A REDUCTION APPROACH TO CONSTRAINED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many applications of reinforcement learning (RL) optimize a long-term reward subject to risk, safety, budget, diversity or other constraints. Though constrained RL problem has been studied to incorporate various constraints, existing methods require randomization among infinitely many policies to approach a feasible solution, which is impractical. In this paper, we present a reduction approach to find *sparse* policies that randomize among a constant number of policies for the constrained RL problem. The key idea is to reduce the constrained RL problem to a distance minimization problem, and a novel variant of Frank-Wolfe type algorithm is proposed for this task. Throughout the learning process, our method maintains a sparse combination of individual policies, and we show that the number of policies stored is worst-case optimal to work with any RL algorithm. Using any off-the-shelf RL algorithm as an *oracle*, our method is shown to strictly reduce the approximation error between each call of the oracle, and improvement in convergence rate over previous results is presented. Experiments on a grid-world navigation task demonstrate our method stores less policies and outperforms previous methods.

1 INTRODUCTION

Contemporary approaches in reinforcement learning (RL) largely focus on optimizing the behavior of an agent against a single reward function. This can be sufficient for simple tasks. However, for complicated applications, designing a reward function that implicitly defines the desired behavior can be challenging. For instance, applications concerning risk (Geibel & Wysotzki, 2005; Chow & Ghavamzadeh, 2014; Chow et al., 2017), safety (Chow et al., 2018) or budget (Boutilier & Lu, 2016; Xiao et al., 2019) have been studied previously. Such tasks are naturally modelled by augmenting the RL problem with constraints.

Constrained Markov decision process (CMDP) is a well-established framework to model RL problems under orthant constraints, and Altman (1999) shows that finite CMDPs with known transition model can be solved by linear programming. To incorporate new classes of constraints, such as to constrain the diversity of the trajectories for better exploration, Miryoosefi et al. (2019) proposed to study the convex constrained reinforcement learning problem (C2RL). In C2RL, instead of a scalar reward, the agent receives a measurement vector at each step. The goal is to find a policy whose expected long-term measurement vector lies in a given convex set. They reformulate the problem to a zero sum game, and solve it with online learning. However, the policy found by this online learning approach requires randomization among all policies found during the iterations. To get an ϵ-approximate solution, it requires randomizing among $O(1/\epsilon^2)$ individual policies, and an exact solution requires randomization among infinitely many policies. This limits the usage of such game-theoretic approach, especially when deep neural networks are used to represent the policies, as storing the parameters of all intermediate deep neural networks is resource-demanding for large-scale usages.

In this paper, we propose a reduction approach to C2RL. For a vector-valued Markov Decision Process (MDP) and any given target convex set, our method finds a mixed policy whose measurement vector lies in the target convex set, using any off-the-shelf RL algorithm that optimizes a scalar reward as a *RL oracle*. To do so, the C2RL problem is reduced to a distance minimization problem between a polytope and a convex set, and a novel variant of Frank-Wolfe type algorithm is proposed

Table 1: Comparison with previous approaches. To find an ϵ-approximate solution, time complexity under orthant or convex constraints are compared using the numbers of RL oracle calls. The memory requirement is measured by the number of individual policies stored for an ϵ-approximate solution.

Method	Orthant constraint	Convex constraint	Converge for any RL algo.	Memory requirement
Tessler et al. (2018)	To a fixed point	✗	✗	1
Le et al. (2019)	$O(1/\epsilon)$	✗	✓	$O(1/\epsilon)$
Miryoosefi et al. (2019)	$O(1/\epsilon^2)$	$O(1/\epsilon^2)$	✓	$O(1/\epsilon^2)$
SparseC2RL (this paper)	$O(1/\epsilon)$	$O(1/\epsilon)$	✓	$\leq m+1$

to solve this distance minimization problem. To find an ϵ-approximate solution, the number of RL oracle invocation reduces from $O(1/\epsilon^2)$ (Miryoosefi et al., 2019) to $O(1/\epsilon)$ (our method). Beside the improved performance, for an m-dimensional vector-valued MDP, our method only stores at most $m+1$ policies, which improves from the quadratic $O(1/\epsilon^2)$ (Miryoosefi et al., 2019) to a constant. The constant is also shown to be worst-case optimal to ensure convergence of RL algorithms using deterministic policies. Since the measurement vector of many constraints are low-dimensional (such as risk, safety or budget), this *sparse* property makes our method appealing for large scale usages. A preliminary experimental comparison with approachability-based policy optimization (ApproPO) (Miryoosefi et al., 2019) and reward constrained policy optimization (RCPO) (Tessler et al., 2018) demonstrates the performance of the proposed method and the sparsity of the policy found. A theoretical comparison between our method and various approaches is provided in Table 1.

2 Related Work

For high dimensional constrained RL, one line of approaches incorporates the constraint as a penalty signal into the reward function, and makes updates in a multiple time-scale scheme (Tessler et al., 2018; Chow & Ghavamzadeh, 2014). When used with policy gradient or actor-critic algorithms (Sutton & Barto, 2018), this penalty signal guides the policy to converge to a constraint satisfying one (Paternain et al., 2019; Chow et al., 2017). However, the convergence guarantee requires the RL algorithm can find a single policy that satisfies the constraint, hence ruling out methods that search for deterministic policies, such as Deep Q-Networks (DQN) (Mnih et al., 2013), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) and their variants (Van Hasselt et al., 2015; Wang et al., 2016; Fujimoto et al., 2018; Barth-Maron et al., 2018).

Another line of approaches uses a game-theoretic framework, and does not tie to specific families of RL algorithm. The constrained problem is relaxed to a zero-sum game, whose equilibrium is solved by online learning (Agarwal et al., 2018). The game is played repeatedly, each time any RL algorithm can be used to find a *best response* policy to play against a no-regret online learner. The *mixed policy* that uniformly distributed among all played policies can be shown to converge to an optimal policy of the constrained problem (Freund & Schapire, 1999; Abernethy et al., 2011). Taking this approach, Le et al. (2019) uses Lagrangian relaxation to solve the orthant constraint case, and Miryoosefi et al. (2019) uses conic duality to solve the convex constraint case. However, since the convergence is established by the no-regret property, the policy found by these methods requires randomization among policies found during the learning process, which limits their prevalence.

3 Preliminaries

A *vector-valued Markov decision process* can be identified by a tuple $\{\mathcal{S}, \mathcal{A}, \beta, P, \boldsymbol{c}\}$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is the set of actions and β is the initial state distribution. At the start of each episode, an initial state s_0 is drawn following the distribution β. Then, at each step $t = 0, 1, \ldots$, the agent observes a state $s_t \in \mathcal{S}$ and makes a decision to take an action a_t. After a_t is chosen, at the next observation the state evolves to state $s_{t+1} \in \mathcal{S}$ with probability $P(s_{t+1}|s_t, a_t)$. However, instead of a scalar reward, in our setting, the agent receives a m-dimensional vector $\boldsymbol{c}_t \in \mathbb{R}^m$ that may implicitly contain measurements of reward, risk or violation of other constraints. The episode ends after a certain number of steps, called the horizon, or when a terminate state is reached.

Actions are typically selected according to a policy π, where $\pi(s)$ is a distribution over actions for any $s \in \mathcal{S}$. Policies that take a single action for any state are *deterministic policies*, and can be identified by the mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$. The set of all deterministic policies is denoted by Π. For a discount factor $\gamma \in [0, 1)$, the discounted long-term *measurement vector* of a policy $\pi \in \Pi$ is defined as

$$\boldsymbol{c}(\pi) := \mathbb{E}(\sum_{t=0}^{T} \gamma^t \boldsymbol{c}_t(s_t, \pi(s_t))), \tag{1}$$

where the expectation is over trajectories generated by the described random process.

Unlike unconstrained setting, for a constrained RL problem, it is possible that all feasible policies are non-deterministic (see Appendix D for an example). This limits the usage of RL algorithms that search for deterministic policies in the setting of constrained RL problem.

