# GREEDY-GQ WITH VARIANCE REDUCTION: FINITE-TIME ANALYSIS AND IMPROVED COMPLEXITY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Greedy-GQ is a value-based reinforcement learning (RL) algorithm for optimal control. Recently, the finite-time analysis of Greedy-GQ has been developed under linear function approximation and Markovian sampling, and the algorithm is shown to achieve an $\epsilon$-stationary point with a sample complexity in the order of $\mathcal{O}(\epsilon^{-3})$. Such a high sample complexity is due to the large variance induced by the Markovian samples. In this paper, we propose a variance-reduced Greedy-GQ (VR-Greedy-GQ) algorithm for off-policy optimal control. In particular, the algorithm applies the SVRG-based variance reduction scheme to reduce the stochastic variance of the two time-scale updates. We study the finite-time convergence of VR-Greedy-GQ under linear function approximation and Markovian sampling and show that the algorithm achieves a much smaller bias and variance error than the original Greedy-GQ. In particular, we prove that VR-Greedy-GQ achieves an improved sample complexity that is in the order of $\mathcal{O}(\epsilon^{-2})$. We further compare the performance of VR-Greedy-GQ with that of Greedy-GQ in various RL experiments to corroborate our theoretical findings.

## 1 INTRODUCTION

In reinforcement learning (RL), an agent interacts with a stochastic environment following a certain policy and receives some reward, and it aims to learn an optimal policy that yields the maximum accumulated reward Sutton & Barto (2018). In particular, many RL algorithms have been developed to learn the optimal control policy, and they have been widely applied to various practical applications such as finance, robotics, computer games and recommendation systems Mnih et al. (2015; 2016); Silver et al. (2016); Kober et al. (2013).

Conventional RL algorithms such as Q-learning Watkins & Dayan (1992) and SARSA Rummery & Niranjan (1994) have been well studied and their convergence is guaranteed in the tabular setting. However, it is known that these algorithms may diverge in the popular off-policy setting under linear function approximation Baird (1995); Gordon (1996). To address this issue, the two time-scale Greedy-GQ algorithm was developed in Maei et al. (2010) for learning the optimal policy. This algorithm extends the efficient gradient temporal difference (GTD) algorithms for policy evaluation Sutton et al. (2009b) to policy optimization. In particular, the asymptotic convergence of Greedy-GQ to a stationary point has been established in Maei et al. (2010). More recently, Wang & Zou (2020) studied the finite-time convergence of Greedy-GQ under linear function approximation and Markovian sampling, and it is shown that the algorithm achieves an $\epsilon$-stationary point of the objective function with a sample complexity in the order of $\mathcal{O}(\epsilon^{-3})$. Such an undesirable high sample complexity is caused by the large variance induced by the Markovian samples queried from the dynamic environment. Therefore, we want to ask the following question.

- *Q1: Can we develop a variance reduction scheme for the two time-scale Greedy-GQ algorithm?*

In fact, in the existing literature, many recent work proposed to apply the variance reduction techniques developed in the stochastic optimization literature to reduce the variance of various TD learning algorithms for policy evaluation, e.g., Du et al. (2017); Peng et al. (2019); Korda & La (2015); Xu et al. (2020). Some other work applied variance reduction techniques to Q-learning algorithms,

e.g., Wainwright (2019); Jia et al. (2020). Hence, it is much desired to develop a variance-reduced Greedy-GQ algorithm for optimal control. In particular, as many of the existing variance-reduced RL algorithms have been shown to achieve an improved sample complexity under variance reduction, it is natural to ask the following fundamental question.

- *Q2: Can variance-reduced Greedy-GQ achieve an improved sample complexity under Markovian sampling?*

In this paper, we provide affirmative answers to these fundamental questions. Specifically, we develop a two time-scale variance reduction scheme for the Greedy-GQ algorithm by leveraging the SVRG scheme Johnson & Zhang (2013). Moreover, under linear function approximation and Markovian sampling, we prove that the proposed variance-reduced Greedy-GQ algorithm achieves an $\epsilon$-stationary point with an improved sample complexity $\mathcal{O}(\epsilon^{-2})$. We summarize our technical contributions as follows.

### 1.1 Our Contributions

We develop a variance-reduced Greedy-GQ (VR-Greedy-GQ) algorithm for optimal control in reinforcement learning. Specifically, the algorithm leverages the SVRG variance reduction scheme Johnson & Zhang (2013) to construct variance-reduced stochastic updates for updating the parameters in both time-scales.

We study the finite-time convergence of VR-Greedy-GQ under linear function approximation and Markovian sampling in the off-policy setting. Specifically, we show that VR-Greedy-GQ achieves an $\epsilon$-stationary point of the objective function $J$ (i.e., $\|\nabla J(\theta)\|^2 \leq \epsilon$) with a sample complexity in the order of $\mathcal{O}(\epsilon^{-2})$. Such a complexity result improves that of the original Greedy-GQ by a significant factor of $\mathcal{O}(\epsilon^{-1})$ Wang & Zou (2020). In particular, our analysis shows that the bias error caused by the Markovian sampling and the variance error of the stochastic updates are in the order of $\mathcal{O}(M^{-1}), \mathcal{O}(\eta_\theta M^{-1})$, respectively, where $\eta_\theta$ is the learning rate and $M$ corresponds to the batch size of the SVRG reference batch update. This shows that the proposed variance reduction scheme can significantly reduce the bias and variance errors of the original Greedy-GQ update (by a factor of $M$) and lead to an improved overall sample complexity.

