Optimizing over a Restricted Policy Class in MDPs

Ershad Banijamali — University of Waterloo
Yasin Abbasi-Yadkori — Adobe
Mohammad Ghavamzadeh — Facebook AI Research
Nikos Vlassis — Netflix

Abstract

We address the problem of finding an optimal policy in a Markov decision process under a restricted policy class defined by the convex hull of a set of base policies. This problem is of great interest in applications in which a number of reasonably good (or safe) policies are already known and we are interested in optimizing in their convex hull. We first prove that solving this problem is NP-hard. We then propose an efficient algorithm that finds a policy whose performance is almost as good as that of the best convex combination of the base policies, under the assumption that the occupancy measures of the base policies have a large overlap. The running time of the proposed algorithm is linear in the number of states and polynomial in the number of base policies. A distinct advantage of the proposed algorithm is that, apart from the computation of the occupancy measures of the base policies, it does not need to interact with the environment during the optimization process. This is especially important (i) in problems that due to concerns such as safety, we are restricted in interacting with the environment only through the (safe) base policies, and (ii) in complex systems where estimating the value of a policy can be a time consuming process.

1 Introduction

In many control and reinforcement learning problems, a number of reasonable (safe) base policies are known. For example, these policies might be provided by an expert. A natural question is whether a combination of these base policies can provide an improvement over a default policy. This problem is especially important when the number of states is large and the exact computation of the optimal policy is not feasible. One way to formulate the problem is to define a policy space that includes all mixtures of the base policies. A policy in this class samples a base policy at each state and acts according to that, as opposed to sampling a base policy at the initial state and running it until the end.

A popular method to optimize a parameterized policy is policy gradient, which typically employs a variant of the gradient descent/ascent method (Williams, 1992; Sutton et al., 2000; Baxter and Bartlett, 2001; Peters et al., 2005; Bhatnagar et al., 2009). Although in some applications the quality of the solution is high, the policy gradient methods often converge to some local minima as the problem is highly non-convex. Further, computing a gradient estimate can be an expensive operation. For example, the finite difference method requires running a number of policies in each iteration and estimating the value of a policy in a complicated system might require a long running time.

In this paper, we show a number of results on the problem of policy optimization in a restricted class of mixture policies. First, we show that solving the optimization problem is NP-hard. The hardness result is obtained by a reduction from the INDEPENDENT-SET problem for graphs and an application of the Motzkin-Straus theorem for optimizing quadratic forms over the simplex (Motzkin and Straus, 1965). This result is somewhat surprising, since the same problem is known to be easy (in the complexity class P), if the space of base policies includes all MDP policies (an exponentially large space!) (Papadimitriou and Tsitsiklis, 1987). The critical difference is that in the unconstrained case an optimal MDP policy is known to be deterministic, in which case linear programming or policy iteration are known to run in polynomial time (Ye, 2005), whereas in the restricted case an optimal policy may need to randomize.

Although this hardness result is somewhat disappointing, we show that an approximately optimal solution can be found in a reasonable time when the occu-

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

pancy measures of the base policies have large overlap. We obtain this result by formulating the problem in the *dual space*. More specifically, instead of searching in the space of mixture policies, we construct a new search space that consists of linear combinations of the occupancy measures of the base policies. Each such linear combination is not an occupancy measure itself, but it defines a policy through a standard normalization. Importantly, this new policy space also contains the base policies and so finding a near optimal policy in this class also provides a policy improvement w.r.t. the initial base policies. The objective function in the dual space is still highly non-convex, but we can exploit the convex relaxation proposed by Abbasi-Yadkori et al. (2014) to have an efficient algorithm with performance guarantees. Abbasi-Yadkori et al. (2014) study the linear programming approach to dynamic programming in the dual space (space of occupancy measures) and propose a penalty method that minimizes the sum of the linear objective and a number of constraint violations.

To demonstrate the idea, consider the problem of controlling the service rate of a queue where jobs arrive at a certain rate and the cost is the sum of the queue length and the chosen service rate. Consider two policies, one that selects low and one that selects high service rates. The space of the mixture of these two policies is rich and is likely to contain a policy with low total cost. We can generate a wide range of service rates as a convex combination of these two base policies. Now let us consider the dual space. The occupancy measures of the first and second policies are concentrated at large and small queue lengths, respectively. It can be shown that the linear combination of occupancy measures can generate a limited set of policies that are either similar to the first policy or the second one. In short, although the space of mixture policies is a rich space (and hence the optimization is NP-hard in that space), the space of dual policies can be more limited. More crucially, if the base policies have some similarities so that their occupancy measures overlap, then we can generate non-trivial policies in the dual space that can be competitive with the mixture policies in the primal space. We show that this is indeed the case in our experiments in Section 5.

Let us compare our algorithm with the traditional policy gradient in the space of mixture of policies. Although the space of the mixture of the base policies is rich and is likely to contain a policy with lower total cost than any policy that is a mixture of the occupancy measures of the base policies, our approach has several advantages. First, policy gradient is more computationally demanding. Gradient descent needs to perform several rollouts in each round to estimate the gradient direction. In a complicated system, the mixing times can be large, and thus, we might need to run a policy for a very long time before we can reliably estimate its gradient. In contrast, and as we will show, apart from the initial rollouts to estimate the occupancy measures of the base policies, the proposed method does not need to interact with the environment when optimizing in the dual space. Second, our approach is *safe*; during the optimization phase, we only need to execute the base policies that are assumed to be safe. In contrast, policy gradient in the primal space evaluates many policies in the intermediate steps that some of them may not be safe to be executed. Furthermore, our method enjoys stronger theoretical guarantees than the policy gradient method.

1.1 Notation

Let $M_{i,:}$ and $M_{:,j}$ denote ith row and jth column of matrix M, respectively. We denote by I an identity matrix, and by 1_n and 0_n, n-dimensional all one and all zero vectors, respectively. We use 0_{mm} to denote the all-zero $m \times m$ matrix and e_i to denote the unit m-vector (1 at position i and 0 elsewhere). We also use $\mathbf{1}\{.\}$ to denote the indicator function, and $\wedge$ and $\vee$ to denote the minimum and maximum. We define $[v]_+ = v \vee 0$ and $[v]_- = v \wedge 0$. For vectors v and w, $v \le w$ means element-wise inequality, i.e., $v_i \le w_i$, for all i. We use $\Delta_{\mathcal{S}}$ to denote the space of probability distributions defined on the set $\mathcal{S}$. For positive integer m, we use $[m]$ to denote the set $\{1, 2, \ldots, m\}$.