One workaround is to use mixed policies. For a set of policies $\mathcal{U}$, a mixed policy is a distribution over $\mathcal{U}$, and the set of all mixed policies over $\mathcal{U}$ is denoted by $\Delta(\mathcal{U})$. To execute a mixed policy $\mu \in \Delta(\mathcal{U})$, we first select a policy $\pi \in \mathcal{U}$ according to $\pi \sim \mu(\pi)$, and then execute π for the entire episode. Altman (1999) shows that any $\boldsymbol{c}(\cdot)$ achievable can be achieved by some *mixed deterministic policies* $\mu \in \Delta(\Pi)$. Therefore, though an off-shelves RL algorithm may not converge to any constraint-satisfying policy, it can be used as a subroutine to find individual policies (possibly deterministic), and a randomization among these policies can converge to a feasible policy. The discounted long-term measurement vector of a mixed policy $\mu \in \Delta(\Pi)$ is defined similarly

$$\boldsymbol{c}(\mu) := \mathbb{E}_{\pi \sim \mu}(\boldsymbol{c}(\pi)) = \sum_{\pi \in \Pi} \mu(\pi) \boldsymbol{c}(\pi). \tag{2}$$

For a mixed policy $\mu \in \Delta(\mathcal{U})$, its *active set* is defined to be the set of policies with non-zero weights $\mathcal{A} := \{\pi \in \mathcal{U} | \mu(\pi) > 0\}$. The memory requirement of storing μ, is then proportion to the size of its active set. Since a mixed policy can be interpreted as a convex combination of policies in its active set, in the following, the term *sparsity* of a mixed policy refers to the sparsity of this combination.

Our learning problem, the convex constrained reinforcement learning (C2RL), is to find a policy whose expected long-term measurement vector lies in a given convex set; i.e., for a given convex target set $\mathcal{C} \subset \mathbb{R}^m$, our target is to

$$\text{find } \mu^* \text{ such that } \boldsymbol{c}(\mu^*) \in \Omega \qquad \text{(C2RL)}. \tag{3}$$

Any policy μ^* that satisfies $\boldsymbol{c}(\mu^*) \in \Omega$ is called a *feasible policy*, and a C2RL problem is *feasible* if there exists some feasible policies. In the following, we assume the C2RL problem is feasible.

4 Approach, Algorithm and Analysis

We now show how the C2RL (3) can be reduced to a distance minimization problem (7) between a polytope and a convex set. A novel variant of Frank-Wolfe-type algorithm is then proposed to solve the distance minimization problem, followed by theoretic analysis about convergence and sparsity of the proposed method.

4.1 Reduce C2RL to a Distance Minimization Problem

Let $||\cdot||$ denotes the Euclidean norm. For a convex set $\Omega \in \mathbb{R}^m$, let $\texttt{Proj}_\Omega(\boldsymbol{x}) \in \arg\min_{\boldsymbol{y} \in \Omega} ||\boldsymbol{x} - \boldsymbol{y}||$ be the projection operator, and $\texttt{dist}$ be the squared Euclidean distance function: $f_\Omega(\boldsymbol{x}) := \frac{1}{2}||\boldsymbol{x} - \texttt{Proj}_\Omega(\boldsymbol{x})||^2$. For the problem

$$\arg\min_{\mu \in \Delta(\Pi)} \texttt{dist}(\boldsymbol{c}(\mu), \Omega), \tag{4}$$

a policy $\mu^* \in \Delta(\Pi)$ is defined to be an *optimal* solution if it minimizes (4). Otherwise, the *approximation error* of $\mu \in \Delta(\Pi)$ is defined as

$$\texttt{err}(\mu) := \texttt{dist}(\boldsymbol{c}(\mu), \Omega) - \texttt{dist}(\boldsymbol{c}(\mu^*), \Omega) \qquad \text{(Approximation Error)} \tag{5}$$

where μ^* is an optimal solution, and a policy is defined to be an ϵ-approximate solution if its approximation error is no larger than ϵ.

When C2RL (3) is feasible, the equivalence of being optimal to (4) and being feasible to C2RL can be easily established. Since a feasible policy of C2RL problem lies inside Ω, it minimizes the non-negative $\texttt{dist}$ function, and hence is optimal to (4). Vice versa, any optimal solution to (4) lies inside Ω and is a feasible solution to C2RL.

From a geometric perspective, let $\boldsymbol{c}(\Pi) := \{\boldsymbol{c}(\pi) | \pi \in \Pi\}$ be the set of all values achievable by deterministic policies. If the MDP has finite states and actions (though may be extremely large), then Π is finite as well, and hence $\boldsymbol{c}(\Pi)$ contains finitely many points in $\mathbb{R}^m$. Then the set of values achievable by mixed deterministic policies is

$$\boldsymbol{c}(\Delta(\Pi)) := \{\boldsymbol{c}(\mu) | \mu \in \Delta(\Pi)\} = \{\sum_{\pi} \mu(\pi)\boldsymbol{c}(\pi) \mid \sum_{\pi} \mu(\pi) = 1, \mu(\pi) \geq 0\} \subset \mathbb{R}^m \tag{6}$$

is the convex hull of $\boldsymbol{c}(\Pi)$; i.e., $\boldsymbol{c}(\Delta(\Pi))$ is a m-dimension polytope whose vertices are $\boldsymbol{c}(\Pi)$. Therefore finding a policy whose value is closest to the target convex set (4) is equivalent to minimizing the distance between the polytope $\boldsymbol{c}(\Delta(\Pi))$ and the convex set Ω

$$\underset{\boldsymbol{c}(\mu) \in \boldsymbol{c}(\Delta(\Pi))}{\arg\min} \ \texttt{dist}(\boldsymbol{c}(\mu), \Omega) \qquad \text{(Distance minimization problem).} \tag{7}$$

Such constrained optimization problem over a polytope can be solved by the Frank-Wolfe algorithm (FW). Before getting into details of our method, we discuss the standard FW and variants of FW-type algorithms over polytope.

4.2 Frank-Wolfe-type Algorithms over Polytope

The Frank-Wolfe algorithm (FW) is a first-order method to minimize a convex function $f : \mathcal{P} \mapsto \mathbb{R}$ over a compact and convex set $\mathcal{P}$, with only access to a *linear minimizer oracle*. When the feasible set is a polytope $\mathcal{P} := \texttt{conv}(\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}) \subset \mathbb{R}^m$ defined as the convex hull of finitely many points, FW-type algorithms are discussed by (Lacoste-Julien & Jaggi, 2015) to optimize

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \quad \text{using} \quad \texttt{Oracle}(\texttt{v}) := \underset{\boldsymbol{s} \in \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}}{\arg\min} \ \boldsymbol{s}^T \boldsymbol{v}. \tag{8}$$

The standard FW (Algorithm 2 in Appendix A.1) consists of making repeated calls to the linear minimizer oracle (line 2), followed by a convex averaging step of the current iterate and the oracle's output (line 3). Though the standard FW can be applied to solve the distance minimization problem (7), an ϵ-approximate solution found by standard FW requires maintaining of an active set of policies of size $O(1/\epsilon)$, which is memory demanding.