The analysis logic of VR-Greedy-GQ partly follows that of the conventional SVRG, but requires substantial new technical developments. Specifically, we must address the following challenges. First, VR-Greedy-GQ involves two time-scale variance-reduced updates that are correlated with each other. Such an extension of the SVRG scheme to the two time-scale updates is novel and requires new technical developments. Specifically, we need to develop tight variance bounds for the two time-scale updates under Markovian sampling. Second, unlike the convex objective functions of the conventional GTD type of algorithms, the objective function of VR-Greedy-GQ is generally non-convex due to the non-stationary target policy. Hence, we need to develop new techniques to characterize the per-iteration optimization progress towards a stationary point under nonconvexity. In particular, to analyze the two time-scale variance reduction updates of the algorithm, we introduce a 'fine-tuned' Lyapunov function of the form $R_t^m = J(\theta_t^{(m)}) + c_t\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2$, where the parameter $c_t$ is fine-tuned to cancel other additional quadratic terms $\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2$ that are implicitly involved in the tracking error terms. The design of this special Lyapunov function is critical to establish the formal convergence of the algorithm. With these technical developments, we are able to establish an improved finite-time convergence rate and sample complexity for VR-Greedy-GQ.

### 1.2 Related Work

**Q-learning and SARSA with function approximation.** The asymptotic convergence of Q-learning and SARSA under linear function approximation were established in Melo et al. (2008); Perkins & Precup (2003), and their finite-time analysis were developed in Zou et al. (2019); Chen et al. (2019). However, these algorithms may diverge in off-policy training Baird (1995). Also, recent works focused on the Markovian setting. Various analysis techniques have been developed to analyze the finite-time convergence of TD/Q-learning under Markovian samples. Specifically, Wang et al. (2020) developed a multi-step Lyapunov analysis for addressing the biasedness of the stochastic approximation in Q-learning. Srikant & Ying (2019) developed a drift analysis to the linear stochas-

tic approximation problem. Besides the linear function approximation, the finite-time analysis of Q-learning under neural network function approximation is developed in Xu & Gu (2019).

**GTD algorithms.** The GTD2 and TDC algorithms were developed for off-policy TD learning. Their asymptotic convergence was proved in Sutton et al. (2009a;b); Yu (2017), and their finite-time analysis were developed recently in Dalal et al. (2018); Wang et al. (2017); Liu et al. (2015); Gupta et al. (2019); Xu et al. (2019). The Greedy-GQ algorithm is an extension of these algorithms to optimal control and involves nonlinear updates.

**RL with variance reduction:** Variance reduction techniques have been applied to various RL algorithms. In TD learning, Du et al. (2017) reformulate the MSPBE problem as a convex-concave saddle-point optimization problem and applied SVRG Johnson & Zhang (2013) and SAGA Defazio et al. (2014) to primal-dual batch gradient algorithm. In Korda & La (2015), the variance-reduced TD algorithm was introduced for solving the MSPBE problem, and later Xu et al. (2020) provided a correct non-asymptotic analysis for this algorithm over Markovian samples. Recently, some other works applied the SVRG , SARAH Nguyen et al. (2017) and SPIDER Fang et al. (2018) variance reduction techniques to develop variance-reduced Q-learning algorithms, e.g., Wainwright (2019); Jia et al. (2020). In these works, TD or TDC algorithms are in the form of linear stochastic approximation, and Q-learning has only a single time-scale update. As a comparison, our VR-Greedy-GQ takes nonlinear two time-scale updates to optimization a nonconvex MSPBE.

## 2 Preliminaries: Policy Optimization and Greedy-GQ

In this section, we review some preliminaries of reinforcement learning and recap the Greedy-GQ algorithm under linear function approximation.

### 2.1 Policy Optimization in Reinforcement Learning

In reinforcement learning, an agent takes actions to interact with the environment via a Markov Decision Process (MDP). Specifically, an MDP is specified by the tuple $(\mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ respectively correspond to the state and action spaces that include finite elements, $r : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0, +\infty)$ denotes a reward function and $\gamma \in (0, 1)$ is the associated reward discount factor.

At any time $t$, assume that the agent is in the state $s_t \in \mathcal{S}$ and takes a certain action $a_t \in \mathcal{A}$ following a stationary policy $\pi$, i.e., $a_t \sim \pi(\cdot|s_t)$. Then, at the subsequent time $t+1$, the current state of the agent transfers to a new state $s_{t+1}$ according to the transition kernel $\mathbf{P}(\cdot|s_t, a_t)$. At the same time, the agent receives a reward $r_t = r(s_t, a_t, s_{t+1})$ from the environment for this action-state transition. To evaluate the quality of a given policy $\pi$, we often use the action-state value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that accumulates the discounted rewards as follows:

$$Q^\pi(s, a) = \mathbb{E}_{s'\sim\mathbf{P}(\cdot|s,a)} \left[r(s, a, s') + \gamma V^\pi(s')\right],$$

where $V^\pi(s)$ is the state value function defined as $V^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\Big]$. In particular, define the Bellman operator $T^\pi$ such that $T^\pi Q(s, a) = \mathbb{E}_{s',a'}[r(s, a, s') + \gamma Q(s', a')]$ for any $Q(s, a)$, where $a' \sim \pi(\cdot|s')$. Then, $Q^\pi(s, a)$ is a fixed point of $T^\pi$, i.e.,

$$T^\pi Q^\pi(s, a) = Q^\pi(s, a), \quad \forall s, a. \tag{1}$$

The goal of policy optimization is to learn the optimal policy $\pi^*$ that maximizes the expected total reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$ for any initial state $s \in \mathcal{S}$, and this is equivalent to learn the optimal value function $Q^*(s, a) = \sup_\pi Q^\pi(s, a), \forall s, a$. In particular, $Q^*$ is a fixed point of the Bellman operator $T$ that is defined as $TQ(s, a) = \mathbb{E}_{s'\sim\mathbf{P}(\cdot|s,a)}[r(s, a, s') + \gamma \max_{b\in\mathcal{A}} Q(s', b)]$.

### 2.2 Greedy-GQ with Linear Function Approximation

The Greedy-GQ algorithm is inspired by the fixed point characterization in eq. (1), and in the tabular setting it aims to minimize the Bellman error $\|T^\pi Q^\pi - Q^\pi\|^2_{\mu_{s,a}}$. Here, $\|\cdot\|^2_{\mu_{s,a}}$ is induced by the state-action stationary distribution $\mu_{s,a}$ (induced by the behavior policy $\pi_b$), and is defined as $\|Q\|^2_{\mu_{s,a}} = \mathbb{E}_{(s,a)\sim\mu_{s,a}}[Q(s, a)^2]$.