2 Preliminaries

In this paper, we study reinforcement learning (RL) problems in which the interaction between the agent and environment has been modeled as a discrete[1] discounted MDP. A discrete MDP is a tuple $\langle \mathcal{X}, \mathcal{A}, c, P, \alpha, \gamma \rangle$, where $\mathcal{X}$ and $\mathcal{A}$ are the sets of S states and A actions, respectively; $c : \mathcal{X} \to [0, 1]$ is the cost function; $P : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}}$ is the transition probability distribution that maps each state-action pair to a distribution over states $\Delta_{\mathcal{X}}$; $\alpha \in \Delta_{\mathcal{X}}$ is the initial state distribution; and $\gamma \in (0, 1)$ is the discount factor. We are primarily interested in the case where the number of states is large. We assume that the cost does not depend on the action, although a number of our results can be extended to action-dependent costs. With an abuse of notation, we also use c to denote a (SA)-dimensional vector such that $c(x, a) = c(x)$ for any $x \in \mathcal{X}$ and $a \in \mathcal{A}$. The restriction that costs are bounded in $[0, 1]$ interval is to simplify the notation and can be relaxed to other bounded ranges. Since we

[1] In a remark at the end of Section 4, we will discuss extension to continuous-state problems.

consider discrete MDPs, all the MDP-related quantities can be written in vector and matrix forms.

We also need to specify the rule according to which the agent selects actions at each state. We assume that this rule does not depend explicitly on time. A stationary policy $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$ is a probability distribution over actions, conditioned on the current state. The MDP controlled by a policy π induces a Markov chain with the transition probability P_π and cost function $c_\pi = c$. We denote by J_π the value function of policy π, i.e., the expected sum of discounted costs of following policy π, and by $\nu_\pi \in \Delta_{\mathcal{X}}$ and $\mu_\pi \in \Delta_{\mathcal{X}\times\mathcal{A}}$ the state and state-action occupancy measures under policy π and w.r.t. the starting distribution α, i.e.,

$$\nu_\pi(x) = (1-\gamma) \sum_{x' \in \mathcal{X}} \alpha(x') \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(X_t = x | X_0 = x'),$$

and $\mu_\pi(x,a) = \nu_\pi(x)\,\pi(a|x)$. Note that when $x_0 \sim \alpha$, we may write $\mathbb{P}(X_t|X_0) = \alpha^\top P_\pi^t$, and thus, $\nu_\pi^\top = (1-\gamma)\alpha^\top (I - \gamma P_\pi)^{-1}$, where I is the identity matrix. Given a policy class Π, the goal is to find a policy $\pi \in \Pi$ that minimizes $J(\pi) = \alpha^\top J_\pi$. It is easy to show that $J(\pi) = \nu_\pi^\top c_\pi = \mu_\pi^\top c$.

In this work, we are interested in the policy optimization problem where the policy class Π is defined as a mixture of m base policies $\{\pi_1, \dots, \pi_m\}$, i.e., $\Pi = \{\pi_w : \pi_w = \sum_{i=1}^m w_i \pi_i,\ w \in \Delta_{[m]}\}$. We assume that the transition dynamics of the base policies are known. This assumption is only used for estimating the stationary distributions and average costs of the base policies. The extension to the unknown dynamics and costs (the RL problem), and the extension to the continuous state problems are discussed in a remark at the end of Section 4. We call the policy space Π the primal space. Executing a policy $\pi_w \in \Pi$ amounts to sampling one of the m base policies from the distribution $w \in \Delta_{[m]}$ at each time step, and then acting according to this policy. Finding the best policy in Π requires solving the following optimization problem

$$\min_{w \in \Delta_{[m]}} J(\pi_w) = \min_{w \in \Delta_{[m]}} \alpha^\top J_{\pi_w} . \qquad (1)$$

We denote by w^* the solution to (1) and use $\pi_* = \pi_{w^*}$. For a positive constant R, we define the dual space of Π as the space of linear combinations of the state-action occupancies of the base policies, i.e.,

$$\Xi = \{\xi_\theta : \xi_\theta = \sum_{i=1}^m \theta_i \mu_{\pi_i},\ \theta \in \Theta\}, \qquad (2)$$

where $\Theta = \{\theta \in \mathbb{R}^m : \sum_{i=1}^m \theta_i = 1,\ \|\theta\|_2 \le R\}$. Here, R is the radius of the parameter space and restricts the size of the policy class. Note that given the definition of Θ, each $\xi_\theta \in \Xi$ is not necessarily a state-action occupancy measure. However, each $\xi_\theta \in \Xi$ corresponds to a policy π_θ, defined as

$$\pi_\theta(a|x) = \frac{[\xi_\theta(x,a)]_+}{\sum_{a' \in \mathcal{A}} [\xi_\theta(x,a')]_+} . \qquad (3)$$

If $\xi_\theta(x,a) \le 0$ for all $a \in \mathcal{A}$, we let $\pi_\theta(.|x)$ be the uniform distribution. If ξ_θ is the state-action occupancy measure of a policy π, it can be shown that $\pi_\theta = \pi$. This implies that the base policies are in the dual space. We denote by μ_{π_θ} the state-action occupancy of policy π_θ. The policy optimization problem in the dual space is defined as

$$\min_{\theta \in \Theta} J(\pi_\theta) = \min_{\theta \in \Theta} \alpha^\top J_{\pi_\theta} = \min_{\theta \in \Theta} \mu_{\pi_\theta}^\top c , \qquad (4)$$

where π_θ is computed from θ using Equation 3.

3 Hardness Result

In this section, we show that the policy optimization problem in the primal space (Eq. 1) is NP-hard. At a high level, the proof involves designing a special MDP and m base policies such that solving Eq. 1 in polynomial time would imply P=NP.

Theorem 3.1. *Given a discounted MDP, a set of policies, and a target cost r, it is NP-hard to decide if there exists a mixture of these policies that has expected cost at most r.*

Proof. We reduce from the INDEPENDENT-SET problem. This problem asks, for a given (undirected and with no self-loops) graph $\mathcal{G}$ with vertex set V, and a positive integer $j \le |V|$, whether $\mathcal{G}$ contains an independent set $V' \subseteq V$ having $|V'| \ge j$. This problem is NP-complete (Garey and Johnson, 1979).

