf{e}$ ,  $\theta = (1 - \gamma)$ ,  $T$ .
- 2: Set  $\mathbf{v} = 0 \in \mathbb{R}^{|\mathcal{S}|}$
- 3: Set  $\xi = \frac{1}{|\mathcal{S}|}\mathbf{e} \in \mathbb{R}^{|\mathcal{S}|}$ ,  $\pi_i = \frac{1}{|\mathcal{A}|}\mathbf{e} \in \mathbb{R}^{|\mathcal{A}|}$  for all  $i \in \mathcal{S}$
- 4: Set  $\beta = (1 - \gamma)\sqrt{\frac{\log(|\mathcal{S}||\mathcal{A}|+1)}{2|\mathcal{S}||\mathcal{A}|T}}$ ,  $\alpha = \frac{|\mathcal{S}|}{2(1-\gamma)^2}\beta$ ,  $M = \frac{1}{1-\gamma}$
- 5: Preprocess the input probabilities  $\mathcal{P} = \{p_{ij}(a)\}$  for sampling (Complexity  $\tilde{O}(S^2A)$ )
- 6: **for**  $t = 1, 2, \dots, T$  **do**
- 7:   Sample  $i$  with probability  $((1 - \theta)\xi_i + \theta q_i)$  (Complexity  $\tilde{O}(1)$ )
- 8:   Sample  $a$  with probability  $\pi_{i,a}$  (Complexity  $\tilde{O}(1)$ )
- 9:   Conditioned on  $(i, a)$ , sample  $j$  with probability  $p_{ij}(a)$  (Complexity  $\tilde{O}(1)$ )
- 10:   Update the iterates by (Complexity  $\tilde{O}(1)$ )

$$\begin{aligned} \Delta &\leftarrow \beta \cdot \frac{\gamma v_j - v_i + r_{ij}(a) - M}{((1 - \theta)\xi_i + \theta q_i)\pi_{i,a}} \\ v_i &\leftarrow \max \left\{ \min \left\{ v_i - \alpha \left( \frac{(1 - \gamma)q_i}{(1 - \theta)\xi_i + \theta q_i} - 1 \right), \frac{1}{1 - \gamma} \right\}, 0 \right\} \\ v_j &\leftarrow \max \left\{ \min \left\{ v_j - \alpha\gamma, \frac{1}{1 - \gamma} \right\}, 0 \right\} \end{aligned}$$

- 11:   Update the iterates by (Complexity  $\tilde{O}(1)$ )

$$\begin{aligned} \xi_i &\leftarrow \xi_i + \xi_i \pi_{i,a} (\exp \{\Delta\} - 1), \xi \leftarrow \xi / \|\xi\|_1 \\ \pi_{i,a} &\leftarrow \pi_{i,a} \cdot \exp \{\Delta\}, \pi_i \leftarrow \pi_i / \|\pi_i\|_1 \end{aligned}$$

- 12: **end for**
  - 13: **Output:** Averaged policy iterate  $\hat{\pi}_i = \frac{1}{T} \sum_{k=1}^T \pi_i^k$  for all  $i \in \mathcal{S}$
- 

## 4.2 The Randomized Primal-Dual Algorithm

Motivated by the saddle point formulation of Bellman's equation, we develop a randomized linear programming method to compute an approximately optimal policy. The method is given in Algorithm 1. It is essentially a randomized primal-dual iteration that makes updates to an explicit primal variable and an *implicit* dual variable. More specifically, it makes iterative coordinate updates to three variables:  $\pi$ ,  $\mathbf{v}$  and  $\xi$ . Here  $\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$  represents a randomized policy and is guaranteed to satisfy  $\mathbf{e}^\top \pi_i = 1$  and  $\pi_i \geq 0$  for all  $i \in \mathcal{S}$  throughout the iterations. The vector  $\mathbf{v} \in \mathbb{R}^{\mathcal{S}}$  is the primal variable (also the value vector) and is guaranteed to satisfy  $0 \leq \mathbf{v} \leq \frac{1}{1-\gamma}\mathbf{e}$  throughout. The vector  $\xi \in \mathbb{R}^{\mathcal{S}}$  represents some distribution over the state space and satisfies  $\mathbf{e}^\top \xi = 1, \xi \geq 0$  throughout. The policy  $\pi$  and the distribution  $\xi$  jointly give the dual variable  $\mu$  according to

$$\mu_{i,a} = ((1 - \theta)\xi_i + \theta q_i) \pi_{i,a}, \quad \forall i \in \mathcal{S}, a \in \mathcal{A}.$$

The dual variable  $\mu$  does not appear in the iteration of Algorithm 1. It is updated implicitly through updates on  $\pi$  and  $\xi$ . We can verify that the implicit dual variable  $\mu$  satisfies  $\mu \in \mathcal{U}_{\theta, \mathbf{q}}$  throughout the iteration.

Also note that Algorithm 1 uses adaptive importance sampling of state-to-state transitions according to the current policy. In particular, it samples a state-action pair  $(i, a)$  with probability  $((1 - \theta)\xi_i + \theta q_i)\pi_{i,a}$  (Steps 7-8). The sampling distribution varies as the algorithm proceeds. Some state-action pairs will be sampled with higher and higher probability, meaning that the action becomes more favorable than other

actions for the corresponding state. In the mean time, some state-action pairs will be sampled less and less frequently, while the corresponding probability  $\pi_{i,a}$  reduces to 0 eventually. This can be viewed as a form of “reinforcement learning”, because the algorithm tend to sample more often the actions that empirically worked well. To account for the nonuniform sampling probability, each sample  $\Delta$  is re-weighted by  $\frac{1}{((1-\theta)\xi_i + \theta q_i)\pi_{i,a}}$ .

Now we analyze the run-time complexity for each step of Algorithm 1. Step 5 preprocesses the input probabilities  $\mathcal{P} = \{p_{ij}(a)\}$  into  $|\mathcal{S}||\mathcal{A}|$  tree samplers, one for each state-action pair. Each of the sampler corresponds to a probability vector of  $|\mathcal{S}|$  dimension. The preprocessing time for each  $m$ -dimensional vector is  $\tilde{\mathcal{O}}(m)$ . Therefore the total processing time is  $\tilde{\mathcal{O}}(|\mathcal{S}|^2|\mathcal{A}|)$ . According to Proposition 1, Steps 7-9 take  $\tilde{\mathcal{O}}(1)$  arithmetic operations each, provided that one updates and sample from  $\xi, \pi$  using the binary-tree data structures. Step 10 updates two entries of the value vector  $\mathbf{v}$  and uses run time  $\tilde{\mathcal{O}}(1)$ . According to Proposition 1, Step 11 can also be implemented using  $\tilde{\mathcal{O}}(1)$  arithmetic operations. Step 13 requires taking average of past iterates  $\{\pi^t\}_{t=1}^T$ . This can be achieved by maintaining an additional variable to record the running multiplicative weights and running average (e.g., using a special binary tree structure). The additional storage overhead is  $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ , and the additional computation overhead is  $\tilde{\mathcal{O}}(1)$  per iteration.

To sum up, the total run-time complexity of Algorithm 1 consists of two parts: complexity of preprocessing and complexity per iteration. Preprocessing takes  $\tilde{\mathcal{O}}(S^2A)$  run time, which can be skipped if the input data are given in some data structure that enables immediate sampling (e.g., sorted arrays [7] or binary trees [30]). Each iteration of Algorithm 1 takes  $\tilde{\mathcal{O}}(1)$  arithmetic operations.

### 4.3 The Meta Algorithm

Finally, we are ready to develop a meta algorithm that computes an approximately optimal policy with probability arbitrarily close to 1. The idea is to run Algorithm 1 for a number of independent trials, perform approximate value evaluation to the output policies, and select the best out of the candidates.

---

#### Algorithm 2 Meta Algorithm

---

- 1: **Input:** DMDP tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$ ,  $\mathbf{q} = \frac{1}{S}\mathbf{e}$ ,  $\theta = (1 - \gamma)$ ,  $T$ .
  - 2: Run Algorithm 1 for  $K = \Omega(\log \frac{1}{\delta})$  independent trials with precision parameter  $\frac{\epsilon}{2}$  and obtain output policies  $\pi^{(1)}, \dots, \pi^{(K)}$ .
  - 3: For each output policy  $\pi^{(k)}$  and initial distribution  $\mathbf{q}$ , conduct approximate value evaluation and obtain an approximate evaluation  $\bar{Y}^{(k)}$  with precision level  $\frac{\epsilon}{2}$  and fail probability  $\frac{\delta}{2K}$ .
  - 4: Output  $\hat{\pi} = \pi^{(k^*)}$  such that  $k^* = \operatorname{argmax}_{k=1, \dots, K} \bar{Y}^{(k)}$ .
  - 5: **Output:** Averaged dual iterate  $\hat{\pi}_i = \frac{1}{T} \sum_{k=1}^T \pi_i^t$  for all  $i \in \mathcal{S}$
- 

In Algorithm 2, Step 3 approximately computes the cumulative discounted rewards for all candidate policies and a fixed initial state distribution  $\mathbf{q}$ . The aim is to find an  $\epsilon$ -approximation to each cumulative reward with probability at least  $1 - \frac{\delta}{2K}$ . Its implementation and complexity will be discussed and analyzed in Section 5.3.

## 5 Complexity Analysis

We develop our main results in a number of steps. We first study the convergence of some duality gap of the randomized primal-dual iteration given by Algorithm 1. Then we study how to quantify the quality of the output randomized policy from the duality gap upper bound. Third we show how to approximately evaluate a randomized policy in a small run time. Finally we connect all the dots and establish the overall run-time complexity of Algorithms 1 and 2.