In practice, the state and action spaces may include a large number of elements that makes tabular approach infeasible. To address this issue, function approximation technique is widely applied. In this paper, we consider approximating the state-action value function $Q(s,a)$ by a linear function. Specifically, consider a set of basis functions $\{\phi^{(i)}: \mathcal{S}\times\mathcal{A}\to\mathbb{R},\ i=1,2,\ldots,d\}$, each of which maps a given state-action pair to a certain value. Define $\phi_{s,a}=[\phi^{(1)}(s,a);...;\phi^{(d)}(s,a)]$ as the feature vector for $(s,a)$. Then, under linear function approximation, the value function $Q(s,a)$ is approximated by $Q_\theta(s,a)=\phi_{s,a}^\top\theta$, where $\theta\in\mathbb{R}^d$ denotes the parameter of the linear approximation. Consequently, Greedy-GQ aims to find the optimal $\theta^*$ that minimizes the following mean squared projected Bellman error (MSPBE).

$$\text{(MSPBE):}\quad J(\theta):=\frac{1}{2}\|\Pi T^{\pi_\theta}Q_\theta-Q_\theta\|_{\mu_{s,a}}^2, \tag{2}$$

where $\mu_{s,a}$ is the stationary distribution induced by the behavior policy $\pi_b$, $\Pi$ is a projection operator that maps an action-value function $Q$ to the space $\mathcal{Q}$ spanned by the feature vectors, i.e., $\Pi Q=\arg\min_{U\in\mathcal{Q}}\|U-Q\|_{\mu_{s,a}}$. Moreover, the policy $\pi_\theta$ is parameterized by $\theta$. In this paper, we consider the class of Lipschitz and smooth policies (see Assumption 4.2).

Next, we introduce the Greedy-GQ algorithm. Define $\overline{V}_{s'}(\theta)=\sum_{a'\in\mathcal{A}}\pi_\theta(a'|s')\phi_{s',a'}^\top\theta$, $\delta_{s,a,s'}(\theta)=r(s,a,s')+\gamma\overline{V}_{s'}(\theta)-\phi_{s,a}^\top\theta$ and denote $\widehat{\phi}_s(\theta)=\nabla\overline{V}_s(\theta)$. Then, the gradient of the objective function $J(\theta)$ in eq. (2) is expressed as

$$\nabla J(\theta)=-\mathbb{E}[\delta_{s,a,s'}(\theta)\phi_{s,a}]+\gamma\mathbb{E}[\widehat{\phi}_{s'}(\theta)\phi_{s,a}^\top]\omega^*(\theta),$$

where $\omega^*(\theta)=\mathbb{E}[\phi_{s,a}\phi_{s,a}^\top]^{-1}\mathbb{E}[\delta_{s,a,s'}(\theta)\phi_{s,a}]$. To address the double-sampling issue when estimating the product of expectations involved in $\mathbb{E}[\widehat{\phi}_{s'}(\theta)\phi_{s,a}^\top]\omega^*(\theta)$, Sutton et al. (2009a) applies a weight doubling trick and constructs the following two time-scale update rule for the Greedy-GQ algorithm: for every $t=0,1,2,...$, sample $(s_t,a_t,r_t,s_{t+1})$ using the behavior policy $\pi_b$ and do

$$\text{(Greedy-GQ):}\quad \begin{cases}\theta_{t+1}=\theta_t-\eta_\theta\big(-\delta_{t+1}(\theta_t)\phi_t+\gamma(\omega_t^\top\phi_t)\widehat{\phi}_{t+1}(\theta_t)\big),\\ \omega_{t+1}=\omega_t-\eta_\omega\big(\phi_t^\top\omega_t-\delta_{t+1}(\theta_t)\big)\phi_t,\\ \pi_{\theta_{t+1}}=\mathcal{P}(\phi^\top\theta_{t+1}).\end{cases} \tag{3}$$

where $\eta_\theta,\eta_\omega>0$ are the learning rates and we denote $\delta_{t+1}(\theta):=\delta_{s_t,a_t,s_{t+1}}(\theta),\phi_t:=\phi_{s_t,a_t}$, $\widehat{\phi}_{t+1}(\theta_t):=\widehat{\phi}_{s_{t+1}}(\theta_t)$ for simplicity. To elaborate, the first two steps correspond to the two time-scale updates for updating the value function $Q_\theta$, whereas the last step is a policy improvement operation that exploits the updated value function to improve the target policy, e.g., greedy, $\epsilon$-greedy, softmax and mellowmax Asadi & Littman (2017).

The above Greedy-GQ algorithm uses a single Markovian sample to perform the two time-scale updates in each iteration. Such a stochastic Markovian sampling often induces a large variance that significantly slows down the overall convergence. This motivates us to develop variance reduction schemes for the two time-scale Greedy-GQ in the next section.

## 3 Greedy-GQ with Variance Reduction

In this section, we propose a variance-reduced Greedy-GQ (VR-Greedy-GQ) algorithm under Markovian sampling by leveraging the SVRG variance reduction scheme Johnson & Zhang (2013). To simplify notations, we define the stochastic updates regarding a sample $x_t=(s_t,a_t,r_t,s_{t+1})$ used in the Greedy-GQ as follows:

$$G_{x_t}(\theta,\omega):=-\delta_{t+1}(\theta)\phi_t+\gamma(\omega^\top\phi_t)\widehat{\phi}_{t+1}(\theta),$$
$$H_{x_t}(\theta,\omega):=\big(\phi_t^\top\omega-\delta_{t+1}(\theta)\big)\phi_t.$$