Let G be the $m \times m$ (symmetric, 0-1) adjacency matrix of an input graph $\mathcal{G}$, and let $A = I + G$, where I is the identity matrix. The reduction constructs a deterministic MDP with $m+3$ states and $m+3$ actions, and m deterministic policies π_i that induce corresponding chains P_{π_i}, for $i = 1, \dots, m$, where each $(m+3)\times(m+3)$ matrix P_{π_i} reads

$$P_{\pi_i} = \begin{bmatrix} 0 & e_i^\top & 0 & 0 \\ 0_m & 0_{mm} & A_{:,i} & 1_m - A_{:,i} \\ 0 & 0_m^\top & 0 & 1 \\ 0 & 0_m^\top & 0 & 1 \end{bmatrix} . \qquad (5)$$

A mixture π_w of the base policies, with weights w, induces the chain

$$P_{\pi_w} = \sum_{i=1}^m w_i P_{\pi_i} = \begin{bmatrix} 0 & w^\top & 0 & 0 \\ 0_m & 0_{mm} & Aw & 1_m - Aw \\ 0 & 0_m^\top & 0 & 1 \\ 0 & 0_m^\top & 0 & 1 \end{bmatrix} . \qquad (6)$$

It is easy to see that, for each $k \geq 3$, the kth power of P_{π_w} is equal to $P^k_{\pi_w} = [\,0_{(m+3)(m+2)},\, 1_{m+3}\,]$ (all zeros except for the last column that is all ones), while its square $P^2_{\pi_w}$ reveals the quadratic form $w^\top A w$ in position $(1, m+2)$. Hence, for initial state distribution $\alpha = [1, 0, \ldots, 0]^\top$ (all mass on the first state), state-only-dependent cost vector $c = [0, \ldots, 0, 1, 0]^\top$ (all zeros except for 1 in the $(m+2)$'th state), and discount factor $\gamma < 1$, the expected discounted cost of π_w is

$$\begin{aligned} J(\pi_w) &= (1-\gamma)\alpha^\top (I - \gamma P_{\pi_w})^{-1} c \\ &= (1-\gamma)\alpha^\top (I + \gamma P_{\pi_w} + \gamma^2 P^2_{\pi_w} + \ldots) c \\ &= (1-\gamma)\gamma^2\, w^\top A w\,. \end{aligned} \tag{7}$$

For any graph $\mathcal{G}$ with m vertices and adjacency matrix G, the following holds (Motzkin and Straus, 1965):

$$\frac{1}{\omega(\mathcal{G})} = \min_{y \in \Delta_{[m]}} y^\top (I + G) y\,, \tag{8}$$

where $\omega(\mathcal{G})$ is the size of the maximum independent set of $\mathcal{G}$. Let the target cost be $r = \frac{(1-\gamma)\gamma^2}{j}$, where j is the target integer of the INDEPENDENT-SET instance. Then the decision question $J(\pi_w) \leq r$ is equivalent to $w^\top (G + I) w \leq \frac{1}{j}$, where we used (7). Hence, it follows from (8) that the existence of a vector w that satisfies $J(\pi_w) \leq r$ would imply $\omega(\mathcal{G}) \geq j$, and thus, $|V'| \geq j$ for some independent set $V' \subseteq V$. This establishes that, deciding the MDP policy optimization problem in polynomial time would also decide the INDEPENDENT-SET problem in polynomial time, implying P=NP. □

Remark 1: The same technique can be used to show NP-hardness of the problem under an average cost criterion. This only requires changing the last two rows of the matrices P_{π_i}, by having the 1's in the first column instead of the last column. In that case, if $\nu_{\pi_w} = [\,x,\, v^\top,\, y,\, z\,]^\top$ is the stationary distribution of P_{π_w}, where v is an m-vector and x, y, z scalars, we can algebraically solve the eigensystem $\nu^\top_{\pi_w} = \nu^\top_{\pi_w} P_{\pi_w}$ (by elementary manipulations), to get $v = w$ and $y = w^\top A w$. Hence, for a cost vector $c = [0, \ldots, 0, 1, 0]^\top$, the average cost of the MDP under w is $w^\top A w$, and by choosing target cost $r = \frac{1}{j}$ the Motzkin-Straus argument applies as above.

Remark 2: Optimizing over a restricted policy class essentially converts the MDP to a POMDP, for which related complexity results are known (Papadimitriou and Tsitsiklis, 1987; Mundhenk et al., 2000; Vlassis et al., 2012; Kumar and Zilberstein, 2015).

4 Reduction to Convex Optimization

In this section, we first propose an algorithm to solve the policy optimization problem in the dual space (Eq. 4) and then prove a bound on the performance of the policy returned by this algorithm compared to the solution of the policy optimization problem in the primal space (Eq. 1).

```
Input: base policies $\{\pi_i\}_{i=1}^m$,; dual parameter space $\Theta$; number of rounds $T$, learning rates $\{\beta_t\}_{t=1}^T$, penalty parameter $H$
Compute occupancy measures of the base policies, i.e., $\{\mu_{\pi_i}\}_{i=1}^m$
Initialize $\theta_1 = 0$
for $t := 1, 2, \ldots, T$ do
    Sample $i \in [m]$ and sample $(x_t, a_t) \sim \mu_{\pi_i}$
    Compute subgradient estimate $g_t(\theta_t)$    (Eq. 10)
    Update $\theta_{t+1} = \Pi_\Theta(\theta_t - \beta_t g_t)$    ($\Pi_\Theta$ is the Euclidean projection onto $\Theta$)
end for
Compute $\widehat{\theta} = \frac{1}{T}\sum_{t=1}^T \theta_t$
Return policy $\pi_{\widehat{\theta}}$    (Eq. 3)
```

Figure 1: Stochastic Subgradient Method for MDPs.

