A Mean-Field Game Approach to Cloud Resource Management with Function Approximation

Weichao Mao
University of Illinois Urbana-Champaign
weichao2@illinois.edu

Haoran Qiu
University of Illinois Urbana-Champaign
haoranq4@illinois.edu

Chen Wang
IBM Research
chen.wang1@ibm.com

Hubertus Franke
IBM Research
frankeh@us.ibm.com

Zbigniew Kalbarczyk
University of Illinois Urbana-Champaign
kalbarcz@illinois.edu

Ravishankar K. Iyer
University of Illinois Urbana-Champaign
rkiyer@illinois.edu

Tamer Başar
University of Illinois Urbana-Champaign
basar1@illinois.edu

Abstract

Reinforcement learning (RL) has gained increasing popularity for resource management in cloud services such as serverless computing. As self-interested users compete for shared resources in a cluster, the multi-tenancy nature of serverless platforms necessitates multi-agent reinforcement learning (MARL) solutions, which often suffer from severe scalability issues. In this paper, we propose a mean-field game (MFG) approach to cloud resource management that is scalable to a large number of users and applications and incorporates function approximation to deal with the large state-action spaces in real-world serverless platforms. Specifically, we present an online natural actor-critic algorithm for learning in MFGs compatible with various forms of function approximation. We theoretically establish its finite-time convergence to the regularized Nash equilibrium under linear function approximation and softmax parameterization. We further implement our algorithm using both linear and neural-network function approximations, and evaluate our solution on an open-source serverless platform, OpenWhisk, with real-world workloads from production traces. Experimental results demonstrate that our approach is scalable to a large number of users and significantly outperforms various baselines in terms of function latency and resource utilization efficiency.

1 Introduction

Serverless computing[1] is an emerging cloud computing paradigm that allows users to develop and run their applications without worrying about the configurations of the infrastructure (containers or virtual machines) [7, 33]. In contrast to traditional cloud computing, serverless users are only charged by the function execution time and their operational cost is reduced since the cloud provider takes care of the resource management (e.g., serverless function provisioning and scheduling) on their behalf. The tension between cloud providers and users incurs a more complex resource management problem: The platform needs to meet various user-defined serverless function quality-of-service (QoS) guarantees while the provider aims to keep the cluster resource utilization at a high level in the face of elastic and bursty function requests. Numerous heuristics-based solutions (e.g., [63, 66, 68, 75]) have been

[1] We focus on Function-as-a-Service (FaaS), which is the most common serverless service in cloud computing.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

utilized, yet they inevitably rely on extensive system-specific domain knowledge and painstaking tuning for specific workloads.

Recently, reinforcement learning (RL) based approaches have attracted increasing attention for dynamic resource management as RL helps automatically adapt to a specific user workload. Each function is managed by an RL agent, leaving human operators out of the loop. The fact that large-scale cloud computing platforms have many users with independent (and sometimes conflicting) requirements adds another layer of complication for serverless function resource management. Since functions from multiple self-interested users coexist and compete for shared resources in a cluster, cloud providers further need to take into account the potential strategic behavior of the users. The multi-agent reinforcement learning (MARL) paradigm [39] is especially well-suited here, as it naturally integrates game-theoretical thinking into the sequential decision-making problems of multi-agent systems. However, a well-known challenge in multi-agent reinforcement learning is *scalability*, as many MARL algorithms suffer from exponential computation & sample complexity in the number of agents, a phenomenon known as "the curse of multiagents" [32, 64, 41, 19, 42, 18]. The scalability bottleneck needs to be properly addressed before applying MARL to real-world cloud serverless platforms, which typically involve a large number of users and functions in a cluster [1].

To circumvent the scalability challenge, we resort to the mean-field approximation [37, 30] and propose a scalable mean-field game (MFG) approach to cloud resource management. The underlying principle of MFGs is to approximate the finite-agent game with an infinite-population limit, where each agent's influence on the overall system becomes infinitesimal. In an MFG with infinite homogeneous agents, the collective behavior of all the agents can be effectively summarized as a population distribution, which is usually specified as the empirical distribution of the agents' states. Such an approximation leads to a tractable solution to the otherwise challenging MARL problem, as each agent no longer needs to keep track of the historical behavior of every other agent to reason about their internal information, which typically incurs a prohibitive combinatorial complexity. Existing theory [53] also shows that the approximation scheme does not lose much of optimality when applying the policies learned from the infinite-agent game back to the original finite-population game.

However, the convergence of learning algorithms in MFGs has been chiefly established either in the tabular setting with small state & action spaces [65, 28], or in the linear-quadratic case with structural assumptions on the system dynamics [23, 72]. Few results have considered *function approximation* in mean-field games with large (and even infinite) state & action spaces, which is the typical case for many real-world application scenarios including cloud computing. In this paper, motivated by the large state & action spaces in cloud resource management problems, we make an initial attempt to understand the effects of function approximation in MFGs with generic system dynamics. While we believe that our proposed methodology is general enough to be applicable to a wide range of scenarios, we demonstrate the effectiveness of our approach in a real-world cloud resource management problem as an important application domain.

Contributions. Our contribution is threefold: (1) Algorithmically, we propose a natural actor-critic (NAC) learning paradigm for MFGs that is compatible with various forms of function approximations. Our method is particularly practical due to an *online* property, in the sense that it need not fix the mean-field state to calculate the best response at each iteration, but instead let the mean-field naturally evolve as the agents learn. (2) Theoretically, we establish the finite-time convergence of NAC with linear function approximation and softmax parameterization as a critical stepping stone. We prove that the resulting algorithm converges to the regularized Nash equilibrium (NE) of the MFG at an $\widetilde{O}(T^{-1/5})$ rate. (3) Empirically, we evaluate NAC on classic MFG benchmarks and demonstrate its convergence behavior. As an important motivating example and test bench, we also incorporate a practical variant of NAC (with both linear and neural-network function approximations) into the resource management module of an open-source serverless platform, OpenWhisk [22]. Our experimental results on real-world production workloads show that NAC is scalable to a large number of agents in serverless resource management and significantly outperforms various baselines in terms of both function latency and resource utilization.

Related Work. Mean-field games have been introduced by [37] and [30] to study continuous-time differential games with infinite identical agents. For discrete-time MFGs, the existence and/or uniqueness of the Nash equilibrium have been studied in [26, 69, 45, 53]. The mean-field regime has been investigated under various settings, including games with linear-quadratic structures [9, 71], major and minor agents [47], partial observability [54], risk-sensitivity [55], and so on. When the

environment dynamics are unknown, learning-based algorithms are commonly used to learn the NE from data [78, 72, 23, 28, 20, 49, 5, 27, 16], but most of these works rely on a "double-loop" fixed-point iterative process that can be time- & sample-inefficient. Single-loop methods like ours have been studied in [65, 48, 76], but they do not consider function approximation to deal with large state and action spaces, and [65] and [48] only yield asymptotic convergence guarantees. On the application side, learning algorithms in MFGs have been applied to economics [6] and animal behavior simulation [50], among others, while in this paper, we identify that large-scale cloud resource management is also an ideal application domain.

Our work is also related to policy optimization in single-agent RL, especially natural policy gradient (NPG) methods [35]. For the tabular setting, the (global) convergence of NPG has been established in a series of works [62, 2, 43, 77, 14]. In particular, [2] has proved that unregularized NPG with softmax parameterization achieves an $O(1/T)$ convergence rate, while [14] has further strengthened the result by showing linear convergence of NPG when equipped with entropy regularization. In the regime of function approximation, [2] has shown that NPG with linear function approximation and softmax parameterization attains a $O(1/\sqrt{T})$ convergence rate subject to some function approximation error. [13] has established a linear convergence when further exploiting entropy regularization, which is most related to our setting. In particular, a standard "double-loop" mean-field approach will lead to a setting very similar to that of [13] because by fixing the population distribution, the learning agent effectively faces a single-agent problem. However, since such double-loop solutions are hardly practical, we instead consider an online setting where the environment simultaneously evolves as the agents update their policies. The environmental non-stationarity adds another layer of complexity and makes [13] not directly applicable. Going beyond linear function approximation, [74] has investigated NPG with over-parameterized two-layer neural networks, a more powerful approximation scheme that is not theoretically pursued but only empirically evaluated in our work. It is also worth mentioning that many successful empirical algorithms, such as TRPO [57] and PPO [58], can be considered as natural variants of NPG.

RL-based resource management approaches [8, 36, 40, 51, 56, 80] have been recently proposed to achieve efficient resource utilization and application quality-of-service (QoS), which outperform baseline rule-based methods. For instance, FIRM [51] is an RL-based resource management framework for cloud microservices to tackle the under-utilization issue and QoS violations. Schuler et al. [56] proposed a Q-learning-based autoscaler that decides the horizontal concurrency for a serverless function. FaaSRank [80] is an RL-based serverless function scheduler that uses PPO [58] to assign function requests to available servers, yet both [56] and [80] only consider the objective of minimizing the function latency. Finally, existing works only consider single-agent RL approaches being trained and evaluated in an isolated environment, while the cloud is multi-tenant environment by nature.

2 Preliminaries

We first consider a classic Markov game with N agents. Each agent has a state space $\mathcal{S}$ and an action space $\mathcal{A}$. At each time step t, the state of agent $i \in [N]$ is denoted by $s_t^i \in \mathcal{S}$, and the agent takes an action $a_t^i \in \mathcal{A}$ according to a certain policy. Given the current state profile $\boldsymbol{s}_t = (s_t^1, \ldots, s_t^N)$, agent i receives a reward determined by a reward function $\tilde{r}^i(\boldsymbol{s}_t, a_t^i)$, and transitions to a new state $s_{t+1}^i \sim \tilde{P}^i(\cdot \mid \boldsymbol{s}_t, a_t^i)$ according to a transition function $\tilde{P}^i$. The goal of each agent is to find a policy that maximizes its expected cumulative reward over time. When N is large, learning in such an N-agent Markov game with generic reward structure is known to be notoriously hard [64, 18]. Mean-field games [37, 30], on the other hand, can be viewed as an infinite-population limit ($N \to \infty$) of the finite-agent game, and provide a tractable approximation to the otherwise challenging problem.

A discrete-time mean-field game (MFG) considers an infinite-number of identical agents [37, 30]. The collective behavior of all the agents is described by population distributon $\mu_t \in \Delta(\mathcal{S})$, also termed a mean-field state, which in practice can be interpreted as the limit of the empirical state distribution, i.e., $\mu_t = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \delta_{s_t^i}$, where $\delta_s \in \Delta(\mathcal{S})$ is the Dirac measure at s. Due to the homogeneity of the agents, we focus on a single representative agent. At each time t, the (representative) agent's state is denoted by $s_t \in \mathcal{S}$, and the mean-field state μ_t describes the probability distribution of s_t. Upon taking an action $a_t \in \mathcal{A}$ at s_t, the agent receives a reward $r(s_t, a_t, \mu_t)$, and transitions to a new state $s_{t+1} \sim P(\cdot \mid s_t, a_t, \mu_t)$, where $r : \mathcal{S} \times \mathcal{A} \times \Delta(\mathcal{S}) \to [0, 1]$ is the reward function and $P : \mathcal{S} \times \mathcal{A} \times \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ is the state transition function. A (Markov) policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$

for the agent is a mapping from the state space to a distribution over the action space, and we use Π to denote the set of all Markov policies. Given a population distribution flow $\boldsymbol{\mu} = (\mu_t)_{t\geq 0}$, we define the *value function* of a policy π as $V_{\boldsymbol{\mu}}^{\pi}(s) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t, \mu_t) \mid s_0 = s\right]$, where $a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim P(\cdot \mid s_t, a_t, \mu_t)$, and $\gamma \in (0,1)$ is the discount factor. In the special case of a time-invariant mean-field, i.e., $\mu_t = \mu, \forall t \geq 0$, we slightly abuse the notation and write $V_{\boldsymbol{\mu}}^{\pi}(s)$ as $V_{\mu}^{\pi}(s)$. For an initial state distribution $\rho \in \Delta(\mathcal{S})$, we define the (discounted) state *visitation distribution* as $d_{\mu}^{\pi}(s) \stackrel{\text{def}}{=} (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}(s_t = s \mid s_0 \sim \rho)$, where $\mathbb{P}(s_t = s | s_0 \sim \rho)$ is the probability that the state s is visited at the t-th time step under policy π and mean-field μ.

Entropy Regularization: We further consider an entropy-regularized value function by augmenting the standard reward objective with an entropy term of the policy. For a fixed mean-field state μ, define

$$V_{\mu}^{\pi,\lambda}(s) \stackrel{\text{def}}{=} V_{\mu}^{\pi}(s) + \lambda H_{\mu}^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \left(r(s_t, a_t, \mu) - \lambda \log \pi(a_t \mid s_t)\right) \mid s_0 = s\right],$$

where $\lambda > 0$ is a parameter that controls the level of regularization, $H_{\mu}^{\pi}(s) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \mathcal{H}(\pi(\cdot|s_t)) \mid s_0 = s\right]$, and $\mathcal{H}(\pi(\cdot|s)) = -\sum_{a\in\mathcal{A}}\pi(a|s)\log\pi(a|s)$ is the Shannon entropy. Entropy regularization has been commonly used to encourage exploration and avoid premature convergence to sub-optimal near-deterministic policies [29, 3]. In mean-field games, [5] has also shown that with regularization, the uniqueness of NE is guaranteed under milder assumptions than the unregularized case. We define the soft Q-function and shifted Q-function, respectively, as

$$Q_{\mu}^{\pi,\lambda}(s,a) \stackrel{\text{def}}{=} r(s,a,\mu) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a,\mu)}\left[V_{\mu}^{\pi,\lambda}(s')\right], \text{ and } q_{\mu}^{\pi,\lambda}(s,a) = Q_{\mu}^{\pi,\lambda}(s,a) - \lambda\log\pi(a|s),$$

which are related to $V_{\mu}^{\pi,\lambda}$ in the sense that $V_{\mu}^{\pi,\lambda}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[q_{\mu}^{\pi,\lambda}(s,a)]$. For a distribution $\rho \in \Delta(\mathcal{S})$, with a slight abuse of notation, we write $V_{\mu}^{\pi,\lambda}(\rho) = \sum_{s\in\mathcal{S}}\rho(s)V_{\mu}^{\pi,\lambda}(s)$.

Policy Parameterization: We consider parametric policy classes. For each state-action pair (s,a), suppose there exists a d-dimensional feature mapping $\phi_{s,a} \in \mathbb{R}^d$, such that $\|\phi_{s,a}\|_2 \leq 1, \forall s \in \mathcal{S}, a \in \mathcal{A}$. A commonly used policy class is softmax parameterization of the form

$$\widetilde{\Pi} = \left\{\pi_\theta(a|s) = \frac{\exp\left(f_\theta(s,a)\right)}{\sum_{a'\in\mathcal{A}}\exp\left(f_\theta(s,a')\right)} : \theta \in \mathbb{R}^d\right\},$$

where θ is a d-dimensional vector that parameterizes the policy, and f_θ is a differentiable function that can be typically instantiated as a linear function or a neural network. In Sections 3 and 4, we focus on "log-linear" policies that use linear function approximation, where f_θ takes the specific linear form of $f_\theta(s,a) = \theta^\top\phi_{s,a}, \forall (s,a) \in \mathcal{S}\times\mathcal{A}$. We will also instantiate f_θ using neural networks later in Section 5. Function approximation helps deal with large state and action spaces, as the feature dimension is usually much smaller, i.e., $d \ll |\mathcal{S}||\mathcal{A}|$, where $|\mathcal{S}|$ can even be infinite in practice. It is worth noting that a parametric policy class may not contain all Markov policies, i.e., $\widetilde{\Pi} \subset \Pi$. Hence, we seek to do as well as the best policy in this class and obtain agnostic results.