Next, consider a single MDP trajectory $\{x_t\}_{t\geq 0}$ obtained by the behavior policy $\pi_b$. In particular, we divide the entire trajectory into multiple batches of samples $\{\mathcal{B}_m\}_{m\geq 1}$ so that $\mathcal{B}_m=\{x_{(m-1)M},...,x_{mM-1}\}$, and our proposed VR-Greedy-GQ uses one batch of samples in every epoch. To elaborate, in the $m$-th epoch, we first initialize this epoch with a pair of reference points $\theta_0^{(m)}=\widetilde{\theta}^{(m)},\omega_0^{(m)}=\widetilde{\omega}^{(m)}$, where $\widetilde{\theta}^{(m)},\widetilde{\omega}^{(m)}$ are set to be the output points $\theta_M^{(m-1)},\omega_M^{(m-1)}$ of

the previous epoch, respectively. Then, we compute a pair of reference batch updates using the reference points and the batch of samples as follows

$$\widetilde{G}^{(m)} = \frac{1}{M}\sum_{k=(m-1)M}^{mM-1} G_{x_k}(\widetilde{\theta}^{(m)}, \widetilde{\omega}^{(m)}), \quad \widetilde{H}^{(m)} = \frac{1}{M}\sum_{k=(m-1)M}^{mM-1} H_{x_k}(\widetilde{\theta}^{(m)}, \widetilde{\omega}^{(m)}). \tag{4}$$

In the $t$-th iteration of the $m$-th epoch, we first query a random sample $x_{\xi_t^m}$ from the batch $\mathcal{B}_m$ uniformly with replacement (i.e., sample $\xi_t^m$ from $\{(m-1)M, ..., mM-1\}$ uniformly). Then, we use this sample to compute the stochastic updates $G_{x_{\xi_t^m}}, H_{x_{\xi_t^m}}$ at both of the points $(\theta_t^{(m)}, \omega_t^{(m)}), (\widetilde{\theta}^{(m)}, \widetilde{\omega}^{(m)})$. After that, we use these stochastic updates and the reference batch updates to construct the variance-reduced updates in Algorithm 1 via the SVRG scheme, where for simplicity we denote the stochastic updates $G_{x_{\xi_t^m}}, H_{x_{\xi_t^m}}$ respectively as $G_t^{(m)}, H_t^{(m)}$. In particular, we project the two time-scale updates onto the Euclidean ball with radius $R$ to stabilize the algorithm updates, and we assume that $R$ is large enough to include at least one stationary point of $J$. Lastly, we further update the policy via the policy improvement operation $\mathcal{P}$.

**Algorithm 1:** Variance-Reduced Greedy-GQ

**Input:** learning rates $\eta_\theta$, $\eta_\omega$, batch size $M$.
**Initialize:** $\widetilde{\theta}^{(1)} = \theta_0, \widetilde{\omega}^{(1)} = \omega_0, \pi_{\widetilde{\theta}^{(1)}} \leftarrow \mathcal{P}(\phi^\top \widetilde{\theta}^{(1)})$.
**for** $m = 1, 2, \ldots$ **do**
  $\theta_0^{(m)} = \widetilde{\theta}^{(m)}, \omega_0^{(m)} = \widetilde{\omega}^{(m)}$. Compute $\widetilde{G}^{(m)}, \widetilde{H}^{(m)}$ according to eq. (4).
  **for** $t = 0, 1, \ldots, M-1$ **do**
    Query a sample from $\mathcal{B}_m$ with replacement.
    $\theta_{t+1}^{(m)} = \Pi_R\Big[\theta_t^{(m)} - \eta_\theta\big(G_t^{(m)}(\theta_t^{(m)}, \omega_t^{(m)}) - G_t^{(m)}(\widetilde{\theta}^{(m)}, \widetilde{\omega}^{(m)}) + \widetilde{G}^{(m)}\big)\Big]$.
    $\omega_{t+1}^{(m)} = \Pi_R\Big[\omega_t^{(m)} - \eta_\omega\big(H_t^{(m)}(\theta_t^{(m)}, \omega_t^{(m)}) - H_t^{(m)}(\tilde{\theta}^{(m)}, \tilde{\omega}^{(m)}) + \widetilde{H}^{(m)}\big)\Big]$.
    **Policy improvement**: $\pi_{\theta_{t+1}^{(m)}} \leftarrow \mathcal{P}(\phi^\top \theta_t^{(m)})$.
  **end**
  Set $\widetilde{\theta}^{(m+1)} = \theta_M^{(m)}, \widetilde{\omega}^{(m+1)} = \omega_M^{(m)}$.
**end**
**Output:** parameter $\theta$ chosen among $\{\theta_t^{(m)}\}_{t,m}$ uniformly at random.

The above VR-Greedy-GQ algorithm has several advantages and uniqueness. First, it takes incremental updates that use a single Markovian sample per-iteration. This makes the algorithm sample efficient. Second, VR-Greedy-GQ applies variance reduction to both of the two time-scale updates. As we show later in the analysis, such a two time-scale variance reduction scheme significantly reduces the variance error of both of the stochastic updates.

We want to further clarify the incrementalism and online property of VR-Greedy-GQ. Our VR-Greedy-GQ is based on the online-SVRG and can be viewed as an incremental algorithm with regard to the batches of samples used in the outer-loops, i.e., in every outer-loop the algorithm samples a new batch of samples and use them to perform variance reduction in the corresponding inner-loops. Therefore, VR-Greedy-GQ can be viewed as an online batch-incremental algorithm. In general, there is a trade-off between incrementalism and variance reduction for SVRG-type algorithms: a larger batch size in the outer-loops enhances the effect of variance reduction, while a smaller batch size makes the algorithm more incremental.

## 4 Finite-Time Analysis of VR-Greedy-GQ

In this section, we analyze the finite-time convergence rate of VR-Greedy-GQ. We adopt the following standard technical assumptions from Wang & Zou (2020); Xu et al. (2020).

**Assumption 4.1** (Feature boundedness). *The feature vectors are uniformly bounded, i.e.,* $\|\phi_{s,a}\| \leq 1$ *for all* $(s, a) \in \mathcal{S} \times \mathcal{A}$.