Figure 1 contains the pseudocode of our proposed algorithm. The algorithm takes the m base policies $\{\pi_i\}_{i=1}^m$ and the dual parameter space Θ as input. It first computes the state-action occupancy measures of the base policies, i.e., $\{\mu_{\pi_i}\}_{i=1}^m$. This is done using the recent results by Cohen et al. (2017) that show it is possible to compute the stationary distribution (occupancy measure) of a Markov chain in time *linear* to the size of the state space.[2] This guarantees that we can compute the state-action occupancy measures of the base policies efficiently. Alternatively, when the size of the state space is very large, these occupancy measures can be estimated by roll-outs.

Motivated by the approach of Abbasi-Yadkori et al. (2014), we propose minimizing a convex surrogate function

$$\mathcal{L}(\theta) = c^\top \xi_\theta + H \sum_{(x,a) \in \mathcal{X} \times \mathcal{A}} |[\xi_\theta(x,a)]_-|\,, \tag{9}$$

where ξ_θ is a linear combination of the occupancy measures and H is a positive parameter that penalizes negative values in ξ_θ. At each time step t, our algorithm first computes an estimate of the sub-gradient $\nabla \mathcal{L}(\theta_t)$ and then feeds it to the projected sub-gradient method to update the policy parameter θ_t. In order to compute an estimate of the sub-gradient $\nabla \mathcal{L}(\theta_t)$, we first sample a state-action pair (x_t, a_t) from $(1/m)\sum_{i=1}^m \mu_{\pi_i}$,

[2]These results can also be applied to the discounted case.

and then compute the function

$$g_t(\theta_t) = c^\top M - H \frac{M(x_t,a_t)\,\mathbf{1}\left\{\xi_{\theta_t}(x_t,a_t) < 0\right\}}{(1/m)\sum_{i=1}^m \mu_{\pi_i}(x_t,a_t)}, \quad (10)$$

where M is a $SA \times m$ matrix, whose j'th column is μ_{π_j}, and $M(x,a)$ is the row of M corresponding to the state-action pair (x,a). Notice that elements of $c^\top M$ are simply average costs of the base policies. To sample (x_t,a_t) from $(1/m)\sum_{i=1}^m \mu_{\pi_i}$, we first select a number $i \in [m]$ uniformly at random and then sample a state-action pair (x_t,a_t) from the state-action occupancy measure of the i'th base policy, i.e., $(x_t,a_t) \sim \mu_{\pi_i}$. If μ_{π_i} is approximated by the historical data generated under policy π_i, then this last sampling amounts to sampling one datapoint uniformly at random from historical observations under policy π_i.

We now show that $g_t(\theta)$ in (10) is an unbiased estimate of $\nabla\mathcal{L}(\theta)$:

$$\begin{aligned}
&\mathbb{E}\left[\frac{M(x_t,a_t)\mathbf{1}\left\{\xi_\theta(x_t,a_t) < 0\right\}}{(1/m)\sum_{i=1}^m \mu_{\pi_i}(x_t,a_t)}\right] \\
&= \sum_{(x,a)\in\mathcal{X}\times\mathcal{A}} \frac{M(x,a)\mathbb{P}\left(x_t = x, a_t = a\right)}{(1/m)\sum_{i=1}^m \mu_{\pi_i}(x,a)} \mathbf{1}\left\{\xi_\theta(x,a) < 0\right\} \\
&= \sum_{(x,a)\in\mathcal{X}\times\mathcal{A}} M(x,a)\mathbf{1}\left\{\xi_\theta(x,a) < 0\right\}.
\end{aligned}$$

After T rounds, the algorithm returns the average of the computed policy parameters $\{\theta_t\}_{t=1}^T$, i.e., $\widehat{\theta} = (1/T)\sum_{t=1}^T \theta_t$. The parameter $\widehat{\theta}$ defines an element $\xi_{\widehat{\theta}}$ of the dual space Ξ (Eq. 2), which in turn defines a policy $\pi_{\widehat{\theta}}$ using Eq. 3. We denote by $\mu_{\pi_{\widehat{\theta}}}$ the state-action occupancy measure of this policy.

The approach of Abbasi-Yadkori et al. (2014) involves minimizing an objective function with an additional term that measures the distance of ξ_θ from the space of occupancy measures. The transition matrix appears in this extra term. In our case, ξ_θ is a linear combination of occupancy measures and hence this extra term is zero, and the transition dynamics does not appear in the objective (9). This results in a simpler optimization problem and improved performance bounds. More importantly, we can use the resulting algorithm in the reinforcement learning setting as no *backward simulator* is needed.

Our main result shows that the policy returned by our algorithm, $\pi_{\widehat{\theta}}$, is near-optimal as long as the occupancy measures of the base policies have large overlap.

Theorem 4.1. *Let $\delta > 0$ be the probability of error, $H = 1/\eta$ be the constraints multiplier in the subgradient estimate* (10)*, and $T = O\big(\frac{R^2}{\eta^2}\log(1/\delta)\big)$ be the number of rounds. Let $\widehat{\theta}$ be the output of the stochastic sub-gradient method after T rounds, the learning rate be $\beta_t = S/(G'\sqrt{T})$, where $G' = \sqrt{m} + Hm$, and $U(\theta) = \sum_{(x,a)\in\mathcal{X}\times\mathcal{A}} |[\xi_\theta(x,a)]_-|$. Then with probability at least $1-\delta$, we have the following bound on the performance of the policy $\pi_{\widehat{\theta}}$ returned by the algorithm in Figure 1:*

$$\begin{aligned}
\overbrace{\alpha^\top J_{\pi_{\widehat{\theta}}}}^{J(\pi_{\widehat{\theta}})} &\le \min_{w\in\Delta_{[m]}} \overbrace{\alpha^\top J_{\pi_w}}^{J(\pi_w)} + O(\eta) \\
&+ \min_{\theta\in\Theta}\left(\sum_i \theta_i(\nu_{\pi_i} - \nu_{\pi_*})^\top c + \left(\frac{1}{\eta} + \frac{9}{1-\gamma}\right)U(\theta)\right).
\end{aligned}$$

Moreover, the computational complexity of the algorithm is $O\big(poly(m)A\big)$ and the constants hidden in $O(\eta)$ are polynomials in R, m, and $1/(1-\gamma)$.

Before proving Theorem 4.1, we show the improvement using a simple example.