## 5.1 Duality Gap Analysis

In this section, we study the convergence of Algorithm 1. We observe that each iteration of Algorithm 1 essentially performs a primal-dual update. To see this, we define the auxiliary dual iterate  $\mu_{i,a}^t$  as

$$\mu_{i,a}^t = ((1 - \theta)\xi_i^t + \theta q_i) \pi_{i,a}^t, \quad \forall i \in \mathcal{S}, a \in \mathcal{A}.$$

Informally speaking, we see that the  $t$ -th iteration of Algorithm 1 takes the form

$$\begin{aligned} \mathbf{v}^{t+1} &= \operatorname{argmin}_{\mathbf{v} \in \mathcal{V}} \left\{ (\partial_{\mathbf{v}} L(\mathbf{v}, \mu^t) + \varepsilon^{t+1})^\top (\mathbf{v} - \mathbf{v}^t) + \frac{1}{2\beta} \|\mathbf{v} - \mathbf{v}^t\|^2 \right\}, \\ \mu^{t+1} &= \operatorname{argmin}_{\mu \in \mathcal{U}_{\theta, \mathbf{q}}} \left\{ (\partial_{\mu} L(\mathbf{v}^t, \mu) + \omega^{t+1})^\top (\mu - \mu^t) + \frac{1}{2\alpha} \Phi(\mu; \mu^t) \right\}, \end{aligned} \quad (4)$$

where  $\varepsilon^{t+1}$  and  $\omega^{t+1}$  are zero-mean random noises due to the sampling step,  $\Phi$  is a Bregman divergence function given by

$$\Phi(\mu; \hat{\mu}) = (1 - \theta) D_{KL}(\lambda \| \hat{\lambda}) + \theta \sum_{i \in \mathcal{S}} q_i D_{KL}(\pi_i \| \hat{\pi}_i),$$

where  $(\lambda, \pi)$  is determined by  $\mu$  and  $(\hat{\lambda}, \hat{\pi})$  is determined by  $\hat{\mu}$ . The dual feasible region  $\mathcal{U}_{\theta, \mathbf{q}}$  plays the role of an ‘‘information set,’’ in which we search for the optimal policy. The information set  $\mathcal{U}_{\theta, \mathbf{q}}$  shall be constructed to characterize properties and additional prior knowledge (if there is any) regarding the DMDP. Clearly, the set  $\mathcal{U}_{\theta, \mathbf{q}}$  and the divergence function  $\Phi$  are determined by the input parameters  $\theta, \mathbf{q}$ . We will specify values of the parameters  $\theta, \mathbf{q}$  in subsequent analysis.

One might attempt to analyze iteration (4) using the general analysis for stochastic mirror-prox iterations by [25, 17]. Unfortunately, this would not work. The general primal-dual convergence analysis given by [25, 17] requires  $\mathbf{E} [\|\partial_{\mu} L(\mathbf{v}^t, \mu) + \omega^{t+1}\|_*^2 | \mathcal{F}_t] < \sigma^2$  for some appropriate norm  $\|\cdot\|_*$  and a finite constant  $\sigma$ , where  $\mathcal{F}_t$  denotes the collection of random variables up to the  $t$ -th iteration. Our Algorithm 1 cannot be treated in this way, because it samples the partial gradients using adaptive weights (Steps 7,8) and re-weighted samples (Step 10). In particular, for a given state  $i$ , the action  $a$  is sampled with probability  $\pi_{i,a}^t$  and the corresponding partial gradient  $\Delta = \beta \cdot \frac{\gamma v_j - v_i + r_{ij}(a) - M}{((1-\theta)\xi_i + \theta q_i)\pi_{i,a}^t}$  has been re-weighted with  $1/\pi_{i,a}^t$  to maintain unbiasedness, leading to large variances on the order of  $\pi_{i,a}^t (1/\pi_{i,a}^t)^2 = 1/\pi_{i,a}^t$ . Therefore, each iteration Algorithm 1 suffers from unbounded noises in the following sense

$$\mathbf{E} \left[ \|\partial_{\mu} L(\mathbf{v}^t, \mu^t) + \omega^t\|_*^2 | \mathcal{F}_t \right] \geq \Theta \left( \min_{i \in \mathcal{S}, a \in \mathcal{A}} \pi_{i,a}^t \left( \frac{1}{\pi_{i,a}^t} \right)^2 \right) \rightarrow \infty, \quad \text{as } t \rightarrow \infty.$$

The lefthand side can take arbitrarily large values and eventually go to infinity. This is because many  $\pi_{i,a}^t$ 's become increasingly close to zero, which is inevitable as the randomized policy  $\pi^t$  converges to the optimal deterministic policy  $\pi^*$ . Due to this reason, the results and analyses given in [25, 17] do not apply.

To tackle this analytical difficulty, we develop an independent proof tailored to Algorithm 1. It involves analyzing the improvement of the relative entropy directly and constructing appropriate martingales. We obtain an finite-iteration upper bound on some expected ‘‘duality gap’’.

**Proposition 2 (Duality Gap Bound).** *Let  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$  be an arbitrary DMDP tuple, let  $\mathbf{q} \in \mathbb{R}^{|\mathcal{S}|}$  be an arbitrary probability vector and let  $\theta \geq 1 - \gamma$ . Assume that the solution of the dual linear program (2) satisfies  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ . Let Algorithm 1 iterate with the input  $(\mathcal{M}, \mathbf{q}, \theta)$ , then the sequence of iterates  $\{(\xi^t, \pi^t)\}_{t=1}^T$  satisfies*

$$\mathbf{E} \left[ \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right) \right] \leq \frac{\sqrt{2|\mathcal{S}|(|\mathcal{A}| + 1) (\log(|\mathcal{S}||\mathcal{A}|) + 1)}}{(1 - \gamma)\sqrt{T}},$$

where  $\hat{\mu}_{i,a} = \frac{1}{T} \sum_{t=1}^T ((1 - \theta)\xi_i^t + \theta q_i)\pi_{i,a}^t$ .

Proposition 2 establishes an expected upper bound on a nonnegative quantity that involves only the averaged dual variable  $\hat{\mu}$  but not the primal variable  $\mathbf{v}$ . It can be essentially viewed as a weighted sum of errors, which characterizes how much the following linear complementarity condition

$$\mu_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right) = 0, \quad \forall a \in \mathcal{A}, i \in \mathcal{S},$$

is violated for the linear programs (1)-(2). Although we refer to this error quantity intuitively as a duality gap, it is actually not the typical minimax duality gap for saddle point problems (which was analyzed in [25, 17]). We defer the detailed proof of Proposition 2 to Section 7.

## 5.2 From Dual Variable To Approximate Policy

In the following, we show that the duality gap of  $\hat{\mu}$  gives an upper bound on the efficiency loss of the randomized policy  $\hat{\pi}$ .

**Proposition 3.** *Let  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$  be an arbitrary DMDP tuple, and let  $\mathbf{q} = \frac{1}{|\mathcal{S}|} \mathbf{e}, \theta = (1 - \gamma)$ . For any vector  $\hat{\mu} \in \mathcal{U}_{\theta, \mathbf{q}} = \{\mathbf{e}^\top \mu = 1, \mu \geq 0, \sum_{a \in \mathcal{A}} \mu_a \geq \theta \mathbf{q}\}$ , we let  $\hat{\pi}$  be the corresponding randomized policy satisfying  $\hat{\pi}_{i,a} = \frac{\hat{\mu}_{i,a}}{\sum_{a \in \mathcal{A}} \hat{\mu}_{i,a}}$  for all  $i \in \mathcal{S}, a \in \mathcal{A}$ . Then*

$$\|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty \leq \frac{|\mathcal{S}|}{(1 - \gamma)^2} \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right).$$

*Proof.* We denote for short that  $Gap = \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right)$ . Using the fact  $\hat{\mu} \in \mathcal{U}_{\theta, \mathbf{q}}$ , we have  $\sum_{a \in \mathcal{A}} \hat{\mu}_{i,a} \geq \theta \mathbf{q} = (1 - \gamma) \mathbf{q}$ . Then we have

$$\begin{aligned} Gap &= \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{i,a} (\mathbf{v}^* - \gamma P_a \mathbf{v}^* - \mathbf{r}_a)_i \\ &\geq (1 - \gamma) \sum_{i \in \mathcal{S}} q_i \sum_{a \in \mathcal{A}} \hat{\pi}_{a,i} (\mathbf{v}^* - \gamma P_a \mathbf{v}^* - \mathbf{r}_a)_i \\ &= (1 - \gamma) \sum_{i \in \mathcal{S}} q_i (\mathbf{v}^* - \gamma P^{\hat{\pi}} \mathbf{v}^* - \mathbf{r}^{\hat{\pi}})_i, \end{aligned}$$

where  $\mathbf{r}^{\hat{\pi}}$  denotes the vector with  $r_i^{\hat{\pi}} = \sum_{a \in \mathcal{A}} \hat{\pi}_i(a) \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a)$ . We note that the Bellman equation for a fixed policy  $\hat{\pi}$  is given by  $\mathbf{v}^{\hat{\pi}} = \gamma P^{\hat{\pi}} \mathbf{v}^{\hat{\pi}} + \mathbf{r}^{\hat{\pi}}$ . Because  $\mathbf{v}^*$  is the optimal value vector of the DMDP, we have  $(\mathbf{v}^* - \gamma P_a \mathbf{v}^* - \mathbf{r}_a)_i \geq 0$  for all  $i \in \mathcal{S}$ . It follows that  $(\mathbf{v}^* - \gamma P^{\hat{\pi}} \mathbf{v}^* - \mathbf{r}^{\hat{\pi}})_i \geq 0$  for  $i \in \mathcal{S}$ , therefore

$$0 \leq \mathbf{v}^* - \gamma P^{\hat{\pi}} \mathbf{v}^* - \mathbf{r}^{\hat{\pi}} = \mathbf{v}^* - \gamma P^{\hat{\pi}} \mathbf{v}^* - (\mathbf{v}^{\hat{\pi}} - \gamma P^{\hat{\pi}} \mathbf{v}^{\hat{\pi}}) = (I - \gamma P^{\hat{\pi}})(\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}).$$

Using the fact  $\mathbf{q} = \frac{1}{|\mathcal{S}|} \mathbf{e}$ , we further have  $Gap = (1 - \gamma)(\mathbf{q})^\top (I - \gamma P^{\hat{\pi}})(\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}) \geq \frac{1 - \gamma}{|\mathcal{S}|} \|(I - \gamma P^{\hat{\pi}})(\mathbf{v}^* - \mathbf{v}^{\hat{\pi}})\|_\infty$ . We use the triangle inequality and the matrix norm inequality  $\|Ax\|_\infty \leq \|A\|_\infty \|x\|_\infty$  to obtain  $\|(I - \gamma P^{\hat{\pi}})(\mathbf{v}^* - \mathbf{v}^{\hat{\pi}})\|_\infty \geq \|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty - \|\gamma P^{\hat{\pi}}(\mathbf{v}^* - \mathbf{v}^{\hat{\pi}})\|_\infty \geq \|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty - \|\gamma P^{\hat{\pi}}\|_\infty \|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty = (1 - \gamma) \|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty$ . It follows that  $Gap \geq \frac{(1 - \gamma)^2}{|\mathcal{S}|} \|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty$ .  $\blacksquare$

Next we will see that, the duality gap provides a sharper upper bound for the policy error when the associated Markov process is ‘‘better’’ behaved. For an arbitrary policy  $\pi$ , we define  $\nu^\pi$  to be the *stationary distribution under policy  $\pi$* , i.e.,  $\nu^\pi = (P^\pi)^\top \nu^\pi$ .