Variants of FW-type algorithms has been studied. In particular, when the objective function is the Euclidean distance to a fixed point, Wolfe's method for minimum norm point has been proposed (Wolfe, 1976). In Wolfe's method (Algorithm 3 in Appendix A.2), the loop in FW is called a *major cycle*, and the convex averaging step (line 3) is replaced by a weight optimization process which is called a *minor cycle*. Wolfe's method maintains an *active set* $\mathcal{S}$, and the current point can represented by a sparse combination of points in the active set. The minor cycles maintain $\mathcal{S}$ to be an affinely independent set such that the affine minimizer is inside $\mathcal{S}^t$, which Wolfe calls *corrals*. Recall an affine minimizer is defined as $\arg\min_{\boldsymbol{s} \in \texttt{aff}(\mathcal{S})} ||\boldsymbol{s}||_2$, where $\texttt{aff}(\mathcal{S}) := \{\boldsymbol{y} | \boldsymbol{y} = \sum_{\boldsymbol{z} \in \mathcal{S}} \alpha_{\boldsymbol{z}}^T \boldsymbol{x}, \sum_{\boldsymbol{z} \in \mathcal{S}} \alpha_{\boldsymbol{z}} = 1\}$ is the affine hull formed by $\mathcal{S}$. Since the active set is affinely independent, the number of active atoms is at most $m + 1$ at any time. Wolfe's method is shown to strictly decrease the approximation error between two major cycles.

In the reminder of this section, we propose a novel variant of FW type algorithm to solve the distance minimization problem (7) between a polytope and a convex set.

4.3 Our Main Algorithm

At the core of our approach, we propose a novel variant of Frank-Wolfe algorithm, to minimize the distance between the polytope of all values achievable by mixed deterministic policy $\boldsymbol{c}(\Delta(\Pi))$ and the target convex set Ω. The proposed method can be seen as an extension of Wolfe's method, whose

Algorithm 1 Convex Constrained Reinforcement Learning (C2RL)

Input. $\texttt{RL_Oracle}$ over policy set $\mathcal{U}$, projection operator to target set $\texttt{Proj}_\Omega$.
Initialize. Policy $\pi \in \mathcal{U}$, value $\boldsymbol{x} = \boldsymbol{c}(\pi)$, active sets $\mathcal{S}_p := [\pi], \mathcal{S}_c := [\boldsymbol{x}]$ and weight $\boldsymbol{\lambda} = [1]$.
Output. Mixed policy μ and its value $\boldsymbol{c}(\mu)$ s.t. $\boldsymbol{c}(\mu)$ minimizes the distance to the target set Ω.

while true **do** // Major cycle
  $\boldsymbol{\omega} \leftarrow \texttt{Proj}_\Omega(\boldsymbol{x})$
  $\pi, \boldsymbol{c}(\pi) \leftarrow \texttt{RL_Oracle}(\boldsymbol{x} - \boldsymbol{\omega})$ // Potential improving point
  if $(\boldsymbol{x} - \boldsymbol{\omega})^T(\boldsymbol{x} - \boldsymbol{c}(\pi)) \leq \epsilon$ **then** break
  if $\mathcal{S}_c \cup \{\boldsymbol{c}(\pi)\}$ is affinely independent **then** $\mathcal{S}_c \leftarrow \mathcal{S}_c \cup \{\boldsymbol{c}(\pi)\}, \mathcal{S}_p \leftarrow \mathcal{S}_p \cup \{\pi\}$
  while true **do** // Minor cycle
    $\boldsymbol{y}, \boldsymbol{\alpha} \leftarrow \texttt{AffineMinimizer}(\mathcal{S}_c, \boldsymbol{\omega})$ // $\boldsymbol{y} = \arg\min_{\boldsymbol{s} \in \texttt{aff}(\mathcal{S}_c)} ||\boldsymbol{s} - \boldsymbol{\omega}||_2$
    if $\alpha_{\boldsymbol{s}} > 0$ for all $\boldsymbol{s}$ **then** break // $\boldsymbol{y} \in \texttt{conv}(\mathcal{S}_c)$
    // If $\boldsymbol{y} \notin \texttt{conv}(\mathcal{S}_c)$, then update $\boldsymbol{y}$ to the intersection of $\texttt{conv}(\mathcal{S}_c)$ and segment joining $\boldsymbol{x}$ and $\boldsymbol{y}$. Then remove points in $\mathcal{S}_c$ unnecessary for describing $\boldsymbol{y}$.
    $\theta \leftarrow \min_{i:\alpha_i \leq 0} \frac{\lambda_i}{\lambda_i - \alpha_i}$ // Recall $\boldsymbol{\lambda}$ satisfies $\boldsymbol{x} = \sum_{\boldsymbol{s} \in \mathcal{S}_c} \lambda_{\boldsymbol{s}} \boldsymbol{s}$
    $\boldsymbol{y} \leftarrow \theta \boldsymbol{y} + (1 - \theta)\boldsymbol{x}, \lambda_i = \theta \alpha_i + (1 - \theta)\lambda_i$
    $\mathcal{S}_c \leftarrow \{\boldsymbol{c}(\pi_i) | \boldsymbol{c}(\pi_i) \in \mathcal{S}_c \text{ and } \lambda_i > 0\}, \mathcal{S}_p \leftarrow \{\pi_i | \pi_i \in \mathcal{S}_p \text{ and } \lambda_i > 0\}$
  end while
  Update $\mu \leftarrow \sum_{\pi \in \mathcal{S}_p} \lambda_\pi \pi$, $\boldsymbol{x} \leftarrow \boldsymbol{y}$, $\boldsymbol{\lambda} \leftarrow \boldsymbol{\alpha}$.
end while
return μ, $\boldsymbol{c}(\mu) \leftarrow \boldsymbol{x}$

objective is to minimize the distance to a fixed point over a polytope. Using any RL algorithm as an oracle, our method minimizes the distance to a target convex set over a polytope, and the found point is maintained as a sparse combination of vertices of the polytope.

We shall define some basic notions. For a set of points in $\mathcal{S} \subset \mathbb{R}^m$, and $\boldsymbol{x} \in \mathbb{R}^m$, we define *affine minimizer with respect to* $\boldsymbol{x}$ as $\arg\min_{\boldsymbol{s} \in \texttt{aff}(\mathcal{S})} ||\boldsymbol{s} - \boldsymbol{x}||_2$. For $\boldsymbol{x}$ being the affine minimizer of $\mathcal{S}$ with respect to $\boldsymbol{\omega}$, the affine minimizer property gives

$$\text{Given } \omega, \forall \boldsymbol{v} \in \texttt{aff}(\mathcal{S}), (\boldsymbol{v} - \boldsymbol{x})^T(\boldsymbol{x} - \boldsymbol{\omega}) = 0 \quad \text{(Affine minimizer property)} \tag{9}$$

Similar to Wolfe's method, our C2RL method (Algorithm 1) contains an outer loop (called major cycle) to find improving policies and their measurement vectors, and an inner loop (called minor cycle) to maintain the affinely independent property of the active set $\mathcal{S}_c$. At the start of each major cycle step, the $\mathcal{S}_c$ is an affinely independent set. Then, the $\texttt{RL_oracle}$ (defined in (10)) finds a potential improving policy $\pi \in \mathcal{U}$, and its long-term measurement vector $\boldsymbol{c}(\pi)$. If the $\boldsymbol{c}(\pi)$ does not get strictly closer to the $\boldsymbol{\omega} := \texttt{Proj}(\boldsymbol{x})$, then we are done, and $\boldsymbol{x}$ is the optimal value. Otherwise, the $\boldsymbol{c}(\pi)$ is added into the active set, and the minor cycle is run to eliminate policies whose measurement vectors are affinely dependent. The elimination is executed as a series of affine projections. The minor cycle terminates if active set $\mathcal{S}_c$ is affinely independent. Though the interleaving of major and minor cycles oscillate the size of active set $\mathcal{S}_c$, the minor cycles keep $|\mathcal{S}_c|$ an affine independent set, and is terminated whenever $\mathcal{S}_c$ contains a single element. Therefore at the start of any major cycle, the size of the active set satisfies $|\mathcal{S}_c| \in [0, m + 1]$.