Example Let λ be such that for any $i,j \in [m]$ and any $(x,a) \in \mathcal{X}\times\mathcal{A}$,

$$\frac{\lambda}{1+\lambda}\mu_{\pi_j}(x,a) \le \mu_{\pi_i}(x,a) \le \frac{1+\lambda}{\lambda}\mu_{\pi_j}(x,a)\,. \quad (11)$$

A large value of λ indicates large overlap of occupancy measures. Let $\varepsilon = \|\mu_{\pi_1} - \mu_{\pi_*}\|_1$. Parameters λ and ε provide ℓ^∞ and ℓ^1 error bounds. We can obtain an easy but non-trivial bound under the condition (11) as follows. Let π_1 and π_m be the policies with the smallest and largest values. Choose $\theta_1 = 1+\lambda$, $\theta_m = -\lambda$, and all other elements are zero. Then

$$\begin{aligned}
U(\theta) &= \sum_{(x,a)} |[\xi_\theta(x,a)]_-| \\
&= \sum_{(x,a)} |[(1+\lambda)\mu_{\pi_1}(x,a) - \lambda\mu_{\pi_m}\ .(x,a)]_-|\ .
\end{aligned}$$

For each term to be negative, we must have $\mu_{\pi_1}(x,a) \le \frac{\lambda}{1+\lambda}\mu_{\pi_m}(x,a)$, which contradicts our assumption, thus $U(\theta) = 0$. Let $\mathcal{E} = \sum_i \theta_i(\nu_{\pi_i} - \nu_{\pi_*})^\top c$. We get that

$$\begin{aligned}
\mathcal{E} &= (1+\lambda)(\mu_{\pi_1} - \mu_{\pi_*})^\top c - \lambda(\mu_{\pi_m} - \mu_{\pi_*})^\top c \\
&= (\mu_{\pi_1} - \mu_{\pi_*})^\top c - \lambda(\alpha^\top J_{\pi_m} - \alpha^\top J_{\pi_1})\,.
\end{aligned}$$

Thus,

$$\alpha^\top J_{\pi_{\widehat{\theta}}} \le \alpha^\top J_{\pi_*} + O(\eta) + \varepsilon - \lambda\alpha^\top(J_{\pi_m} - J_{\pi_1})\,. \quad (12)$$

Let's compare the above result to the simple bound

$$\alpha^\top J_{\pi_1} \le \alpha^\top J_{\pi_*} + \varepsilon\,. \quad (13)$$

The term $O(\eta)$ is due to the error of gradient descent in a convex problem and can be made arbitrarily small by increasing the number of iterations. Thus, the term $\lambda\alpha^\top(J_{\pi_m} - J_{\pi_1})$ in (12) shows the amount of improvement compared to the bound in (13). This term is positive since π_m has a larger value than π_1.

Next, we state a number of results that will be used to prove Theorem 4.1. Theorem. 1 of Abbasi-Yadkori et al. (2014) adapted to the dual space Ξ implies that $\pi_{\widehat{\theta}}$ is near-optimal in the dual space Ξ.

Theorem 4.2 (Abbasi-Yadkori et al. 2014)**.** *Let δ, T, H, and $\widehat{\theta}$ be defined as in Theorem 4.1. Then with probability at least $1-\delta$, we have*

$$J(\widehat{\theta}) = \alpha^\top J_{\pi_{\widehat{\theta}}} \leq \min_{\theta\in\Theta}\Bigg(\alpha^\top J_{\pi_\theta} + \frac{6\sqrt{m}CR\eta}{1-\gamma} + O(\eta) + \left(\frac{1}{\eta}+\frac{6}{1-\gamma}\right)\sum_{(x,a)}|[\xi_\theta(x,a)]_-|\Bigg),$$

where the constants hidden in $O(\eta)$ are polynomials in R, m, and C.

Our main technical tool is the following lemma.

Lemma 4.3. *Let $\varepsilon = \max_{i,j\in[m]}\left\|\nu_{\pi_i}-\nu_{\pi_j}\right\|_1$. Then, for any $i\in[m]$ and any policy π_w in the primal space Π, we have $\|\nu_{\pi_i}-\nu_{\pi_w}\|_1 \leq \frac{\varepsilon(1+\gamma)}{1-\gamma}$.*

Proof. For any $i,j\in[m]$, there exists a vector $v_{i,j}$ with $\|v_{i,j}\|_1\leq\varepsilon$, such that $\nu_{\pi_i}-\nu_{\pi_j}=v_{i,j}$. We may rewrite this equation as $\alpha^\top(I-\gamma P_{\pi_i})^{-1} = \alpha^\top(I-\gamma P_{\pi_j})^{-1} + v_{i,j}^\top$, and further as

$$\alpha^\top(I-\gamma P_{\pi_i})^{-1}(I-\gamma P_{\pi_j}) = \alpha^\top + v_{i,j}^\top(I-\gamma P_{\pi_j})\ . \quad (14)$$

Now for a policy $\pi_w\in\Pi$ corresponding to the weight vector $w\in\Delta_{[m]}$, we may write

$$\begin{aligned}\alpha^\top(I-\gamma P_{\pi_i})^{-1}\Big(I-\gamma\sum_{k=1}^m w_k P_{\pi_k}\Big) &= \sum_{k=1}^m w_k\alpha^\top(I-\gamma P_{\pi_i})^{-1}(I-\gamma P_{\pi_k})\\ &\overset{(a)}{=} \alpha^\top\sum_{k=1}^m w_k + \sum_{k=1}^m w_k v_{i,k}^\top(I-\gamma P_{\pi_k})\\ &\overset{(b)}{=} \alpha^\top + \sum_{k=1}^m w_k v_{i,k}^\top(I-\gamma P_{\pi_k})\ , \quad (15)\end{aligned}$$

where **(a)** is from Eq. 14 and **(b)** is because $w\in\Delta_{[m]}$, and thus, $\sum_{k=1}^m w_k = 1$. From Eq. 15, we have

$$\alpha^\top(I-\gamma P_{\pi_i})^{-1} = \alpha^\top\Big(I-\gamma\sum_{k=1}^m w_k P_{\pi_k}\Big)^{-1} + \sum_{k=1}^m w_k v_{i,k}^\top(I-\gamma P_{\pi_k})\Big(I-\gamma\sum_{l=1}^m w_l P_{\pi_l}\Big)^{-1}. \quad (16)$$