Single-agent Policy Optimization. Given a fixed mean-field state μ, the representative agent faces a single-agent policy optimization problem: $\max_{\pi\in\Pi} V_{\mu}^{\pi,\lambda}(s)$, which is equivalent to finding the (entropy-regularized) optimal policy for a single-agent Markov decision process (MDP) induced by μ. It has been shown that the optimal policy is unique whenever $\lambda > 0$ [25]. Hence, we can use $\pi_{\mu}^{\star,\lambda}$ to denote the (unique) optimal solution to the optimization problem, and define a mapping $\Gamma_1^{\lambda} : \Delta(\mathcal{S}) \to \Pi$ such that $\Gamma_1^{\lambda}(\mu) = \pi_{\mu}^{\star,\lambda}$. We refer to Γ_1^{λ} as the *policy optimization operator*, which maps a mean-field state μ to the optimal policy $\pi_{\mu}^{\star,\lambda}$ of the induced MDP.

Mean-field Dynamics. Since all agents follow the same policy π, we can define another mapping $\Gamma_2 : \Pi \times \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ to describe the evolution of the mean-field. Specifically, the *mean-field dynamics operator* Γ_2 is defined by $\Gamma_2(\pi,\mu) = \mu^+$, where

$$\mu^+(\cdot) = \int_{\mathcal{S}\times\mathcal{A}} P(\cdot|s,a,\mu)\mu(s)\pi(a|s)\,\mathrm{d}a\,\mathrm{d}s.$$

Intuitively, Γ_2 characterizes the next mean-field state, given the current mean-field state and the current policy adopted by all the agents. For notational convenience, we further introduce a composite mapping $\Lambda^{\lambda} : \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ as $\Lambda^{\lambda}(\mu) = \Gamma_2(\Gamma_1^{\lambda}(\mu), \mu)$, which simply combines Γ_1^{λ} and Γ_2.

Mean-field Equilibrium. In the following, we introduce the main learning objective of the paper.

Algorithm 1: Natural Actor-Critic for MFGs with Linear Function Approximation

Input: Initial mean-field state μ_0, and regularization level parameter λ;
Initialize $\theta_0 \leftarrow \mathbf{0}$;
for *iteration* $t \leftarrow 0$ *to* T **do**
  Policy evaluation: Calculate an $\varepsilon_{\text{critic}}$-accurate estimate $\hat{q}_t^\lambda$ of $q_{\mu_t}^{\pi_t,\lambda}$ that satisfies Equation (2) using, e.g. Algorithm 2;
  Gradient estimation: Use $\hat{q}_t^\lambda$ to calculate an $\varepsilon_{\text{actor}}$-accurate estimate $\hat{w}_t$ of the gradient (using, e.g., Algorithm 3) such that $\|\hat{w}_t\|_2 \leq R$ for some $R > 0$, and Equation (3) holds;
  Mean-field update: $\mu_{t+1} \leftarrow (1-\beta_t)\mu_t + \beta_t \Gamma_2(\pi_t, \mu_t)$, where $\beta_t = O(T^{-4/5})$;
  Policy update: $\theta_{t+1} \leftarrow \theta_t + \eta_t g_t$, where $g_t = \hat{w}_t - \lambda\theta_t$, and $\eta_t = O(T^{-2/5})/\lambda$;

Definition 1. *A policy-population pair* $(\pi^\star, \mu^\star) \in \Pi \times \Delta(\mathcal{S})$ *is a stationary (time-independent) entropy-regularized Nash equilibrium for the mean-field game if it satisfies: (1) Rationality:* $\pi^\star = \Gamma_1^\lambda(\mu^\star)$, *and (2) consistency:* $\mu^\star = \Gamma_2(\pi^\star, \mu^\star)$.

When $\lambda = 0$, the above definition reduces to that of the standard (unregularized) NE in MFGs [53, 28]. For $\lambda > 0$, the regularized NE $(\pi^\star, \mu^\star)$ also serves as a good approximation of the unregularized one, as characterized by the following error bound [14] (established using the fact that $H_\mu^\pi(s) \leq \frac{\log|\mathcal{A}|}{1-\gamma}$):

$$V_{\mu^\star}^{\pi^\star}(\rho) \leq \max_{\pi\in\Pi} V_{\mu^\star}^{\pi}(\rho) \leq V_{\mu^\star}^{\pi^\star}(\rho) + \frac{\lambda \log|\mathcal{A}|}{1-\gamma}. \tag{1}$$

When the composite mapping Λ^λ is a contraction, one can show (using a standard Banach-fixed point theorem) that the regularized NE exists and is unique [5]. Accordingly, the policy-population pairs $\{(\pi_t, \mu_t) : t \geq 0\}$ given by the simple iterates $\pi_t \leftarrow \Gamma_1^\lambda(\mu_t)$ and $\mu_{t+1} \leftarrow \Gamma_2(\pi_t, \mu_t)$ converge to the regularized NE at a linear rate, assuming that the exact optimal policy can be computed for Γ_1^λ.

3 Natural Actor-Critic for MFGs with Function Approximation

In this section, we present an online natural actor-critic (NAC) algorithm with function approximation to learn the regularized NE of a mean-field game (Algorithm 1). Four major steps are involved: policy evaluation, gradient estimation, mean-field update, and policy update.

For the policy update step, policy gradient methods improve the policy parameter θ by ascending along the direction of the gradient of the policy, i.e., $\nabla_\theta V_\mu^{\pi_\theta,\lambda}(\rho)$. A direct extension of the policy gradient theorem [67] shows that the policy gradient with entropy-regularization can be expressed as

$$\nabla_\theta V_\mu^{\pi_\theta,\lambda}(\rho) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a|s) q_\mu^{\pi_\theta,\lambda}(s,a)\right].$$

Compared with the "vanilla" policy gradient method that follows the steepest direction in the parameter space, the natural policy gradient (NPG) approach [35] proceeds along the steepest direction with respect to the Fisher metric. NPG has the advantage of being invariant to the parameterization of the policy [4] and enjoys faster convergence as it follows a more direct path to the optimal solution. For a policy π_θ parameterized by θ, NPG [35] defines a Fisher information matrix under policy π_θ as: $F^\theta = \mathbb{E}_{s\sim d_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a|s) \left(\nabla_\theta \log \pi_\theta(a|s)\right)^\top\right]$. NPG then performs gradient updates along the steepest direction induced by this matrix: $\theta \leftarrow \theta + \eta (F^\theta)^\dagger \nabla_\theta V_\mu^{\pi,\lambda}(\rho)$, where η is the learning rate, and $(F^\theta)^\dagger$ denotes the Moore-Penrose pseudoinverse of F^θ. Leveraging the notion of compatible function approximation, it can be shown that the above update rule can be equivalently expressed as (see [35] for a proof of the unregularized case and [13] for the regularized counterpart) $\theta \leftarrow \theta + \frac{\eta}{1-\gamma} w_\lambda^\theta$, where w_λ^θ is a minimizer of the following regression problem:

$$w_\lambda^\theta \in \operatorname*{argmin}_{w\in\mathbb{R}^d} \bar{L}(w,\theta), \text{ where } \bar{L}(w,\theta) \stackrel{\text{def}}{=} \mathbb{E}_{s\sim d_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\left(w^\top \nabla_\theta \log \pi_\theta(a|s) - q_\mu^{\pi_\theta,\lambda}(s,a)\right)^2\right].$$

For variance reduction purposes, Algorithm 1 further subtracts a baseline from the Q-function and

solves a variant of the regression problem at the t-th iteration instead:

$$w_t \in \operatorname*{argmin}_{w\in\mathbb{R}^d:\|w\|_2\le R} L(w,\theta_t), \text{ where } L(w,\theta) \stackrel{\text{def}}{=} \mathbb{E}_{s\sim d_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\left(w^\top \nabla_\theta \log \pi_\theta(a|s) - A_\mu^{\pi_\theta,\lambda}(s,a)\right)^2\right],$$

where $A_\mu^{\pi,\lambda}(s,a) \stackrel{\text{def}}{=} Q_\mu^{\pi,\lambda}(s,a) - \mathbb{E}_{a'\sim\pi(\cdot|s)}[Q_\mu^{\pi,\lambda}(s,a')]$, and $R>0$ is the gradient clipping radius. Since the target function may not be perfectly represented by a linear function, we use the *function approximation error* $\varepsilon_{\text{approx}}$ to denote the minimal possible error for our parametric class[2]:

$$\varepsilon_{\text{approx}} \stackrel{\text{def}}{=} \sup_{t\ge 0} \min_{w\in\mathbb{R}^d:\|w\|_2\le R} L(w,\theta_t).$$

In practice, the values of $A_\mu^{\pi_\theta,\lambda}$ and w_t in the above regression problem cannot be calculated precisely and need to be computed from a finite number of samples, which further introduce *statistical errors* (or excess risk). In particular, we use the policy evaluation step (Line 4 of Algorithm 1) to compute an approximation $\hat{q}_t^\lambda$ of the shifted Q-function $q_{\mu_t}^{\pi_t,\lambda}$, where $\pi_t \stackrel{\text{def}}{=} \pi_{\theta_t}$ denotes the policy at the t-th iteration of Algorithm 1, and we apply the gradient estimation step (Line 5) to further use $\hat{q}_t^\lambda$ to calculate a gradient estimate $\hat{w}_t$. For ease of presentation, in this section, we will assume that we can obtain the estimated values $\hat{q}_t^\lambda$ and $\hat{w}_t$ from two black-box oracles. In Appendix C, we show that such oracles can be accomplished by standard sample-based estimation techniques. Specifically, we assume for now the existence of a policy evaluation oracle that returns an estimate $\hat{q}_t^\lambda$ such that

$$\mathbb{E}_{s\sim d_{\mu_t}^{\pi_t}, a\sim\pi_t(\cdot|s)}\left[\left(\hat{q}_t^\lambda(s,a) - q_{\mu_t}^{\pi_t,\lambda}(s,a)\right)^2\right] \le \varepsilon_{\text{critic}}, \tag{2}$$

for some critic error $\varepsilon_{\text{critic}} \ge 0$. Let $\hat{Q}_t^\lambda(s,a) = \hat{q}_t^\lambda(s,a) + \lambda \log \pi_t(a|s)$, and $\hat{A}_t^\lambda(s,a) = \hat{Q}_t^\lambda(s,a) - \mathbb{E}_{a\sim\pi_t(\cdot|s)}[\hat{Q}_t^\lambda(s,a')]$. We further assume that a gradient estimation oracle provides an estimate $\hat{w}_t$ that satisfies:

$$\mathbb{E}\left[(\hat{w}_t^\top \nabla \log \pi_t(a|s) - \hat{A}_t^\lambda(s,a))^2\right] - \min_w \mathbb{E}\left[(w^\top \nabla \log \pi_t(a|s) - \hat{A}_t^\lambda(s,a))^2\right] \le \varepsilon_{\text{actor}}, \tag{3}$$

for some actor error $\varepsilon_{\text{actor}} \ge 0$, where the expectation is over $s\sim d_{\mu_t}^{\pi_t}$ and $a\sim\pi_t(\cdot|s)$. $\varepsilon_{\text{critic}}$ and $\varepsilon_{\text{actor}}$ together represent the statistical error that can be driven to 0 when we have more samples, while $\varepsilon_{\text{approx}}$ accounts for the inherent modeling error due to the potential lack of expressiveness of the parametric policy class. We further use $\varepsilon_{\text{total}} = \varepsilon_{\text{approx}} + \varepsilon_{\text{actor}} + \varepsilon_{\text{critic}}$ to sum up all sources of errors.

A final remark on the policy update step is that we adopt the idea of gradient averaging from [13] to encourage exploration. Specifically, the parameter update in Line 7 is effectively $\theta_{t+1} \leftarrow (1-\eta_t\lambda)\theta_t + \eta_t\lambda\cdot\frac{\hat{w}_t}{\lambda}$, which can be viewed as a convex combination of θ_t and $\hat{w}_t/\lambda$. Since our gradient estimation step "clips" the gradient estimate to ensure that $\|\hat{w}_t\|_2 \le R$ for some $R>0$ (Line 5), we can show by induction that the policy parameter is uniformly bounded, i.e., $\|\theta_t\|_2 \le R/\lambda, \forall t\ge 0$. Together with the softmax parameterization of the policy, this condition ensures that our policy always explores the action space with some positive probability; that is, there exists $p_{\min}>0$, such that $\pi_t(a|s)\ge p_{\min}, \forall t\ge 0, (s,a)\in\mathcal{S}\times\mathcal{A}$. Such a property is essential in establishing the convergence of the policy. See Lemma 2 in Appendix A for a formal treatment.

In each iteration of the algorithm, we update the mean-field state (Line 6) as $\mu_{t+1} \leftarrow (1-\beta_t)\mu_t + \beta_t\Gamma_2(\pi_t,\mu_t)$, which can be considered as a "soft" step of mean-field evolution along the direction of $\Gamma_2(\pi_t,\mu_t)$ with a step size β_t. Similar to existing works [65, 28, 72, 76], this step assumes access to a simulator that returns the new population distribution given the current mean-field and policy. In practice, one can approximate the simulator by estimating the new population distribution through randomly sampling the actual states of a large number N of agents, which incurs an estimation error $O(1/\sqrt{N})$ that decays as more agents are sampled [76].

Finally, we emphasize that our algorithm enjoys the additional advantage of being *online* (also termed "learning while playing" in [76]), in the sense that it lets the mean-field simultaneously evolve as the agents play. Specifically, we perform only a single step of policy update for each iteration of mean-field evolution, instead of computing the exact optimal policy with respect to the current mean-field. This is achieved by carefully tuning the step sizes of the policy update ($\eta_t \approx T^{-2/5}$) and mean-field

[2]It has been shown that the approximation error is 0 for the realizable cases such as tabular (finite state-action) MDPs or linear MDPs [31], where the value functions are linear in the given features.

update ($\beta_t \approx T^{-4/5}$) so as to let the policy converge at a faster timescale than the mean-field does. Our online approach is in sharp contrast to the "double-loop" methods [23, 28, 5, 20, 49, 27] that have to fix the mean-field state as an outer loop and use an iterative process as the inner loop to learn an optimal policy with respect to each fixed population distribution, which can be more time consuming and incurs oscillations in the learned policies (as evidenced by our simulations).

4 Convergence Analysis

In this section, we theoretically establish the convergence of the natural actor-critic algorithm with linear function approximation to the regularized Nash equilibrium $(\pi^\star, \mu^\star)$ of the mean-field game.

We start by introducing a few notations. For any $t \geq 0$, we use $\pi_t^\star$ to denote the optimal policy of the representative agent w.r.t the mean-field state μ_t, i.e., $\pi_t^\star = \Gamma_1^\lambda(\mu_t)$. Let $d_t^\star \stackrel{\text{def}}{=} d_{\mu_t}^{\pi_t^\star}$ be the visitation distribution of $\pi_t^\star$ under μ_t, and $d^\star \stackrel{\text{def}}{=} d_{\mu^\star}^{\pi^\star}$ be the one induced by the NE $(\pi^\star, \mu^\star)$. Our first result establishes the improvement of the policy under natural actor-critic updates in terms of the KL divergence. All missing proofs are deferred to Appendix B.