**Proposition 4.** *Suppose that the Markov decision process specified by  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$  is ergodic in the sense that  $c_1 \mathbf{q} \leq \nu^\pi \leq c_2 \mathbf{q}$  for some distribution vector  $\mathbf{q}$  and any policy  $\pi$ . Let  $\hat{\mu} \in \mathcal{U}_{\theta, \mathbf{q}}$  where  $\theta = 1 - \gamma + \gamma \frac{c_1}{c_2}$ , and let  $\hat{\pi}_{i,a} = \frac{\hat{\mu}_{i,a}}{\sum_{a \in \mathcal{A}} \hat{\mu}_{i,a}}$ . Then*

$$\mathbf{q}^\top \mathbf{v}^* - \mathbf{q}^\top \mathbf{v}^{\hat{\pi}} \leq \frac{c_2^2}{(1 - \gamma)c_1^2} \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right).$$

*Proof.* We have

$$\begin{aligned}
\sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i}(\mathbf{v}^* - \gamma P_a \mathbf{v}^* - \mathbf{r}_a)_i &\geq \left(1 - \gamma + \gamma \frac{c_1}{c_2}\right) \sum_{i \in \mathcal{S}} q_i \sum_{a \in \mathcal{A}} \hat{\pi}_{i,a}(\mathbf{v}^* - \gamma P_a \mathbf{v}^* - \mathbf{r}_a)_i \\
&= \left(1 - \gamma + \gamma \frac{c_1}{c_2}\right) \sum_{i \in \mathcal{S}} q_i (\mathbf{v}^* - \gamma P^{\hat{\pi}} \mathbf{v}^* - \mathbf{r}^{\hat{\pi}})_i \\
&\geq \left(1 - \gamma + \gamma \frac{c_1}{c_2}\right) \sum_{i \in \mathcal{S}} \frac{1}{c_2} \nu_i^{\hat{\pi}} (\mathbf{v}^* - \gamma P^{\hat{\pi}} \mathbf{v}^* - \mathbf{r}^{\hat{\pi}})_i \\
&= \left(1 - \gamma + \gamma \frac{c_1}{c_2}\right) \frac{1}{c_2} (\nu^{\hat{\pi}})^\top (I - \gamma P^{\hat{\pi}}) (\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}) \\
&= \left(1 - \gamma + \gamma \frac{c_1}{c_2}\right) \frac{1}{c_2} (1 - \gamma) (\nu^{\hat{\pi}})^\top (\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}) \\
&\geq \frac{c_1^2}{c_2^2} (1 - \gamma) \mathbf{q}^\top (\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}),
\end{aligned}$$

where the first inequality uses  $\hat{\mu} \in \mathcal{U}_{\theta, \mathbf{q}}$ , the second and third inequalities use  $\nu^{\hat{\pi}} \in \mathcal{U}_{\theta, \mathbf{q}}$ .  $\blacksquare$

Prop. 4 suggests that, when all ergodic distributions of the Markov decision process under stationary policies belong to a certain range, we get a sharper bound for the cumulative value of the output policy from the duality gap bound.

### 5.3 Approximate Policy Evaluation

In Algorithm 2, we run Algorithm 1 for multiple independent trials and identify the most successful one, in order to boost the probability of finding a good policy to be arbitrarily close to 1. Before we can do that, we need to be able to evaluate multiple candidate policies and select the best one out of many with high probability (Step 3 of Algorithm 2). In fact, we show that it is possible to approximately evaluate any policy  $\pi$  within  $\epsilon$  precision in run time  $\tilde{O}\left(\frac{1}{\epsilon^2(1-\gamma)^2}\right)$  for a pre-specified initial distribution.

**Proposition 5 (Approximate Policy Evaluation).** *Suppose we are given a DMDP tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$ , a fixed randomized policy  $\pi$ , and an initial distribution  $\mathbf{q}$ . Suppose that a state transition of the DMDP under  $\pi$  can be sampled in  $\tilde{O}(1)$  time, there exists an algorithm that outputs an approximate value  $\bar{Y}$  such that  $\mathbf{q}^\top \mathbf{v}^\pi - \epsilon \leq \bar{Y} \leq \mathbf{q}^\top \mathbf{v}^\pi$  with probability at least  $1 - \delta$  in run time  $\tilde{O}\left(\frac{1}{\epsilon^2(1-\gamma)^2} \log\left(\frac{1}{\delta}\right)\right)$ .*

*Proof.* The approximate policy evaluation algorithm runs as follows: For a given policy  $\pi$ , we simulate the Markov decision process under policy  $\pi$  from the initial distribution  $\mathbf{q}$  for  $n$  transitions and calculate the cumulative discounted reward  $Y$ . We repeat the simulation for  $K$  independent times and return the average cumulative reward  $\bar{Y} = \frac{1}{K}(Y_1 + \dots + Y_K)$ .

We observe that the unknown value  $\mathbf{q}^\top \mathbf{v}^\pi$  is the expectation of the infinite cumulative discounted reward. We pick  $n$  sufficiently large such that expected  $n$ -period cumulative discounted reward is sufficiently close to  $\mathbf{q}^\top \mathbf{v}^\pi$ . Since  $r_{ij}(a) \in [\frac{1}{2}, 1]$  for  $i, j, a$ , the cumulative discounted reward starting from the  $(n+1)$ th period is bounded by  $\sum_{t=n}^{\infty} \gamma^t = \frac{\gamma^n}{1-\gamma}$  with probability 1. In particular, we pick  $n$  such that  $\frac{\epsilon}{2} = \frac{\gamma^n}{1-\gamma}$ , which suggests that  $n = \left(\log_\gamma\left(\frac{\epsilon(1-\gamma)}{2}\right)\right) = \tilde{O}(1)$ . Therefore we obtain

$$\mathbf{q}^\top \mathbf{v}^\pi - \frac{\epsilon}{2} \leq \mathbf{E}[Y_1] \leq \mathbf{q}^\top \mathbf{v}^\pi.$$

Note that  $Y_1, \dots, Y_K$  are i.i.d. random variables and  $Y_k \in [0, \frac{1}{1-\gamma}]$  for all  $k$ . By using the Azuma-Hoeffding inequality, we obtain that  $\bar{Y} = \frac{1}{K} \sum_{t=1}^K Y_t$  satisfies for any  $\epsilon > 0$  that

$$\mathbf{P}\left(|\bar{Y} - \mathbf{E}[Y_1]| \geq \epsilon\right) \leq 2 \exp\left(-\frac{\epsilon^2 K (1-\gamma)^2}{2}\right).$$

By letting  $\varepsilon = \frac{\epsilon}{2}$  and  $K = \mathcal{O}(\frac{1}{\epsilon^2(1-\gamma)^2} \log(1/\delta))$ , we obtain that  $|\bar{Y} - \mathbf{E}[Y_1]| \leq \frac{\epsilon}{2}$  with probability at least  $1 - \delta$ . It follows that

$$\mathbf{P}(\mathbf{q}^\top \mathbf{v}^\pi - \epsilon \leq \bar{Y} \leq \mathbf{q}^\top \mathbf{v}^\pi) \geq 1 - \delta.$$

The number of sample state transitions is  $K \cdot n = \tilde{\mathcal{O}}(\frac{1}{\epsilon^2(1-\gamma)^2} \log(\frac{1}{\delta}))$ , which equals to the total run time. ■

## 5.4 Run-Time Complexity For Algorithms 1 and 2

Finally, we are ready to develop the main results of this paper. Our first main result is given in Theorem 1. It establishes the run-time complexity for computing an  $\epsilon$ -optimal policy for arbitrary DMDP using the randomized linear programming methods given by Algorithms 1 and 2.