Since $P_{\pi_w} = \sum_{k=1}^m w_k P_{\pi_k}$, we may rewrite Eq. 16 as

$$\begin{aligned}\nu_{\pi_i}-\nu_{\pi_w} &= \alpha^\top(I-\gamma P_{\pi_i})^{-1} - \alpha^\top(I-\gamma P_{\pi_w})^{-1}\\ &= \sum_{k=1}^m w_k v_{i,k}^\top(I-\gamma P_{\pi_k})\Big(I-\gamma\sum_{l=1}^m w_l P_{\pi_l}\Big)^{-1}.\end{aligned}$$

Let

$$\varepsilon' = \max_{i\in[m]}\left\|\sum_{k=1}^m w_k v_{i,k}^\top(I-\gamma P_{\pi_k})\Big(I-\gamma\sum_{l=1}^m w_l P_{\pi_l}\Big)^{-1}\right\|_1,$$

$z_k = (I-\gamma P_{\pi_k})^\top v_{i,k}$, $Q=\sum_{l=1}^m w_l P_{\pi_l}$, and $M^{-1} = I-\gamma Q$. We may now write

$$\begin{aligned}\|\nu_{\pi_i}-\nu_{\pi_w}\|_1 \leq \varepsilon' &\leq \left\|M^\top\sum_{k=1}^m w_k z_k\right\|_1\\ &\leq \left\|M^\top\right\|_1\left\|\sum_{k=1}^m w_k z_k\right\|_1\\ &\leq \underbrace{\left\|I+\gamma Q^\top+\gamma^2 Q^{2\top}+\dots\right\|_1}_{\leq 1/(1-\gamma)}\\ &\quad\times\sum_{k=1}^m w_k\overbrace{\left\|(I-\gamma P_{\pi_k})^\top\right\|_1}^{\leq(1+\gamma)}\overbrace{\|v_{i,k}\|_1}^{\leq\varepsilon}\\ &\leq \frac{\varepsilon(1+\gamma)}{1-\gamma}\ .\end{aligned}$$

This concludes the proof. ☐

Theorem 4.1 follows immediately from the above two results.

Proof of Theorem 4.1. From Lemma 2 of Abbasi-Yadkori et al. (2014), for any $\theta\in\Theta$, we have $\|\xi_\theta-\mu_{\pi_\theta}\|_1 \leq \frac{3U(\theta)}{1-\gamma}$. From Lemma 4.3, we have $\|\nu_{\pi_i}-\nu_{\pi_*}\|_1 = O(\varepsilon)$, for any $i\in[m]$ and $\pi^*=\pi_{w^*}$. Thus, for any $\theta\in\Theta$, we may write

$$\begin{aligned}\alpha^\top J_{\pi_\theta} &= \alpha^\top J_{\pi_*} + \mu_{\pi_\theta}^\top c - \mu_{\pi_*}^\top c\\ &= \alpha^\top J_{\pi_*} + \xi_\theta^\top c - \mu_{\pi_*}^\top c + \mu_{\pi_\theta}^\top c - \xi_\theta^\top c\\ &\leq \alpha^\top J_{\pi_*} + (\xi_\theta-\mu_{\pi_*})^\top c + \frac{3U(\theta)}{1-\gamma}\\ &= \alpha^\top J_{\pi_*} + \sum_i\theta_i(\mu_{\pi_i}-\mu_{\pi_*})^\top c + \frac{3U(\theta)}{1-\gamma}\\ &= \alpha^\top J_{\pi_*} + \sum_i\theta_i(\nu_{\pi_i}-\nu_{\pi_*})^\top c + \frac{3U(\theta)}{1-\gamma}\ .\end{aligned}$$

We used the fact that $c(x,a)=c(x)$ in the 5th step. This last inequality together with Theorem 4.2 gives us the desired result. ☐

The main theoretical contribution of this section is Lemma 4.3, which allows us to relate the primal and dual spaces. Although Theorem 4.2 is taken from Abbasi-Yadkori et al. (2014), this particular presentation using stationary distributions as features is new and provides an interesting application of the previous result.

Remark: When the reward function and transition probabilities are unknown (the RL problem), we can run the base policies to estimate their occupancy measures and average costs. These rollouts introduce an estimation error that will also appear in our performance bounds. Nevertheless, as we will show in our experiments, the rollouts provide an effective way to approximate the occupancy measures. Alternatively, we might have access to historical data from base policies that can be used for this estimation.

To simplify the presentation, we assumed that the state and action spaces are finite. However, the proposed algorithm can be easily extended to continuous problems; the occupancy measures can still be approximated using rollouts along with state aggregation or other function approximation techniques.

5 Experiments

We compare the results of combining policies in the policy space (w) and occupancy measure space (θ), where the base policies have overlapping occupancy measures. We consider three different queuing problems: *1-Queue*, *4-Queue*, and *8-Queue systems*. These problems have been studied before by de Farias and Van Roy (2003) with slightly different parameters. The results for *1-Queue* and *8-Queue systems* are shown next, and the results for *4-Queue* are shown in Appendix A.

In all the experiments, the proposed policy gradient in the dual space produces policies whose average costs are comparable with solutions of the policy gradient in the primal space. The notable difference is that the proposed policy gradient is significantly more resource efficient. In both the 4-queue and 8-queue problems, the time complexity of the proposed method is less than 0.001 of the time complexity of policy gradient in the primal space. In a RL setting, the proposed policy gradient would be similarly more sample efficient.

5.1 Queuing Problem: 1-Queue

Consider a queue of length L. We denote the state of the system at time t by x_t that shows the number of jobs in the queue. Jobs arrive at the queue with rate p. Action at each time, a_t, is chosen from the finite set $\{0.1625, 0.325, 0.4875, 0.65\}$ that represents the service rate or departure probability. The transition function of the system is then defined as: $x_{t+1} = x_t - 1$ w.p. a_t; $x_{t+1} = x_t + 1$ w.p. p; and $x_{t+1} = x_t$, otherwise. The system goes from state 0 to 1 w.p. p and stays in 0 w.p. $1 - p$. Also, transition from state L to $L - 1$ has probability $a(L)$ and the system stays in L w.p. $1 - a(L)$. The cost incurred by being in state x and taking action a is given by $c(x, a) = x^2 + 2500a^2$.