Lemma 1. *(Policy improvement). For any time $t \geq 0$, the policy update $\theta_{t+1} = \theta_t + \eta_t g_t$ with softmax parameterization and linear function approximation leads to the following policy improvement:*

$$\mathbb{E}_{s\sim d_t^\star}\left[KL(\pi_t^\star(\cdot|s)\|\pi_{t+1}(\cdot|s))\right] \leq (1-\eta_t\lambda)\mathbb{E}_{s\sim d_t^\star}\left[KL(\pi_t^\star(\cdot|s))\|\pi_t(\cdot|s)\right] - \eta_t \mathbb{E}_{s\sim d_t^\star}[V_{\mu_t}^{\pi_t,\lambda}(s)] - \eta_t \mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top \nabla_\theta \log \pi_t(a|s) - q_\mu^{\pi_t,\lambda}(s,a)\right] + \frac{1}{2}\eta_t^2 \|g_t\|_2^2 .$$

The proof follows from the performance difference lemma [34] in the regularized case and the smoothness of $\log \pi_\theta(a|s)$. Similar results have also been shown in [13, 76]. To proceed further, we impose the following regularity assumptions on the visitation distribution.

Assumption 1. *There exists a constant $d_0 > 0$, such that for any mean-field states μ and μ', the state visitation distributions under their corresponding optimal policies $\pi^\star$ and $\pi^{\star\prime}$ satisfy $\|d_\mu^{\pi^\star} - d_{\mu'}^{\pi^{\star\prime}}\|_1 \leq d_0 \|\mu - \mu'\|_1$.*

Assumption 2. *(Finite concentrability coefficients). There exist constants $C_1, C_2, C_3 > 0$, such that for any $t \geq 0$ and any encountered mean-field state μ_t,*

$$\sup_{s\in\mathcal{S}} \frac{d_t^\star(s)}{d^\star(s)} \leq C_1, \quad \mathbb{E}_{s\sim d_t^\star}\left[\left|\frac{d^\star(s)}{d_t^\star(s)}\right|^2\right] \leq C_2^2, \quad \textit{and} \quad \mathbb{E}_{s\sim d_{\mu_t}^{\pi_t}}\left[\left|\frac{d_t^\star(s)}{d_{\mu_t}^{\pi_t}(s)}\right|^2\right] \leq C_3^2.$$

Assumption 1 states that the visitation distributions are smooth w.r.t the mean fields, which is consistent with [76] and is reminiscent of the smooth visitation assumption for RL in non-stationary environments [21]. Assumption 2 is a standard assumption in policy optimization [34, 62, 2, 76, 13] that captures the difficulty of strategic exploration by requiring the visitation distributions of certain policies to adequately cover that of an optimal policy. Together with Lemma 1, these assumptions allow us to establish a recursive relationship of $\text{KL}(\pi_t^\star\|\pi_t)$ over time (Lemma 8 in Appendix B).

To characterize the convergence of π_t, we use $D(\pi,\pi') \stackrel{\text{def}}{=} \mathbb{E}_{s\sim d^\star}[\|\pi(\cdot|s) - \pi'(\cdot|s)\|_1]$ as a measure of distance between two policies. The following assumption is standard in the literature [28, 72, 76, 16] that imposes the Lipschitzness of the two operators Γ_1^λ and Γ_2 with respect to the $D(\cdot,\cdot)$ metric.

Assumption 3. *(Lipschitz Operators). There exist constants $d_1, d_2, d_3 > 0$, such that for any policies π, π' and mean-field states μ, μ', it holds that: (1) $D\left(\Gamma_1^\lambda(\mu), \Gamma_1^\lambda(\mu')\right) \leq d_1 \|\mu - \mu'\|_1$, $\|\Gamma_2(\pi,\mu) - \Gamma_2(\pi',\mu)\|_1 \leq d_2 D(\pi,\pi')$, and (2) $\|\Gamma_2(\pi,\mu) - \Gamma_2(\pi,\mu')\|_1 \leq d_3 \|\mu - \mu'\|_1$.*

The first condition states that $\Gamma_1^\lambda(\mu)$ is Lipschitz with respect to the mean-field state, and the second and third conditions stipulate that $\Gamma_2(\pi,\mu)$ is Lipschitz in each of its arguments when fixing the other. Assumption 3 immediately implies that the composite operator Λ^λ is contractive when the Lipschitz constants are small enough (Lemma 6 in Appendix A). We are now ready to state our main theoretical guarantees on the convergence of the policy-population sequence $\{(\pi_t,\mu_t)\}_{t\geq 0}$ to the NE $(\pi^\star,\mu^\star)$.

Theorem 1. *Suppose that Assumptions 1 – 3 hold with $\bar{d} = 1 - d_1 d_2 - d_3 > 0$, and the policy evaluation and gradient estimation oracles satisfy* (2) *and* (3)*, respectively. The policy-population*

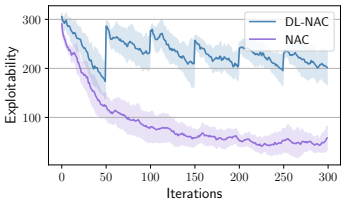

Figure 1: SIS Exploitability

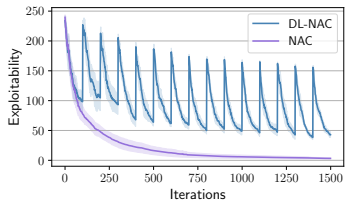

Figure 2: LQ Exploitability

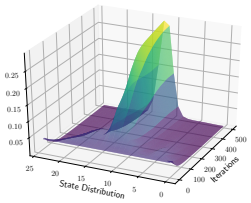

Figure 3: LQ Evolution

sequence $\{(\pi_t, \mu_t)\}_{t\geq 0}$ *generated by Algorithm 1 satisfies*

$$D\left(\pi^\star, \frac{1}{T}\sum_{t=0}^{T-1}\pi_t\right) + \left\|\mu^\star - \frac{1}{T}\sum_{t=0}^{T-1}\mu_t\right\|_1 \leq \widetilde{O}\left(\frac{1}{\lambda T^{1/5}} + \sqrt{\frac{|\mathcal{A}|\exp(1/\lambda)\varepsilon_{total}^{1/2}}{\lambda}}\right)$$

Theorem 1 establishes the finite-time convergence of Algorithm 1. After T iterations, the averaged policy-population pair $(\frac{1}{T}\sum_{t=0}^{T-1}\pi_t, \frac{1}{T}\sum_{t=0}^{T-1}\mu_t)$ constitutes an $\widetilde{O}(T^{-1/5})$-approximate NE, subject to the approximation and statistical errors. As we collect more samples, the statistical errors can be driven to 0. However, the approximation error $\varepsilon_{\text{approx}}$ is an inherent mis-modeling cost due to using a (potentially not expressive enough) linear function to approximate the policy, which might not diminish to 0 regardless of the number of iterations we run. While this seems to imply that linear function approximation can be restrictive, our experimental results (Section 5) instead suggest that linear functions are already satisfactory in most application scenarios that we are interested in. The convergence rate in Theorem 1 is independent of the size of the state space $|\mathcal{S}|$, implying that our approach is applicable to MFGs with large state spaces. The dependence on the action space $|\mathcal{A}|$ is also mild, and in mostly RL-related tasks the action space is adequately small [44]. The convergence rate is also inverse proportional to the regularization parameter λ. This indicates that a higher regularization level can accelerate the convergence of the algorithm while introducing a larger regularization error according to (1) on the other hand. In practice, one can choose a regularization level that balances the convergence rate and accuracy.

5 Experimental Results

5.1 Simulations on Classic MFGs

We first evaluate our natural actor-critic algorithm on two classic mean-field games considered in the literature, including an SIS epidemics model [16, 38], and a linear-quadratic MFG [49, 38, 12, 45]. We refer to these two tasks as SIS and LQ, respectively. Detailed descriptions of the tasks and the simulation setups are deferred to Appendix D. We implement Algorithm 1 with linear function approximation ("NAC" for short) and use temporal difference (TD) learning also with linear function approximation as the critic for policy evaluation. We utilize the standard notion of *exploitability* [82, 49, 16] to measure the sub-optimality of a policy. Intuitively, a higher degree of exploitability suggests that an individual agent can be much better off by deviating from the given policy. As a comparison baseline, we also implement a "double-loop" version of natural actor-critic ("DL-NAC") that uses a fixed point iteration in a similar fashion as existing works [28, 5, 27].

Simulation results on SIS and LQ are given in Figures 1 and 2, respectively. All results are averaged over 10 runs. For both tasks, we can observe that the exploitability of NAC converges to 0, indicating that NAC can efficiently learn the NE. DL-NAC also converges in general, but it suffers "zigzag" fluctuations due to the fact that double-loop methods update the mean-field abruptly and hence nullify the policies learned from the past. See Appendix D for a detailed discussion of this phenomenon. Similar patterns have also been observed in the literature [16]. Hence, our online method enjoys faster convergence and more smooth learning behavior than the fixed-point iteration. In Figure 3, we further plot the evolution of the state distribution over time when applying NAC to LQ. The population starts from a uniform distribution over the state space, and as time goes by, it quickly concentrates in a small neighborhood of the state space, a desired behavior for linear-quadratic tracking-type problems [71].

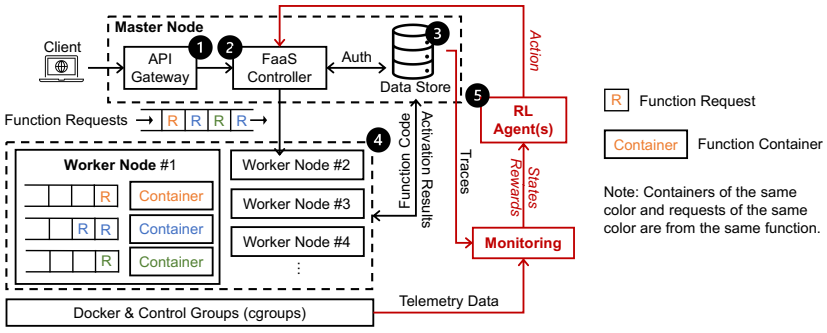

Figure 4: OpenWhisk cluster architecture and function container resource management with reinforcement learning. Each RL agent is assigned to one function.

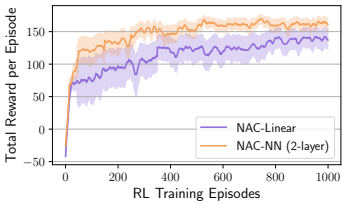

Figure 5: Policy Training

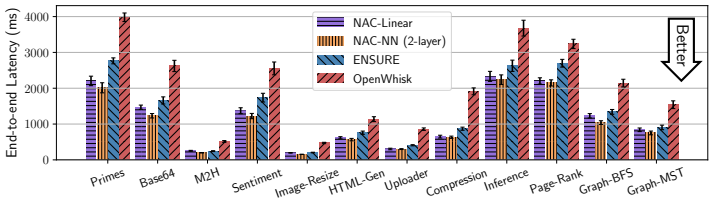

Figure 6: Policy Serving.

5.2 Experiments on a Serverless Platform

We apply our approach (with both linear and neural-network function approximations) to a resource management problem in a serverless computing environment. Here, we directly consider a finite-agent game and collect the empirical average of the local states as the mean-field.

Serverless Platform and Workload. We use a production-grade open-source serverless platform, OpenWhisk [22], and deploy it on IBM Cloud with 22 VMs. The OpenWhisk cluster consists of one master node (that runs the serverless controller) and 21 worker nodes (each of which runs the function containers). Figure 4 shows the architecture of a distributed OpenWhisk cluster and how RL agents work with the OpenWhisk cluster to manage resources of each function. The master node runs the API gateway (labeled as ❶), FaaS controller (❷), data store (❸), and other management modules. Each of the worker nodes (labeled as ❹) runs the function containers. We deploy the workload generator and RL agents from two separate nodes in the same cluster and use FaaSProfiler [59] to trace requests for function latency measurements. We select diverse function benchmarks from widely used open-source FaaS benchmark suites [15, 59, 81], including web applications (e.g., `HTML-Gen`), multimedia application (e.g., `Image-Resize`), scientific functions (e.g., `PageRank`), and ML-model serving (e.g., `Sentiment-Analysis`). To drive the benchmarks, we sample and replay the function invocations from Azure function traces [60]. Unlike [51], however, we do not explicitly consider function dependencies, but we believe that our approach is orthogonal to [51] and can be potentially integrated with the critical component localization of [51] to model function dependencies. More detailed descriptions of the OpenWhisk setup are relegated to Appendix E.

RL Pipeline in OpenWhisk. We model the resource management for each serverless function as a sequential decision-making problem that can be captured by RL. At each step in the sequence, an RL agent (labeled as ❺ in Figure 4) monitors the system and application conditions from both the OpenWhisk data store and the Linux cgroups. The collected measurements include function-level performance statistics (i.e., tail latencies on execution time, waiting time, and cold-start time for serving function requests) and system-level resource utilization statistics (e.g., CPU utilization of function containers). These measured telemetry data are pre-processed and used to define RL states and rewards, which is then mapped to a resource management decision by the RL agent. The decision is finally executed by FaaS controller and takes effect on the OpenWhisk platform. After applying the action, the RL agent receives another state and reward in the next time step.

MARL Formulation. We formulate the function resource management problem on a multi-tenant serverless platform as a Markov game where each function is assigned to one agent. At each step, the agent perceives available system and application conditions (e.g., resource utilization and function latencies) from the platform monitor as the state. The action space includes both horizontal scaling (i.e., scaling the number of function containers) and vertical scaling (i.e., scaling the container size). Since the objective is to meet QoS objectives while keeping the resource utilization at a high level, we consider a reward function as $r_t = \alpha \cdot QP(t) + (1-\alpha)/2 \cdot (RU_{cpu}(t) + RU_{mem}(t)) + penalty$, where $QP(t)$ and $RU(t)$ are the QoS preservation ratio and resource utilization at time t, and $penalty$ is set to -1 (and 0 otherwise) for illegal or undesired actions (e.g., dangling decisions). We implement and evaluate two variants of our method: one exploits linear function approximation for both the actor and the critic ("NAC-Linear" for short), and the other one leverages a two-layer fully-connected neural network as function approximation (i.e., "NAC-NN"). Since serverless functions usually have diverse resource requirements and behaviors, we resort to a multi-type formulation of MFGs (see [48, 73, 24] for more extensive discussions) to allow for heterogeneous agents.

Policy Training. For training, we create 50 functions[3] on OpenWhisk, each of which is randomly selected from the function benchmarks. Figure 5 shows the total reward achieved by NAC-Linear and NAC-NN at each training iteration. We observe that NAC-NN takes about 300 fewer episodes to converge and achieves an 18.8% higher reward after convergence compared to NAC-Linear. We attribute this to the fact that linear function approximation cannot fully capture the complex system dynamics and decision-making policies compared to neural networks.

Policy Serving. We measure the policy-serving performance using two metrics: the 99th percentile function latency (commonly used in user-defined QoS objectives) and the number of function containers in use. We take the model checkpoints of NAC agents after convergence and leverage the learned policies to manage resources for each function. Figure 6 shows the function performance managed by NAC-Linear and NAC-NN compared with two heuristics-based approaches ENSURE [66] and OpenWhisk's original resource manager (Appendix E). As shown in Figure 6, NAC-NN achieves the best performance and has 33.6% to 67.3% (for `Image-Resize`) lower latency compared to OpenWhisk's original algorithm. NAC-NN also has 14.8% to 29.6% (for `Sentiment-Analysis`) lower latency compared to ENSURE. Additionally, NAC-NN uses 29% fewer function containers than ENSURE and 37% fewer than OpenWhisk's design, indicating a higher resource utilization level. NAC-Linear also outperforms ENSURE regarding function tail latency (by up to 25.5%, for `Uploader`) and resource utilization (by up to 24% fewer function containers overall), which suggests that the simple linear function approximation can already lead to reasonable performances in practice.