**Assumption 4.2** (Policy smoothness). *The mapping* $\theta \mapsto \pi_\theta$ *is* $k_1$*-Lipschitz and* $k_2$*-smooth.*

We note that the above class of smooth policies covers a variety of practical policies, including softmax and mellowmax policies Asadi & Littman (2017); Wang & Zou (2020).

**Assumption 4.3** (Problem solvability). *The matrix $C := \mathbb{E}[\phi_{s,a}\phi_{s,a}^\top]$ is non-singular.*

**Assumption 4.4** (Geometric uniform ergodicity). *There exists $\Lambda > 0$ and $\rho \in (0,1)$ such that*

$$\sup_{s\in\mathcal{S}} d_{TV}\big(\mathbb{P}(s_t|s_0 = s), \mu\big) \leq \Lambda\rho^t,$$

*for any $t > 0$, where $d_{TV}$ is the total-variation distance.*

Based on the above assumptions, we obtain the following finite-time convergence rate result.

**Theorem 4.5** (Finite-time convergence). *Let Assumptions 4.1– 4.4 hold and consider the VR-Greedy-GQ algorithm. Choose learning rates $\eta_\theta$, $\eta_\omega$ and the batch size $M$ that satisfy the conditions specified in eqs. (15) to (19). Then, after $T$ epochs, the output of the algorithm satisfies*

$$\mathbb{E}\|\nabla J(\theta_\xi^{(\zeta)})\|^2 \leq \mathcal{O}\Big(\frac{1}{\eta_\theta TM} + \frac{1}{T}\big(\eta_\omega + \frac{\eta_\theta^2}{\eta_\omega^2}\big) + \big(\eta_\omega + \frac{\eta_\theta^2}{\eta_\omega^2}\big)^2 + \frac{1}{M}\Big),$$

*where $\xi, \zeta$ are random indexes that are sampled from $\{0, ..., M-1\}$ and $\{1, ..., T\}$ uniformly at random, respectively.*

Theorem 4.5 shows that VR-Greedy-GQ asymptotically converges to a neighborhood of a stationary point at a sublinear rate. In particular, the size of the neighborhood is in the order of $\mathcal{O}(M^{-1} + \eta_\theta^4\eta_\omega^{-4} + \eta_\omega^2)$, which can be driven arbitrarily close to zero by choosing a large batch size and sufficiently small learning rates that satisfy the two time-scale condition $\eta_\theta/\eta_\omega \to 0$. Moreover, the convergence error terms implicitly include a bias error $\mathcal{O}(\frac{1}{M})$ caused by the Markovian sampling and a variance error $\mathcal{O}(\frac{\eta_\theta}{M})$ caused by the stochastic updates, both of which are substantially reduced by the large batch size $M$. This shows that the SVRG scheme can effectively reduce the bias and variance error of the two time-scale stochastic updates.

By further optimizing the choice of hyper-parameters, we obtain the following characterization of sample complexity of VR-Greedy-GQ.

**Corollary 4.6** (Sample complexity). *Under the same conditions as those of Theorem 4.5, choose learning rates so that $\eta_\theta = \mathcal{O}(\frac{1}{M}), \eta_\omega = \mathcal{O}(\eta_\theta^{2/3})$,, and set $T, M = \mathcal{O}(\epsilon^{-1})$). Then, the required sample complexity for achieving $\mathbb{E}\|\nabla J(\theta_\xi^{(\zeta)})\|^2 \leq \epsilon$ is in the order of $TM = \mathcal{O}(\epsilon^{-2})$.*

Such a complexity result is orderwise lower than the complexity $\mathcal{O}(\epsilon^{-3})$ of the original Greedy-GQ Wang & Zou (2020). Therefore, this demonstrates the advantage of applying variance reduction to the two time-scale updates of VR-Greedy-GQ. We also note that for online stochastic non-convex optimization, the sample complexity of the SVRG algorithm is in the order of $\mathcal{O}(\epsilon^{-5/3})$ Li & Li (2018), which is slightly better than our result. This is reasonable as the SVRG in stochastic optimization is unbiased due to the i.i.d. sampling. In comparison, VR-Greedy-GQ works on a single MDP trajectory that induces Markovian noise, and the two-timescale updates of the algorithm also introduces additional tracking error.

## 5 Sketch of the Technical Proof

In this section, we provide an outline of the technical proof of the main Theorem 4.5 and highlight the main technical contributions. The details of the proof can be found in the appendix.

We note that our proof logic partly follows the that of the conventional SVRG, i.e., exploiting the objective function smoothness and introducing a Lyapunov function. However, our analysis requires substantial new developments to address the challenges of off-policy control, two time-scale updates of VR-Greedy-GQ and correlation of Markovian samples.

The key step of the proof is to develop a proper Lyapunov function that drives the parameter to a stationary point along the iterations. In addition, we also need to develop tight bounds for the bias error, variance error and tracking error. We elaborate the key steps of the proof below.

- Step 1: Construct an appropriate Lyapunov function $R_t^m$ and develop its upper bound.

- Step 2: Choose a proper parameterization $c_t$ of the Lyapunov function $R_t^m$. Bound the per-iteration progress $R_{t+1}^m - R_t^m$ and sum up the progress over one epoch.
- Step 3: Develop a tight bound for the tracking error.
- Step 4: Combine all the bounds to obtain the finite-time convergence rate.