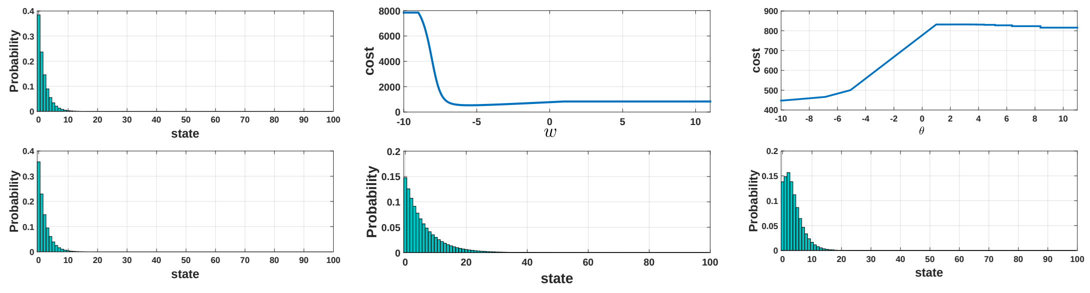
Consider a queue with $L = 99$ and two base policies that are independent of the states and have the following distributions over the set of actions: $\pi_1 = [0, 0, 0.50, 0.50]$ and $\pi_2 = [0, 0.1, 0.45, 0.45]$. Consider two scenarios: **1)** Mixing the original policies in the space of w, and **2)** Building a policy based on combining the corresponding occupancy measures (optimization in the space of θ).

Figure 2 shows the results of optimization in the two spaces. The costs associated with π_1 and π_2 are $J(\pi_1) = 831.91$ and $J(\pi_2) = 777.36$. Suppose $\pi_w = w\pi_1 + (1 - w)\pi_2$ is our mixture policy. Then for $0 \le w \le 1$, $J(\pi_w)$ is a monotonically increasing function of w. In other words, there is no gain in mixing the policies when w is between 0 and 1. For this experiment, we let w to take values outside this interval to examine the lowest possible cost. The best mixing weight is $w = -5.49$ and the associated cost is $J(\pi_w) = 533.60$. However, in the space of occupancy measures, we can build a better policy. Suppose $\xi_\theta = \theta\mu_1 + (1 - \theta)\mu_2$. Then the cost associated with the policy that is built based on ξ_θ decreases monotonically as we decrease θ and saturates at almost $J(\pi_\theta) = 447$, which is much lower than the cost of the optimal policy mixing in the space of w.

5.2 Queuing Problem: 8-Queues

Consider a system with three servers and eight queues. Similar to 4-queue problem, each server is responsible of serving multiple queues (see Appendix A). There are two pipelines. Jobs in the first pipeline enter the system from the first queue by rate λ_1 and exit the system from the third queue. Jobs in the second pipeline enter the system from the forth queue by rate λ_2 and exit the system from the eighth queue. Fig. 3 demonstrates the system. The service rate is denoted by r_i, $i \in \{1, 2, ..., 8\}$. There is no limit on the length of queues. State of the system is determined by an 8-dimensional vector that shows the number of jobs in each queue. Actions are also 8-dimensional binary vectors, where each dimension shows if the associated queue is served or not. The cost at each time step is equal to the total number of jobs in the system. Each server can serve only one queue at each time. We choose three base policies from a family of policies that is described below.

Family of base policies: The base policies are parameterized by three components: p_1, p_2, and p_3, which are associated to each server and satisfy the these properties: **1)** If all of the queues associated to



(a) (b) (c)

Figure 2: (a) Stationary distribution of the two initial policies. (b) Top: Cost of the mixture policy versus w. Bottom: Stationary distribution of the best mixture policy found in the space of w. (c) Top: Cost of the mixture policy versus θ. Bottom: Stationary distribution of the best mixture policy found in the space of θ.

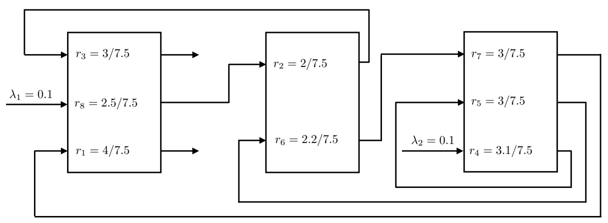


Figure 3: A system with 8 queues and 3 servers.

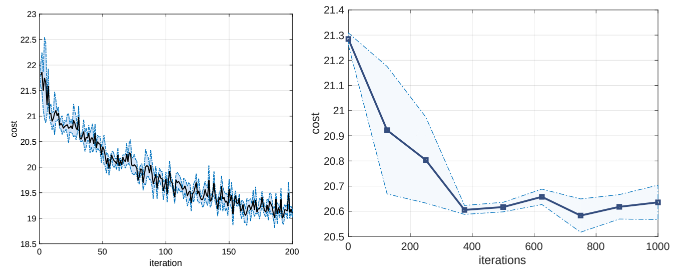
server i are empty the server is idle. **2)** If only one of the queues for server i is non-empty, that queue is served with probability p_i. **3)** If server i has more than one non-empty queue, it serves the longest queue with probability p_i and the rest of the non-empty queues with probability $(1 - p_i)/(n_q - 1)$, where n_q is the number of non-empty queues.

The three base policies and their costs are shown in Table 1. The cost is computed by running each policy for 10,000,000 iterations, starting from empty system, and averaging over number of jobs at each iteration.

Table 1: Base Policies for eight queue problem

	p_1	p_2	p_3	cost
1	0.8	0.5	0.5	21.78± 0.16
2	0.5	0.8	0.5	22.47±0.08
3	0.5	0.5	0.8	22.98 ± 0.12

Solving the problem in the policy space: As opposed to the previous problems, the number of states in this domain is not limited. We solve the problem in the primal space using policy gradient, and specifically a finite differences method. We start from a random initial weight. At each iteration, we approximate the gradient surface using four points around the current point and the weights are updated accordingly. For each point the cost of the mixed policy should be computed by running the policy for a long time, e.g. 10,0000,000 iterations. Therefore updating the weights in each iteration is computationally expensive and finding the optimum mixing weight in this space is very tedious. After 200 iterations the weight converges to $w^* = [0.35, 0.33, 0.32]$, which corresponds to $J(\pi^*) = 19.14$ (on average).



(a) (b)

Figure 4: Mean and standard deviation of cost per iteration in the eight queue problem. (a) Primal space. (b) Dual Space.