6 Concluding Remarks

In this paper, we have proposed a mean-field game approach to large-scale cloud resource management. We have presented an online natural actor-critic algorithm and proved its finite-time convergence to the regularized Nash equilibrium with linear function approximation. We have evaluated our solution on a serverless platform using both linear and neural-network function approximations, which has demonstrated superior performances in terms of scalability, latency, and resource utilization. Interesting future directions include further tightening the convergence rate and investigating alternative forms of function approximations for other application scenarios.

Acknowledgments and Disclosure of Funding

This work was partially supported by the National Science Foundation (NSF) under grant CCF 20-29049; by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR), a research collaboration that is part of the IBM AI Horizon Network; by the IBM-ILLINOIS Discovery Accelerator Institute (IIDAI); and by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-19-1-0353. Any opinions, findings or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or IBM or AFOSR.

[3]We do not experiment with even more functions because the serverless platform architecture is a centralized model where the central manager usually becomes the bottleneck for scalability. To scale beyond thousands of functions in the production environment, cloud datacenters can use a two-tier model where a large cluster is divided into a couple of sub-clusters [17]. In this case, our method can be applied to each system sub-cluster to address the scalability issue.

References

[1] A. Agache, M. Brooker, A. Iordache, A. Liguori, R. Neugebauer, P. Piwonka, and D.-M. Popa. Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 419–434, 2020.

[2] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22 (98):1–76, 2021.

[3] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160. PMLR, 2019.

[4] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[5] B. Anahtarci, C. D. Kariksiz, and N. Saldi. Q-learning in regularized mean-field games. *arXiv preprint arXiv:2003.12151*, 2020.

[6] A. Angiuli, J.-P. Fouque, and M. Lauriere. Reinforcement learning for mean field games, with applications to economics. *arXiv preprint arXiv:2106.13755*, 2021.

[7] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, et al. *Serverless Computing: Current Trends and Open Problems*, pages 1–20. 2017.

[8] S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer. Inductive-bias-driven reinforcement learning for efficient scheduling in heterogeneous clusters. In *Proceedings of the 37th International Conference on Machine Learning*, pages 629–641, 2020.

[9] A. Bensoussan, K. Sung, S. C. P. Yam, and S.-P. Yung. Linear-quadratic mean field games. *Journal of Optimization Theory and Applications*, 169(2):496–529, 2016.

[10] J. Bhandari, D. Russo, and R. Singal. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on Learning Theory*, pages 1691–1692, 2018.

[11] S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[12] R. Carmona, J.-P. Fouque, and L.-H. Sun. Mean field games and systemic risk. *Communications in Mathematical Sciences*, 13(4):911–933, 2015.

[13] S. Cayci, N. He, and R. Srikant. Linear convergence of entropy-regularized natural policy gradient with linear function approximation. *arXiv preprint arXiv:2106.04096*, 2021.

[14] S. Cen, C. Cheng, Y. Chen, Y. Wei, and Y. Chi. Fast global convergence of natural policy gradient methods with entropy regularization. *Operations Research*, 2021.

[15] M. Copik, G. Kwasniewski, M. Besta, M. Podstawski, and T. Hoefler. SeBS: A serverless benchmark suite for Function-as-a-Service computing. In *Proceedings of the 22nd International Middleware Conference (Middleware 2021)*, page 64–78, 2021.

[16] K. Cui and H. Koeppl. Approximately solving mean field games via entropy-regularized deep reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1909–1917. PMLR, 2021.

[17] C. Curino, S. Krishnan, K. Karanasos, S. Rao, G. M. Fumarola, B. Huang, K. Chaliparambil, A. Suresh, Y. Chen, S. Heddaya, et al. Hydra: a federated resource manager for data-center scale analytics. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 177–192, 2019.

[18] C. Daskalakis, N. Golowich, and K. Zhang. The complexity of Markov equilibrium in stochastic games. *arXiv preprint arXiv:2204.03991*, 2022.

[19] D. Ding, C.-Y. Wei, K. Zhang, and M. R. Jovanović. Independent policy gradient for large-scale Markov potential games: Sharper rates, function approximation, and game-agnostic convergence. *arXiv preprint arXiv:2202.04129*, 2022.

[20] R. Elie, J. Perolat, M. Laurière, M. Geist, and O. Pietquin. On the convergence of model free learning in mean field games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7143–7150, 2020.

[21] Y. Fei, Z. Yang, Z. Wang, and Q. Xie. Dynamic regret of policy optimization in non-stationary environments. *Advances in Neural Information Processing Systems*, 33:6743–6754, 2020.

[22] A. S. Foundation. OpenWhisk. `https://github.com/apache/openwhisk`, 2022. Accessed: 2022-05-10.

[23] Z. Fu, Z. Yang, Y. Chen, and Z. Wang. Actor-critic provably finds Nash equilibria of linear-quadratic mean-field games. In *International Conference on Learning Representations*, 2020.

[24] S. Ganapathi Subramanian, P. Poupart, M. E. Taylor, and N. Hegde. Multi type mean field reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 411–419, 2020.

[25] M. Geist, B. Scherrer, and O. Pietquin. A theory of regularized Markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR, 2019.

[26] D. A. Gomes, J. Mohr, and R. R. Souza. Discrete time, finite state space mean field games. *Journal de mathématiques pures et appliquées*, 93(3):308–328, 2010.

[27] H. Gu, X. Guo, X. Wei, and R. Xu. Mean-field controls with Q-learning for cooperative MARL: convergence and complexity analysis. *SIAM Journal on Mathematics of Data Science*, 3(4): 1168–1196, 2021.

[28] X. Guo, A. Hu, R. Xu, and J. Zhang. A general framework for learning mean-field games. *arXiv preprint arXiv:2003.06069*, 2020.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[30] M. Huang, R. P. Malhamé, and P. E. Caines. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252, 2006.

[31] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[32] C. Jin, Q. Liu, Y. Wang, and T. Yu. V-learning–a simple, efficient, decentralized algorithm for multiagent RL. *arXiv preprint arXiv:2110.14555*, 2021.

[33] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, et al. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*, 2019.

[34] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002.

[35] S. M. Kakade. A natural policy gradient. *Advances in Neural Information Processing Systems*, 14, 2001.

[36] S. Kardani-Moghaddam, R. Buyya, and K. Ramamohanarao. ADRL: A hybrid anomaly-aware deep reinforcement learning-based resource scaling in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 32(3):514–526, 2020.

[37] J.-M. Lasry and P.-L. Lions. Mean field games. *Japanese Journal of Mathematics*, 2(1):229–260, 2007.

[38] M. Laurière, S. Perrin, S. Girgin, P. Muller, A. Jain, T. Cabannes, G. Piliouras, J. Pérolat, R. Élie, O. Pietquin, et al. Scalable deep reinforcement learning algorithms for mean field games. *arXiv preprint arXiv:2203.11973*, 2022.

[39] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning*, pages 157–163. Elsevier, 1994.

[40] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNet 2016)*, pages 50–56, 2016.

[41] W. Mao and T. Başar. Provably efficient reinforcement learning in decentralized general-sum Markov games. *Dynamic Games and Applications*, pages 1–22, 2022.

[42] W. Mao, L. Yang, K. Zhang, and T. Başar. On improving model-free algorithms for decentralized multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 15007–15049. PMLR, 2022.

[43] J. Mei, C. Xiao, C. Szepesvari, and D. Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pages 6820–6829. PMLR, 2020.

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

[45] J. Moon and T. Başar. Discrete-time LQG mean field games with unreliable communication. In *53rd IEEE Conference on Decision and Control*, pages 2697–2702. IEEE, 2014.

[46] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in Neural Information Processing Systems*, 2017.

[47] M. Nourian and P. E. Caines. ε-Nash mean field game theory for nonlinear stochastic dynamical systems with major and minor agents. *SIAM Journal on Control and Optimization*, 51(4): 3302–3331, 2013.

[48] J. Perolat, S. Perrin, R. Elie, M. Laurière, G. Piliouras, M. Geist, K. Tuyls, and O. Pietquin. Scaling up mean field games with online mirror descent. *arXiv preprint arXiv:2103.00623*, 2021.

[49] S. Perrin, J. Pérolat, M. Laurière, M. Geist, R. Elie, and O. Pietquin. Fictitious play for mean field games: Continuous time analysis and applications. *Advances in Neural Information Processing Systems*, 33:13199–13213, 2020.

[50] S. Perrin, M. Laurière, J. Pérolat, M. Geist, R. Élie, and O. Pietquin. Mean field games flock! The reinforcement learning way. *arXiv preprint arXiv:2105.07933*, 2021.

[51] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer. FIRM: An intelligent fine-grained resource management framework for SLO-oriented microservices. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2020)*, pages 805–825, Berkeley, CA, USA, Nov. 2020. USENIX Association.

[52] H. Qiu, W. Mao, A. Patke, C. Wang, H. Franke, Z. T. Kalbarczyk, T. Başar, and R. K. Iyer. SIMPPO: A scalable and incremental online learning framework for serverless resource management. In *Proceedings of the 13th ACM Symposium on Cloud Computing*, 2022.

[53] N. Saldi, T. Başar, and M. Raginsky. Markov–Nash equilibria in mean-field games with discounted cost. *SIAM Journal on Control and Optimization*, 56(6):4256–4287, 2018.

[54] N. Saldi, T. Başar, and M. Raginsky. Approximate Nash equilibria in partially observed stochastic games with mean-field interactions. *Mathematics of Operations Research*, 44(3): 1006–1033, 2019.

[55] N. Saldi, T. Başar, and M. Raginsky. Approximate Markov-Nash equilibria for discrete-time risk-sensitive mean-field games. *Mathematics of Operations Research*, 45(4):1596–1620, 2020.

[56] L. Schuler, S. Jamil, and N. Kühl. AI-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2021)*, pages 804–811, 2021.

[57] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[58] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[59] M. Shahrad, J. Balkind, and D. Wentzlaff. Architectural implications of Function-as-a-Service computing. In *Proceedings of the 52nd International Symposium on Microarchitecture (MICRO 2019)*, pages 1063–1075, 2019.

[60] M. Shahrad, R. Fonseca, Í. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *Proceedings of the USENIX 2020 Annual Technical Conference (ATC 2020)*, pages 205–218, 2020.

[61] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[62] L. Shani, Y. Efroni, and S. Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5668–5675, 2020.

[63] A. Singhvi, A. Balasubramanian, K. Houck, M. D. Shaikh, S. Venkataraman, and A. Akella. Atoll: A scalable low-latency serverless platform. In *Proceedings of the 12th ACM Symposium on Cloud Computing (SoCC 2021)*, pages 138–152, 2021.

[64] Z. Song, S. Mei, and Y. Bai. When can we learn general-sum Markov games with a large number of players sample-efficiently? *arXiv preprint arXiv:2110.04184*, 2021.

[65] J. Subramanian and A. Mahajan. Reinforcement learning in stationary mean-field games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 251–259, 2019.

[66] A. Suresh, G. Somashekar, A. Varadarajan, V. R. Kakarla, H. Upadhyay, and A. Gandhi. ENSURE: Efficient scheduling and autonomous resource management in serverless environments. In *International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS 2020)*, pages 1–10, 2020.

[67] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[68] A. Tariq, A. Pahl, S. Nimmagadda, E. Rozner, and S. Lanka. Sequoia: Enabling Quality-of-Service in serverless computing. In *Proceedings of the 11th ACM Symposium on Cloud Computing (SoCC 2020)*, pages 311–327, 2020.

[69] H. Tembine and M. Huang. Mean field difference games: Mckean-Vlasov dynamics. In *50th IEEE Conference on Decision and Control and European Control Conference*, pages 1006–1011. IEEE, 2011.

[70] J. Tsitsiklis and B. Van Roy. Analysis of temporal-diffference learning with function approximation. *Advances in Neural Information Processing Systems*, 9, 1996.

[71] M. A. uz Zaman, K. Zhang, E. Miehling, and T. Başar. Approximate equilibrium computation for discrete-time linear-quadratic mean-field games. In *2020 American Control Conference (ACC)*, pages 333–339. IEEE, 2020.

[72] M. A. uz Zaman, K. Zhang, E. Miehling, and T. Başar. Reinforcement learning in non-stationary discrete-time linear-quadratic mean-field games. In *2020 59th IEEE Conference on Decision and Control*, pages 2278–2284. IEEE, 2020.

[73] M. A. uz Zaman, E. Miehling, and T. Başar. Reinforcement learning for non-stationary discrete-time linear–quadratic mean-field games in multiple populations. *Dynamic Games and Applications*, pages 1–47, 2022.

[74] L. Wang, Q. Cai, Z. Yang, and Z. Wang. Neural policy gradient methods: Global optimality and rates of convergence. In *International Conference on Learning Representations*, 2020.

[75] Z. Wen, Y. Wang, and F. Liu. StepConf: SLO-aware dynamic resource configuration for serverless function workflows. In *IEEE Conference on Computer Communications (INFOCOM 2022)*, pages 1–10, Washington, DC, USA, 2022. IEEE Computer Society.

[76] Q. Xie, Z. Yang, Z. Wang, and A. Minca. Learning while playing in mean-field games: Convergence and optimality. In *International Conference on Machine Learning*, pages 11436–11447. PMLR, 2021.

[77] T. Xu, Z. Wang, and Y. Liang. Improving sample complexity bounds for (natural) actor-critic algorithms. *Advances in Neural Information Processing Systems*, 33:4358–4369, 2020.

[78] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018.

[79] Z. Yang, P. Nguyen, H. Jin, and K. Nahrstedt. MIRAS: Model-based reinforcement learning for microservice resource allocation over scientific workflows. In *IEEE 39th International Conference on Distributed Computing Systems*, pages 122–132, 2019.

[80] H. Yu, A. A. Irissappane, H. Wang, and W. J. Lloyd. FaaSRank: Learning to schedule functions in serverless platforms. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS 2021)*, pages 31–40, 2021.

[81] T. Yu, Q. Liu, D. Du, Y. Xia, B. Zang, Z. Lu, P. Yang, C. Qin, and H. Chen. ServerlessBench (socc 2020). `https://github.com/SJTU-IPADS/ServerlessBench`, 2020.

[82] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. *Advances in neural Information Processing Systems*, 20, 2007.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We have discussed the potential lack of expressiveness of linear function approximation in Sections 3 and 4. We have also pointed a few directions for improvement in our concluding remarks.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Applying our resource management solution to real-world serverless platforms might lead to privacy concerns as it relies on collecting behavior data of the user functions. Nevertheless, as discussed in Section 5, our simulations themselves are evaluated on anonymized open-source function traces and should not raise such concerns.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Assumptions 1– 3 in Section 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendices A and B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendices D and E.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figures 1, 2, 5, and 6.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5 and Appendix E.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We have properly cited the creators of the datasets [60] and benchmarks [15, 59, 81] in Section 5.

