Trajectory-wise Control Variates for Variance Reduction in Policy Gradient Methods

Ching-An Cheng[*]
Georgia Tech
cacheng@gatech.edu

Xinyan Yan[*]
Georgia Tech
xinyan.yan@cc.gatech.edu

Byron Boots
Georgia Tech
bboots@cc.gatech.edu

Abstract: Policy gradient methods have demonstrated success in reinforcement learning tasks with high-dimensional continuous state and action spaces. But they are also notoriously sample inefficient, which can be attributed, at least in part, to the high variance in estimating the gradient of the task objective with Monte Carlo methods. Previous research has endeavored to contend with this problem by studying control variates (CVs) that can reduce the variance of estimates without introducing bias, including the early use of baselines, state dependent CVs, and the more recent state-action dependent CVs. In this work, we analyze the properties and drawbacks of previous CV techniques and, surprisingly, we find that these works have overlooked an important fact that Monte Carlo gradient estimates are generated by *trajectories* of states and actions. We show that ignoring the correlation across the trajectories can result in suboptimal variance reduction, and we propose a simple fix: a class of *trajectory-wise* CVs, that can further drive down the variance. The trajectory-wise CVs can be computed recursively and require only learning state-action value functions like the previous CVs for policy gradient. We further prove that the proposed trajectory-wise CVs are optimal for variance reduction under reasonable assumptions.

Keywords: Reinforcement Learning, Policy Gradient, Control Variate

1 Introduction

Policy gradient methods [1–6] are a popular class of model-free reinforcement learning (RL) algorithms. They have many advantages, including simple update rules and convergence guarantees [2, 7–9]. However, basic policy gradient methods, like REINFORCE [1], are also notoriously sample inefficiency. This can be attributed, at least in part, to the high variance in Monte Carlo gradient estimates, which stems from both policy stochasticity necessary for exploration as well as stochastic environmental dynamics. The high variance is further exacerbated as the horizon becomes longer and the dimension becomes higher. If the variance of gradient estimates can be reduced, then the learning speed of policy gradient methods can be accelerated [10, 6].

Variance reduction has been studied since early work of policy gradient methods. For example, function approximators (critics) have been adopted to (partially) replace the Monte Carlo estimates, which reduces variance but at the expense of bias in the search direction [2, 11–15]. This bias-variance tradeoff can work well in practice, but can also diverge when not tuned carefully [14, 16, 9].

Another line of research uses the control variate (CV) method from statistics, which can reduce variance in Monte Carlo methods without introducing bias [2, 17–24]. For policy gradient algorithms, specialized CV methods have been proposed to take advantage of structures inherent in RL. Especially, the state dependent CVs (also known as baselines or reward reshaping [17, 19]) have been thoroughly investigated [18]. Common state dependent CVs are constructed as approximators of the policy's value function, which admits update rules based on policy evaluation techniques. Overall, state dependent CVs are simple to implement and have been found to be quite effective, but they can still lead to detrimentally high variance, especially in problems that has a long horizon. This

[*] Equal contribution.

issue has motivated a recent development of state-action dependent CVs [20–23, 25] that can further reduce the variance due to randomness in the *actions* the previous state-only CVs fails to manage.

Considering the decades-long development of CV methods, one might wonder if there is a need for new policy gradient CV techniques. In this paper, we argue that the past development of CVs for policy gradients has overlooked an important fact that the Monte Carlo gradient estimates are generated by rolling out a policy and collecting statistics along a *trajectory* of states and actions. Instead the focus has been on sampling *pairs* of states and actions, ignoring the correlation between states and actions *across* time steps. Recently Tucker et al. [23] empirically analyzed the variance of instantaneous state-action pairs and compared this to the variance correlations across time steps in multiple simulated robot locomotion tasks. They found that the variance due to long-term trajectories is often larger than the variance due to instantaneous state-action pairs. This finding implies that there is potential room for improvement.

In this paper, we theoretically analyze the properties of previous CVs, and show that indeed the variance due to long-term trajectories can have non-negligible effects. Motivated by this observation, we propose a family of trajectory-wise CVs, called *TrajCV*, which recursively augments existing CVs with extra terms to *additionally* cancel this long-term variance. We show that TrajCV is particularly effective when the transition dynamics, despite unknown, is close to deterministic. Like existing CVs, TrajCV requires only approximates of the state-action value function (i.e. Q-function) of a policy. Therefore, the performance of TrajCV can benefit directly from the development of policy evaluation techniques. Moreover, we prove that TrajCV is optimal for variance reduction under reasonable assumptions. These theoretical insights are validated in simulation.