**Theorem 1 (Run-Time Complexity for Arbitrary DMDP).** *Let  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$  be an arbitrary DMDP. For any  $\epsilon \in (0, 1)$ ,  $\delta \in (0, 1)$ , let  $\mathbf{q} = \frac{1}{|\mathcal{S}|} \mathbf{e}$ ,  $\theta = (1 - \gamma)$ ,  $T = \Omega\left(\frac{|\mathcal{S}|^3 |\mathcal{A}| \log(|\mathcal{S}| |\mathcal{A}|)}{(1-\gamma)^6 \epsilon^2}\right)$ . Then:*

- (i) *Algorithm 1 outputs a policy  $\hat{\pi}$  satisfying  $\mathbf{v}^{\hat{\pi}}(i) \geq \mathbf{v}^*(i) - \epsilon$  for all  $i \in \mathcal{S}$  with probability at least  $2/3$ .*
- (ii) *Algorithm 2 outputs a policy  $\hat{\pi}$  such that  $\mathbf{q}^\top \mathbf{v}^{\hat{\pi}} \geq \mathbf{q}^\top \mathbf{v}^* - \epsilon$  in run time*

$$\tilde{\mathcal{O}}\left(|\mathcal{S}|^2 |\mathcal{A}| + \frac{|\mathcal{S}|^3 |\mathcal{A}|}{(1-\gamma)^6 \epsilon^2} \log\left(\frac{1}{\delta}\right) + \frac{1}{\epsilon^2 (1-\gamma)^2} \left(\log\frac{1}{\delta}\right)^2\right)$$

*with probability at least  $1 - \delta$ .*

*Proof.* (i) We first show that the dual optimal solution  $\mu^*$  to the linear program (2) satisfies  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$  when  $\theta = 1 - \gamma$ ,  $\mathbf{q} = \frac{1}{|\mathcal{S}|} \mathbf{e}$ . To see this, we note the dual feasibility constraint  $\sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^* = (1 - \gamma) \mathbf{q}$ , which implies a lower bound to the dual variable  $\mu$ , given by  $\sum_{a \in \mathcal{A}} \mu_a^* \geq (1 - \gamma) \mathbf{q}$ . Therefore  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$  and the assumption of Proposition 2 is satisfied.

We denote for short that  $Gap = \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right)$ . Now we apply Prop. 3 and the Markov inequality to obtain  $\|\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}\|_\infty \leq \frac{|\mathcal{S}|}{(1-\gamma)^2} Gap \leq \mathcal{O}\left(\frac{|\mathcal{S}|}{(1-\gamma)^2}\right) \mathbf{E}[Gap]$  with probability at least  $2/3$ . Therefore  $\mathbf{v}^\pi \geq \mathbf{v}^* - \epsilon \mathbf{e}$  with probability  $2/3$  when  $\mathbf{E}[Gap] \leq \mathcal{O}\left(\frac{\epsilon(1-\gamma)^2}{|\mathcal{S}|}\right)$ , which holds if we let  $T = \Omega\left(\frac{|\mathcal{S}|^3 |\mathcal{A}| \log(|\mathcal{S}| |\mathcal{A}|)}{(1-\gamma)^6 \epsilon^2}\right)$  and apply Prop. 2.

(ii) Let us analyze Algorithm 2 step by step.

1. In Step 2 of Algorithm 2, it runs Algorithm 1 for  $K$  independent trials with precision parameter  $\frac{\epsilon}{2}$  and generates output policies  $\pi^{(1)}, \dots, \pi^{(K)}$ . The total run time is  $\mathcal{O}(|\mathcal{S}|^2 |\mathcal{A}|) + K \cdot T_{\frac{\epsilon}{2}}$ , where  $T_{\frac{\epsilon}{2}}$  is the time complexity for each run of Algorithm 1. According to (i), each trial generates an  $\epsilon/2$ -optimal policy with probability at least  $2/3$ .
2. In Step 3 of Algorithm 2, for each output policy  $\pi^{(k)}$ , we conduct approximate value evaluation and obtain an approximate evaluation  $\bar{Y}^{(k)}$  with precision level  $\frac{\epsilon}{2}$  and fail probability  $\frac{\delta}{2K}$ . According to Lemma 5, we have

$$\bar{Y}^{(k)} - \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} \in \left[-\frac{\epsilon}{2}, 0\right],$$

with probability at least  $1 - \frac{\delta}{2K}$ . This step takes  $K \cdot \tilde{\mathcal{O}}\left(\frac{1}{\epsilon^2(1-\gamma)^2} \log\left(\frac{K}{\delta}\right)\right)$  run time.

3. Step 4 of Algorithm 2 outputs  $\hat{\pi} = \pi^{(k^*)}$  such that  $k^* = \operatorname{argmax}_{k=1, \dots, K} \bar{Y}^{(k)}$ . This step takes  $\mathcal{O}(K)$  run time.

Now we verify that  $\hat{\pi}$  is indeed near-optimal with probability at least  $1 - \delta$  when  $K$  is chosen appropriately. Let  $\mathcal{K} = \left\{k \mid \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} \geq \mathbf{q}^\top \mathbf{v}^* - \frac{\epsilon}{2}\right\}$ , which can be interpreted as indices of successful trails of Algorithm 1. Consider the event where  $\mathcal{K} \neq \emptyset$  and all policy evaluation errors belong to the small interval  $[-\frac{\epsilon}{2}, 0]$ . In this

case, we have  $\bar{Y}^{(k)} \leq \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}}$  for all  $k$  and  $\bar{Y}^{(k)} \geq \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} - \frac{\epsilon}{2} \geq \mathbf{q}^\top \mathbf{v}^* - \epsilon$  if  $k \in \mathcal{K}$ . As a result, the output policy  $\hat{\pi}$  which has the largest value of  $\bar{Y}^{(k)}$  must be  $\epsilon$ -optimal. We use the union bound to obtain

$$\begin{aligned} \mathbf{P}(\mathbf{q}^\top \mathbf{v}^{\hat{\pi}} < \mathbf{q}^\top \mathbf{v}^* - \epsilon) &\leq \mathbf{P}\left(\{\mathcal{K} = \emptyset\} \cup \left\{\exists k : \bar{Y}^{(k)} - \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} \notin \left[-\frac{\epsilon}{2}, 0\right]\right\}\right) \\ &\leq \mathbf{P}(\mathcal{K} = \emptyset) + \mathbf{P}\left(\exists k : \bar{Y}^{(k)} - \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} \notin \left[-\frac{\epsilon}{2}, 0\right]\right) \\ &\leq \prod_{k=1}^K \mathbf{P}\left(\mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} < \mathbf{q}^\top \mathbf{v}^* - \frac{\epsilon}{2}\right) + \sum_{k=1}^K \mathbf{P}\left(\bar{Y}^{(k)} - \mathbf{q}^\top \mathbf{v}^{\pi^{(k)}} \notin \left[-\frac{\epsilon}{2}, 0\right]\right) \\ &\leq (1/3)^K + K \cdot \frac{\delta}{2K}. \end{aligned}$$

By choosing  $K = \Omega(\log(1/\delta))$ , we obtain  $\mathbf{P}(\mathbf{q}^\top \mathbf{v}^{\hat{\pi}} < \mathbf{q}^\top \mathbf{v}^* - \epsilon) \leq \delta$ . Then the output policy is  $\epsilon$ -optimal when initiated at distribution  $\mathbf{q}$  with probability at least  $1 - \delta$ .

According to Section 4, preprocessing of Algorithms 1-2 takes  $\tilde{\mathcal{O}}(|\mathcal{S}|^2|\mathcal{A}|)$  arithmetic operations and each iteration takes  $\tilde{\mathcal{O}}(1)$  arithmetic operations. The total run time is  $\tilde{\mathcal{O}}(|\mathcal{S}|^2|\mathcal{A}| + T_{\frac{\epsilon}{2}} \log \frac{1}{\delta} + \frac{1}{\epsilon^2(1-\gamma)^2} (\log \frac{1}{\delta})^2)$ . ■

Theorem 1 establishes a worst-case complexity for Algorithms 1 and 2. The results apply to all instances of DMDP with  $|\mathcal{S}|$  states,  $|\mathcal{A}|$  actions per state, and a fixed discount factor  $\gamma$ . Theorem 1 does not require any additional property such as irreducibility or aperiodicity of the associated Markov chains. In fact, the results even hold for problems with transient states and/or multiple optimal policies.

The cubic dependence  $|\mathcal{S}|^3$  in Theorem 1 is not very satisfying. One factor of  $|\mathcal{S}|$  comes from the duality gap bound (Prop. 2), and two more factors come from rounding the duality gap to the policy error  $\|\mathbf{v}^{\hat{\pi}} - \mathbf{v}^*\|_\infty$  (Prop. 3). We conjecture that the term  $|\mathcal{S}|^3$  can be improved to  $|\mathcal{S}|^2$  (or even  $|\mathcal{S}|$ ) using an improved algorithm and analysis. In addition, we conjecture that the complexity result should have a better dependence on  $\frac{1}{1-\gamma}$ , especially when all  $P^\pi$ 's have relatively large spectral gaps. These two questions are left open for future investigation.

Next we will see that, the DMDP becomes easier to solve when the associated Markov process is ergodic. In what follows, we focus on the class of Markov decision processes where every stationary policy generates an ergodic Markov chain. For an arbitrary policy  $\pi$ , we define  $\nu^\pi$  to be the *stationary distribution under policy  $\pi$* , i.e.,  $\nu^\pi = (P^\pi)^\top \nu^\pi$ . Our next main result shows that Algorithms 1-2 have significantly improved complexity for ergodic DMDP.