(b) Did you mention the license of the assets? [Yes] The dataset [60] uses the Creative Commons Attribution 4.0 International Public License. The benchmark FaaSProfiler uses MIT License; [15] uses BSD 3-Clause License; [81] uses Mulan Permissive Software License.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The dataset [60] and benchmarks [15, 59, 81] that we use have been open-sourced by their corresponding creators on GitHub.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data we are using does not contain personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No crowdsourcing research has been involved in this work.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Supplementary Materials for "A Mean-Field Game Approach to Cloud Resource Management with Function Approximation"

A Technical Lemmas

Lemma 2. *(Persistence of excitation, Lemma 2 of [13]). For any $\lambda > 0$, the entropy-regularized natural actor-critic update with averaging satisfies*

$$\|\theta_t\|_2 \leq R/\lambda, \forall t \geq 0, \quad \textit{and} \quad p_{\min} = \inf_{t\geq 0} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} \pi_t(a|s) \geq \frac{\exp(-2R/\lambda)}{|\mathcal{A}|} > 0.$$

Proof. We prove the first statement by induction. The statement holds for $t = 0$ due to our initialization $\theta_0 = \mathbf{0}$. Expanding the recursive policy update rule

$$\theta_{t+1} = (1 - \eta_t\lambda)\theta_t + \eta_t\hat{w}_t = (1 - \eta_t\lambda)\theta_t + \eta_t\lambda \cdot \frac{\hat{w}_t}{\lambda}.$$

Applying the triangle inequality,

$$\|\theta_{t+1}\|_2 \leq (1 - \eta_t\lambda) \|\theta_t\|_2 + \eta_t\lambda \|\hat{w}_t/\lambda\|_2 \leq R/\lambda,$$

where the last step holds because of the induction hypothesis, the fact that our gradient estimation step guarantees $\|\hat{w}_t\|_2 \leq R$, and that $0 < \eta_t\lambda < 1$. Invoking the induction completes the proof of $\|\theta_t\|_2 \leq R/\lambda, \forall t \geq 0$.

Further, using the condition that $\|\phi_{s,a}\|_2 \leq 1$, the Cauchy-Schwarz inequality implies that $\left|\theta_t^\top \phi_{s,a}\right| \leq R/\lambda, \forall (s, a) \in \mathcal{S} \times \mathcal{A}$. Under softmax parameterization,

$$p_{\min} = \inf_{t\geq 0} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} \pi_t(a|s) \geq \frac{\exp(-R/\lambda)}{|\mathcal{A}| \exp(R/\lambda)} = \frac{\exp(-2R/\lambda)}{|\mathcal{A}|} > 0.$$

This completes the proof of the lemma. □

Lemma 3. *Let $Q_{\max} = \frac{1+\gamma\lambda \log|\mathcal{A}|}{1-\gamma}$. For any mean-field state μ, the optimal policy $\pi_\mu^{\star,\lambda}$ with respect to the MDP induced by μ satisfies that*

$$\pi_\mu^{\star,\lambda}(a|s) \geq \frac{1}{|\mathcal{A}| \exp(Q_{\max}/\lambda)}, \forall (s, a) \in \mathcal{S} \times \mathcal{A}.$$

Proof. It has been shown [46] that the optimal policy $\pi_\mu^{\star,\lambda}$ can be expressed as a Boltzmann distribution of the form

$$\pi_\mu^{\star,\lambda}(a|s) \propto \exp\left(\frac{Q_\mu^{\star,\lambda}(s, a)}{\lambda}\right),$$

where $Q_\mu^{\star,\lambda}(s, a)$ is the optimal soft Q-function. From the definition of $Q_\mu^{\pi,\lambda}$ and the facts that $r(s, a, \mu) \in [0, 1]$ and $\mathcal{H}(p) \leq \log|\mathcal{A}|$ for any distribution p over $\mathcal{A}$, we can easily see that $Q_\mu^{\star,\lambda}(s, a) \leq Q_{\max} = \frac{1+\gamma\lambda \log|\mathcal{A}|}{1-\gamma}$. Therefore, for any $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$\pi_\mu^{\star,\lambda}(a|s) = \frac{\exp\left(Q_\mu^{\star,\lambda}(s, a)/\lambda\right)}{\sum_{b\in\mathcal{A}} \exp\left(Q_\mu^{\star,\lambda}(s, b)/\lambda\right)} \geq \frac{1}{\sum_{b\in\mathcal{A}} \exp\left(Q_{\max}/\lambda\right)} = \frac{1}{|\mathcal{A}| \exp(Q_{\max}/\lambda)}.$$

□

Lemma 4. *(Lemma 3 of [76]). For any $x, y, z \in \mathcal{A}$, if $x(a) \geq \alpha_1, y(a) \geq \alpha_1$, and $z(a) \geq \alpha_2, \forall a \in \mathcal{A}$, then*

$$KL(x\|z) - KL(y\|z) \leq \left(1 + \log\frac{1}{\min\{\alpha_1, \alpha_2\}}\right) \cdot \|x - y\|_1.$$

Lemma 5. *Under the same conditions as Lemma 8, it holds that*

$$\sigma_{t+1}^{\pi} \leq \mathbb{E}_{d_t^\star}\left[KL(\pi_t^\star \| \pi_{t+1})\right] + (1 + C_1) \cdot \kappa d_0 \left\|\mu_{t+1} - \mu_t\right\|_1,$$

where $\kappa = \frac{2\log|\mathcal{A}|}{1-\gamma} + \frac{1+2R(1-\gamma)}{\lambda(1-\gamma)}$.

Proof. From the definition of σ_{t+1}^{π},

$$\begin{aligned}
&\sigma_{t+1}^{\pi} = \mathbb{E}_{d_{t+1}^\star}\left[\mathrm{KL}(\pi_{t+1}^\star \| \pi_{t+1})\right] \\
\leq &\mathbb{E}_{d_{t+1}^\star}\left[\mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right] + \left|\mathbb{E}_{d_{t+1}^\star}\left[\mathrm{KL}(\pi_{t+1}^\star \| \pi_{t+1}) - \mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right]\right| \\
= &\mathbb{E}_{d_t^\star}\left[\mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right] + \left(\mathbb{E}_{d_{t+1}^\star} - \mathbb{E}_{d_t^\star}\right)\left[\mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right] + \left|\mathbb{E}_{d_{t+1}^\star}\left[\mathrm{KL}(\pi_{t+1}^\star \| \pi_{t+1}) - \mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right]\right|
\end{aligned} \tag{4}$$

In the following, we upper bound each term in (4) separately. We first define $p_{\min} \stackrel{\text{def}}{=} \inf_{t\geq 0} \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} \pi_t(a|s)$, and apply the persistence of excitation condition from Lemma 2 to obtain that $p_{\min} \geq \frac{\exp(-2R/\lambda)}{|\mathcal{A}|} > 0$. To upper bound the second term in (4), we first show that for any $s \in \mathcal{S}$,

$$\mathrm{KL}(\pi_t^\star(\cdot|s) \| \pi_{t+1}(\cdot|s)) = \sum_{a\in\mathcal{A}} \pi_t^\star(a|s) \log \frac{\pi_t^\star(a|s)}{\pi_{t+1}(a|s)} \leq \sum_{a\in\mathcal{A}} \pi_t^\star(a|s) \log \frac{1}{p_{\min}} \leq \log|\mathcal{A}| + 2R/\lambda.$$

If we define $\mathrm{KL}_{\max} \stackrel{\text{def}}{=} \log|\mathcal{A}| + 2R/\lambda$, we will have that

$$\begin{aligned}
\left(\mathbb{E}_{d_{t+1}^\star} - \mathbb{E}_{d_t^\star}\right)\left[\mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right] =&\mathbb{E}_{s\sim d^\star}\left[\frac{d_{t+1}^\star(s) - d_t^\star(s)}{d^\star(s)} \cdot \mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right] \\
\leq &\mathrm{KL}_{\max} \cdot \mathbb{E}_{s\sim d^\star}\left[\frac{\left|d_{t+1}^\star(s) - d_t^\star(s)\right|}{d^\star(s)}\right] \\
\leq &\mathrm{KL}_{\max} \cdot d_0 \left\|\mu_{t+1} - \mu_t\right\|_1,
\end{aligned} \tag{5}$$

where the last step follows from Assumption 1. This gives an upper bound of the second term.

We proceed to upper bound the third term in (4). Let $\tau = \frac{1}{|\mathcal{A}|}\exp\left(-\frac{1+\gamma\lambda\log|\mathcal{A}|}{\lambda(1-\gamma)}\right)$. From Lemma 3, we know that

$$\pi_t^\star(a|s) \geq \tau, \text{ and } \pi_{t+1}^\star(a|s) \geq \tau, \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

Since both $\pi_t(a|s)$ and $\pi_t^\star(a|s)$ are lower bounded, we can apply the Lipschitzness of KL-divergence (Lemma 4) and obtain

$$\begin{aligned}
&\left|\mathbb{E}_{d_{t+1}^\star}\left[\mathrm{KL}(\pi_{t+1}^\star \| \pi_{t+1}) - \mathrm{KL}(\pi_t^\star \| \pi_{t+1})\right]\right| \\
\leq &\left(1 + \log\frac{1}{\min\{\tau, p_{\min}\}}\right)\mathbb{E}_{s\sim d_{t+1}^\star}\left[\left\|\pi_t^\star(\cdot|s) - \pi_{t+1}^\star(\cdot|s)\right\|_1\right] \\
\leq &\kappa\mathbb{E}_{s\sim d^\star}\left[\frac{d_{t+1}^\star(s)}{d^\star(s)} \cdot \left\|\pi_t^\star(\cdot|s) - \pi_{t+1}^\star(\cdot|s)\right\|_1\right] \\
\leq &\kappa C_1 D(\pi_t^\star, \pi_{t+1}^\star) \\
= &\kappa C_1 D(\Gamma_1^\lambda(\mu_t), \Gamma_1^\lambda(\mu_{t+1})) \\
\leq &\kappa C_1 d_0 \left\|\mu_t - \mu_{t+1}\right\|_1,
\end{aligned} \tag{6}$$

where the third inequality is by Assumption 2, the last step is due to Assumption 1, and

$$\begin{aligned}
\kappa \stackrel{\text{def}}{=} &\frac{2\log|\mathcal{A}|}{1-\gamma} + \frac{1+2R(1-\gamma)}{\lambda(1-\gamma)} \\
\geq &1 + \max\left\{\log|\mathcal{A}| + \frac{1+\gamma\lambda\log|\mathcal{A}|}{\lambda(1-\gamma)}, \log|\mathcal{A}| + 2R/\lambda\right\} \\
\geq &1 + \log\frac{1}{\min\{\tau, p_{\min}\}}.
\end{aligned}$$

This gives an upper bound of the third term in (4). Combining (4), (5), and (6) completes the proof of the lemma. □

Lemma 6. *Under Assumption 3, it holds for any mean-field states μ, μ' that*

$$\left\|\Lambda^{\lambda}(\mu)-\Lambda^{\lambda}\left(\mu'\right)\right\|_1 \leq \left(d_1 d_2+d_3\right)\left\|\mu-\mu'\right\|_1 .$$

In particular, Λ^{λ} is a contraction if $d_1 d_2 + d_3 < 1$.

Proof. By the definition of the composite operator,

$$\begin{aligned}
\left\|\Lambda^{\lambda}(\mu)-\Lambda^{\lambda}\left(\mu'\right)\right\|_1 &= \left\|\Gamma_2\left(\Gamma_1^{\lambda}(\mu), \mu\right)-\Gamma_2\left(\Gamma_1^{\lambda}\left(\mu'\right), \mu'\right)\right\|_1 \\
&\leq \left\|\Gamma_2\left(\Gamma_1^{\lambda}(\mu), \mu\right)-\Gamma_2\left(\Gamma_1^{\lambda}\left(\mu'\right), \mu\right)\right\|_1+\left\|\Gamma_2\left(\Gamma_1^{\lambda}\left(\mu'\right), \mu\right)-\Gamma_2\left(\Gamma_1^{\lambda}\left(\mu'\right), \mu'\right)\right\|_1 \\
&\leq d_2 D\left(\Gamma_1^{\lambda}(\mu), \Gamma_1^{\lambda}\left(\mu'\right)\right)+d_3\left\|\mu-\mu'\right\|_1 \\
&\leq \left(d_1 d_2+d_3\right)\left\|\mu-\mu'\right\|_1 ,
\end{aligned}$$

where the second inequality uses the Lipschitzness of Γ_2, and the last inequality is due to the Lipschitzness of Γ_1^{λ}. □

Lemma 7. *(Lemma 8 of [76]). Suppose that Assumptions 2 and 3 hold with $\bar{d} = 1 - d_1 d_2 - d_3 > 0$, we then have*

$$\left\|\mu_{t+1}-\mu^{\star}\right\|_1 \leq (1-\beta_t \bar{d})\left\|\mu_t-\mu^{\star}\right\|_1+d_2 C_2 \beta_t \sqrt{\sigma_t^{\pi}}, \forall t \geq 0.$$

Proof. For notational convenience, define $\sigma_t^{\mu} \stackrel{\text{def}}{=} \left\|\mu_t-\mu^{\star}\right\|_1$. Since Algorithm 1 updates the mean-field state μ_t in the same way as [76], our σ_t^{μ} also exhibits the same recursive behavior as characterized in Lemma 8 of [76], and we reproduce the proof here for completeness. Using the update rule of the mean-field state in Algorithm 1,

$$\begin{aligned}
&\left\|\mu_{t+1}-\mu^{\star}\right\|_1 \\
=&\left\|(1-\beta_t)\mu_t+\beta_t \Gamma_2(\pi_t, \mu_t)-\mu^{\star}\right\|_1 \\
=&\left\|\left(1-\beta_t\right)\left(\mu_t-\mu^{\star}\right)+\beta_t\left(\Gamma_2\left(\Gamma_1^{\lambda}\left(\mu_t\right), \mu_t\right)-\mu^{\star}\right)-\beta_t\left(\Gamma_2\left(\Gamma_1^{\lambda}\left(\mu_t\right), \mu_t\right)-\Gamma_2\left(\pi_t, \mu_t\right)\right)\right\|_1 \\
\leq&\left(1-\beta_t\right)\left\|\mu_t-\mu^{\star}\right\|_1+\beta_t\left\|\Gamma_2(\Gamma_1^{\lambda}(\mu_t), \mu_t)-\Gamma_2(\Gamma_1^{\lambda}(\mu^{\star}), \mu^{\star})\right\|_1 \\
&+\beta_t\left\|\Gamma_2(\Gamma_1^{\lambda}(\mu_t), \mu_t)-\Gamma_2(\pi_t, \mu_t)\right\|_1 , \\
\leq&(1-\beta_t \bar{d})\left\|\mu_t-\mu^{\star}\right\|_1+\beta_t d_2 D(\pi_t^{\star}, \pi_t),
\end{aligned} \tag{7}$$

where the first inequality uses the fact that $\Gamma_2(\Gamma_1^{\lambda}(\mu^{\star}), \mu^{\star}) = \mu^{\star}$, the second inequality follows from the Lipschitzness of the operators Λ^{λ} (Lemma 6) and Γ_2. To further upper bound the second term on the RHS of (7), we recall the definition that

$$\begin{aligned}
D(\pi_t^{\star}, \pi_t) =&\mathbb{E}_{s \sim d^{\star}}\left[\left\|\pi_t^{\star}(\cdot|s)-\pi_t(\cdot|s)\right\|_1\right] \\
=&\mathbb{E}_{s \sim d_t^{\star}}\left[\frac{d^{\star}(s)}{d_t^{\star}(s)} \cdot\left\|\pi_t^{\star}(\cdot|s)-\pi_t(\cdot|s)\right\|_1\right] \\
\leq&\left(\mathbb{E}_{s \sim d_t^{\star}}\left[\left|\frac{d^{\star}(s)}{d_t^{\star}(s)}\right|^2\right] \cdot \mathbb{E}_{s \sim d_t^{\star}}\left[\left\|\pi_t^{\star}(\cdot|s)-\pi_t(\cdot|s)\right\|_1^2\right]\right)^{\frac{1}{2}} \\
\leq&C_2 \sqrt{\mathbb{E}_{s \sim d_t^{\star}}\left[\mathrm{KL}(\pi_t^{\star}(\cdot|s)) \| \pi_t(\cdot|s)\right]},
\end{aligned} \tag{8}$$

where the last step follows from Assumption 2 and Pinsker's inequality. Plugging (8) back to (7) completes the proof. □