Upon finishing this work we discovered a recent technical report [24] that is motivated similarly and details exactly the equation (11) that TrajCV uses. Their empirical results on simulated LQG tasks are encouraging too: TrajCV demonstrated superior performance compared with previous state and state-action dependent CVs. By contrast, we derive TrajCV following a completely different route, which brings extra insight into the previous deficiency and suggests natural ways for improvement. In addition, we analyze other potential trajectory-wise CVs and prove the proposed idea is optimal.

2 Problem Setup and Background

We consider episodic policy optimization in a finite-horizon Markov Decision Process (MDP) [26, 27] with horizon h, state space $\mathcal{S}$, action space $\mathcal{A}$, instantaneous cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution p_{init}, and dynamics $\mathcal{P}$.[1] Given a parameterized *stochastic* policy class Π the goal is to search for a policy in Π that achieves low accumulated costs averaged over trajectories

$$J(\pi) := \mathbf{E}[\textstyle\sum_{t=1}^{h} C_t], \quad \text{where } C_t := c(S_t, A_t), \quad S_1 \sim p_{\text{init}}, \quad A_t \sim \pi_{S_t}, \quad S_{t+1} \sim \mathcal{P}_{S_t, A_t} \tag{1}$$

where $\mathcal{P}_{s,a}$ denotes the distribution of the next state after applying action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$, and π_s denotes the distribution of action at state $s \in \mathcal{S}$. Note that S_t and A_t are the sampled state and action at step t. For simplicity of writing, we embed the time information into the definition of state, e.g., $c(S_t, A_t)$ can represent non-stationary functions. The randomness in (1) consists of the randomness in the start state, policy, and dynamics. In this work, we focus on the case where the dynamics $\mathcal{P}$ and the start state distribution p_{init} are unknown, but the instantaneous cost c is known.

Notation We use uppercases to denote random variables, such as S_t and A_t, with the exception of J. We will use subscript $_{i..j}$ to denote the set of random variables, i.e. $X_{1..5} = \{X_1, \dots, X_5\}$, and $_{i:j}$ to denote summation (i.e. $C_{1:h} = \sum_{t=1}^{h} C_t$). As we will be frequently manipulating conditional distributions, we adopt the subscript notation below to write conditional expectation and variance. For $\mathbf{E}_{X|Y}[f(X, Y)]$ of some function f, X denotes the random variable where the expectation is defined and Y denotes the conditioned one. Furthermore, for $f(X_{1..N}, Y)$, we use $\mathbf{E}_{|Y}[f(X_{1..N}, Y)]$ as a shorthand to denote taking the expectation over all other random variables (i.e. $X_{1..N}$) conditioned on Y. This subscript notation also applies to variance, which is denoted as $\mathbf{Var}$.

2.1 Policy Gradient Methods: Pros and Cons

The goal of this paper is to improve the learning performance of policy gradient methods [1, 7, 2–4, 13, 5, 6]. These algorithms treat minimizing (1) as a first-order stochastic non-convex optimization problem, where noisy, unbiased gradient estimates of J in (1) are used to inform policy search.

[1] We use one-based indexing throughout the manuscript.

The basic idea is to apply the likelihood-ratio method to derive the gradient of (1). Let us define $N_t := \nabla \log \pi_{S_t}(A_t)$, where ∇ is the derivative with respect to the policy parameters, and define q^π as the Q-function of π; that is, $q^\pi(S_t, A_t) = \mathbf{E}[C_{t:h}]$ where the expectation is generated by taking A_t at S_t and then π afterwards. Define $G := G_{1:h}$ and $G_t := N_t C_{t:h}$. Then it follows [1]

$$\nabla J(\pi) = \mathbf{E}[\textstyle\sum_{t=1}^{h} N_t q^\pi(S_t, A_t)] = \mathbf{E}\left[G\right], \tag{2}$$

where the second equality is due to $q^\pi(S_t, A_t) = \mathbf{E}[C_{t:h}]$. Eq. (2) is an expectation over trajectories generated by running π. Thus, we can treat the random vector G as an unbiased estimate of $\nabla J(\pi)$, which can be computed by executing the policy π starting from initial distribution and then recording the statistics G_t, for $t \in \{1, \ldots, h\}$. This technique is known as the Monte Carlo estimate, which samples i.i.d. trajectories from the trajectory distribution in (1) to approximate the expectation.