**Theorem 2 (Linear Run Time for Ergodic DMDP).** *Suppose that the Markov decision process specified by  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \gamma)$  is ergodic in the sense that  $c_1 \mathbf{q} \leq \nu^\pi \leq c_2 \mathbf{q}$  for some distribution vector  $\mathbf{q}$  and any policy  $\pi$ . For any  $\epsilon \in (0, 1)$ ,  $\delta \in (0, 1)$ , let  $\theta = 1 - \gamma + \gamma \frac{c_1}{c_2}$ ,  $T = \Omega\left(\left(\frac{c_2}{c_1}\right)^4 \frac{|\mathcal{S}||\mathcal{A}| \log(|\mathcal{S}||\mathcal{A}|)}{(1-\gamma)^4 \epsilon^2}\right)$ . Then:*

(i) *Algorithm 1 outputs a policy  $\hat{\pi}$  satisfying  $\mathbf{q}^\top \mathbf{v}^{\hat{\pi}} \geq \mathbf{q}^\top \mathbf{v}^* - \epsilon$  with probability at least  $2/3$ .*

(ii) *Algorithm 2 outputs a policy  $\hat{\pi}$  such that  $\mathbf{q}^\top \mathbf{v}^{\hat{\pi}} \geq \mathbf{q}^\top \mathbf{v}^* - \epsilon$  in run time*

$$\tilde{\mathcal{O}}\left(|\mathcal{S}|^2|\mathcal{A}| + \left(\frac{c_2}{c_1}\right)^4 \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4 \epsilon^2} \log\left(\frac{1}{\delta}\right) + \frac{1}{\epsilon^2(1-\gamma)^2} \left(\log \frac{1}{\delta}\right)^2\right) \quad (5)$$

*with probability at least  $1 - \delta$ .*

*Proof.* (i) We first verify that  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$  with the given vector  $\mathbf{q}$  and  $\theta = 1 - \gamma + \gamma \frac{c_1}{c_2}$ . We note that all eigenvalues of a probability transition matrix  $P^\pi$  belong to the unit circle and  $\gamma \in (0, 1)$ , therefore the eigenvalues  $I - \gamma P^\pi$  belong to the positive half plane. As a result, the matrix  $I - \gamma P^\pi$  is invertible for all  $\pi$ ,

including  $I - \gamma P^*$ . We have  $\sum_{a \in \mathcal{A}} \mu_a^* = (I - \gamma(P^*)^\top)^{-1} (1 - \gamma) \mathbf{q}$ , therefore

$$\begin{aligned}
\sum_{a \in \mathcal{A}} \mu_a^* &= (1 - \gamma) \left( \sum_{k=0}^{\infty} (\gamma P^*)^k \right)^\top \mathbf{q} = (1 - \gamma) \mathbf{q} + (1 - \gamma) \left( \sum_{k=1}^{\infty} (\gamma P^*)^k \right)^\top \mathbf{q} \\
&\geq (1 - \gamma) \mathbf{q} + \frac{1}{c_2} (1 - \gamma) \sum_{k=1}^{\infty} ((\gamma P^*)^k)^\top \nu^* \\
&= (1 - \gamma) \mathbf{q} + \frac{1}{c_2} (1 - \gamma) \left( \sum_{k=1}^{\infty} \gamma^k \right) \nu^* \\
&\geq (1 - \gamma) \mathbf{q} + \frac{c_1}{c_2} \gamma \mathbf{q} \\
&\geq \left( 1 - \gamma + \gamma \frac{c_1}{c_2} \right) \mathbf{q}.
\end{aligned}$$

As a result, we have verified  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$  and the assumption and results of Prop. 2 hold.

When Algorithm 1 is applied with  $\mathbf{q}$  and  $\theta = 1 - \gamma + \gamma \frac{c_1}{c_2}$ . By the nature of the algorithm, we have  $\sum_{a \in \mathcal{A}} \hat{\mu}_a \geq \theta \mathbf{q}$ . Then we have and  $c_1 \mathbf{q} \geq \nu^\pi \geq c_2 \mathbf{q}$  for any policy  $\pi$ , so the assumptions of Prop. 4 hold. We denote for short that  $Gap = \sum_{a \in \mathcal{A}, i \in \mathcal{S}} \hat{\mu}_{a,i} \left( v_i^* - \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^* - \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) \right)$ . By applying Prop. 4, we obtain that  $\mathbf{q}^\top (\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}) \leq \frac{c_2^2}{c_1^2(1-\gamma)} Gap$ . Then we may use the Markov inequality to show that  $\mathbf{q}^\top (\mathbf{v}^* - \mathbf{v}^{\hat{\pi}}) \leq \epsilon$  with probability  $2/3$  as long as  $\frac{c_2^2}{c_1^2(1-\gamma)} \mathbf{E}[Gap] \leq 2/3\epsilon$ , which requires  $T = \Omega \left( \left( \frac{c_2}{c_1} \right)^4 \frac{|\mathcal{S}||\mathcal{A}| \log(|\mathcal{S}||\mathcal{A}|)}{(1-\gamma)^4 \epsilon^2} \right)$  according to Prop. 2.

(ii) By using a similar analysis as in the proof of Theorem 1, we finish the proof.  $\blacksquare$

Theorem 2 shows that the iteration complexity of Algorithms 1-2 substantially improves when the DMDP is ergodic under every stationary policy. More specifically, the complexity reduces when all policies generate “similar” stationary distributions. Comparing the preceding complexity result with the input size  $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$  of the DMDP, we conclude that Algorithms 1-2 have nearly-linear run-time complexity in the worst case. For large-scale problems, as long as  $|\mathcal{S}| \gg \frac{1}{\epsilon}$  and  $|\mathcal{A}| \gg \frac{1}{\epsilon}$ , Algorithm 1 is able to compute an approximate policy more efficiently than any known deterministic method.

The ratio  $\frac{c_2}{c_1}$  characterizes a notion of complexity of ergodic DMDP, i.e., the range of possible ergodic distributions under different policies. Intuitively, dynamic programs are harder to solve when different policies lead to vastly different state trajectories. For example, suppose that there is a critical state that can only show up after a particular sequence of correct actions (this typical happens in aperiodic Markov chains). In this case, any algorithm would need to search over the space of all policies to identify the critical state. For another example, consider that all policies only affect the immediate reward but will lead to the same outgoing transition probabilities. In this case, one would be able learn the optimal action at each state much more efficiently, where  $c_1/c_2 = 1$ .

Our last result shows that it is possible to skip the preprocessing step, as long as the DMDP tuple is specified using special formats. When the input data are given in a way that enables immediate sampling, one can skip Step 5 in Algorithm 1 and remove the first term  $|\mathcal{S}||\mathcal{A}|$  from the overall run time.

**Theorem 3 (Sublinear Time Complexity for Ergodic DMDP In Special Formats).** *Suppose that the assumptions of Theorem 2 hold and the collection of transition probabilities  $\mathcal{P} = (P_a)_{a \in \mathcal{A}}$  are specified in any one of the following formats:*

- (a) *Matrices of cumulative probabilities  $C_a$  of dimension  $\mathcal{S} \times \mathcal{S}$ , for all  $a \in \mathcal{A}$ , where  $C_a(i, j) = \sum_{k=1}^j P_a(i, k)$  for all  $i \in \mathcal{S}, a \in \mathcal{A}$  and  $j \in \mathcal{S}$ .*
- (b) *Transition probability distributions  $P_a(i, \cdot)$  that are encoded in binary trees. There are  $|\mathcal{S}||\mathcal{A}|$  trees, one for each state-action pair  $(i, a) \in \mathcal{S} \times \mathcal{A}$ . Each binary tree is of depth  $\log |\mathcal{S}|$  and has  $|\mathcal{S}|$  leaves that store the values of  $P_a(i, j)$ ,  $j \in \mathcal{S}$ . Each inner node of the tree stores the sum of its two children.*

Then for any  $\epsilon \in (0, 1)$ ,  $\delta \in (0, 1)$ , Algorithm 2 outputs an approximately optimal policy  $\hat{\pi}$  such that  $\mathbf{q}^\top \mathbf{v}^{\hat{\pi}} \geq \mathbf{q}^\top \mathbf{v}^* - \epsilon$  in run time

$$\tilde{\mathcal{O}} \left( \left( \frac{c_2}{c_1} \right)^4 \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^4 \epsilon^2} \log \left( \frac{1}{\delta} \right) + \frac{1}{\epsilon^2 (1-\gamma)^2} \left( \log \frac{1}{\delta} \right)^2 \right)$$

with probability at least  $1 - \delta$ .

*Proof.* In both cases (a) and (b), it is possible to draw a sample coordinate  $j \mid (i, a)$  with probability  $p_{ij}(a)$  using  $\mathcal{O}(\log |\mathcal{S}|)$  run time, because the corresponding data structure storing the vector  $P_a(i, \cdot)$  can be readily used as a sampler [7, 30]. Therefore one can skip the preprocessing step (Step 5) in Algorithm 1 and remove the  $\tilde{\mathcal{O}}(|\mathcal{S}|^2|\mathcal{A}|)$  term from the run-time result of Theorem 2. ■

When the input is given in suitable data structures, the run-time complexity of Algorithms 1-2 reduces from nearly-linear to sublinear with respect to the input size. The reduced run time is almost linear in  $|\mathcal{S} \times \mathcal{A}|$ , i.e., the number of state-action pairs. It means that for fixed values of  $\epsilon, \gamma$ , each state-action pair is queried for a constant number of times on average regardless of the dimension of the DMDP. This result is sublinear with respect to the input size  $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ . It suggests that one can find an approximately optimal policy without even reading a significant portion of the input entries. We recall that the input data mainly consist of transition probabilities  $p_{ij}(a)$ . An explanation for the sublinear complexity is that certain small probabilities in the input data can be safely ignored, without deteriorating the quality of the approximate policy significantly.

Theorems 1, 2, 3 established new complexity upper bounds for computing an approximate-optimal policy of the DMDP. Let us compare the upper bounds given by Theorems 1, 2, 3 with recent lower bound results for DMDP [9]. For DMDP that is specified in the standard way (arrays of transition probabilities), it shows that any randomized algorithm needs at least  $\Omega(|\mathcal{S}|^2|\mathcal{A}|)$  run time to get any  $\epsilon$ -approximate policy with high probability; see Theorem 1 of [9]. It also shows that the lower bound reduces to  $\Omega(\frac{|\mathcal{S}||\mathcal{A}|}{\epsilon})$  when the input data are in the format of binary trees or cumulative sums (for which the preprocessing step can be skipped); see Theorems 2-3 of [9]. Our upper bound results nearly match the lower bound results in both cases. Both the upper and lower bounds suggest that the computational complexity of DMDP indeed depends on the input data structure. The overall complexity for approximately solving the MDP is dominated by the preprocessing time. Once the data is preprocessed, the remaining computation problem becomes significantly easier.

## 6 Remarks

We have developed a novel randomized method that exploits the linear duality between the value function and the policy function for solving DMDPs. It is related to several fundamental methods in linear programming, stochastic optimization and online learning. It is easy to implement, uses sublinear space complexity and nearly-linear (sometimes sublinear) run-time complexity.