The construction of our RL oracle can use any off-the-shelf RL algorithm that minimizes a scalar reward. For any given $\boldsymbol{\lambda} \in \mathbb{R}^m$, we define any algorithm that finds a policy minimizing the linear function $\boldsymbol{\lambda}^T \boldsymbol{c}(\cdot)$ as a *RL oracle*, that is

$$\texttt{RL_oracle}_{\texttt{p}}(\boldsymbol{\lambda}) \in \arg\min_{\pi \in \Pi} \boldsymbol{\lambda}^T \boldsymbol{c}(\pi). \tag{10}$$

The construction of RL oracle relies on the following linear property

$$\arg\min_{\pi \in \Pi} \boldsymbol{\lambda}^T \boldsymbol{c}(\pi) = \arg\min_{\pi \in \Pi} \boldsymbol{\lambda}^T \mathbb{E}(\sum_{t=0}^{T} \gamma^t \boldsymbol{c}_t) = -\arg\max_{\pi \in \Pi} \mathbb{E}(\sum_{t=0}^{T} \gamma^t (-\boldsymbol{\lambda}^T \boldsymbol{c}_t)), \tag{11}$$

which shows that a linear minimizer of $\boldsymbol{\lambda}^T \boldsymbol{c}(\cdot)$ can be solved by training any RL algorithms to maximize the scalar reward $(-\boldsymbol{\lambda}^T \boldsymbol{c}_t)$. Therefore any RL algorithm that suits the underlying problem can be used to construct a RL oracle.

Certifying constraint satisfaction amounts to evaluate the measurement vector of the current policy. This is handy in online settings, where simulations can be used to evaluate the measurement vector of the policy directly. Otherwise, in batch settings, various off-policy evaluation methods, such as importance sampling (Precup, 2000; Precup et al., 2001) or doubly robust (Jiang & Li, 2016; Dudík et al., 2011), can be used to evaluate the policy.

$$\texttt{RL_oracle}_{\texttt{c}}(\boldsymbol{\lambda}) := \boldsymbol{c}(\arg\min_{\pi\in\Pi} \boldsymbol{\lambda}^T \boldsymbol{c}(\pi)) = \arg\min_{\boldsymbol{c}(\pi),\pi\in\Pi} \boldsymbol{\lambda}^T \boldsymbol{c}(\pi). \tag{12}$$

With a bit abuse of notation, we assume a $\texttt{RL_Oracle}$ returns a policy as well as the measurement vector of the policy. The process $\texttt{AffineMinimizer}(S, \boldsymbol{x})$ returns the $(\boldsymbol{y}, \boldsymbol{\alpha})$ the affine minimizer of S with respect to $\boldsymbol{x}$ where $\boldsymbol{y}$ is the affine minimizer and $\boldsymbol{\alpha} := \{\alpha_{\boldsymbol{s}} | \forall \boldsymbol{s} \in \mathcal{S}_c\}$ is the set of coefficient expressing $\boldsymbol{y}$ as an affine combination of points in S, that is $\boldsymbol{y} = \sum_{\boldsymbol{s}\in\mathcal{S}_c} \alpha_{\boldsymbol{s}} \boldsymbol{s}$, where $\alpha_{\boldsymbol{s}}$ is the weight associated with $\boldsymbol{s}$. The process $\texttt{AffineMinimizer}(S, \boldsymbol{x})$ can be straightforwardly implemented using linear algebra. Wolfe (1976) also provides a more efficient implementation that uses a triangular array representation of the active set.

4.4 Convergence and Sparsity

We first show that approximation error of Algorithm 1 strictly decreases among any two major cycle steps (Thm. 4.1). Then, the convergence is studied. We show that after $O(1/\epsilon)$ rounds of iteration, Algorithm 1 returns a policy $\mu \in \Delta(\Pi)$ satisfying $\texttt{err}(\mu) \leq \epsilon$ (Thm. 4.2). That is, the Algorithm (1) converges in $O(1/t)$ rate. Finally we show that maintaining an active policy set of $m+1$ is worst case optimal to ensure the convergence of arbitrary RL algorithm, especially those searching for deterministic policies (Thm. 4.5.1).

A major obstacle to analyze the proposed Algorithm 1 is the mixture of major and minor cycles, which makes the sizes of the active sets $\mathcal{S}_c$ and $\mathcal{S}_p$ fluctuate during iterations. In our analysis, we consider the approximation error as defined in (5). We use superscript t to denote the variable in t-th major cycle before executing any minor cycle. To simplify notions, we let $x^t := \boldsymbol{c}(\mu^t)$ and $\boldsymbol{s}^t := \boldsymbol{c}(\pi^t)$. Since $\boldsymbol{y}$ is used only within each major cycle, when discussing one step with t fixed, we let $\boldsymbol{y}^i$ denote the affine minimizer found in i-th minor cycle (line 6 of Algorithm 1). We first show that the Algorithm 1 strictly reduces approximation error between two calls of the RL oracle.

Theorem 4.1 (Approximation Error Strictly Decreases)**.** *For any non-terminal step t, we have $\texttt{err}(\mu^{t+1}) < \texttt{err}(\mu^t)$. That is, the measurement vector of μ^t found by Algorithm 1 gets strictly closer to the convex set Ω after major cycle step.*

The proof is provided in Appendix B. The idea is to consider the distance between $\boldsymbol{x}^t$ and $\boldsymbol{\omega}^t$. When the major cycle has no minor cycle, the non-terminal condition and the affine minimizer property implies $\texttt{dist}(\boldsymbol{x}^{t+1}, \boldsymbol{\omega}^t) < \texttt{dist}(\boldsymbol{x}^t, \boldsymbol{\omega}^t)$. Otherwise we show that the first minor cycle strictly reduces the $\texttt{dist}(\boldsymbol{x}^t, \boldsymbol{\omega}^t)$ by moving along the segment joining $\boldsymbol{x}$ and $\boldsymbol{y}$, and the subsequent minor cycle cannot increase it. Since $\boldsymbol{\omega}^t \in \Omega$, we conclude $\texttt{err}(\boldsymbol{x}^{t+1}) \leq \texttt{dist}(\boldsymbol{x}^{t+1}, \omega^t) < \texttt{dist}(\boldsymbol{x}^t, \omega^t) = \texttt{err}(\boldsymbol{x}^t)$, and the approximation error strictly decreases.

Given the approximation error strictly decreases, Wolfe's method for minimum norm point can be shown to terminate finitely (Wolfe, 1976). However, this finitely terminating property does not hold for our algorithm. Since a changed ω^t may yield a lower distance to the same active set $\mathcal{S}_c^t$, the active set may stay unchanged across major cycles (see Figure 2 Middle for an example). We establish the convergence rate of our method in the following theorem.

Theorem 4.2 (Convergence in Approximation Error)**.** *For $t \geq 1$, the mixed policy μ^t found by Algorithm 1 satisfies*

$$\texttt{err}(\mu^t) \leq 16Q^2/(t+2), \tag{13}$$

where $Q := \max_{\mu\in\Delta(\mathcal{U})} ||\boldsymbol{c}(\mu)||$ is the maximum norm of a measurement vector.

The proof is provided in Appendix C, which relies on the following two lemmas. We briefly discuss the main idea here. Define major cycle steps with at most one minor cycle as "non-drop step" and major cycle steps with more than one minor cycles as "drop steps". We show that in each non-drop step, Algorithm 1 is guaranteed to make enough progress in the following lemma.

Lemma 4.3. *For a non-drop step, we have $\texttt{err}(\mu^t) - \texttt{err}(\mu^{t+1}) \geq \texttt{err}^2(\mu^t)/8Q^2$.*

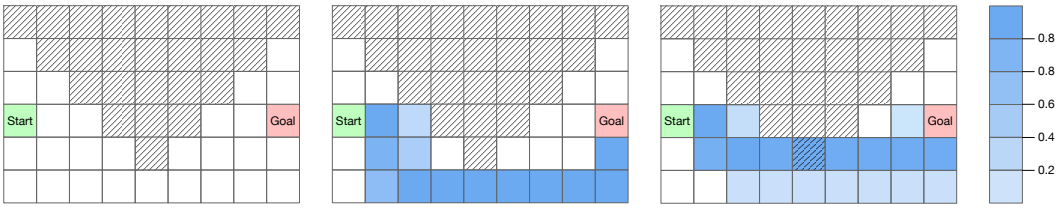

Figure 1: *Left*: The *Risky Mars Rover* environment. The agent is required to navigate from the starting point to reach the goal point without staying long (0.5 steps in expectation) in the risky area (cross-hatching region). *Middle, Right*. Example of an optimal mixed policy found by C2RL in a single run. After 10k samples, C2RL finds a mixed policy that randomizes among two policies with weight 0.49 and 0.51. The visitation probabilities of the two policies are plotted.