The policy gradient methods (e.g. REINFORCE [1]) optimize policies based on gradient estimates constructed using the above idea. They have numerous advantages, such as straightforward update rules and convergence guarantee [2, 7–9]. But simply using the Monte Carlo estimate G in policy optimization (i.e. the vanilla implementation of REINFORCE) can result in poor performance due to excessive variance [13, 14]. Therefore, while ideally one can apply standard first-order optimization algorithms, such as mirror descent [28], with the random estimate G to optimize policies, this often is not viable in practice. They would require a tremendous amount of trajectory rollouts in order to attenuate the high variance, making learning sample inefficient.

The high variance of G is due to the exploration difficulty in RL: in the worst-case, the variance of G can grow exponentially in the problem's horizon [29, 30], as it becomes harder for the policy to visit meaningful states and get useful update information. Intuitively we can then imagine that policy optimization progress can be extremely slow, when the gradient estimates are noisy. From an optimization perspective, variance is detrimental to the convergence rate in stochastic optimization. For example, the number of iterations for mirror descent to converge to an ϵ-approximate stationary point is $O((\mathbf{Tr}\left(\mathbf{Var}[G]\right) + 1)/\epsilon^2)$, increasing as the problem becomes more noisy [10]. Therefore, if the variance of estimates of (2) can be reduced, the policy gradient methods can be accelerated.

2.2 Variance Reduction and Control Variate

A powerful technique for reducing the variance in the Monte Carlo estimates is the CV method [31, 32]. Leveraging correlation between random estimates, the CV method has formed the backbone of many state-of-the-art stochastic optimization algorithms [33–35], in particular, practical policy gradient methods for RL [18] because of the high-variance issue of G discussed in the previous section. Below, we review the basics of the CV method as well as previous CV techniques designed for reducing the variance of G. Without loss of generality, we suppose only one trajectory is sampled from the MDP to construct the estimate of (2) and study the variance of different single-sample estimates. We remind that the variance can be always further reduced, when more i.i.d. trajectories are sampled (i.e. using mini batches).

2.3 Control Variate Method and Difference Estimator

Consider the problem of estimating the expectation $\mathbf{E}[X]$, where X is a (possibly multivariate) random variable. The CV method [31, 32] is a technique for synthesizing unbiased estimates of $\mathbf{E}[X]$ that potentially have lower variance than the naive sample estimate X. It works as follows: Assume that we have access to another random variable Y, called the *CV*, whose expectation $\mathbf{E}[Y]$ is cheaper to estimate than $\mathbf{E}[X]$. Then we can devise this new estimate by a linear combination,

$$X - \Omega^\top (Y - \mathbf{E}[Y]), \tag{3}$$

where Ω is a properly-shaped matrix. Due to the linearity of expectation, the estimate in (3) is unbiased. Suppose Y is in the same dimension as X. One can show that the optimal Ω is $\Omega^\star = \frac{1}{2}\mathbf{Var}[Y]^{-1}\left(\mathbf{Cov}[X, Y] + \mathbf{Cov}[Y, X]\right)$. When data are too scarce to estimate $\Omega^\star$, Ω usually set as the identity, which often works well when Y is positively correlated with X. The resulting estimate $X - (Y - \mathbf{E}\left[Y\right])$ is known as the *difference estimator* [32] and has variance $\mathbf{Var}\left[X - Y\right]$, meaning that if Y is close to X then the variance becomes smaller. In the following, we concentrate on the design of difference estimators; we note that designing a good Ω is an orthogonal research direction.

2.4 Common Control Variates for Policy Gradient Methods

The art to various CV methods lies in the design of the correlated random variable Y. The choice is often domain-dependent, based on how X is generated. When estimating the policy gradient in (2),

many structures (e.g. the Markov property) can be leveraged to design CVs. We discuss properties of these designs below. Following previous works (e.g. [18, 23]), here we focus on the policy gradient component G_t of G given in (2) for simplicity of exposition.[2]

The most commonly used CVs for policy gradient [1, 17, 18] are state-dependent functions $\widehat{v} : \mathcal{S} \to \mathbb{R}$, which leads to the difference estimator

$$\widetilde{G}_t^{\mathrm{S}} := G_t - \left(N_t \widehat{V}_t - \mathbf{E}_{A_t|S_t}[N_t \widehat{V}_t]\right) = G_t - N_t \widehat{V}_t, \quad \text{where } \widehat{V}_t := \widehat{v}(S_t), \tag{4}$$

and the expectation vanishes as $\mathbf{E}_{A_t|S_t}[N_t \widehat{V}_t] = \widehat{V}_t \nabla \mathbf{E}_{A_t|S_t}[1] = 0$.[3] Recently, *state-action CVs* $\widehat{q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ have also been proposed [20–23, 25, 36], in an attempt to reduce more variance through CVs that better correlate with G_t. The state-action CVs yields the difference estimator

$$\widetilde{G}_t^{\mathrm{SA}} := G_t - \left(N_t \widehat{Q}_t - \mathbf{E}_{A_t|S_t}[N_t \widehat{Q}_t]\right), \quad \text{where } \widehat{Q}_t := \widehat{q}(S_t, A_t). \tag{5}$$