Solving the problem in the space of occupancy measures: To solve the problem in this space we need to run the base policies for a long time (around 10,0000,000 iterations) only once and obtain the stationary distributions. From there, running the induced mixed policies to estimate their values are not needed and the optimal θ is obtained using the described algorithm. The optimal value of θ for this problem is: $\theta = [1.1246, -0.1143, -0.0102]$ and the cost of the induced policy is $J(\pi_\theta) = 20.59$. In fact, by mixing the policies in this space we obtain a lower cost compared to the base policies. Although, this cost is slightly worse than the cost of the policy obtained in the primal space, the gain in computation is extremely significant. Figure 4 shows the cost per iteration in the two spaces.

References

Y. Abbasi-Yadkori, P. Bartlett, and A. Malek. Linear programming for large-scale Markov decision problems. In *International Conference on Machine Learning (ICML)*, 2014.

J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

Rong-Rong Chen and Sean Meyn. Value iteration and optimization of multiclass queueing networks. *Queueing Systems*, 32(1-3):65–97, 1999.

Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *STOC*, 2017.

Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6): 850–865, 2003.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

A. Kumar and S. Zilberstein. History-based controller design and optimization for partially observable MDPs. In *In International Conference on Automated Planning and Scheduling (ICAPS)*, 2015.

PR Kumar and Thomas I Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, 35(3):289–298, 1990.

T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, 17:533–540, 1965.

M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. *Journal of ACM*, 47:681–720, 2000.

C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 280–291, 2005.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

N. Vlassis, M. L. Littman, and D. Barber. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory*, 4(4):12, 2012.

R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Y. Ye. A new complexity result on solving the Markov decision problem. *Mathematics of Operations Research*, 30:733–749, 2005.

A Queuing Problem: 4-Queues

Consider a system with four queues, each with capacity $L = 9$, shown in Figure 5. This problem has been studied in several papers (e.g., Chen and Meyn 1999; Kumar and Seidman 1990; de Farias and Van Roy 2003). There are two servers in the system: Server 1 serves queue 1 and 4 with rates r_1 and r_4, and Server 2 serves queue 2 and 3 with rates r_2 and r_3. Each server serves only one of its associated queues at each time. Jobs arrive at queue 1 and 3 with rate λ. A job leaves the systems after being served at either queue 1 and 2 or queue 3 and 4. The state of the system is denoted by a 4-dimensional vector $[x_1, x_2, x_3, x_4]$, where x_i represents the number of jobs in queue i at each time. A controller can choose a 4-dimensional action from $\{0,1\}^4$ such that $a_1 + a_4 \leq 1$ and $a_2 + a_3 \leq 1$. The cost at each time is equal to the total number of jobs in the system: $c(x) = \sum_{i=1}^{4} x_i$.

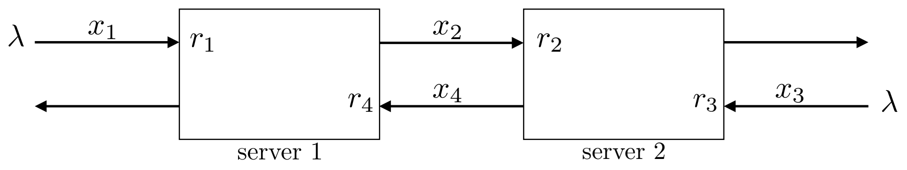


Figure 5: A system with four queues and two servers

Assume $r_1 = r_2 = 0.12$, $r_3 = r_4 = 0.28$, and $\lambda = 0.08$. We choose our base policies from a family of policies for which server i serves its longer queue w.p. p_i and its shorter queue w.p. $1 - p_i$, $i = \{1, 2\}$. Five base policies and their associated costs are shown in Table 2.

Table 2: Base Policies for the 4-Queue problem.

	p_1	p_2	cost
1	0.9	0.9	16.2950
2	0.9	0.7	17.3926
3	0.8	0.8	15.1535
4	0.7	0.9	13.6525
5	0.7	0.7	14.3266

Solving this problem in the space of policies using policy gradient will result in the optimal weight vector $w^* = [0, 0, 0, 1, 0]$, i.e., giving all the weight to the policy with the lowest cost. Therefore, the cost of the mixture policy will be $J(\pi^*) = 13.6525$. Interestingly, if we solve the problem in the dual space (space of stationary distributions) using policy gradient and Eq. 3, after 200 iterations, the optimal resulting policy will have cost $J(\pi_\theta) = 12.84$ with $\theta = [-0.32, -0.68, -0.4, 2.16, 0.24]$ (note that based on the definition of Θ, this vector can have negative values). In either of the spaces, in order to approximate the gradient surface for the policy gradient method, we need to compute the cost of each mixed policy a couple of times (depending on the number of base policies) in each iteration. Computing the cost involves an eigen-decomposition of the transition matrix, which makes the whole process computationally costly. Using our stochastic sub-gradient method, we can approximate the cost very fast and efficiently. Our method needs only one eigen-decomposition for the final mixing weights. Therefore, the computational cost is much lower (in this particular example $1/(8 \times 200)$ of the policy gradient method, where 8 is the number of points we use for approximating the gradient surface and 200 is the number of iterations). The policy resulted from our method will have cost $J(\pi_\theta) = 13.00$ with $\theta = [-0.23, -1.28, 0.09, 1.54, 0.88]$. Figure 6 shows the difference between the costs of policy gradient and the stochastic sub-gradient method at each iteration.

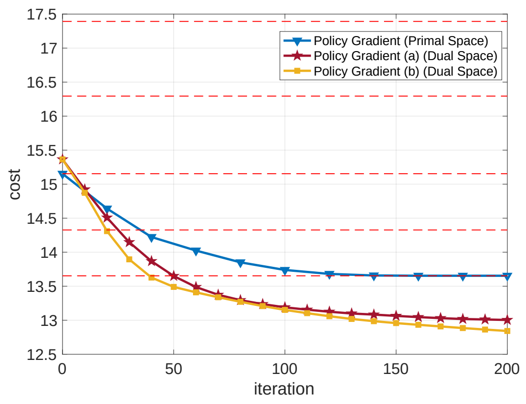


Figure 6: Cost per iteration for the primal and dual spaces. The policy gradient (b) for the dual space is the method described in Algorithm 1. Horizontal dashed lines are the costs of the base policies.