However, this does not hold for drop step. We use the following lemma to bound the frequency of drop steps.

Lemma 4.4. *After t major cycle steps of Algorithm 1, the number of drop steps is less than $t/2$.*

Since the approximation error strictly decreases (Thm. 4.1), and in more than half of the major cycles steps Algorithm 1 makes significantly progress. The Thm. (4.2) can then be proved using an induction argument (Appendix C).

We then discuss the sparsity of mixed policy for constrained RL problem, and we give an constructive proof in Appendix D.

Theorem 4.5 (Memory Complexity Bound)**.** *For an constrained RL problem with m-dimensional measurement vector, in the worst case, a mixed policy needs to randomize among $m+1$ individual policies to ensure convergence of RL oracles that search for deterministic policies.*

Corollary 4.5.1. *The Algorithm 1 that randomize among at most $m+1$ policies is worst-case optimal to ensure convergence of any RL oracle.*

5 Experiments

We evaluate the performance of C2RL in a grid-world navigation task (Figure 1, and demonstrate its ability to efficiently find sparse policy. The proposed method is compared with approachability-based policy optimization (ApproPO) (Miryoosefi et al., 2019) and with reward constrained policy optimization (RCPO) (Tessler et al., 2018). ApproPO solves the same convex constrained RL problem by using an RL oracle to play against a no-regret online learner (Hazan et al., 2008; Zinkevich, 2003). Since ApproPO and C2RL both use a RL oracle, ApproPO is a natural baseline to be compared with our method. Besides, we also compare with RCPO, which takes a Lagrangian approach to incorporate the constraints as a penalty signal into the reward. Using an advantage actor critic (A2C) Mnih et al. (2016), RCPO has been shown to converge to a fixed point. For a fair comparison, C2RL and ApproPO uses an A2C agent as the RL oracle, with the same hyperparameter as used in RCPO. The approximation errors are compared after training for the same number of samples.

We evaluate the method on a *Risky Mars Rover* grid-world environment. In this environment, the agent navigates from the starting point to the goal. The episodes terminate when the goal point is reached or after 300 steps. The environment contains a risky area. The agent receives a measurement vector to indicate the steps it takes (0.1 for every step), and whether it stays in the risky region (0.1 for every risky step, and 0 otherwise), with discount factor $\gamma = 0.99$. We constrain the agent to reach the goal point with expected cumulative steps measure within 1.1 and the expected cumulative risky steps within 0.05.

We first showcase the consequences of our theoretical results using an *optimal RL oracle*. For any $\boldsymbol{x} \in \mathbb{R}^m$, an optimal policy can be easily found via Dijkstra's algorithm. If multiple optimal paths exist, one is randomly picked to form a deterministic policy. Using this as an optimal RL oracle, the convergence property of C2RL and ApproPo are compared. Figure 2 Middle shows the value of policies $\boldsymbol{c}(\mu^t)$ found after each call to the oracle. In Figure 2 Right, when approaching the boundary of the feasible set, the iterations of approachability-based methods start to zigzag. Lacoste-

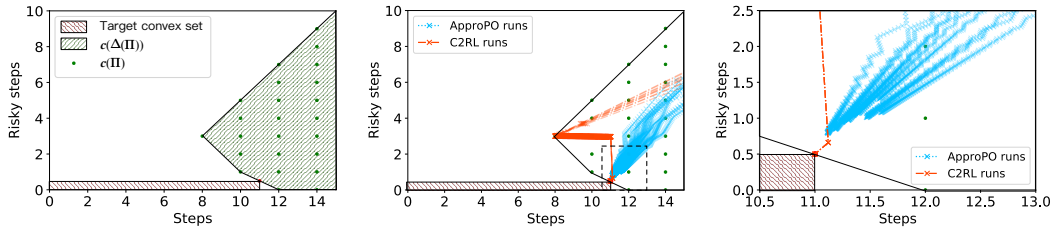

Figure 2: *Left*: Visualization of the distance minimization problem (7) in $\mathbb{R}^2$, where the number of steps and the number of steps in risky zone are measured. The green hatched region is the polytope formed by values achievable by mixed deterministic policies $\boldsymbol{c}(\Delta(\Pi))$, and the red hatched region is the target set. *Middle*: Using an *optimal RL oracle*, 10 paths are sampled to showcase the convergence property of C2RL and ApproPO, where each cross on the dashed line corresponds to a call to the oracle. *Right*: If we zoom in, ApproPO suffers from the zig-zagging problem.

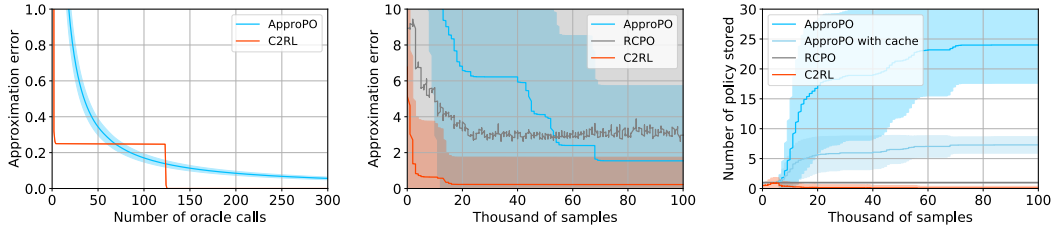

Figure 3: *Left*: Time complexity measured by number of calling an optimal RL oracle. *Middle, Right*: Using A2C to approximate an RL oracle, time complexity measured by thousands of samples and memory complexity measured by the number of policies stored are compared.

Julien & Jaggi (2015) discuss this zig-zagging problem in the context of FW-type algorithms and shows that, intuitively, reducing weights from "bad atoms" alleviates this problem, such as away-step FW (Wolfe, 1970; Jaggi, 2013), pairwise FW (Mitchell et al., 1974), and Wolfe's method (Wolfe, 1976). Since C2RL contains a minor cycle to re-optimize the weights among the active set, C2RL progresses quickly to reach the exact optimal solution. In Figure 3 Left, the approximation error is shown for 300 calls of the optimal RL oracle.

We then compare C2RL, ApproPO and RCPO using the same A2C agent. ApproPO has a parameter to project the convex constraint to a cone constraint, and both ApproPO and RCPO has an learning rate in the outer loop. We use the value as suggested by their paper. We run each algorithm for 50 times, and each run for a maximum of 100 thousands of samples. The mean and standard deviation of the results are presented in Figure 3. The original paper of ApproPO suggests using a cache to save memory, and the memory requirement of this variant is also presented. Figure 3 demonstrates that C2RL converges to an optimal policy faster than previous methods, and a sparse combination of individual policies are maintained throughout the iteration process.

6 Conclusion

In this paper, we introduce C2RL, an algorithm to solve RL problems under convex constraints. Our method reduces the constrained RL problem to a distance minimization problem, and a novel variant of Frank-Wolfe type algorithm is proposed to solve this. Our method comes with rigorous theoretical guarantees. To find an ϵ-approximation solution, C2RL takes $O(1/\epsilon)$ calls of any RL oracle, which improves from previous $O(1/\epsilon^2)$. Moreover, C2RL strictly reduces the approximation error between consecutive calls of RL oracle. Beside the improved performance, for m-dimensional constraints, the memory requirement is reduce from storing infinitely many policies ($O(1/\epsilon^2)$) to storing at most constantly many $(m+1)$ polices. We further show that the constant is worst-case optimal to ensure the convergence of any RL algorithms that search for deterministic policies. Experimentally, we demonstrate that C2RL finds sparse solution efficiently, and outperforms previous methods.