Usually $\widehat{v}$ and $\widehat{q}$ are constructed as function approximators of the value function v^π and the Q-function q^π of the current policy π, respectively, and learned by policy evaluation, e.g., variants of TD(λ) [37], during policy optimization. Therefore, these methods can also be viewed as unbiased actor-critic approaches. In practice, it has been observed that these CVs indeed accelerate policy optimization, especially in simulated robot control tasks [20, 21, 23, 25, 36].

3 Why We Need New Control Variates

Given the decades-long development of CVs for policy gradient reviewed above, one might wonder if there is a need for new CV techniques. If so, what is the additional gain we can potentially have? To answer this question, let us first analyze the variance of policy gradient component G_t and how the CVs above reduce it. By the law of total variance[4], $\mathbf{Var}[G_t]$ can be decomposed into *three terms*

$$\mathbf{Var}_{S_t}\mathbf{E}_{|S_t}\left[N_t C_{t:h}\right] + \mathbf{E}_{S_t}\mathbf{Var}_{A_t|S_t}\left[N_t \mathbf{E}_{|S_t,A_t}\left[C_{t:h}\right]\right] + \mathbf{E}_{S_t,A_t}\mathbf{Var}_{|S_t,A_t}\left[N_t C_{t:h}\right], \tag{6}$$

where the first term is due to the randomness of policy and dynamics before getting to S_t, the second term is due to policy randomness alone at step t, i.e. selecting A_t, and the third term is due to again both the policy and the dynamics randomness in the future trajectories, i.e. after S_t and A_t. We can measure the size of these three terms by their trace and define

$$\begin{gathered}\mathbb{V}_{S_t} := \mathbf{Tr}\left(\mathbf{Var}_{S_t}\mathbf{E}_{|S_t}\left[N_t C_{t:h}\right]\right), \quad \mathbb{V}_{A_t|S_t} := \mathbf{Tr}\left(\mathbf{E}_{S_t}\mathbf{Var}_{A_t|S_t}\left[N_t \mathbf{E}_{|S_t,A_t}\left[C_{t:h}\right]\right]\right), \\ \mathbb{V}_{|S_t,A_t} := \mathbf{Tr}\left(\mathbf{E}_{S_t,A_t}\mathbf{Var}_{|S_t,A_t}\left[N_t C_{t:h}\right]\right).\end{gathered} \tag{7}$$

Hence, $\mathbf{Tr}\left(\mathbf{Var}[G_t]\right) = \mathbb{V}_{S_t} + \mathbb{V}_{A_t|S_t} + \mathbb{V}_{|S_t,A_t}$. The following theorem shows the size of each term when the policy is Gaussian, which is commonly the case for problems with continuous actions.

Theorem 3.1. Suppose the policy π is Gaussian such that $\pi_{S_t}(A_t) = \mathcal{N}(A_t|\mu_\theta(S_t), \sigma I)$, where μ_θ is the mean function, and θ and $\sigma > 0$ are learnable parameters. Assume the cost function c is bounded and the Q-function $q^\pi(s,a)$ is analytic in a. Then for small enough σ, it holds that

$$\mathbb{V}_{S_t} = O(h^2), \quad \mathbb{V}_{A_t|S_t} = O\left(\tfrac{h^2}{\sigma^4}\right), \quad \text{and} \quad \mathbb{V}_{|S_t,A_t} = O\left(\tfrac{h^2}{\sigma^4}\right).$$

Here we focus on the effects due to the problem horizon h and the policy variance σ. Theorem 3.1 shows that, when the stochasticity in policy decreases (e.g. when it passes the initial exploration phase) the terms $\mathbb{V}_{A_t|S_t}$ and $\mathbb{V}_{|S_t,A_t}$ will dominate variance in policy gradients. An intuitive explanation to this effect is that, as the policy becomes more deterministic, it becomes harder to approximate the derivative through zero-order feedback (i.e. accumulated costs). In particular, one can expect that $\mathbb{V}_{|S_t,A_t}$ is likely to be larger than $\mathbb{V}_{A_t|S_t}$ when the variation of $C_{t:h}$ is larger than the variation of $q^\pi(S_t, A_t) = \mathbf{E}_{|S_t,A_t}[C_{t:h}]$. After understanding the composition of $\mathbf{Var}[G_t]$, let us analyze $\mathbf{Var}[\widetilde{G}_t^{\mathrm{SA}}]$ to see why using Q-function estimates as CVs (in Section 2.4) can reduce the variance.[5] Akin to the derivation of (6), one can show that $\mathbf{Var}[\widetilde{G}_t^{\mathrm{SA}}]$ can be written as