B Proofs for Section 4

B.1 Proof for Lemma 1

Proof. First, notice that for any fixed $(s, a) \in \mathcal{S} \times \mathcal{A}$, the log-linear policy $\log \pi_\theta(a|s)$ is a 1-smooth function in θ:

$$\left\|\nabla_\theta \log \pi_\theta(a|s)-\nabla_\theta \log \pi_{\theta'}(a|s)\right\|_2 \leq\left\|\theta-\theta'\right\|_2, \forall \theta, \theta' \in \mathbb{R}^d.$$

A standard result for an L-smooth function f on $\mathbb{R}^n$ is that (e.g., Lemma 3.4 of [11])

$$f(y) \geq f(x) + \nabla f(x)^\top (y-x) - \frac{L}{2}\|y-x\|_2^2, \forall x, y \in \mathbb{R}^n.$$

We hence obtain that

$$\begin{aligned}
&\mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_{t+1}(\cdot|s))\right] - \mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_t(\cdot|s))\right] \\
\leq &\mathbb{E}_{s\sim d_t^\star}\left[\sum_{a\in\mathcal{A}} \pi_t^\star(a|s)\left(\log \pi_t(a|s) - \log \pi_{t+1}(a|s)\right)\right] \\
\leq &-\eta_t \mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top \nabla_\theta \log \pi_t(a|s)\right] + \frac{\eta_t^2}{2}\|g_t\|_2^2 \\
= &-\eta_t \mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top \nabla_\theta \log \pi_t(a|s) - q_\mu^{\pi_t,\lambda}(s,a)\right] - \eta_t \mathbb{E}_{d_t^\star\circ\pi_t^\star}\left[q_\mu^{\pi_t,\lambda}(s,a)\right] + \frac{\eta_t^2}{2}\|g_t\|_2^2.
\end{aligned}$$

The performance difference lemma in the regularized case (e.g., Lemma 5 of [13]) implies that for any two policies π and π', we have that

$$V_\mu^{\pi,\lambda}(\rho) - V_\mu^{\pi',\lambda}(\rho) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi, a\sim\pi(\cdot|s)}\left[q_\mu^{\pi',\lambda}(s,a) - V_\mu^{\pi',\lambda}(s)\right] - \frac{\lambda}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi}\left[\mathrm{KL}(\pi(\cdot|s)\|\pi'(\cdot|s))\right].$$

By letting $\mu = \mu_t$, $\pi' = \pi_t$ and $\pi = \pi_t^\star$, we further have that

$$\begin{aligned}
&\mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_{t+1}(\cdot|s))\right] - \mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_t(\cdot|s))\right] \\
\leq &-\eta_t \mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top \nabla_\theta \log \pi_t(a|s) - q_\mu^{\pi_t,\lambda}(s,a)\right] - \eta_t(1-\gamma)\left(V_{\mu_t}^{\pi_t^\star,\lambda}(\rho) - V_{\mu_t}^{\pi_t,\lambda}(\rho)\right) \\
&- \eta_t\lambda \mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_t(\cdot|s))\right] - \eta_t \mathbb{E}_{s\sim d_t^\star}\left[V_{\mu_t}^{\pi_t,\lambda}(s)\right] + \frac{\eta_t^2}{2}\|g_t\|_2^2 \\
\leq &-\eta_t \mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top \nabla_\theta \log \pi_t(a|s) - q_\mu^{\pi_t,\lambda}(s,a)\right] - \eta_t\lambda \mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_t(\cdot|s))\right] \\
&- \eta_t \mathbb{E}_{s\sim d_t^\star}\left[V_{\mu_t}^{\pi_t,\lambda}(s)\right] + \frac{\eta_t^2}{2}\|g_t\|_2^2,
\end{aligned}$$

where the last step uses the fact that $\pi_t^\star$ is the optimal (regularized) policy with respect to μ_t, and thus $V_{\mu_t}^{\pi_t^\star,\lambda}(\rho) \geq V_{\mu_t}^{\pi_t,\lambda}(\rho)$. Rearranging the terms completes the proof. □

B.2 Recursive Relationship of $\mathrm{KL}(\pi_t^\star\|\pi_t)$

We define $\sigma_t^\pi \stackrel{\text{def}}{=} \mathbb{E}_{s\sim d_t^\star}[\mathrm{KL}(\pi_t^\star(\cdot|s))\|\pi_t(\cdot|s)] = \sum_{s\in\mathcal{S}} d_t^\star(s)\sum_{a\in\mathcal{A}}\pi_t^\star(a|s)\log\frac{\pi_t^\star(a|s)}{\pi_t(a|s)}$ as a measure of distance between π_t and $\pi_t^\star$. Built upon Assumptions 1 and 2 and the policy improvement lemma, our next result characterizes the recursive relationship of σ_t^π in terms of the approximation and statistical errors as well as the evolution of the mean-field. Such a recursion is critical to establish the convergence of the policy.

Lemma 8. *Under Assumptions 1 and 2, it holds that for every iteration $t \geq 0$ of Algorithm 1:*

$$\sigma_{t+1}^\pi \leq (1-\eta_t\lambda)\sigma_t^\pi + 3C_3\eta_t(1+\frac{1}{p_{\min}})\sqrt{\varepsilon_{total}} + 2\eta_t^2R^2 + (1+C_1)\cdot\kappa d_0\|\mu_{t+1}-\mu_t\|_1,$$

where $\kappa = \frac{2\log|\mathcal{A}|}{1-\gamma} + \frac{1+2R(1-\gamma)}{\lambda(1-\gamma)}$, and $p_{\min} \geq \frac{\exp(-2R/\lambda)}{|\mathcal{A}|}$.

Proof. From Lemma 1, we know that

$$\begin{aligned}
\mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_{t+1}(\cdot|s))\right] \leq &(1-\eta_t\lambda)\sigma_t^\pi - \eta_t\mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top\nabla_\theta\log\pi_t(a|s) - q_\mu^{\pi_t,\lambda}(s,a)\right] \\
&- \eta_t\mathbb{E}_{s\sim d_t^\star}[V_{\mu_t}^{\pi_t,\lambda}(s)] + \frac{1}{2}\eta_t^2\|g_t\|_2^2.
\end{aligned}$$

In order to show the recursive relationship between σ_t^π and σ_{t+1}^π, we first need to establish the relationship between σ_{t+1}^π and $\mathbb{E}_{s\sim d_t^\star}\left[\mathrm{KL}(\pi_t^\star(\cdot|s)\|\pi_{t+1}(\cdot|s))\right]$, which is shown in Lemma 5 of

Appendix A. By applying the result of Lemma 5, we obtain that

$$\sigma_{t+1}^{\pi} \leq (1-\eta_t\lambda)\sigma_t^{\pi} \underbrace{-\eta_t \mathbb{E}_{s\sim d_t^\star, a\sim\pi_t^\star(\cdot|s)}\left[g_t^\top \nabla_\theta \log \pi_t(a|s) - q_\mu^{\pi_t,\lambda}(s,a)\right]}_{①}$$

$$\underbrace{-\eta_t \mathbb{E}_{s\sim d_t^\star}[V_{\mu_t}^{\pi_t,\lambda}(s)]}_{②} + \frac{1}{2}\eta_t^2 \left\|g_t\right\|_2^2 + (1+C_1)\cdot\kappa d_0 \left\|\mu_{t+1}-\mu_t\right\|_1. \tag{9}$$

In the following, we upper bound each term on the RHS separately. With the compatible function approximation condition, we have that (see, e.g., [67])

$$\nabla_\theta \log \pi_\theta(a|s) = \phi_{s,a} - \sum_{a'\in\mathcal{A}} \pi_\theta(a'|s)\phi_{s,a'}.$$

We can hence rewrite ① as

$$\begin{aligned}
① =& -\eta_t \sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\left(\phi_{s,a}^\top g_t - \sum_{a'\in\mathcal{A}} \pi_\theta(a'|s)\phi_{s,a'}^\top g_t - q_\mu^{\pi_t,\lambda}(s,a)\right)\\
=& -\eta_t \sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\left(\phi_{s,a}^\top g_t - \sum_{a'\in\mathcal{A}} \pi_\theta(a'|s)\phi_{s,a'}^\top g_t - Q_\mu^{\pi_t,\lambda}(s,a) + \lambda\theta_t^\top\phi_{s,a}\right)\\
&+ \lambda\eta_t \sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\sum_{a'\in\mathcal{A}} \pi_t(a'|s)\theta_t^\top\phi_{s,a'},
\end{aligned}$$

where in the last step we used the fact that $q_\mu^{\pi_t,\lambda}(s,a) = Q_\mu^{\pi_t,\lambda}(s,a) - \lambda\log\pi_t(a|s)$ and the expression of $\log\pi_t(a|s)$. Similarly, by using the relation that

$$V_\mu^{\pi,\lambda}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q_\mu^{\pi,\lambda}(s,a) - \lambda\log\pi(a|s)],$$

we can also rewrite ② as

$$\begin{aligned}
② =& -\eta_t \sum_{s,a} d_t^\star(s)\pi_t(a|s)\left(Q_{\mu_t}^{\pi_t,\lambda}(s,a) - \lambda\log\pi_t(a|s)\right)\\
=& -\eta_t \sum_{s,a} d_t^\star(s)\pi_t(a|s)\left(Q_{\mu_t}^{\pi_t,\lambda}(s,a) - \lambda\theta_t^\top\phi_{s,a}\right) - \lambda\eta_t \sum_{s,a} d_t^\star(s)\pi_t(a|s)\sum_{a'\in\mathcal{A}}\pi_t(a'|s)\theta_t^\top\phi_{s,a'}.
\end{aligned}$$

Since the value of $\sum_{a'\in\mathcal{A}}\pi_t(a'|s)\theta_t^\top\phi_{s,a'}$ is independent of a, we have that

$$\sum_{s,a} d_t^\star(s)\left(\pi_t^\star(a|s) - \pi_t(a|s)\right)\sum_{a'\in\mathcal{A}}\pi_t(a'|s)\theta_t^\top\phi_{s,a'} = 0.$$

We can hence combine the expressions of ① and ②, and deduce that

$$\begin{aligned}
①+② =& -\eta_t \sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\left(\phi_{s,a}^\top g_t - \sum_{a'\in\mathcal{A}} \pi_\theta(a'|s)\phi_{s,a'}^\top g_t - Q_{\mu_t}^{\pi_t,\lambda}(s,a) + \lambda\theta_t^\top\phi_{s,a}\right)\\
&- \eta_t \sum_{s,a} d_t^\star(s)\pi_t(a|s)\left(Q_{\mu_t}^{\pi_t,\lambda}(s,a) - \lambda\theta_t^\top\phi_{s,a}\right)\\
=& -\eta_t \sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\left(\phi_{s,a}^\top g_t - Q_{\mu_t}^{\pi_t,\lambda}(s,a) + \lambda\theta_t^\top\phi_{s,a}\right) + \eta_t \sum_{s,a,a'} d_t^\star(s)\pi_t^\star(a|s)\pi_\theta(a'|s)\phi_{s,a'}^\top g_t\\
&- \eta_t \sum_{s,a} d_t^\star(s)\pi_t(a|s)\left(-\phi_{s,a}^\top g_t + Q_{\mu_t}^{\pi_t,\lambda}(s,a) - \lambda\theta_t^\top\phi_{s,a}\right) - \eta_t \sum_{s,a} d_t^\star(s)\pi_t(a|s)\phi_{s,a}^\top g_t\\
=& -\eta_t \sum_{s,a} d_t^\star(s)\left(\pi_t^\star(a|s) - \pi_t(a|s)\right)\left(\phi_{s,a}^\top g_t - Q_{\mu_t}^{\pi_t,\lambda}(s,a) + \lambda\theta_t^\top\phi_{s,a}\right),
\end{aligned}$$

where the last step holds because

$$\begin{aligned}
&\sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\sum_{a'}\pi_t(a'|s)\phi_{s,a'}^\top g_t - \sum_{s,a} d_t^\star(s)\pi_t(a|s)\phi_{s,a}^\top g_t\\
=&\sum_s d_t^\star(s)\left(\sum_a \pi_t^\star(a|s) - 1\right)\sum_{a'}\pi_t(a'|s)\phi_{s,a'}^\top g_t\\
=&0.
\end{aligned}$$

Using the update rule that $g_t = \hat{w}_t - \lambda\theta_t$, we further have

$$\begin{aligned}
①+② =& -\eta_t \sum_{s,a} d_t^\star(s)\left(\pi_t^\star(a|s) - \pi_t(a|s)\right)\left(\phi_{s,a}^\top \hat{w}_t - Q_{\mu_t}^{\pi_t,\lambda}(s,a)\right) \\
=& -\eta_t \sum_{s,a} d_t^\star(s)\left(\pi_t^\star(a|s) - \pi_t(a|s)\right)\left(\phi_{s,a}^\top \hat{w}_t - \mathbb{E}_{a'\sim\pi_t(\cdot|s)}\left[\phi_{s,a'}^\top \hat{w}_t - Q_{\mu_t}^{\pi_t,\lambda}(s,a')\right] - Q_{\mu_t}^{\pi_t,\lambda}(s,a)\right) \\
=& -\eta_t \sum_{s,a} d_t^\star(s)\left(\pi_t^\star(a|s) - \pi_t(a|s)\right)\left(\hat{w}_t^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right), \qquad (10)
\end{aligned}$$

where the second step uses the fact that

$$\sum_a \left(\pi_t^\star(a|s) - \pi_t(a|s)\right)\mathbb{E}_{a'\sim\pi_t(\cdot|s)}\left[\phi_{s,a'}^\top \hat{w}_t - Q_{\mu_t}^{\pi_t,\lambda}(s,a')\right] = 0,$$

and the third step is again due to $\nabla \log \pi_t(a|s) = \phi_{s,a} - \sum_{a'\in\mathcal{A}} \pi_t(a'|s)\phi_{s,a'}$ and the definition of $A_{\mu_t}^{\pi_t,\lambda}(s,a)$. We define

$$\begin{aligned}
L_t(w) &\overset{\text{def}}{=} \mathbb{E}_{s\sim d_{\mu_t}^{\pi_t}, a\sim\pi_t(\cdot|s)}\left[\left(w^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right)^2\right], \\
\text{and } \hat{L}_t(w) &\overset{\text{def}}{=} \mathbb{E}_{s\sim d_{\mu_t}^{\pi_t}, a\sim\pi_t(\cdot|s)}\left[\left(w^\top \nabla \log \pi_t(a|s) - \hat{A}_t^\lambda(s,a)\right)^2\right],
\end{aligned}$$

where recall that $\hat{A}_t^\lambda(s,a) = \hat{Q}_t^\lambda(s,a) - \mathbb{E}_{a\sim\pi_t(\cdot|s)}[\hat{Q}_t^\lambda(s,a')]$ is an estimate of $A_{\mu_t}^{\pi_t,\lambda}(s,a)$ calculated using the policy evaluation oracle. From Jensen's inequality,

$$\begin{aligned}
& -\sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\left(\hat{w}_t^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right) \\
\le& \sum_{s,a} d_{\mu_t}^{\pi_t}(s)\pi_t(a|s)\sqrt{\left(\hat{w}_t^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right)^2}\cdot \frac{d_t^\star(s)}{d_{\mu_t}^{\pi_t}(s)}\cdot\frac{\pi_t^\star(a|s)}{\pi_t(a|s)} \\
\le& \frac{C_3}{p_{\min}}\sqrt{L_t(\hat{w}_t)}, \qquad (11)
\end{aligned}$$

where the last step uses Assumption 2 and the definition of $L_t(w)$. Recall that the policy evaluation oracle satisfies

$$\mathbb{E}\left[\left(\hat{q}_t^\lambda(s,a) - q_{\mu_t}^{\pi_t,\lambda}(s,a)\right)^2\right] \le \varepsilon_{\text{critic}}, \forall (s,a)\in\mathcal{S}\times\mathcal{A},$$

which immediately implies that

$$\mathbb{E}\left[\left(\hat{A}_t^\lambda(s,a) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right)^2\right] \le \varepsilon_{\text{critic}}, \forall (s,a)\in\mathcal{S}\times\mathcal{A}.$$