Algorithm 1 can be viewed as a randomized version of the simplex method, which is also equivalent to a version of the policy iteration method for solving MDP. It maintains a primal variable (value) and a dual variable (policy). Each update  $\pi_{i,a} \leftarrow \pi_{i,a} \cdot \exp\{\Delta\}$  mimics a pivoting step towards a neighboring basis. Instead of moving from one basis to another one, each update in the dual variable can be viewed as a “soft” pivot. Our theoretical results show that the algorithm finds an approximate policy in  $\tilde{\mathcal{O}}(SA)$  iterations. This is somewhat similar to the simplex method, which also terminates in  $\tilde{\mathcal{O}}(SA)$  (which is the number of constraints) iterations on average. What makes our randomized algorithm different is its  $\tilde{\mathcal{O}}(1)$  run time per iteration. It avoids explicitly solving any linear system, which is unavoidable in the simplex method.

Algorithm 1 can also be viewed as a stochastic approximation method for solving a saddle point problem. It utilizes the structures of specially crafted primal and dual constraint sets to make each update as simple and efficient as possible. The update of the dual variable (policy) uses a special Bregman divergence function which is related to the relative entropy between randomized policies.

It is worth noting that Algorithm 1 is related to the exponentiated gradient method for the multi-arm bandit problem in the online learning setting. When there is a single state, Algorithm 1 reduces to the basic

exponentiated gradient method. This observation provides a hint that we might be able to adapt Algorithm 1 to apply to online reinforcement learning. This is a direction for future research.

The new method and complexity results of this paper suggest a promising direction that awaits further research. The current results leave open many questions. We conjecture that the complexity result should have a better dependence on  $\frac{1}{1-\gamma}$ , especially when all  $P^\pi$ 's have relatively large spectral gaps. A related notion of complexity metric for MDP is the diameter, i.e., the maximal expected time to move from any state to any other state. We conjecture that the diameter should play a key role in an improved complexity analysis and replace at least one factor of  $\frac{1}{1-\gamma}$ . We also conjecture that the sublinear-time complexity result of Proposition 3 hold for more general DMDPs without prior knowledge about  $\frac{c_2}{c_1}$ . Another direction for future research is to consider the run-time complexity for finite-horizon MDP and average-reward MDP. It remains unclear what roles the mixing rate and the horizon play in the run-time complexity. In the mean time, an equally important (if not more) question is to establish the computation complexity lower bound for approximating optimal policies of MDP.

## 7 Proof of Proposition 2

We provide the complete proof of Proposition 2 in this last section. Readers who are not interested in the technical details are free to skip this part.

### 7.1 Technical Lemmas

In this section, we analyze the convergence of Algorithm 1. We denote the  $t$ -th iterates generated by Algorithm 1 by  $\pi^t$ ,  $\xi^t$ , and  $\mathbf{v}^t$ . We define the auxiliary variables  $\lambda = (\lambda_{i,a}^t)_{i \in \mathcal{S}, a \in \mathcal{A}}$  as

$$\lambda_{i,a}^t = \xi_i^t \pi_{i,a}^t.$$

According to Algorithm 1, we can verify that  $\xi^t \in \mathfrak{R}^{|\mathcal{S}|}$  and  $\pi_i^t \in \mathfrak{R}^{|\mathcal{A}|}$  are vectors of probability distributions. It follows that  $\lambda^t$  is always a  $|\mathcal{S}||\mathcal{A}|$ -dimensional vector of probability distribution. In addition, the updates on  $\xi^t$  and  $\pi^t$  can be equivalently written as updates on  $\lambda^t$  and  $\pi^t$ , given by

$$\lambda_{i,a}^{t+1} = \frac{\lambda_{i,a}^t \cdot \exp(\Delta_{i,a}^{t+1})}{\sum_{i',a'} \lambda_{i',a'}^t \cdot \exp(\Delta_{i',a'}^{t+1})}, \quad \pi_{i,a}^{t+1} = \frac{\pi_{i,a}^t \cdot \exp(\Delta_{i,a}^{t+1})}{\sum_{a'} \pi_{i,a'}^t \cdot \exp(\Delta_{i,a'}^{t+1})} \quad \forall i \in \mathcal{S}, a \in \mathcal{A}, \quad (6)$$

where

$$\Delta_{i,a}^{t+1} = \begin{cases} \beta \cdot \frac{(\gamma v_j^t - v_i^t + r_{ij^t(a)} - M)}{((1-\theta)\xi_i^t + \theta q_i)\pi_{i,a}^t} & \text{if } i = i_{t+1}, a = a_{t+1} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

In what follows, we denote by  $\mathcal{F}_t$  the collection of random variables that are revealed up to the end of the  $t$ -th iteration. We denote by  $\mu_{i,a}^t$  the dual iterates given by

$$\mu_{i,a}^t = ((1-\theta)\xi_i^t + \theta q_i)\pi_{i,a}^t = (1-\theta)\lambda_{i,a}^t + \theta q_i \pi_{i,a}^t.$$

According to Algorithm 1, we can verify that  $\mu^t \in \mathcal{U}_{\theta, \mathbf{q}}$  and  $\mathbf{v}^t \in \mathcal{V}$  for all  $t$  with probability 1.

**Lemma 1.** *If  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ , there exists probability distribution vectors  $\lambda^* \in \mathfrak{R}^{|\mathcal{S}||\mathcal{A}|}$  and  $\pi_i^* \in \mathfrak{R}^{|\mathcal{A}|}$ ,  $i \in \mathcal{S}$ , such that*

$$\mu_{i,a}^* = (1-\theta)\lambda_{i,a}^* + \theta q_i \pi_{i,a}^*, \quad \forall i \in \mathcal{S}, a \in \mathcal{A}.$$

*Proof.* The proof is straightforward by the definition of  $\mathcal{U}_{\theta, \mathbf{q}}$ . ■

**Lemma 2.** *The iterates generated by Algorithm 1 satisfy*

$$\mathbf{E} [D_{KL}(\lambda^* || \lambda^{t+1}) | \mathcal{F}_t] - D_{KL}(\lambda^* || \lambda^t) \leq \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} (\lambda_{i,a}^t - \lambda_{i,a}^*) \mathbf{E} [\Delta_{i,a}^{t+1} | \mathcal{F}_t] + \frac{1}{2} \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \mathbf{E} [(\Delta_{i,a}^{t+1})^2 | \mathcal{F}_t], \quad (8)$$

for all  $t$ , with probability 1.

*Proof.* By using the relation (6), we have

$$\begin{aligned}
D_{KL}(\lambda^* || \lambda^{t+1}) - D_{KL}(\lambda^* || \lambda^t) &= \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \log \frac{\lambda_{i,a}^*}{\lambda_{i,a}^{t+1}} - \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \log \frac{\lambda_{i,a}^*}{\lambda_{i,a}^t} \\
&= \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \log \frac{\lambda_{i,a}^t}{\lambda_{i,a}^{t+1}} \\
&= \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \log \frac{Z}{\exp(\Delta_{i,a}^{t+1})} \\
&= \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \log(Z) - \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \Delta_{i,a}^{t+1} \\
&= \log Z - \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^* \Delta_{i,a}^{t+1},
\end{aligned} \tag{9}$$

where  $Z = \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \exp(\Delta_{i,a}^{t+1})$ . According to (7), we have  $\gamma v_j^t - v_i^t + r_{ij^t}(a) - M \leq \frac{\gamma}{1-\gamma} - 0 + 1 - \frac{1}{1-\gamma} \leq 0$  because  $v_i \in [0, \frac{1}{1-\gamma}]$ ,  $r_{ij^t}(a) \in [\frac{1}{2}, 1]$  and  $M = \frac{1}{1-\gamma}$ . It follows that  $\Delta_{i,a}^{t+1} \leq 0$  for all  $i \in \mathcal{S}, a \in \mathcal{A}$  with probability 1. Then we derive

$$\begin{aligned}
\log Z &= \log \left( \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \exp(\Delta_{i,a}^{t+1}) \right) \leq \log \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \left( 1 + \Delta_{i,a}^{t+1} + \frac{1}{2} (\Delta_{i,a}^{t+1})^2 \right) \\
&= \log \left( 1 + \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \Delta_{i,a}^{t+1} + \frac{1}{2} \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t (\Delta_{i,a}^{t+1})^2 \right) \\
&\leq \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \Delta_{i,a}^{t+1} + \frac{1}{2} \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t (\Delta_{i,a}^{t+1})^2,
\end{aligned} \tag{10}$$

where the first inequality uses the fact  $e^x \leq 1 + x + \frac{1}{2}x^2$  if  $x \leq 0$  and the second inequality uses the fact  $\log(1+x) \leq x$  for all  $x$ . We combine (9) and (10) and take conditional expectation  $\mathbf{E}[\cdot | \mathcal{F}_t]$  on both sides, then we obtain (8).  $\blacksquare$

**Lemma 3.** For any  $i \in \mathcal{S}$ , the iterates generated by Algorithm 1 satisfy

$$\mathbf{E} \left[ D_{KL}(\pi_i^* || \pi_i^{t+1}) \mid \mathcal{F}_t \right] - D_{KL}(\pi_i^* || \pi_i^t) \leq \sum_{a \in \mathcal{A}} (\pi_{i,a}^t - \pi_{i,a}^*) \mathbf{E} [\Delta_{i,a}^{t+1} | \mathcal{F}_t] + \frac{1}{2} \sum_{a \in \mathcal{A}} \pi_{i,a}^t \mathbf{E} [(\Delta_{i,a}^{t+1})^2 | \mathcal{F}_t], \tag{11}$$

for all  $t \geq 1$ , with probability 1.

*Proof.* The proof is similar to Lemma 2. We omit it for simplicity.  $\blacksquare$

**Lemma 4.** The iterates generated by Algorithm 1 satisfy

$$\sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t \mathbf{E} [(\Delta_{i,a}^{t+1})^2 | \mathcal{F}_t] \leq \frac{4|\mathcal{S}||\mathcal{A}|\beta^2}{(1-\gamma)^2},$$

for all  $t \geq 1$  with probability 1.