**Step 1:** We first define the following Lyapunov function with certain $c_t > 0$ to be determined later.

$$R_t^m := J(\theta_t^{(m)}) + c_t \|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2. \tag{5}$$

To explain the motivation, note that unlike the analysis of variance-reduced TD learning Xu et al. (2020) where the term $\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2$ can be decomposed into $\|\theta_t^{(m)} - \theta^*\|^2 + \|\widetilde{\theta}^{(m)} - \theta^*\|^2$ to get the desired upper bound, here we do not have $\theta^*$ due to the non-convexity of $J(\theta)$. Hence, we need to properly merge this term into the Lyapunov function $R_t^m$. By leveraging the smoothness of $J(\theta)$ and the algorithm update rule, we obtain the following bound for the Lyapunov function $R_t^m$ (see eq. (10) in the appendix for the details).

$$\begin{aligned}\mathbb{E}[R_{t+1}^m] \leq \mathbb{E}[J(\theta_t^{(m)})] + \mathcal{O}\big(\mathbb{E}[\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2] + \mathbb{E}[\|\nabla J(\theta_t^{(m)})]\|^2 + M^{-1}\big) \\ + \mathcal{O}\big(\mathbb{E}[\|\widetilde{\omega}^{(m)} - \omega^*(\widetilde{\theta}^{(m)})\|^2] + \mathbb{E}[\|\omega_t^{(m)} - \omega^*(\theta_t^{(m)})\|^2]\big).\end{aligned} \tag{6}$$

In particular, the error term $\frac{1}{M}$ is due to the noise of Markovian sampling and the variance of the stochastic updates, and the last two terms correspond to tracking errors.

**Step 2:** To telescope the Lyapunov function over $t$ based on eq. (6), one may want to define $J(\theta_t^{(m)})] + \mathcal{O}\big(\mathbb{E}[\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2]\big) = R_t^m$ by choosing a proper $c_t$ of $R_t^m$. However, note that eq. (6) involves the last two tracking error terms, which also implicitly depend on $\mathbb{E}\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2$ as we show later in the Step 3. Therefore, we need to carefully define the $c_t$ of $R_t^m$ so that after applying the tracking error bounds developed in the Step 3, the right hand side of eq. (6) can yield an $R_t^m$ without involving the term $\mathbb{E}\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2$. It turns out that we need to define $c_t$ via the recursion specified in eq. (11) in the appendix. We rigorously show that the sequence $\{c_t\}_t$ is uniformly bounded by a small constant $\widehat{c} = \frac{1}{8}$. Then, plugging these bounds into eq. (6) and summing over one epoch, we obtain the following bound (see eq. (13) in the appendix for the details).

$$\begin{aligned}\eta_\theta \sum_{t=0}^{M-1} \mathbb{E}[\|\nabla J(\theta_t^{(m)})\|^2] \leq \mathbb{E}[R_0^m] - \mathbb{E}[R_M^m] + \mathcal{O}\big(\eta_\theta + \eta_\theta^2 M \mathbb{E}[\|\widetilde{\omega}^{(m)} - \omega^*(\widetilde{\theta}^{(m)})\|^2]\big) \\ + \mathcal{O}\Big(\eta_\theta \sum_{t=0}^{M-1} \mathbb{E}\|\omega_t^{(m)} - \omega^*(\theta_t^{(m)})\|^2 - \eta_\theta \widehat{c}\Big(\eta_\omega + \frac{\eta_\theta^2}{\eta_\omega^2}\Big) \sum_{t=0}^{M-1} \mathbb{E}[\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2]\Big).\end{aligned}$$

**Step 3:** We derive bounds for the tracking error terms $\sum_{t=0}^{M-1} \mathbb{E}\|\omega_t^{(m)} - \omega^*(\theta_t^{(m)})\|^2$ and $\mathbb{E}\|\widetilde{\omega}^{(m)} - \omega^*(\widetilde{\theta}^{(m)})\|^2$ in the above bound in Lemma D.7 and Lemma D.8.

**Step 4:** Lastly, by substituting the tracking error bounds obtained in Step 3 into the bound obtained in Step 2, the resulting bound does not involve the term $\sum_{t=0}^{M-1} \mathbb{E}\|\theta_t^{(m)} - \widetilde{\theta}^{(m)}\|^2$. Then, summing this bound over the epochs $m = 1, ..., T$, we obtain the desired finite-time convergence rate result.

## 6 Experiments

In this section, we conduct two reinforcement learning experiments, namely, Garnet problem Archibald et al. (1995) and Frozen Lake game Brockman et al. (2016), to test the performance of VR-Greedy-GQ in the off-policy setting, and compare it with Greedy-GQ in the Markovian setting.

### 6.1 Garnet Problem

For the Garnet problem, we refer to Appendix F for the details of the problem setup. We set the default learning rates as $\eta_\theta = 0.02$ and $\eta_\omega = 0.01$ for both VR-Greedy-GQ and Greedy-GQ algorithm. For VR-Greedy-GQ, we set the default batch size as $M = 3000$.

In Figure 1 (left), we plot the minimum gradient norm *v.s.* the number of pseudo stochastic gradient computations for both algorithms using $40$ Garnet MDP trajectories, and each trajectory contains $10$k samples. The upper and lower envelopes of the curves correspond to the $95\%$ and $5\%$ percentiles of the $40$ curves, respectively. It can be seen that VR-Greedy-GQ outperforms Greedy-GQ and achieves a significantly smaller asymptotic gradient norm.

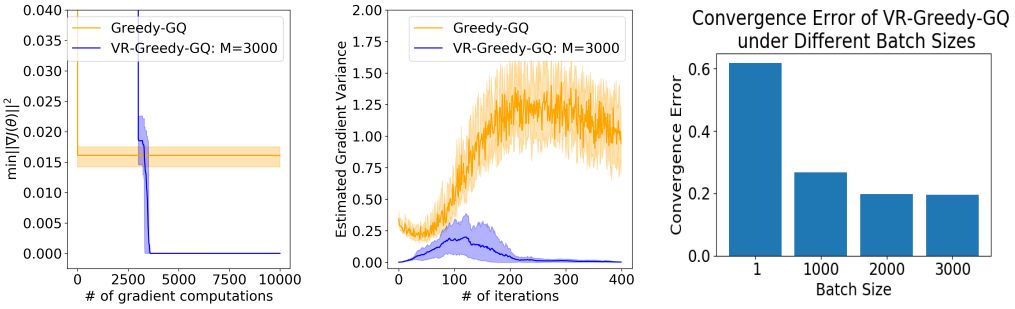


Figure 1: Comparison of Greedy-GQ and VR-Greedy-GQ in solving the Garnet problem.