Let $w_t^\star = \operatorname{argmin}_{w\in\mathbb{R}^d} L_t(w)$. From the simple fact that $(x+y)^2 \le 2x^2 + 2y^2$, we have

$$\begin{aligned}
\hat{L}_t(w_t^\star) =& \mathbb{E}_{s\sim d_{\mu_t}^{\pi_t}, a\sim\pi_t(\cdot|s)}\left[\left((w_t^\star)^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a) + A_{\mu_t}^{\pi_t,\lambda}(s,a) - \hat{A}_t^\lambda(s,a)\right)^2\right] \\
\le& 2L_t(w_t^\star) + 2\varepsilon_{\text{critic}}.
\end{aligned}$$

A similar argument shows that

$$L_t(\hat{w}_t) \le 2\hat{L}_t(\hat{w}_t) + 2\varepsilon_{\text{critic}}.$$

Further, the gradient estimation oracle (3) guarantees that $\hat{L}_t(\hat{w}_t) - \min_w \hat{L}_t(w) \le \varepsilon_{\text{actor}}$, and we hence obtain

$$\begin{aligned}
L_t(\hat{w}_t) \le& 2\hat{L}_t(\hat{w}_t) + 2\varepsilon_{\text{critic}} \le 2\min_w \hat{L}_t(w) + 2\varepsilon_{\text{actor}} + 2\varepsilon_{\text{critic}} \\
\le& 2\hat{L}_t(w_t^\star) + 2\varepsilon_{\text{actor}} + 2\varepsilon_{\text{critic}} \le 4L_t(w_t^\star) + 2\varepsilon_{\text{actor}} + 6\varepsilon_{\text{critic}} \\
\le& 4\varepsilon_{\text{approx}} + 2\varepsilon_{\text{actor}} + 6\varepsilon_{\text{critic}},
\end{aligned}$$

where the last step follows from the definition of $\varepsilon_{\text{approx}}$. Plugging the above inequality back to (11), and using the fact that $\varepsilon_{\text{total}} = \varepsilon_{\text{approx}} + \varepsilon_{\text{actor}} + \varepsilon_{\text{critic}}$, we obtain that

$$-\sum_{s,a} d_t^\star(s)\pi_t^\star(a|s)\left(\hat{w}_t^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right) \leq \frac{3C_3}{p_{\min}}\sqrt{\varepsilon_{\text{total}}}. \tag{12}$$

Similarly, we can also get

$$\sum_{s,a} d_t^\star(s)\pi_t(a|s)\left(w_t^\top \nabla \log \pi_t(a|s) - A_{\mu_t}^{\pi_t,\lambda}(s,a)\right) \leq 3C_3\sqrt{\varepsilon_{\text{total}}}. \tag{13}$$

Substituting (10), (12), and (13) back to (9), we obtain that

$$\begin{aligned}\sigma_{t+1}^\pi \leq& (1-\eta_t\lambda)\sigma_t^\pi + 3C_3\eta_t(1+\frac{1}{p_{\min}})\sqrt{\varepsilon_{\text{total}}} + \frac{1}{2}\eta_t^2\left\|g_t\right\|_2^2 + (1+C_1)\cdot\kappa d_0\left\|\mu_{t+1}-\mu_t\right\|_1\\ \leq& (1-\eta_t\lambda)\sigma_t^\pi + 3C_3\eta_t(1+\frac{1}{p_{\min}})\sqrt{\varepsilon_{\text{total}}} + 2\eta_t^2R^2 + (1+C_1)\cdot\kappa d_0\left\|\mu_{t+1}-\mu_t\right\|_1,\end{aligned}$$

where the second step holds because $\|g_t\|_2 \leq \|w_t\|_2 + \lambda\|\theta_t\|_2 \leq 2R$ due to Lemma 2.. □

B.3 Proof for Theorem 1

Proof. First, we know from Lemma 8 that

$$\sigma_{t+1}^\pi \leq (1-\eta_t\lambda)\sigma_t^\pi + 3C_3\eta_t(1+\frac{1}{p_{\min}})\sqrt{\varepsilon_{\text{total}}} + 2\eta_t^2R^2 + (1+C_1)\cdot\kappa d_0\left\|\mu_{t+1}-\mu_t\right\|_1, \tag{14}$$

where $\kappa = \frac{2\log|\mathcal{A}|}{1-\gamma} + \frac{1+2R(1-\gamma)}{\lambda(1-\gamma)}$, and $p_{\min} \geq \frac{\exp(-2R/\lambda)}{|\mathcal{A}|}$. Using the mean-field state update rule that $\mu_{t+1} = (1-\beta_t)\mu_t + \beta_t\Gamma_2(\pi_t,\mu_t)$, we have that

$$\left\|\mu_{t+1}-\mu_t\right\|_1 = \beta_t\left\|\mu_t - \Gamma_2(\pi_t,\mu_t)\right\|_1 \leq 2\beta_t.$$

Substituting the above equation back to (14) and rearranging,

$$\sigma_t \leq \frac{1}{\eta_t\lambda}(\sigma_t - \sigma_{t+1}) + \frac{3C_3}{\lambda}(1+\frac{1}{p_{\min}})\sqrt{\varepsilon_{\text{total}}} + \frac{2\eta_tR^2}{\lambda} + \frac{2(1+C_1)\cdot\kappa d_0\beta_t}{\eta_t\lambda}$$

Let $\eta_t = \eta = O(T^{-2/5})/\lambda, \beta_t = \beta = O(T^{-4/5})$. Summing over $t = 0, 1, \ldots, T-1$ leads to

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\sigma_t^\pi \leq& \frac{\sigma_0}{T\eta\lambda} + \frac{3C_3}{\lambda}(1+\frac{1}{p_{\min}})\sqrt{\varepsilon_{\text{total}}} + \frac{2\eta R^2}{\lambda} + \frac{2(1+C_1)\cdot\kappa d_0\beta}{\eta\lambda}\\ \leq& \widetilde{O}\left(\frac{1}{\lambda^2T^{2/5}} + \frac{|\mathcal{A}|\exp(1/\lambda)}{\lambda}\sqrt{\varepsilon_{\text{total}}}\right).\end{aligned} \tag{15}$$

We can then apply the Cauchy-Schwarz inequality and Pinsker's inequality to obtain that

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} D(\pi_t,\pi_t^\star) =& \mathbb{E}_\tau\left[D(\pi_\tau,\pi_\tau^\star)\right]\\ =& \mathbb{E}_\tau\mathbb{E}_{s\sim d_\tau^\star}\left[\frac{d^\star(s)}{d_\tau^\star(s)}\cdot\left\|\pi_\tau^\star(\cdot|s)-\pi_\tau(\cdot|s)\right\|_1\right]\\ \leq& \sqrt{\mathbb{E}_\tau\mathbb{E}_{s\sim d_t^\star}\left[\left|\frac{d^\star}{d_\tau^\star}\right|^2\right]\cdot\mathbb{E}_\tau\mathbb{E}_{s\sim d_\tau^\star}\left[\left\|\pi_\tau^\star(\cdot|s)-\pi_\tau(\cdot|s)\right\|_1^2\right]}\\ \leq& C_2\sqrt{\mathbb{E}_\tau\mathbb{E}_{s\sim d_t^\star}\left[2\mathbf{KL}\left(\pi_\tau^\star(\cdot|s)\|\pi_\tau(\cdot|s)\right)\right]}\\ \leq& \widetilde{O}\left(\frac{1}{\lambda T^{1/5}} + \sqrt{\frac{|\mathcal{A}|\exp(1/\lambda)\varepsilon_{\text{total}}^{1/2}}{\lambda}}\right),\end{aligned}$$

where the last step follows from (15). This characterizes the convergence of the policy π_t. In the following, we follow a similar analysis as in [76] and analyze the convergence behavior of the mean-field state μ_t. From Lemma 7, we know that

$$\|\mu_{t+1} - \mu^\star\|_1 \leq (1 - \beta_t \bar{d}) \|\mu_t - \mu^\star\|_1 + d_2 C_2 \beta_t \sqrt{\sigma_t^\pi}, \forall t \geq 0.$$

Rearranging and using the definition that $\sigma_t^\mu = \|\mu_t - \mu^\star\|_1$, we have

$$\sigma_t^\mu \leq \frac{1}{\beta_t \bar{d}} (\sigma_t^\mu - \sigma_{t+1}^\mu) + \frac{d_2 C_2}{\bar{d}} \sqrt{\sigma_t^\pi}.$$

With $\beta_t = \beta = O(T^{-4/5})$, we sum over $t = 0, 1, \ldots, T-1$ and obtain

$$\begin{aligned}
\frac{1}{T} \sum_{t=0}^{T-1} \sigma_t^\mu \leq & \frac{1}{T\beta\bar{d}} (\sigma_0^\mu - \sigma_T^\mu) + \frac{d_2 C_2}{T\bar{d}} \sum_{t=0}^{T-1} \sqrt{\sigma_t^\pi} \\
\leq & \frac{\sigma_0^\mu}{T\beta\bar{d}} + \frac{d_2 C_2}{\bar{d}} \sqrt{\frac{1}{T} \sum_{t=0}^{T-1} \sigma_t^\pi} \\
\leq & \widetilde{O}\left(\frac{1}{T^{1/5}} + \sqrt{\frac{1}{\lambda^2 T^{2/5}} + \frac{|\mathcal{A}| \exp(1/\lambda)}{\lambda} \sqrt{\varepsilon_{\text{total}}}} \right) \qquad (16) \\
\leq & \widetilde{O}\left(\frac{1}{\lambda T^{1/5}} + \sqrt{\frac{|\mathcal{A}| \exp(1/\lambda) \varepsilon_{\text{total}}^{1/2}}{\lambda}} \right), \qquad (17)
\end{aligned}$$

where the second step uses the Cauchy-Schwarz inequality, and the third step follows from (15). Finally, using the triangle inequality,

$$\begin{aligned}
D(\pi_t, \pi^\star) \leq & D(\pi_t, \pi_t^\star) + D(\pi_t^\star, \pi^\star) \\
= & D(\pi_t, \pi_t^\star) + D(\Gamma_1^\lambda(\mu_t), \Gamma_1^\lambda(\mu^\star)) \\
\leq & D(\pi_t, \pi_t^\star) + d_1 \|\mu_t - \mu^\star\|_1, \qquad (18)
\end{aligned}$$

where the last step holds due to Assumption 3. Combining (14), (15), and (17),

$$\begin{aligned}
D\left(\pi^\star, \frac{1}{T} \sum_{t=0}^{T-1} \pi_t\right) + \left\| \mu^\star - \frac{1}{T} \sum_{t=0}^{T-1} \mu_t \right\|_1 \leq & \frac{1}{T} \sum_{t=0}^{T-1} D(\pi^\star, \pi_t) + \frac{1}{T} \sum_{t=0}^{T-1} \|\mu^\star - \mu_t\|_1 \\
\leq & \frac{1}{T} \sum_{t=0}^{T-1} (D(\pi_t, \pi_t^\star) + d_1 \|\mu_t - \mu^\star\|_1) + \frac{1}{T} \sum_{t=0}^{T-1} \|\mu^\star - \mu_t\|_1 \\
\leq & \widetilde{O}\left(\frac{1}{\lambda T^{1/5}} + \sqrt{\frac{|\mathcal{A}| \exp(1/\lambda) \varepsilon_{\text{total}}^{1/2}}{\lambda}} \right).
\end{aligned}$$

This completes the proof of the theorem. □

C Instantiation of the Oracles

Section 3 assumes access to two black-box oracles that can return relatively accurate evaluations of a policy and estimations of the policy gradient. In this appendix, we discuss possible ways that the two oracles can be instantiated using standard techniques.

We start with the policy evaluation oracle, which provides an $\varepsilon_{\text{critic}}$-accurate estimate $\hat{q}$ of the shifted Q-function q^π given a policy π. One viable approach is to instantiate such a critic oracle using temporal difference (TD) learning with linear function approximation [10, 70], a simple and widely used iterative method for policy evaluation. Specifically, we consider the case where the shifted Q-function is approximated as $q^\pi(s, a) = \psi^\top \phi_{s,a}$, where $\phi_{s,a} \in \mathbb{R}^d$ is the d-dimensional feature vector, and $\psi \in \mathbb{R}^d$ is the parameter vector to be optimized. The optimal ψ should minimize the mean-squared projected Bellman error. A formal description of the projected TD(0) algorithm is

Algorithm 2: Projected Temporal Difference Learning with Linear Function Approximation

Input: Policy π to be evaluated;
Initialize $\psi_0 \leftarrow \mathbf{0}$;
for *iteration* $k \leftarrow 0$ *to* $K-1$ **do**
  Execute policy π to collect sample $(s_k, a_k, r_k, s_{k+1}, a_{k+1})$;
  $\tilde{\psi}_{k+1} \leftarrow \psi_k + \alpha_k \left(r_k - \lambda \log \pi(a_k|s_k) + \gamma \psi_k^\top \phi_{s_{k+1},a_{k+1}} - \psi_k^\top \phi_{s_k,a_k}\right) \phi_{s_k,a_k}$;
  $\psi_{k+1} \leftarrow \operatorname{argmin}_{\psi \in \mathbb{R}^d, \|\psi\|_2 \le B} \|\psi - \tilde{\psi}_{k+1}\|_2^2$;
Output: $\hat{q}(s,a) = \frac{1}{K} \sum_{k=0}^{K-1} \phi_{s,a}^\top \psi_k$.

Algorithm 3: Stochastic Gradient Descent for Gradient Estimation

Input: Shifted Q-function estimates $\hat{q}$ from Algorithm 2;
Initialize $\bar{w}_0 \leftarrow \mathbf{0}$;
for *iteration* $k \leftarrow 0$ *to* $K-1$ **do**
  Sample $s_k \sim d^\pi$ and $a_k \sim \pi(\cdot|s_k)$ using a sampler;
  $\bar{w}_{k+1} \leftarrow \bar{w}_k - 2\bar{\alpha}_k \left(\bar{w}_k^\top \nabla \log \pi(a_k|s_k) - \hat{A}(s_k,a_k)\right) \nabla \log \pi(a_k|s_k)$;
  $\bar{w}_{k+1} \leftarrow \operatorname{argmin}_{w \in \mathbb{R}^d : \|w\|_2 \le R} \|w - \bar{w}_{k+1}\|_2^2$;
Output: $\hat{w} = \frac{1}{K} \sum_{k=0}^{K-1} \bar{w}_k$.

presented in Algorithm 2. It starts with an initial ψ_0 parameter. At each iteration k, it executes the given policy π, and observe a sample $O_k = (s_k, a_k, r_k, s_{k+1}, a_{k+1})$ of the current state and action, the current reward, and the next state and action. The algorithm then takes a step in the direction along the negative gradient of the squared Bellman error induced by the sample O_k. In the entropy-regularized case, it can be shown [13] that the negative gradient is expressed as

$$g_k = \left(r_k - \lambda \log \pi(a_k|s_k) + \gamma \psi^\top \phi_{s_{k+1},a_{k+1}} - \psi^\top \phi_{s_k,a_k}\right) \phi_{s_k,a_k}.$$

The algorithm further projects the parameter ψ back to a Euclidean ball of radius B to ensure that the gradient norms are uniformly bounded over time. Finally, Algorithm 2 outputs an estimate of the shifted Q-function using the averaged iterate of the parameter.