*Proof.* We note that  $\mathbf{P}(i_{t+1} = i, a_{t+1} = a \mid \mathcal{F}_t) = \mu_{i,a}^t$ . Then we have

$$\begin{aligned} \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t \mathbf{E} \left[ (\Delta_{i,a}^{t+1})^2 \mid \mathcal{F}_t \right] &= \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t \cdot \mu_{i,a}^t \cdot \sum_{j \in \mathcal{S}} p_{ij}(a) \left( \beta \cdot \frac{(\gamma v_j^t - v_i^t + r_{ij}(a) - M)}{\mu_{i,a}^t} \right)^2 \\ &= \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sum_{j \in \mathcal{S}} p_{ij}(a) (\beta \cdot (\gamma v_j^t - v_i^t + r_{ij}(a) - M))^2 \\ &\leq \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \left( \beta \cdot \frac{2}{1-\gamma} \right)^2 \\ &= \frac{4|\mathcal{S}||\mathcal{A}|\beta^2}{(1-\gamma)^2}, \end{aligned}$$

where the inequality uses the fact that  $\mathbf{v}^t$  belongs to the  $\|\cdot\|_\infty$  ball of radius  $\frac{1}{1-\gamma}$  and  $M = \frac{1}{1-\gamma}$ .  $\blacksquare$

**Proposition 6.** We let  $\Phi(\mu^t)$  be the divergence function given by

$$\Phi(\mu^t) = (1-\theta)D_{KL}(\lambda^* \parallel \lambda^t) + \theta \sum_{i \in \mathcal{S}} q_i D_{KL}(\pi_i^* \parallel \pi_i^t).$$

The iterates generated by Algorithm 1 satisfy

$$\mathbf{E} [\Phi(\mu^{t+1}) \mid \mathcal{F}_t] \leq \Phi(\mu^t) + \beta \sum_{a \in \mathcal{A}} (\mu_a^t - \mu_a^*)^\top ((\gamma P_a - I)\mathbf{v}^t + \mathbf{r}_a) + \frac{2|\mathcal{S}||\mathcal{A}|\beta^2}{(1-\gamma)^2}, \quad (12)$$

for all  $t$ , with probability 1.

*Proof.* We take the weighted sum between (8) and (11), so we have

$$\begin{aligned} \mathbf{E} [\Phi(\mu^{t+1}) \mid \mathcal{F}_t] &\leq \Phi(\mu^t) + (1-\theta) \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} (\lambda_{i,a}^t - \lambda_{i,a}^*) \mathbf{E} [\Delta_{i,a}^{t+1} \mid \mathcal{F}_t] + \frac{1-\theta}{2} \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda_{i,a}^t \mathbf{E} [(\Delta_{i,a}^{t+1})^2 \mid \mathcal{F}_t] \\ &\quad + \theta \sum_{i \in \mathcal{S}} q_i \sum_{a \in \mathcal{A}} (\pi_{i,a}^t - \pi_{i,a}^*) \mathbf{E} [\Delta_{i,a}^{t+1} \mid \mathcal{F}_t] + \frac{\theta}{2} \sum_{i \in \mathcal{S}} q_i \sum_{a \in \mathcal{A}} \pi_{i,a}^t \mathbf{E} [(\Delta_{i,a}^{t+1})^2 \mid \mathcal{F}_t] \\ &= \Phi(\mu^t) + \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} (\mu_{i,a}^t - \mu_{i,a}^*) \mathbf{E} [\Delta_{i,a}^{t+1} \mid \mathcal{F}_t] + \frac{1}{2} \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t \mathbf{E} [(\Delta_{i,a}^{t+1})^2 \mid \mathcal{F}_t], \end{aligned}$$

where the equality uses the relations  $\mu_{i,a}^t = (1-\theta)\lambda_{i,a}^t + \theta q_i \pi_{i,a}^t$  and  $\mu_{i,a}^* = (1-\theta)\lambda_{i,a}^* + \theta q_i \pi_{i,a}^*$  (by Lemma 1 since  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ ). For arbitrary  $i \in \mathcal{S}$  and  $a \in \mathcal{A}$ , we have

$$\frac{1}{\beta} \cdot \mathbf{E} [\Delta_{i,a}^{t+1} \mid \mathcal{F}_t] = \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) v_j^t - v_i^t + \sum_{j \in \mathcal{S}} p_{ij}(a) r_{ij}(a) - M = (\gamma P_a \mathbf{v}^t - \mathbf{v}^t + \mathbf{r}_a)_i - M.$$

It follows that

$$\begin{aligned} \frac{1}{\beta} \cdot \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} (\mu_{i,a}^t - \mu_{i,a}^*) \mathbf{E} [\Delta_{i,a}^{t+1} \mid \mathcal{F}_t] &= \sum_{a \in \mathcal{A}} \sum_{i \in \mathcal{S}} (\mu_{i,a}^t - \mu_{i,a}^*) [(\gamma P_a \mathbf{v}^t - \mathbf{v}^t + \mathbf{r}_a)_i - M] \\ &= \sum_{a \in \mathcal{A}} (\mu_a^t - \mu_a^*)^\top ((\gamma P_a - I)\mathbf{v}^t + \mathbf{r}_a), \end{aligned}$$

where the second equality comes from the fact  $\sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t = \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^* = 1$  (because  $\mu^t \in \mathcal{U}_{\theta, \mathbf{q}}$ ,  $\mu^* \in \mathcal{U}_{\theta, \mathbf{q}}$ ). According to Lemma 4, we also have  $\mathbf{E} \left[ \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t (\Delta_{i,a}^{t+1})^2 \mid \mathcal{F}_t \right] \leq \frac{4|\mathcal{S}||\mathcal{A}|\beta^2}{(1-\gamma)^2}$ . We apply the preceding two relations and complete the proof.  $\blacksquare$

**Proposition 7.** *The iterates generated by Algorithm 1 satisfy for all  $t$  with probability 1 that*

$$\mathbf{E} [\|\mathbf{v}^{t+1} - \mathbf{v}^*\|^2 \mid \mathcal{F}_t] \leq \|\mathbf{v}^t - \mathbf{v}^*\|^2 + 2\alpha(\mathbf{v}^t - \mathbf{v}^*)^\top \left( \sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^t - (1 - \gamma)\mathbf{q} \right) + 4\alpha^2. \quad (13)$$

*Proof.* We let  $\mathbf{v}^{t+1/2}$  be the vector such that

$$\begin{aligned} v_{i_{t+1}}^{t+1/2} &= v_{i_{t+1}}^t - \alpha \left( \frac{(1 - \gamma)q_{i_{t+1}}}{(1 - \theta)\xi_{i_{t+1}}^t + \theta q_{i_{t+1}}} - 1 \right), \\ v_{j_{t+1}}^{t+1/2} &= v_{j_{t+1}}^t - \alpha\gamma, \\ v_i^{t+1/2} &= v_i^t, \quad \text{if } i \notin \{i_{t+1}, j_{t+1}\}. \end{aligned}$$

Then we can verify that  $\mathbf{v}^{t+1} = \Pi_{\mathcal{V}}\mathbf{v}^{t+1/2}$ , where  $\Pi$  denotes the Euclidean projection. We note that  $\mathbf{P}(i_{t+1} = i \mid \mathcal{F}_t) = (1 - \theta)\xi_i^t + \theta q_i = \sum_{a \in \mathcal{A}} \mu_{i,a}^t$ ,  $\mathbf{P}(j_{t+1} = j \mid \mathcal{F}_t) = \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t p_{ij}(a)$ . Then we can verify that

$$\mathbf{E} [\mathbf{v}^{t+1/2} - \mathbf{v}^t \mid \mathcal{F}_t] = -\alpha \left( (1 - \gamma)\mathbf{q} - \sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^t \right).$$

Also since  $\theta \geq 1 - \gamma$  we have  $\frac{(1 - \gamma)q_{i_{t+1}}}{(1 - \theta)\xi_{i_{t+1}}^t + \theta q_{i_{t+1}}} \in [0, 1]$ . Then we have  $|v_{i_{t+1}}^{t+1/2} - v_{i_{t+1}}^t| < \alpha$  and  $|v_{j_{t+1}}^{t+1/2} - v_{j_{t+1}}^t| < \alpha$ . Then we can verify that  $\|\mathbf{v}^{t+1/2} - \mathbf{v}^t\|^2 \leq 4\alpha^2$  for all  $t$  with probability 1. Finally, by using the nonexpansive property of  $\Pi_{\mathcal{V}}$  and  $\mathbf{v}^* \in \mathcal{V}$ , we further obtain

$$\begin{aligned} \mathbf{E} [\|\mathbf{v}^{t+1} - \mathbf{v}^*\|^2 \mid \mathcal{F}_t] &= \mathbf{E} [\|\Pi_{\mathcal{V}}\mathbf{v}^{t+1/2} - \mathbf{v}^*\|^2 \mid \mathcal{F}_t] \\ &\leq \mathbf{E} [\|\mathbf{v}^{t+1/2} - \mathbf{v}^*\|^2 \mid \mathcal{F}_t] \\ &= \|\mathbf{v}^t - \mathbf{v}^*\|^2 + 2(\mathbf{v}^t - \mathbf{v}^*)^\top \mathbf{E} [\mathbf{v}^{t+1/2} - \mathbf{v}^t \mid \mathcal{F}_t] + \mathbf{E} [\|\mathbf{v}^{t+1/2} - \mathbf{v}^t\|^2 \mid \mathcal{F}_t], \\ &\leq \|\mathbf{v}^t - \mathbf{v}^*\|^2 + 2\alpha(\mathbf{v}^t - \mathbf{v}^*)^\top \left( \sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^t - (1 - \gamma)\mathbf{q} \right) + 4\alpha^2, \end{aligned}$$

for all  $t$  with probability 1. ■

**Proposition 8.** *We define for short that*

$$\mathcal{E}^t = \Phi(\mu^t) + \frac{(1 - \gamma)^2}{|\mathcal{S}|} \|\mathbf{v}^t - \mathbf{v}^*\|^2$$

and

$$\mathcal{G}^t = \sum_{a \in \mathcal{A}} (\mu_a^t)^\top ((I - \gamma P_a)\mathbf{v}^* - \mathbf{r}_a) = \sum_{i \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{i,a}^t (\mathbf{v}^* - \gamma P_a \mathbf{v}^* - \mathbf{r}_a)_i.$$