In Figure 1 (middle), we track the estimated variance of the stochastic update for both algorithms along the iterations. Specifically, we query $500$ Monte Carlo samples per iteration to estimate the pseudo gradient variance $\mathbb{E}\|G_t^{(m)}(\theta_t^{(m)}, \omega_t^{(m)}) - \nabla J(\theta_t^{(m)})\|^2$. It can be seen from the figure that the stochastic updates of VR-Greedy-GQ induce a much smaller variance than Greedy-GQ. This demonstrates the effectiveness of the two time-scale variance reduction scheme of VR-Greedy-GQ.

We further study the asymptotic convergence error of VR-Greedy-GQ under different batch sizes $M$. We use the default learning rate setting that is mentioned previously and run $100$k iterations for one Garnet trajectories. We use the mean of the convergence error of the last $10$k iterations as an estimate of the asymptotic convergence error (the training curves are already saturated and flattened). Figure 1 (right) shows the asymptotic convergence error of VR-Greedy-GQ under different batch sizes $M$. It can be seen that VR-Greedy-GQ achieves a smaller asymptotic convergence error with a larger batch size, which matches our theoretical result.

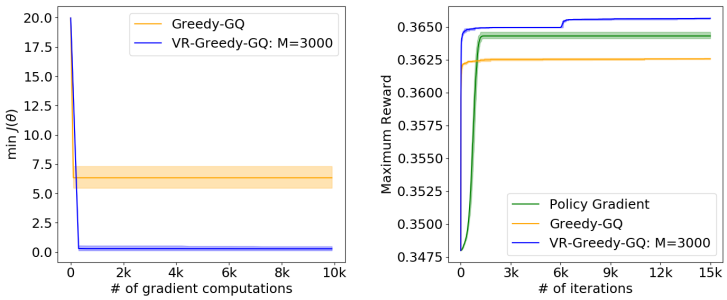


Figure 2: Comparison of MSPBE and reward obtained by Greedy-GQ, VR-Greedy-GQ and PG.

Lastly, we plot the MSPBE $J(\theta)$ *v.s.* number of gradient computations for both Greedy-GQ and VR-Greedy-GQ in Figure 2 (Left), where one can see that VR-Greedy-GQ achieves a much smaller MSPBE than Greedy-GQ. In addition, we also plot the estimated expected maximum reward (see Appendix F for details) *v.s.* number of iterations for Greedy-GQ, VR-Greedy-GQ and policy gradient (PG) in Figure 2 (Right). For the policy gradient, we apply the standard off-policy policy gradient algorithm. For each update, we sample $30$ independent trajectories with a fixed length $60$ to estimate the expected discounted return. The learning rate of policy gradient is set as $\eta_\theta$. We note that each iteration of policy gradient consumes $1800$ samples and hence it is very sample inefficient. Hence we set the x-axis to be number of iterations for a clear presentation. One can see that VR-Greedy-GQ achieves a much higher expected reward than both Greedy-GQ and policy gradient.

### 6.2 FROZEN LAKE GAME

We further test these algorithms in solving the more complex frozen lake game. We generate a Gaussian feature matrix with dimension 8 to linearly approximate the value function and we aim to evaluate a target policy based on a behavior policy. The target policy is generated via the uniform distribution on $[0, 1]$ with proper normalization and the behavior policy is the uniform policy. We set the learning rates as $\eta_\theta = 0.2$ and $\eta_\omega = 0.1$ for both algorithms and set the batch size as $M = 3000$ for the VR-Greedy-GQ. We run 200k iterations for each of the 10 trajectories.

Figure 3 shows the comparison between VR-Greedy-GQ and Greedy-GQ, and one can make consistent observations with those made in the Garnet experiment. Specifically, Figure 3 (left) shows that VR-Greedy-GQ achieves a much more stationary policy than Greedy-GQ. Figure 3 (middle) shows that the stochastic updates of VR-Greedy-GQ induce a much smaller variance than those of Greedy-GQ. Moreover, Figure 3 (right) verifies our theoretical result that VR-Greedy achieves a smaller asymptotic convergence error with a larger batch size.

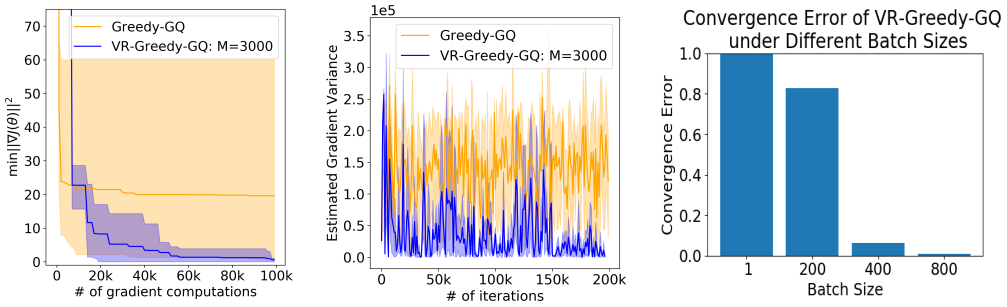


Figure 3: Comparison of Greedy-GQ and VR-Greedy-GQ in solving the Frozen Lake problem.

We further plot the objective function value *v.s.* number of gradient computations for both Greedy-GQ and VR-Greedy-GQ in Figure 4 (Left), where one can see that VR-Greedy-GQ outperforms Greedy-GQ. In Figure 2 (Right), we also plot the estimated expected maximum reward *v.s.* number of iterations for Greedy-GQ, VR-Greedy-GQ and policy gradient. For policy gradient, we use the same parameter settings as before. One can see that VR-Greedy-GQ achieves a much higher expected reward than both Greedy-GQ and policy gradient.

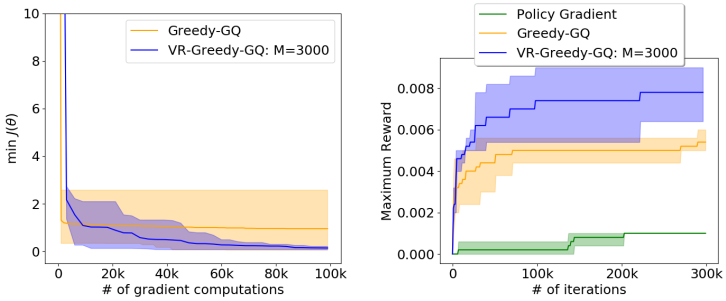


Figure 4: Comparison of MSPBE and reward obtained by Greedy-GQ, VR-Greedy-GQ and PG.