Under proper assumptions (realizability and uniform mixing of the induced Markov chain, see [10] for an extensive discussion), the finite-time convergence of projected TD learning with linear function approximation is characterized in the following proposition.

Proposition 1. *(Theorem 3 of [10]). Under certain regularity assumptions, Algorithm 2 with a decaying step size* $\alpha_k = \frac{1}{\omega(k+1)(1-\gamma)}$ *ensures that*

$$\mathbb{E}\left[\|\hat{q} - q^\pi\|_{d^\pi \times \pi}^2\right] \le \widetilde{O}\left(\frac{\tau^{mix}(\alpha_K)}{K(1-\gamma)^2\omega}\right),$$

where $\tau^{mix}(\varepsilon)$ *is the* ε*-mixing time of the induced Markov chain, and* ω *is the smallest eigenvalue of the steady-state feature covariance matrix* $\sum_{s,a} d^\pi(s)\pi(a|s)\phi_{s,a}\phi_{s,a}^\top$.

Therefore, in order to obtain an $\varepsilon_{\text{critic}}$-accurate estimate of the shifted Q-function in expectation, it suffices to run Algorithm 2 for $\widetilde{O}(1/\varepsilon_{\text{critic}}^2)$ iterations.

Next, we instantiate the gradient estimation oracle in Algorithm 1, which provides an $\varepsilon_{\text{actor}}$-accurate estimate $\hat{w}$ of the gradient w, given a policy π and an estimated value function $\hat{A}$. Since (3) solves a standard convex optimization problem, we can simply use a stochastic gradient descent (SGD) method for the actor update, which is formally described in Algorithm 3. Specifically, we first initialize the gradient estimate as $\bar{w}_0 = \mathbf{0}$. At each iteration k, Algorithm 3 takes a step along the opposite direction of the gradient of loss function. The gradient is given by

$$\bar{g}_k = 2\left(\bar{w}_k^\top \nabla \log \pi(a_k|s_k) - \hat{A}(s_k,a_k)\right) \nabla \log \pi(a_k|s_k),$$

where (s_k, a_k) is drawn from the distribution $d^\pi \times \pi$ (for simplicity of notations dropped the dependence on the population distribution) using a sampler (e.g., [2]), and $\hat{A}$ is calculated from $\hat{q}$

provided by the critic. The algorithm finally averages $\bar{w}_k$ over the iterations as the output. A standard result shows that Algorithm 3 with a learning step size of $\bar{\alpha}_k = \frac{R}{Q_{\max}\sqrt{K}}$ finds the optimum at a rate of $O(1/\sqrt{K})$, where recall that $Q_{\max} = \frac{1+\gamma\lambda \log|\mathcal{A}|}{1-\gamma}$:

Proposition 2. *(Combining Theorem 14.8 and Lemma 14.9 of [61]). Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, and let $x^\star = \operatorname{argmin}_{x\in\mathbb{R}^d:\|x\|_2\leq R} f(x)$. Assume that the gradient norm at each step is bounded by $\rho > 0$ with probability 1. Suppose the (projected) stochastic gradient descent algorithm is run for K iterations with the learning step size $\bar{\alpha}_k = \frac{R}{\rho\sqrt{K}}$. Then,*

$$\mathbb{E}\left[f\left(\frac{1}{K}\sum_{k=1}^{K} x_k\right)\right] - f\left(x^*\right) \leq \frac{B\rho}{\sqrt{K}}.$$

The above result immediately implies that, in order to obtain an $\varepsilon_{\text{actor}}$-accurate gradient estimation in expectation, it suffices to run Algorithm 3 for $O(1/\varepsilon^2_{\text{actor}})$ iterations.

D Simulations Setup

In Subsection 5.1, we adopt two classic mean-field game tasks from the literature, including an SIS epidemics model [16, 38], and a linear-quadratic MFG [49, 38, 12, 45]. Simulations are done in an episodic setting. In our implementation, we use the collected empirical trajectory to estimate the policy gradient and the Fisher information matrix (instead of formally calculating the state visitation distribution), which turns out to serve as accurate estimates of the true values. The mean-field states are not directly observed by the learning agent, but instead only influence the environment implicitly as a parameter of the transition and reward functions.

SIS Epidemics Model. The SIS task describes a toy mean-field game model for epidemics. In our simulations, we consider the same setting as has been proposed in [16]. This task has two states: susceptible (S) and infected (I). At each time step, each agent may choose between two actions: social distancing (D) or going out (U). A susceptible agent will not get infected if it practices social distancing, i.e., $P(s_{t+1} = I \mid s_t = S, a_t = D, \mu_t) = 0$. When a susceptible agent chooses to go out, it has a higher probability of becoming infected if a larger proportion of the population is infected. Specifically, the state transition is given by $P(s_{t+1} = I \mid S_t = S, A_t = U, \mu_t) = 0.9^2 \cdot \mu_t(I)$, where $\mu_t(I)$ denotes the ratio of the population that is infected at time step t. An infected agent has a constant probability of recovery at each step, regardless of its choice of action, i.e., $P(s_{t+1} = S \mid S_t = I, A_t = U, \mu_t) = P(s_{t+1} = S \mid S_t = I, A_t = D, \mu_t) = 0.3$. For each individual agent, both practicing social distancing and being in the infected state have an associated cost, regardless of the rest of the population. Specifically, the reward function is given by $r(s, a, \mu) = -\mathbb{1}\{s = I\} - 0.5 \cdot \mathbb{1}\{a = D\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function. It is worth remarking that even though this task has only two states, the transitions are also influenced by the population distribution, which is a real-valued quantity that makes this task significantly more challenging than a simple tabular MDP.

Linear-Quadratic MFG. The second task we consider is a 1D linear-quadratic mean-field game. We adopt the same discrete setting as has been utilized in [49, 38], which is in turn an approximation of the classic linear-quadratic MFG formulations [12, 45]. For each individual agent, the transition function of this task is given by:

$$s_{t+1} = s_t + a_t\Delta_t + \sigma\varepsilon_t\sqrt{\Delta_t},$$

where Δ_t is the time duration, and ε_t is the i.i.d noise taking values from $\{-3, \ldots, 3\}$ approximately following a normal distribution $\mathcal{N}(0, 1)$. Let $\bar{\mu}_t$ denote the empirical average of the population states at time step t. The reward function for each agent is then specified as:

$$r(s_t, a_t, \mu_t) = \left(-\frac{1}{2}|a_t|^2 + qa_t(\bar{\mu}_t - s_t) - \frac{\kappa}{2}|\bar{\mu}_t - s_t|^2\right)\Delta_t.$$

Intuitively, this reward function incentivizes agents to track and stay close to the mean state of the population (despite the random drift ε_t), but discourages agents from taking large-magnitude actions. We set the parameters as $\Delta_t = 1, \sigma = 1, q = 0.01, \kappa = 0.5$, and $|\mathcal{S}| = 25$. We do not consider terminal costs in our simulations.

Exploitability. We utilize the standard notion of exploitability [82, 49, 16] to measure the sub-optimality of the algorithm. Specifically, the exploitability of a policy π is defined as

$$\mathcal{E}(\pi) = \max_{\pi^\star} V_{\mu_\pi}^{\pi^\star,\lambda}(\rho) - V_{\mu_\pi}^{\pi,\lambda}(\rho),$$

where μ_π is the mean-field distribution generated by following the policy π, and recall that ρ is the initial state distribution. Intuitively, a higher degree of exploitability suggests that an individual agent can be much better off by deviating from the given policy. On the other hand, an exploitability of 0 suggests that the policy π and its induced mean-field distribution constitute a Nash equilibrium of the mean-field game.

Hyperparameter Configuration. In the task of SIS, we set $\beta_t = 0.01$ for NAC, and $\eta_t = 0.05$ for both NAC and DL-NAC. This corroborates our theoretical findings in Algorithm 1 that the policy should evolve at a faster rate than the mean-field estimate. The learning rate of the critic is set to 0.001 for both algorithms. In the task of LQ, we choose $\beta_t = 0.05$ for NAC, and $\eta_t = 0.1$, and a critic learning rate of 0.001 for both NAC and DL-NAC. We use dynamic programming and the model information to calculate the exploitabilities of the policies exactly, but our algorithms do not have access to these values as they reveal information about the underlying environment. All simulation results are averaged over 10 runs.

"Zigzags" for Fixed-Point Iterations. The "zigzag" fluctuations of DL-NAC in Figures 1 and 2 are due to the fact that double-loop methods update the mean-field abruptly: In each "segment" of the zigzag plot, DL-NAC fixes the population distribution and learns an approximately optimal policy with respect to it. At the end of the segment, DL-NAC abruptly updates the mean-field estimate by applying one step of the mean-field dynamics operator Γ_2 under the learned policy. Such an abrupt change in the environment dynamics nullifies the policies learned from the past, and the algorithm needs to learn a new policy from scratch. This accounts for the sharp spikes in the plots of DL-NAC. Similar patterns have also been observed in the literature [16] for other fixed-point iteration methods. Our online method hence enjoys a more smooth learning behavior than standard fixed-point iteration. It is also worth remarking that the seemingly converging behavior of DL-NAC on each zigzag segment does not imply its convergence to NE, because DL-NAC fixes the mean-field estimate for each segment and does not let it get closer to the equilibrium mean-field state. In fact, even if we run each segment of DL-NAC for a sufficiently long time, there will still be a non-zero exploitability gap due to the inherent error of the mean-field estimate.

E Serverless FaaS Platform Setup

OpenWhisk Cluster Setup. A serverless Function-as-a-Service (FaaS) platform runs functions in response to invocations (i.e., function requests) from end-users or clients. Since all serverless platforms have similar master-worker architectures, we choose to use an production-grade open-source serverless platform, OpenWhisk [22], and deploy a distributed OpenWhisk cluster on IBM Cloud with 22 VMs in us-south-2. OpenWhisk manages the infrastructure, servers and scaling using Docker containers. Figure 4 shows the architecture of a distributed OpenWhisk cluster and how RL agents work with the OpenWhisk cluster to manage resources of each function. The OpenWhisk cluster that we deploy consists of one master node and 21 worker nodes. The master node runs the API gateway (labeled as ❶), FaaS controller (❷), data store (❸), and other management modules. Each of the worker nodes (labeled as ❹) runs the function containers. All nodes have 8 cores and 16-32GB RAM, running Ubuntu 20.04 LTS. There is no interference from external jobs. We run the workload generator [59] and the RL agents from two separate nodes in the same cluster and use FaaSProfiler [59] to trace requests to measure function latencies.

Serverless Function Workflow. The FaaS controller creates function containers, allocates CPU and RAM for each function container, and assigns the containers to worker nodes. When end-user requests arrive via the API gateway, the controller distributes the requests to worker nodes. If the function code exists on the worker node, the worker node will execute the function after it receives a request and the execution results are written to the data store; otherwise, the worker node will first pull function code from the data store before function execution. A container is evicted after an idle timeout of 10 minutes (the default value set in OpenWhisk).

Serverless Workloads. We select diverse function benchmarks from widely used open-source FaaS benchmark suites [15, 59, 81][4]. These benchmarks include web applications (`HTML-Gen`, `Uploader`), machine learning model serving (`Sentiment-Analysis`,`Image-Inference`), multimedia applications (`Image-Resize, Compression`), scientific functions (`Primes`, `PageRank`, `Graph-BFT`, `Graph-MST`), and utility functions (`Base64`, `Markdown2HTML`). The basic description of each function benchmark is listed here:

i. `Base64`: Encode and decode an input string with the Base64 algorithm.
ii. `Primes`: Find the list of prime numbers less than 10^7.
iii. `Markdown2HTML`: Render a Base64 uploaded text string as HTML.
iv. `Sentiment-Analysis`: Generate a sentiment analysis score for the input text.
v. `Image-Resize`: Resize the input Base64-coded image with new sizes.
vi. `HTML-Gen`: Generate HTML files randomly from templates.
vii. `Uploader`: Upload a file from a given URL to Cloud storage.
viii. `Compression`: Compress given images and upload to Cloud storage.
ix. `Image-Inference`: Image recognition on a given image with a pre-trained ResNet-50 model.
x. `Page-Rank`: Calculates the Google PageRank for a specified graph.
xi. `Graph-BFT`: Traverse the given graph with breath-first search.
xii. `Graph-MST`: Generate the minimum spanning tree given a graph.

These function benchmarks have different runtime behaviors and resource demands in terms of CPU, memory, and I/O utilization. For example, `Image-Resize` and `Image-Inference` are computation-intensive functions; `Base64` and `Markdown2HTML` are memory-intensive functions; `Uploader` and `Compression` are I/O-bound functions; `Page-Rank` and `Graph-BFT/MST` are data-intensive functions (cpu- and memory-intensive). The functions are written in either Python or Java. Function-latency-based QoS objectives are defined on a per-function basis. In our experiments, we follow the common practice and use the 99th percentile latency when running in isolation on the serverless platform with 15% relaxation as the QoS latency. To drive the benchmarks, we sample and replay the function invocations from Azure function traces [60][5].

NAC Implementation. In the implementation of NAC-Linear and NAC-NN, due to the complexity of computing the Fisher information matrix in a large-scale environment, we use a standard gradient descent method with adaptive KL divergence penalty to approximate the policy update step, which leads to a similar procedure as the Proximal Policy Optimization algorithm [58]. We set the learning rate for both the actor and the critic networks to 3×10^{-4}. The discount factor is set to 0.99. NAC-NN has one hidden layer that consists of 64 hidden units. We set the mini-batch size and number of SGD epochs to be both 5. The reward coefficient α is set to 0.3, which results in the highest reward after convergence in our sensitivity study.

Comparison Baselines. We compare our approach with a heuristics-based approach ENSURE [66] and OpenWhisk's original resource manager. ENSURE allocates $R + c\sqrt{R}$ containers to a function with function request arrival rate R, scales the resources within a worker node based on a latency degradation threshold, and scales the number of worker nodes based on a memory capacity threshold. OpenWhisk sets CPU shares for each container proportional to its requested memory capacity and tries to place as many containers as possible on the same worker node to maximize the utilization. We do not include the comparison results with single-agent RL algorithms proposed for resource management (e.g., FIRM [51], MIRAS [79], and FaaSRank [80]) because these solutions are proposed under different assumptions from ours and it is hardly possible to make fair comparisons. Specially, single-agent RL solutions typically assume that the agent is in an isolated environment where there is only the application that the agent manages, but do not address the competitions for shared resources in a cluster. In fact, a very recent work [52] has shown that applying single-agent RL algorithms to the multi-agent domain leads to severe performance degradation due to the environment's non-stationarity, which makes the single-agent solution even worse than the heuristics-based baseline that we choose.

[4]The benchmark FaaSProfiler uses MIT License; [15] uses BSD 3-Clause License; [81] uses Mulan Permissive Software License.

[5]The dataset [60] uses the Creative602 Commons Attribution 4.0 International Public License