Let  $\alpha = \frac{|\mathcal{S}|}{2(1 - \gamma)^2} \beta$ . The iterates generated by Algorithm 1 satisfy for all  $t$  with probability 1 that

$$\mathbf{E} [\mathcal{E}^{t+1} \mid \mathcal{F}_t] \leq \mathcal{E}^t - \beta \mathcal{G}^t + \beta^2 \frac{2|\mathcal{S}|(|\mathcal{A}| + 1)}{(1 - \gamma)^2}. \quad (14)$$

*Proof.* Let  $\alpha = \frac{|\mathcal{S}|}{2(1 - \gamma)^2} \beta$ . We multiply (13) with  $\frac{(1 - \gamma)^2}{|\mathcal{S}|}$  and takes its sum with (12), obtaining

$$\begin{aligned} \mathbf{E} [\mathcal{E}^{t+1} \mid \mathcal{F}_t] &\leq \mathcal{E}^t + \beta^2 \frac{2|\mathcal{S}|(|\mathcal{A}| + |\mathcal{S}|)}{(1 - \gamma)^2} \\ &\quad + \beta \left( \sum_{a \in \mathcal{A}} (\mu_a^t - \mu_a^*)^\top ((\gamma P_a - I)\mathbf{v}^t + \mathbf{r}_a) + (\mathbf{v}^t - \mathbf{v}^*)^\top \left( \sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^t - (1 - \gamma)\mathbf{q} \right) \right). \end{aligned}$$

We have

$$\begin{aligned}
& \sum_{a \in \mathcal{A}} (\mu_a^t - \mu_a^*)^\top ((\gamma P_a - I) \mathbf{v}^t + \mathbf{r}_a) + (\mathbf{v}^t - \mathbf{v}^*)^\top \left( \sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^t - (1 - \gamma) \mathbf{q} \right) \\
&= \sum_{a \in \mathcal{A}} (\mu_a^t - \mu_a^*)^\top ((\gamma P_a - I) \mathbf{v}^t + \mathbf{r}_a) + (\mathbf{v}^t - \mathbf{v}^*)^\top \sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top (\mu_a^t - \mu_a^*) \\
&= \sum_{a \in \mathcal{A}} (\mu_a^t - \mu_a^*)^\top ((\gamma P_a - I) \mathbf{v}^* + \mathbf{r}_a) \quad (\text{by the dual feasibility of } \mu^*) \\
&= \sum_{a \in \mathcal{A}} (\mu_a^t)^\top ((\gamma P_a - I) \mathbf{v}^* + \mathbf{r}_a) \quad (\text{by the linear complementarity condition for } \mathbf{v}^*, \mu^*)
\end{aligned}$$

where the second equality uses the dual feasibility of  $\mu^*$  that  $\sum_{a \in \mathcal{A}} (I - \gamma P_a)^\top \mu_a^* = (1 - \gamma) \mathbf{q}$  and the fourth equality uses the complementary condition  $\mu_{a,i}^* ((\gamma P_a - I) \mathbf{v}^* + \mathbf{r}_a)_i = 0$  for all  $i \in \mathcal{S}, a \in \mathcal{A}$ . Combining the preceding relations, we obtain (14).  $\blacksquare$

## 7.2 Proof of Proposition 2

*Proof.* We claim that  $\mathcal{E}^1 \leq \log(|\mathcal{S}||\mathcal{A}|) + 1$ . To see this, we note that  $\lambda^1$  and  $\pi_i^1$ 's are uniform distributions (according to Step 3 of Algorithm 1). Therefore we have  $D_{KL}(\lambda^* || \lambda^1) \leq \log(SA)$  and  $D_{KL}(\pi_i^* || \pi_i^1) \leq \log(S)$  for  $i$ , and  $\|\mathbf{v}^t - \mathbf{v}^*\|^2 \leq \frac{S}{(1-\gamma)^2}$  for all  $t$ . Then we have  $\mathcal{E}^1 \leq (1 - \theta) D_{KL}(\lambda^* || \lambda^1) + \theta \sum_{i \in \mathcal{S}} q_i D_{KL}(\pi_i^* || \pi_i^1) + \frac{(1-\gamma)^2}{|\mathcal{S}|} \|\mathbf{v}^1 - \mathbf{v}^*\|^2 \leq \log(|\mathcal{S}||\mathcal{A}|) + 1$ .

We rearrange the terms of (14) and obtain

$$\mathcal{G}^t \leq \frac{1}{\beta} (\mathcal{E}^t - \mathbf{E} [\mathcal{E}^{t+1} | \mathcal{F}_t]) + \frac{2\beta|\mathcal{S}|(|\mathcal{A}| + 1)}{(1 - \gamma)^2}.$$

Summing over  $t = 1, \dots, T$  and taking average, we have

$$\begin{aligned}
\mathbf{E} \left[ \sum_{t=1}^T \mathcal{G}^t \right] &\leq \frac{1}{\beta} \sum_{t=1}^T (\mathbf{E} [\mathcal{E}^t] - \mathbf{E} [\mathcal{E}^{t+1}]) + \frac{2\beta|\mathcal{S}|(|\mathcal{A}| + 1)T}{(1 - \gamma)^2} \\
&= \frac{\mathbf{E} [\mathcal{E}^1] - \mathbf{E} [\mathcal{E}^T]}{\beta} + \frac{2\beta|\mathcal{S}|(|\mathcal{A}| + 1)T}{(1 - \gamma)^2} \\
&\leq \frac{1}{\beta} (\log(|\mathcal{S}||\mathcal{A}|) + 1) + \frac{2\beta|\mathcal{S}|(|\mathcal{A}| + 1)T}{(1 - \gamma)^2}.
\end{aligned}$$

where the inequality is based on the fact  $\mathcal{E}^1 \leq \log(|\mathcal{S}||\mathcal{A}|) + 1$  and  $\mathcal{E}^T \geq 0$ . Therefore by taking  $\beta = (1 - \gamma) \sqrt{\frac{\log(|\mathcal{S}||\mathcal{A}| + 1)}{2|\mathcal{S}||\mathcal{A}|T}}$ , we obtain  $\mathbf{E} \left[ \frac{1}{T} \sum_{t=1}^T \mathcal{G}^t \right] \leq \frac{\sqrt{2|\mathcal{S}|(|\mathcal{A}| + 1)(\log(|\mathcal{S}||\mathcal{A}| + 1))}}{(1 - \gamma)\sqrt{T}}$ .  $\blacksquare$

## References

- [1] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
- [2] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.
- [3] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific, Belmont, MA, 1995.
- [4] Dimitri P Bertsekas. *Abstract dynamic programming*. Athena Scientific, Belmont, MA, 2013.
- [5] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 1, pages 560–564. IEEE, 1995.
- [6] Vivek S. Borkar. *Stochastic approximation: a dynamical systems viewpoint*. Cambridge University Press, Cambridge, 2008.

- [7] Karl Bringmann and Konstantinos Panagiotou. Efficient sampling methods for discrete distributions. In *International Colloquium on Automata, Languages, and Programming*, pages 133–144. Springer, 2012.
- [8] Yichen Chen and Mengdi Wang. Stochastic primal-dual methods and sample complexity of reinforcement learning. *arXiv preprint arXiv:1612.02516*, 2016.
- [9] Yichen Chen and Mengdi Wang. Lower bound on the computational complexity of discounted markov decision problems. *arXiv preprint arXiv:1705.07312*, 2017.
- [10] Kenneth L Clarkson, Elad Hazan, and David P Woodruff. Sublinear optimization for machine learning. *Journal of the ACM (JACM)*, 59(5):23, 2012.
- [11] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 2016.
- [12] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.
- [13] Guy De Ghellinck. Les problemes de decisions sequentielles. *Cahiers du Centre dEtudes de Recherche Opérationnelle*, 2(2):161–179, 1960.
- [14] F d’Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, 1963.
- [15] Eugene A Feinberg and Jefferson Huang. The value iteration algorithm is not strongly polynomial for discounted dynamic programming. *Operations Research Letters*, 42(2):130–131, 2014.
- [16] Ronald A. Howard. *Dynamic programming and Markov processes*. The MIT press, Cambridge, MA, 1960.
- [17] Anatoli Juditsky, Arkadi Nemirovski, Claire Tauvel, et al. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.
- [18] Michael J Kearns and Satinder P Singh. Finite-sample convergence rates for q-learning and indirect algorithms. In *Advances in neural information processing systems*, pages 996–1002, 1999.
- [19] Harold J Kushner and George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003.
- [20] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in  $o(\text{vrnk})$  iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014.
- [21] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 230–249. IEEE, 2015.
- [22] Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 394–402. Morgan Kaufmann Publishers Inc., 1995.
- [23] Yishay Mansour and Satinder Singh. On the complexity of policy iteration. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 401–408. Morgan Kaufmann Publishers Inc., 1999.
- [24] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.
- [25] Arkadi Nemirovski and Reuven Y Rubinfeld. An efficient stochastic approximation algorithm for stochastic saddle point problems. In *Modeling Uncertainty*, pages 156–184. Springer, 2005.
- [26] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [27] Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. In *Advances in Neural Information Processing Systems*, pages 386–394, 2013.
- [28] Paul Tseng. Solving h-horizon, stationary markov decision problems in time proportional to  $\log(h)$ . *Operations Research Letters*, 9(5):287–297, 1990.
- [29] Mengdi Wang and Yichen Chen. An online primal-dual method for discounted Markov decision processes. In *IEEE Conference of Decisions and Control*, 2016.
- [30] Chak-Kuen Wong and Malcolm C. Easton. An efficient method for weighted sampling without replacement. *SIAM Journal on Computing*, 9(1):111–113, 1980.
- [31] Yinyu Ye. A new complexity result on solving the Markov decision problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.
- [32] 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.