$$\mathbf{Var}_{S_t}\mathbf{E}_{|S_t}[N_t C_{t:h}] + \mathbf{E}_{S_t}\mathbf{Var}_{A_t|S_t}[N_t(\mathbf{E}_{|S_t,A_t}[C_{t:h}] - \widehat{Q}_t)] + \mathbf{E}_{S_t,A_t}\mathbf{Var}_{|S_t,A_t}[N_t C_{t:h}]. \tag{8}$$

[2] Without any assumption of the MDP, the variance of G can be bounded by the variance of G_t (Appendix A.3). Tighter bounds can be derived when assumptions on the MDP is made, e.g., faster mixing rate [18].

[3] State dependent functions naturally include non-stationary constant baselines in our notation.

[4] The law of total variance: $\mathbf{Var}[f(X,Y)] = \mathbf{E}_X\mathbf{Var}_{Y|X}[f(X,Y)] + \mathbf{Var}_X\mathbf{E}_{Y|X}[f(X,Y)]$ [38].

[5] Discussion on $\mathbf{Var}[\widetilde{G}_t^{\mathrm{S}}]$ is omitted in that $\widetilde{G}_t^{\mathrm{S}}$ is subsumed by $\widetilde{G}_t^{\mathrm{SA}}$.

Comparing (6) and (8), we can see that the CVs in the literature have been focusing on reducing *the second term* $\mathbf{Var}_{A_t|S_t}$. Apparently, from the decomposition (8), the optimal choice of the state-action CV $\widehat{q}$ is the Q-function of the current policy q^π, because $q^\pi(S_t, A_t) := \mathbf{E}_{|S_t,A_t}[C_{t:h}]$, which explains why $\widehat{q}$ can be constructed by policy evaluation. When $\widehat{q} = q^\pi$, the effect of $\mathbf{Var}_{A_t|S_t}$ can be completely removed. In practice, $\widehat{q}$ is never perfect (let alone the state-dependent version); nonetheless, improvement in learning speed has been consistently reported.

However, Theorem 3.1 suggests that $\mathbf{Var}_{|S_t,A_t}$ can be in the similar magnitude as $\mathbf{Var}_{A_t|S_t}$, implying that even when we completely remove the second term $\mathbf{Var}_{A_t|S_t}$, the variance of the gradient estimate can still be significant. Indeed, recently Tucker et al. [23] empirically analyzed the three variance components in (8) in LQG and simulated robot locomotion tasks. They found that the third term $\mathbf{Var}_{|S_t,A_t}$ is often close to the second term $\mathbf{Var}_{A_t|S_t}$, and both of them are several orders of magnitude larger than the first term $\mathbf{Var}_{S_t}$. Our Theorem 3.1 supports their finding and implies that there is a potential for improvement by reducing $\mathbf{Var}_{|S_t,A_t}$. We discuss exactly how to do this next.

4 Trajectory-wise Control Variates

We propose a new family of trajectory-wise CVs, called TrajCV, that improves upon existing state or state-action CV techniques by tackling *additionally* $\mathbf{Var}_{|S_t,A_t}$, the variance due to randomness in trajectory after step t (cf. Section 3). While this idea sounds intuitively pleasing, a technical challenge immediately arises. Recall in designing CVs, we need to know the expectation of the proposed CV function over the randomness that we wish to reduce (see (3)). In this case, suppose we propose a CV $\phi(S_{t..h}, A_{t..h})$, we would need to know its conditional expectation $\mathbf{E}_{|S_t,A_t}[\phi(S_{t..h}, A_{t..h})]$. This need makes reducing $\mathbf{Var}_{|S_t,A_t}$ fundamentally different from reducing $\mathbf{Var}_{A_t|S_t}$, the latter of which has been the main focus in the literature: Because the dynamics $\mathcal{P}$ is unknown, we do not have access to the distribution of trajectories after step t and therefore cannot compute $\mathbf{E}_{|S_t,A_t}$; by contrast, reducing $\mathbf{Var}_{A_t|S_t}$ only requires knowing the policy π.