## 7 CONCLUSION

In this paper, we develop a variance-reduced two time-scale Greedy-GQ algorithm for optimal control by leveraging the SVRG variance reduction scheme. Under linear function approximation and Markovian sampling, we establish the sublinear finite-time convergence rate of the algorithm to a stationary point and prove an improved sample complexity bound over that of the original Greedy-GQ. The RL experiments well demonstrated the effectiveness of the proposed two time-scale variance reduction scheme. Our algorithm design may inspire new developments of variance reduction for two time-scale RL algorithms. In the future, we will explore Greedy-GQ with other nonconvex variance reduction schemes to possibly further improve the sample complexity.

## REFERENCES


TW Archibald, KIM McKinnon, and LC Thomas. On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46(3):354–361, 1995.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 243–252. JMLR. org, 2017.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Zaiwei Chen, Sheng Zhang, Thinh T Doan, Siva Theja Maguluri, and John-Paul Clarke. Performance of Q-learning with linear function approximation: Stability and finite-time analysis. *arXiv preprint arXiv:1905.11425*, 2019.

Gal Dalal, Balázs Szörényi, Gugan Thoppe, and Shie Mannor. Finite sample analysis of two-timescale stochastic approximation with applications to reinforcement learning. *Proceedings of Machine Learning Research*, 75:1–35, 2018.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1646–1654, 2014.

Simon S Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. In *Proc. International Conference on Machine Learning (ICML)*, pp. 1049–1058, 2017.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 689–699, 2018.

Geoffrey J. Gordon. Chattering in SARSA ($\lambda$)-a CMU learning lab internal report. *Citeseer*, 1996.

Harsh Gupta, R Srikant, and Lei Ying. Finite-time performance bounds and adaptive learning rate selection for two time-scale reinforcement learning. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4706–4715, 2019.

Haonan Jia, Xiao Zhang, Jun Xu, Wei Zeng, Hao Jiang, Xiaohui Yan, and Ji-Rong Wen. Variance reduction for deep q-learning using stochastic recursive gradient. *arXiv:2007.12817*, 07 2020.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Nathaniel Korda and Prashanth La. On td (0) with function approximation: Concentration bounds and a centered variant with exponential convergence. In *International Conference on Machine Learning*, pp. 626–634, 2015.

Zhize Li and Jian Li. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. In *Advances in neural information processing systems*, pp. 5564–5574, 2018.

Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-sample analysis of proximal gradient td algorithms. In *Proc. International Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 504–513. Citeseer, 2015.

Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proc. International Conference on Machine Learning (ICML)*, 2010.

Francisco S Melo, Sean P Meyn, and M Isabel Ribeiro. An analysis of reinforcement learning with function approximation. In *Proc. International Conference on Machine Learning (ICML)*, pp. 664–671. ACM, 2008.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2613–2621. JMLR. org, 2017.

Zilun Peng, Ahmed Touati, Pascal Vincent, and Doina Precup. SVRG for policy evaluation with fewer gradient evaluations. *arXiv:1906.03704*, 2019.

Theodore J Perkins and Doina Precup. A convergent form of approximate policy iteration. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1627–1634, 2003.

G. A. Rummery and M. Niranjan. Online Q-learning using connectionist systems. *Technical Report, Cambridge University Engineering Department*, September 1994.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Rayadurgam Srikant and Lei Ying. Finite-time error bounds for linear stochastic approximation andtd learning. In *Conference on Learning Theory*, pp. 2803–2830. PMLR, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, Second Edition*. The MIT Press, Cambridge, Massachusetts, 2018.

Richard S Sutton, Hamid R Maei, and Csaba Szepesvári. A convergent $O(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in neural information processing systems*, pp. 1609–1616, 2009a.

Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proc. International Conference on Machine Learning (ICML)*, pp. 993–1000, 2009b.

Martin Wainwright. Variance-reduced q-learning is minimax optimal. *arXiv:1906.04697*, 06 2019.

Gang Wang, Bingcong Li, and Georgios B. Giannakis. A multistep lyapunov approach for finite-time analysis of biased stochastic approximation. *arXiv:1909.04299*, 2020.

Yue Wang and Shaofeng Zou. Finite-sample analysis of greedy-gq with linear function approximation under markovian noise. In *Proc. Machine Learning Research*, volume 124, pp. 11–20, Aug 2020.

Yue Wang, Wei Chen, Yuting Liu, Zhi-Ming Ma, and Tie-Yan Liu. Finite sample analysis of the gtd policy evaluation algorithms in markov setting. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5504–5513, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Yue Wu, Weitong Zhang, Pan Xu, and Quanquan Gu. A finite time analysis of two time-scale actor critic methods. *arXiv preprint arXiv:2005.01350*, 2020.

Pan Xu and Quanquan Gu. A finite-time analysis of q-learning with neural network function approximation. *arXiv preprint arXiv:1912.04511*, 2019.

Tengyu Xu, Shaofeng Zou, and Yingbin Liang. Two time-scale off-policy TD learning: Non-asymptotic analysis over Markovian samples. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10633–10643, 2019.

Tengyu Xu, Zhe Wang, Yi Zhou, and Yingbin Liang. Reanalysis of variance reduced temporal difference learning. In *Proc. International Conference on Learning Representations (ICLR)*, 2020.

Huizhen Yu. On convergence of some gradient-based temporal-differences algorithms for off-policy learning. *arXiv preprint arXiv:1712.09652*, 2017.

Shaofeng Zou, Tengyu Xu, and Yingbin Liang. Finite-sample analysis for SARSA with linear function approximation. In *Advances in Neural Information Processing Systems*, pp. 8665–8675, 2019.