REFERENCES

Jacob Abernethy, Peter L Bartlett, and Elad Hazan. Blackwell approachability and no-regret learning are equivalent. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 27–46, 2011.

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453*, 2018.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Amir Beck and Shimrit Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming*, 164(1-2):1–27, 2017.

Craig Boutilier and Tyler Lu. Budget allocation using weakly coupled, constrained markov decision processes. 2016.

Deeparnab Chakrabarty, Prateek Jain, and Pravesh Kothari. Provable submodular minimization using wolfe's algorithm. In *Advances in Neural Information Processing Systems*, pp. 802–809, 2014.

Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for cvar optimization in mdps. In *Advances in neural information processing systems*, pp. 3509–3517, 2014.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18 (1):6070–6120, 2017.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in neural information processing systems*, pp. 8092–8101, 2018.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Dan Garber and Elad Hazan. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *arXiv preprint arXiv:1301.4666*, 2013a.

Dan Garber and Elad Hazan. Playing non-linear games with linear oracles. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 420–428. IEEE, 2013b.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

Elad Hazan, Alexander Rakhlin, and Peter L Bartlett. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems*, pp. 65–72, 2008.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th international conference on machine learning*, number CONF, pp. 427–435, 2013.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661. PMLR, 2016.

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in neural information processing systems*, pp. 496–504, 2015.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712, 2019.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Sobhan Miryoosefi, Kianté Brantley, Hal Daume III, Miro Dudik, and Robert E Schapire. Reinforcement learning with convex constraints. In *Advances in Neural Information Processing Systems*, pp. 14093–14102, 2019.

BF Mitchell, Vladimir Fedorovich Dem'yanov, and VN Malozemov. Finding the point of a polyhedron closest to the origin. *SIAM Journal on Control*, 12(1):19–26, 1974.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, pp. 7555–7565, 2019.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pp. 417–424, 2001.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003, 2016.

Philip Wolfe. Convergence theory in nonlinear programming. *Integer and nonlinear programming*, pp. 1–36, 1970.

Philip Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.

Shuai Xiao, Le Guo, Zaifan Jiang, Lei Lv, Yuanbo Chen, Jun Zhu, and Shuang Yang. Model-based constrained mdp for budget allocation in sequential incentive marketing. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 971–980, 2019.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

A MORE ON FRANK-WOLFE-TYPE ALGORITHMS

A.1 STANDARD FRANK-WOLFE ALGORITHM

Algorithm 2 Frank-Wolfe algorithm (Frank et al., 1956)

Input: obj. $f : \mathcal{Y} \mapsto \mathbb{R}$, oracle $O(\cdot)$, init. $\boldsymbol{w}_0 \in \mathcal{Y}$

for t=1, 2, 3 ..., T **do**
  $\boldsymbol{v}_t \leftarrow \texttt{Oracle}(\nabla f(\boldsymbol{w}_{t-1})) = \arg\min_{\boldsymbol{v} \in \mathcal{Y}} \boldsymbol{v}^T \nabla f(\boldsymbol{w}_{t-1})$
  $\boldsymbol{w}_t \leftarrow (1 - \eta_t \boldsymbol{w}_{t-1}) + \eta_t \boldsymbol{v}_t$, for $\eta := \frac{2}{t+2}$
end for
return $\boldsymbol{w}_T$

For a convex function $f : \mathcal{X} \mapsto \mathbb{R}$ the Frank-Wolfe algorithm (FW) solves the constrained optimization problem over a compact and convex set $\mathcal{X}$. The standard FW is known to have a sublinear convergence rate, and various methods are proposed to improve the performance. For example, when the underlying feasible set is a polytope, and the objective function is strongly convex, multiple variants, such as away-step FW (Wolfe, 1970; Jaggi, 2013), pairwise FW (Mitchell et al., 1974), and Wolfe's method (Wolfe, 1976) are shown to enjoy linear convergence rate. Linear convergence under other conditions are also studied (Beck & Shtern, 2017; Garber & Hazan, 2013a;b).

A.2 WOLFE'S METHOD FOR MINIMUM NORM POINT

Algorithm 3 Wolfe's Method for Minimum Norm Point

Initialize $\boldsymbol{x} \in \mathcal{P}$, active set $\mathcal{S} = [\boldsymbol{x}]$ and weight $\boldsymbol{\lambda} = [1]$.
Output: $\boldsymbol{x} \in \mathcal{P}$ that has the minimum Euclidean norm.

while true **do** // Major cycle
  $\boldsymbol{s} \leftarrow \texttt{Oracle}(\boldsymbol{x})$ // Potential improving point
  if $||\boldsymbol{x}||^2 \leq \boldsymbol{x}^T \boldsymbol{s} + \epsilon$ **then** break
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{s}\}$
  while true **do** // Minor cycle
    $\boldsymbol{y}, \boldsymbol{\alpha} \leftarrow \texttt{AffineMinimizer}(\mathcal{S})$ // $\boldsymbol{y} = \arg\min_{\boldsymbol{s} \in \texttt{aff}(\mathcal{S})} ||\boldsymbol{s}||_2$
    if $\alpha_{\boldsymbol{s}} > 0$ for all $\boldsymbol{s}$ **then** break // $\boldsymbol{y} \in \texttt{conv}(\mathcal{S})$
    // If $\boldsymbol{y} \notin \texttt{conv}(\mathcal{S})$, then update $\boldsymbol{y}$ to the intersection of $\texttt{conv}(\mathcal{S})$ and segment joining $\boldsymbol{x}$ and $\boldsymbol{y}$. Then remove points in $\mathcal{S}$ unnecessary for describing $\boldsymbol{y}$.
    $\theta \leftarrow \min_{i:\alpha_i \leq 0} \frac{\lambda_i}{\lambda_i - \alpha_i}$ // Recall $\boldsymbol{\lambda}$ satisfies $\boldsymbol{x} = \sum_{\boldsymbol{s} \in \mathcal{S}} \lambda_{\boldsymbol{s}} \boldsymbol{s}$
    $\boldsymbol{y} \leftarrow \theta \boldsymbol{y} + (1 - \theta)\boldsymbol{x}, \lambda_i = \theta \alpha_i + (1 - \theta)\lambda_i$
    $\mathcal{S} \leftarrow \{\boldsymbol{s}_i | \boldsymbol{s}_i \in \mathcal{S} \text{ and } \lambda_i > 0\}$
  end while
  Update $\boldsymbol{x} = \boldsymbol{y}$ and $\boldsymbol{\lambda} = \boldsymbol{\alpha}$.
end while
return $\boldsymbol{x}$

Wolfe's method is an iterative algorithm for finding the point with minimum Euclidean norm in a polytope, which is defined as the convex hull of a set of finitely many points.

The Wolfe's method consists of a finite number of major cycles, each of which consists of a finite number of minor cycles. At the start of each major cycle, let $H(\boldsymbol{x}) := \{\boldsymbol{y}^T \boldsymbol{x} = \boldsymbol{x}^{\boldsymbol{x}}\}$ be the hyperplane defined by $\boldsymbol{x}$. If $H(\boldsymbol{x})$ separates the polytope from the origin, then the major cycle is terminated. Otherwise, it invokes an oracle to find any point on the near side of the hyperplane. The point is then added into the active set $\mathcal{S}$, and starts a minor cycle.