At first glance this seems like an impossible quest. But we will show that by a clever divide-and-conquer trick, an unbiased CV can actually be devised to reduce the variance $\mathbf{Var}_{|S_t,A_t}$. The main idea is to *1)* decompose $\mathbf{Var}_{|S_t,A_t}$ through repeatedly invoking the law of total variance and then *2)* attack the terms that are *amenable* to reduction using CVs. As expected, the future variance cannot be completely removed, because of the unknown dynamics. But we should be able to reduce the randomness due to known distributions, namely, the future uses of policy π.

4.1 A Divide-and-Conquer Strategy

Before giving the details, let us first elucidate our idea using a toy problem. Consider estimating $\mathbf{E}[f(X_{1..5})]$, the expectation of a function f of 5 random variables. We can apply the law of total variance repeatedly, in the order indicated by the subscript, and decompose the variance into

$$\mathbf{Var}[f(X_{1..5})] = \sum_{k=1}^{5} \mathbf{E}_{X_{1..k-1}} \mathbf{Var}_{X_k|X_{1..k-1}} \mathbf{E}_{X_{k+1..n}|X_{1..k}}[f(X_{1..5})] \tag{9}$$

For example, suppose we wish to reduce $\mathbf{Var}_{X_3|X_{1..2}}$ we simply need to consider a CV in the form $\phi(X_{1..3})$, which does not depends on random variables with larger indices. With the difference estimator $f(X_{1..5}) - \phi(X_{1..3}) + \mathbf{E}_{X_3|X_{1..2}}[\phi(X_{1..3})]$, the variance $\mathbf{Var}_{X_3|X_{1..2}}$ changes into $\mathbf{E}_{X_{1..2}}\mathbf{Var}_{X_3|X_{1..2}}[\mathbf{E}_{X_{4..5}|X_{1..3}}[f(X_{1..5})] - \phi(X_{1..3})]$. Apparently when g is optimally chosen as $\phi^\star(X_{1..3}) := \mathbf{E}_{X_{4..5}|X_{1..3}}[f(X_{1..5})]$, this term vanishes.

Fact 1 A key of designing CVs by the recursive decomposition above is that the inclusion of the extra term, e.g. $\phi(X_{1..3}) - \mathbf{E}_{X_3|X_{1..2}}[\phi(X_{1..3})]$, in the difference estimator only affects a single component $\mathbf{Var}_{X_3|X_{1..2}}$ in the total variance, *without influencing the other terms*. This separation property hence allows for a divide-and-conquer strategy: we can design CVs for each term separately and then combine them; the reduction on each term will add up and reduce the total variance.

Fact 2 There is still one missing piece before we can adopt the above idea to design CVs for estimating policy gradients: the ordering of random variables. In the example above, we need to know $\mathbf{E}_{X_3|X_{1..2}}[\phi(X_{1..3})]$ to compute the difference estimator. Namely, it implicitly assumes the knowledge about $p(X_3|X_{1..2})$, which may or may not be accessible. Suppose $p(X_3|X_{1..2})$ is not available but $p(X_3|X_{4..5})$ is. We can consider instead invoking the law of total variance in a different order, e.g. $X_4 \to X_5 \to X_3 \to X_1 \to X_2$, and utilize the information $p(X_3|X_{4..5})$ to construct a difference estimator to reduce $\mathbf{Var}_{X_3|X_{4..5}}$. Therefore, the design of CVs hinges also on the information available. Recall that we only know about the policy but not the dynamics in RL.

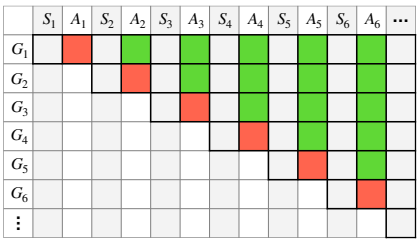


Figure 1: Comparison of state-action CV and TrajCV

Figure 2: An illustration of the effects of state-action CV (5) and TrajCV (11). One row corresponds to one policy component $G_t := N_t C_{t:h}$, and thick borders indicate the random variables of which G_t is a function. State-action CV reduces the variance due to the random variables in red, whereas TrajCV *additionally* affect the variance stemming from the random variables in green.