In a minor cycle, let $\boldsymbol{y}$ be the point of smallest norm in of the affine hull $\texttt{aff}(\mathcal{S})$. If $\boldsymbol{y}$ is in the relative interior of the convex hull $\texttt{conv}(\mathcal{S})$, then $\boldsymbol{x}$ is updated to $\boldsymbol{y}$ and the minor cycle is terminated. Otherwise, $\boldsymbol{y}$ is updated to the nearest point to $\boldsymbol{y}$ on the line segment $\texttt{conv}(\mathcal{S}) \cap [\boldsymbol{x}, \boldsymbol{y}]$. Thus $\boldsymbol{y}$ is updated to a boundary point of $\texttt{conv}(\mathcal{S})$, and any point that is not on the face of $\texttt{conv}(\mathcal{S})$ in which $\boldsymbol{y}$ lies is deleted. The minor cycles are executed repeatedly until $\mathcal{S}$ becomes a *corral*, that is, a set

whose affine minimizer lies inside its convex hull. Since a set of one point is always a corral, the minor cycles is terminated after a finite number of runs.

B PROOF OF THEOREM 4.1

Theorem 4.1 (Approximation Error Strictly Decreases)**.** *For any non-terminal step t, we have* $\texttt{err}(\mu^{t+1}) < \texttt{err}(\mu^t)$. *That is, the measurement vector of μ^t found by Algorithm 1 gets strictly closer to the convex set Ω after major cycle step.*

Proof. If the current step is a major cycle with no minor cycle, then $\boldsymbol{x}^{t+1}$ is the affine minimizer of $\texttt{aff}(\mathcal{S} \cup \{\boldsymbol{s}^t\})$ with respect to $\boldsymbol{\omega}^t$. Then the affine minimizer property implies $(\boldsymbol{s}^t - \boldsymbol{x}^{t+1})(\boldsymbol{x}^{t+1} - \omega^t) = 0$. Since iteration does not terminate at step t, we have $(\boldsymbol{x}^t - \boldsymbol{\omega}^t)^T(\boldsymbol{x}^t - \boldsymbol{s}^t) > 0$, and therefore $\boldsymbol{x}^{t+1}$ not equal to $\boldsymbol{x}^t$. Then $\boldsymbol{x}^{t+1}$ is the unique affine minimizer implies $f_\Omega(\boldsymbol{x}^{t+1}) = \min_{\omega \in \Omega} ||\boldsymbol{x}^{t+1} - \omega||^2 \leq ||\boldsymbol{x}^{t+1} - \omega^t||^2 < ||\boldsymbol{x}^t - \omega^t||^2 = f_\Omega(\boldsymbol{x}^t)$.

Otherwise the current step contains one or more minor cycles. In this case, we show that the first minor cycle strictly reduces the approximation error, and the (possibly) following minor cycles cannot increase it. For the first minor cycle, the affine minimizer $\boldsymbol{y}^0$ of $\texttt{aff}(\mathcal{S} \cup \{\boldsymbol{s}^t\})$ with respect to $\boldsymbol{\omega}^t$ is outside $\texttt{conv}(\mathcal{S} \cup \{\boldsymbol{s}^t\})$. Let $\boldsymbol{z} = \theta \boldsymbol{y}^0 + (1-\theta)\boldsymbol{x}^t$ be the intersection of $\texttt{conv}(\mathcal{S} \cup \{\boldsymbol{s}^t\})$ and segment joining $\boldsymbol{x}$ and $\boldsymbol{y}$. Let $\mathcal{V}^0 := \mathcal{S}^t$ and $\mathcal{V}^i$ denotes the active set after the i-th minor cycle. Then since $\boldsymbol{y}^1$ is the affine minimizer of $\mathcal{V}^1$ with respect to ω^t, we have

$$||\boldsymbol{z} - \omega^t|| = ||\theta \boldsymbol{y}^0 + (1-\theta)\boldsymbol{x}^t - \omega^t|| \leq \theta||\boldsymbol{y}^0 - \omega^t|| + (1-\theta)||\boldsymbol{x}^t - \omega^t|| < ||\boldsymbol{x}^t - \omega^t||, \quad (14)$$

where the second step uses the triangle inequality and the last step follows since the segment $\boldsymbol{x}^t\boldsymbol{y}^0$ intersects the interior of $\texttt{conv}(\mathcal{S} \cup \{\boldsymbol{s}^t\})$, and the distance to ω^t strictly decreases along this segment. Therefore the point $\boldsymbol{z}$ found by first minor cycle satisfies

$$f_\Omega(\boldsymbol{z}) = \min_{\omega \in \Omega} ||\boldsymbol{z} - \omega||^2 \leq ||\boldsymbol{z} - \omega^t||^2 < ||\boldsymbol{x}^t - \omega^t|| = f_\Omega(\boldsymbol{x}^t). \quad (15)$$

Hence $h(\boldsymbol{y}^1) < h(\boldsymbol{x}^t)$, and the first minor cycle strictly decreases the approximation error. By a similar argument, in subsequent minor cycles the approximation error cannot be increased. However, after the first minor cycle, the iterating point may already at the intersection point and the strict inequality in last step of Eq. 14 need to be replaced by non-strict inequality.

Therefore any major cycle either finds an improving point and continue, or enter minor cycles where the first minor cycle finds an improving point, and the subsequent minor cycles does not increase the distance. Adding both side of $f_\Omega(\boldsymbol{x}^{t+1}) < f_\Omega(\boldsymbol{x}^t)$ by $f_\Omega(\boldsymbol{x}^*)$ and we have the approximation error $h(\boldsymbol{x}^{t+1}) < h(\boldsymbol{x}^t)$ strictly decreases. □

C PROOF OF THEOREM 4.2

We first prove the Theorem 4.2, using Lemma 4.3 and Lemma 4.4. Then present the proof of the lemmas.

Theorem 4.2 (Convergence in Approximation Error)**.** *For $t \geq 1$, the mixed policy μ^t found by Algorithm 1 satisfies*

$$\texttt{err}(\mu^t) \leq 16Q^2/(t+2). \quad (16)$$

where $Q := \max_{\mu \in \Delta(\mathcal{U})} ||\mathbf{c}(\mu)||$ is the maximum norm of a measurement vector.

Proof. Since Lemma 4.4 shows that drop steps are no more than half of total major cycle steps, and Theorem 4.1 guarantees these drop steps reducing the approximation error, we can safely skip these step, and re-index the step numbers to include non-drop steps only using k.

For these non-drop steps, we claim that $\texttt{err}(\mu^k) \leq 8Q^2/(k+1)$. Using Lemma 4.3, we prove the convergence rate using induction. We first bound the error of any $\texttt{err}(\mu^k)$. For any $k \geq 1$

$$\begin{aligned}
\texttt{err}(\mu^k) &= \texttt{dist}(\boldsymbol{c}(\mu^k), \Omega) - \texttt{dist}(\boldsymbol{c}(\mu^*), \Omega) && (17)\\
&= 1/2||\boldsymbol{c}(\mu^k) - \texttt{Proj}_\Omega(\boldsymbol{c}(\mu^k))||^2 - 1/2||\boldsymbol{c}(\mu^*) - \texttt{Proj}_\Omega(\boldsymbol{c}(\mu^*))||^2 && (18)\\
&\leq 1/2(||\boldsymbol{c}(\mu^k)||^2 + ||\texttt{Proj}_\Omega(\boldsymbol{c}(\mu^k))||^2 - ||\boldsymbol{c}(\mu^*)||^2 - ||\texttt{Proj}_\Omega(\boldsymbol{c}(\mu^*))||^2) && (19)\\
&\leq ||\boldsymbol{c}(\mu^k)||^2 - ||\boldsymbol{c}(\mu^*)||^2 && (20)\\
&\leq ||\boldsymbol{c}(\mu^k)||^2 && (21)\\
&\leq Q^2, && (22)
\end{aligned}$$

where Eq. 18 uses the definition of our squared Euclidean distance function. Eq. 19 follows from triangle inequality, and Eq. 20 is by the contractive property of the Euclidean distance.