4.2 Design of TrajCV

After fleshing out the idea in the example above, we are now ready to present TrajCVs for policy gradient. Again we will focus on the component G_t for transparency. Recall that G_t is a function of $S_{t..h}$ and $A_{t..h}$. Given the information we know about these random variables (i.e. the policy) and the Markovian structure in MDP, a natural ordering of them for applying law of total variance is

$$S_t \to A_t \to S_{t+1} \to A_{t+1} \to \cdots \to S_h \to A_h. \tag{10}$$

Suppose now we want to reduce $\mathbf{Var}_{A_k|S_{t..k},A_{t..k-1}}$ for some $k > t$. Based on Section 4.1, we may consider a CV in the form $\phi_t(S_{t..k}, A_{t..k})$, whose the optimal choice is $\phi_t^\star(S_{t..k}, A_{t..k}) = \mathbf{E}_{|S_{t..k},A_{t..k}}[N_t C_{t:h}] = N_t\left(C_{t:k-1} + \mathbf{E}_{|S_{t..k},A_{t..k}}[C_{k:h}]\right) = N_t\left(C_{t:k-1} + q^\pi(S_k, A_k)\right)$, where the last equality is due to the Markovian structure and the definition of q^π. This suggests practically we can use $\phi_t(S_{t..k}, A_{t..k}) := N_t(C_{t:k-1} + \widehat{Q}_k)$, where $\widehat{Q}_k := \widehat{q}(S_k, A_k)$ and $\widehat{q} \approx q^\pi$ as was in (5).

In other words, we showed that finding the optimal CV for reducing variance in policy gradient can be reduced to learning a good Q-function estimate; this enables us to take advantage of existing policy evaluation algorithms. Now we combine[6] $\{\phi_t(S_{t..k}, A_{t..k})\}_{k=t}^h$ to build the CV for G_t. Because these terms do not interfere with each other (cf. Section 4.1), we can simply add them together into $\sum_{k=t}^h \phi_t(S_{t..k}, A_{t..k})$ as the TrajCV. Equivalently, we have devised a difference estimator:

$$\begin{aligned}\widetilde{G}_t^{\text{Traj}} &:= G_t - \textstyle\sum_{k=t}^h \left(\phi_t(S_{t..k}, A_{t..k}) - \mathbf{E}_{A_k|S_{t..k},A_{t..k-1}}[\phi_t(S_{t..k}, A_{t..k})]\right) \\ &= G_t - \textstyle\sum_{k=t}^h \left(N_t \widehat{Q}_k - \mathbf{E}_{A_k|S_k}[N_t \widehat{Q}_k]\right)\end{aligned} \tag{11}$$

Comparing TrajCV in (11) and state-action CV in (4), we see that the state-action CV only contains the first term in the summation of TrajCV.[7] The remaining terms with $k > t$ can be viewed as multiplying N_t with estimates of future advantage functions: i.e., we have $N_t\widehat{Q}_k - \mathbf{E}_{A_k|S_k}[N_t\widehat{Q}_k] = N_t(\widehat{Q}_k - \mathbf{E}_{A_k|S_k}[\widehat{Q}_k])$. Appealing to law of total variance, $\mathbf{Var}[\widetilde{G}_t^{\text{Traj}}]$ can be decomposed into

$$\underbrace{\mathbf{Var}_{S_t}\mathbf{E}_{|S_t}[N_t C_{t:h}]}_{\text{due to } S_t} + \underbrace{\mathbf{E}_{S_t}\mathbf{Var}_{A_t|S_t}[N_t(\mathbf{E}_{|S_t,A_t}[C_{t:h}] - \widehat{Q}_t)]}_{\text{due to } A_t} + \tag{12}$$

$$\underbrace{\sum_{k=t}^h \mathbf{E}_{S_k,A_k}\mathbf{Var}_{S_{k+1}|S_k,A_k}\left[N_t \mathbf{E}_{|S_{k+1}}[C_{k:h}]\right]}_{\text{due to dynamics randomness after step } t} + \underbrace{\sum_{k=t}^h \mathbf{E}_{S_{k+1}}\mathbf{Var}_{A_{k+1}|S_{k+1}}[N_t(\mathbf{E}_{|S_{k+1},A_{k+1}}[C_{k:h}] - \widehat{Q}_{k+1})]}_{\text{due to policy randomness after step } t}$$

where we further decompose the effect of $\mathbf{Var}_{|S_t,A_t}$ in the second line into the randomness in dynamics and actions, respectively. Therefore, suppose the underlying dynamics is deterministic (i.e. $\mathbf{Var}_{S_{k+1}|S_k,A_k}$ vanishes), and $\widehat{q} = q^\pi$, then using TrajCV (11) would *completely* remove $\mathbf{Var}_{A_t|S_t}$ and $\mathbf{Var}_{|S_t,A_t}$, the latter of which previous CVs (4) and (5) cannot affect. In Fig. 1, we visualize effects of TrajCV and state-action CV on each policy gradient component G_t. State-action CV only influences the diagonal terms, while TrajCV is able to affect the full upper-triangle

[6] When $k = t$, the CV is the same as state-action CV in (5), i.e. we define $\phi_t(S_t, A_t) = N_t\widehat{Q}_t$.