When $k = 1$, the Eq. 22 established the based case. Now for $k \geq 1$, assume that $\texttt{err}(\mu^k) \leq 8Q^2/(k+1)$ for $k \geq 1$, then Lemma 4.3 gives $\texttt{err}(\mu^{k+1}) \leq \texttt{err}(\mu^k) - \texttt{err}^2(\mu^k)/8Q^2$. Since the quadratic function of the right hand side is monotonically increasing on $(-\infty, 4Q^2]$, using the inductive hypothesis

$$\texttt{err}(\mu^{k+1}) \leq \texttt{err}(\mu^k) - \texttt{err}^2(\mu^k)/8Q^2 \leq 8Q^2/(k+1) - 8Q^2/(k+1)^2 \leq Q^2/(k+2) \quad (23)$$

Then since for t steps of major cycle steps, the number of non-drop steps $k > t/2$, we conclude that $\texttt{err}(\mu^t) \leq 16Q^2/(t+2)$.

□

Then we prove the lemmas.

Lemma 4.3. *For a non-drop step, we have* $\texttt{err}(\mu^t) - \texttt{err}(\mu^{t+1}) \geq \texttt{err}^2(\mu^t)/8Q^2$.

Proof. The non-drop step contains either no minor cycle or one minor cycle. We first consider the no minor cycle case.

If a major cycle contains no minor cycle, then $\boldsymbol{x}^{t+1}$ is the affine minimizer of the $\mathcal{S} \cup \{\boldsymbol{s}^t\}$.

$$\begin{aligned}
\texttt{err}(\mu^t) - \texttt{err}(\mu^{t+1}) &= \texttt{dist}(\boldsymbol{x}^t, \Omega) - \texttt{dist}(\boldsymbol{x}^{t+1}, \Omega) && (24)\\
&= 1/2(||\boldsymbol{x}^t - \boldsymbol{\omega}^t||^2 - \min_{\boldsymbol{\omega}\in\Omega} ||\boldsymbol{x}^{t+1} - \boldsymbol{\omega}||^2) && (25)\\
&\geq 1/2(||\boldsymbol{x}^t - \boldsymbol{\omega}^t||^2 - ||\boldsymbol{x}^{t+1} - \boldsymbol{\omega}^t||^2) && (26)\\
&= 1/2(||\boldsymbol{x}^t - \boldsymbol{\omega}^t||^2 + ||\boldsymbol{x}^{t+1} - \boldsymbol{\omega}^t||^2 - 2||\boldsymbol{x}^{t+1} - \boldsymbol{\omega}^t||^2) && (27)\\
&= 1/2(||\boldsymbol{x}^t - \boldsymbol{\omega}^t||^2 + ||\boldsymbol{x}^{t+1} - \boldsymbol{\omega}^t||^2 - 2(\boldsymbol{x}^t - \boldsymbol{\omega}^t)^T(\boldsymbol{x}^{t+1} - \boldsymbol{\omega}^t)) && (28)\\
&= 1/2(||\boldsymbol{x}^t - \boldsymbol{x}^{t+1}||^2), && (29)
\end{aligned}$$

where the equation (28) follows from the affine minimizer property Eq. (9). For $||\boldsymbol{x}^t - \boldsymbol{x}^{t+1}||$ in the last equation, and $\forall \boldsymbol{q} \in \texttt{aff}(\mathcal{S} \cup \{\boldsymbol{s}^t\})$, we have

$$\begin{aligned}
||\boldsymbol{x}^t - \boldsymbol{x}^{t+1}|| &\geq ||\boldsymbol{x}^t - \boldsymbol{x}^{t+1}|| \frac{||\boldsymbol{x}^t|| + ||\boldsymbol{q}||}{2Q} && (\text{ Definition of } Q) && (30)\\
&\geq ||\boldsymbol{x}^t - \boldsymbol{x}^{t+1}|| \frac{||\boldsymbol{x}^t - \boldsymbol{q}||}{2Q} && (\text{ Triangle inequality}) && (31)\\
&\geq \frac{1}{2Q}(\boldsymbol{x}^t - \boldsymbol{x}^{t+1})(\boldsymbol{x}^t - \boldsymbol{q}) && (\text{ Cauchy-Schwarz inequality}) && (32)\\
&= \frac{1}{2Q}(\boldsymbol{x}^t - \boldsymbol{\omega}^t)(\boldsymbol{x}^t - \boldsymbol{q}) && (\text{ Affine minimizer property}). && (33)
\end{aligned}$$

Then it suffices to show that $(\boldsymbol{x}^t - \boldsymbol{\omega}^t)(\boldsymbol{x}^t - \boldsymbol{q}) \geq \texttt{err}(\mu^t)$.

Since Ω is a convex set, the squared Euclidean distance function $\texttt{dist}(\boldsymbol{x}, \Omega)$ is convex for $\boldsymbol{x}$, which implies

$$\texttt{dist}(\boldsymbol{x}^t, \Omega) + (\boldsymbol{q} - \boldsymbol{x}^t)\nabla\texttt{dist}(\boldsymbol{x}^t, \Omega) \leq \texttt{dist}(\boldsymbol{q}, \Omega). \tag{34}$$

Putting in $\nabla\texttt{dist}(\boldsymbol{x}^t, \Omega) = (\boldsymbol{x}^t - \texttt{Proj}_\Omega(\boldsymbol{x}^t)) = (\boldsymbol{x}^t - \boldsymbol{\omega}^t)$, we get $(\boldsymbol{x}^t - \boldsymbol{\omega}^t)(\boldsymbol{x}^t - \boldsymbol{q}) \geq \texttt{err}(\mu^t)$, which together with Eq. 29 and Eq. 33 concludes that for non-drop step with no minor cycles, we have $\texttt{err}(\mu^t) - \texttt{err}(\mu^{t+1}) \geq \texttt{err}^2(\mu^t)/8Q^2$.

For non-drop step with one minor cycle, we use the Theorem 6 of (Chakrabarty et al., 2014). By a linear translation of adding all points with $-\omega^t$, it gives

$$||\boldsymbol{x}^t - \boldsymbol{\omega}^t||^2 - ||\boldsymbol{x}^{t+1} - \boldsymbol{\omega}^t||^2 \geq ((\boldsymbol{x}^t - \boldsymbol{\omega}^t)(\boldsymbol{x}^t - \boldsymbol{q}))^2/8Q^2. \tag{35}$$

Then applying the same argument as Eq. 34, and we finished our proof.

□

Lemma 4.4. *After t major cycle steps of Algorithm 1, the number of drop steps is less than $t/2$.*

Proof. Recall that at the termination of a minor cycle, the size of the active set $|\mathcal{S}_c| \in [1, m]$. Since in each major cycle steps, the size of active set $\mathcal{S}_t$ increases by one, and each drop step reduces the size of $\mathcal{S}_t$ by at least one, the number of drop steps is always less than half of total number of the major cycle steps. □

D Proof of Theorem 4.5

Theorem 4.5 (Memory Complexity Bound)**.** *For an constrained RL problem with m-dimensional measurement vector, in the worst case, a mixed policy needs to randomize among $m + 1$ individual policies to ensure convergence of RL oracles that search for deterministic policies.*

Proof. We give a constructive proof. Consider a m-dimensional vector-valued MDP with a single state, $m + 1$ actions, and $\boldsymbol{c}(a_i) := \boldsymbol{e}_i$ is the unit vector of i-th dimension for $i \in [1, m]$, and $\boldsymbol{c}(a_{m+1}) := \mathbf{0}$, and the episode terminates after 1 steps. The constrained RL problem is to find a policy whose measurement vector lies in the convex set of a single point $\{\mathbf{1}/2m\}$. By linear programming, it is clear that the only feasible mixed deterministic policy is to select a_{m+1} with $1/2$ probability, and the rest m actions with $1/2m$ probability; i.e. the unique feasible policy to this problem has an active set containing $m + 1$ deterministic policies. Therefore any method randomize among less than $m + 1$ individual policies does not ensure convergence when used with RL algorithms searching for deterministic policies. □