[7] For brevity, we use CV to mean the difference estimator of that CV when there is no confusion.

parts. Note that in implementation of TrajCV for $G_{1:h}$, we only need to compute quantities $\widehat{Q}_t$, $\mathbf{E}_{A_t|S_t}[\widehat{Q}_t]$ and $\mathbf{E}_{A_t|S_t}[N_t\widehat{Q}_t]$ along a trajectory (done in $O(h)$ linear time) and they can be used to compute $\{\widetilde{G}_t^{\text{Traj}}\}_{t=1}^h$ in (11). In addition, we remark that when $\widehat{q}(s,a) = \widehat{v}(s)$, TrajCV reduces to the state-dependent CVs. We provide an example implementation of TrajCV in Appendix B.

4.3 The Natural Ordering in (10) is Optimal

Recall in Section 4.1 we mentioned that the admissible ordering of random variables used in invoking the law of total variance depends on the information available. Here we show that the chosen ordering (10) is indeed the best ordering to adopt, as we only know the policy, not the dynamics.

We compare (10) against other potential orderings constructed by reparameterizing the policy such that its randomness in action becomes independent of the input state. We suppose the policy $\pi \in \Pi$ can be reparameterized by a function $\omega : \mathcal{S} \times \mathcal{R} \to \mathcal{A}$ and a distribution p_R, so that for all $s \in \mathcal{S}$, $\omega(s, R)$ and π_s are equal. This is not a restricted assumption, which applies to, e.g., policies based on Gaussian [1, 5, 39] and Boltzmann [40, 2] distributions.

Reparameterization makes designing a larger family of TrajCVs possible. Because the component G_t becomes a function of $R_{t..h}$ and $S_{t..h}$, the ordering the random variables in applying the law of total variance now can have many possibilities. In one extreme case, the randomness of actions can be ordered before states (except S_t) as

$$S_t \to R_t \to \cdots \to R_h \to S_{t+1} \to \cdots \to S_h, \tag{13}$$

leading to a CV that's a function of $R_{t..h}$ bearing the optimal choice $\mathbf{E}_{S_{t+1..h}|R_{t..h},S_t}[N_tC_{t:h}]$. Note that $\mathbf{E}_{S_{t+1..h}|R_{t..h},S_t}[N_tC_{t:h}]$ is a function that inputs the observable action randomness $R_{t..h}$, not the randomness of the unknown dynamics. Therefore, it can be approximated, e.g., if we have a biased simulator of the dynamics.[8] One might ask, given all possible orderings of random variables, which ordering we should pick to design the CV. Interestingly, to this question, the most natural one and the optimal one coincide. The proof is deferred to Appendix A.

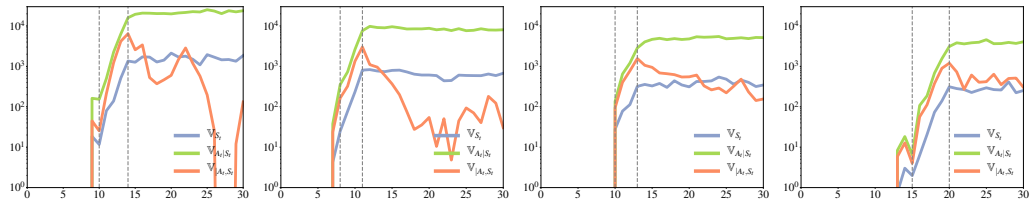
Theorem 4.1. Suppose that policy specified by ω and p_R is known, but the dynamics $\mathcal{P}$ is unknown. Assume the optimal CV of a given ordering of random variables $S_{t..h}$ and $R_{t..h}$ can be obtained. Then the optimal ordering that minimizes the residue variance is the natural ordering in (10) .

Theorem 4.1 tells us that if the optimal CVs are attainable (i.e. we can compute the Q-function exactly), then the natural ordering is optimal. However, we also remark that using this exact Q-function in the actor-critic rule can actually compute a gradient estimate that does not depend on any future randomness, which is better than using any of the CV techniques above as they yield noisy gradient estimates that always depend on unobservable randomness due to the stochastic dynamics. However, in practice, we can get neither the exact Q-function, nor the optimal CV of the other orderings. Consequently, further trade-off between bias and variance should be considered, which can have large effects in practice. For example, when using a biased Q-function estimate, it may be good to start with the the biased actor-critic gradient and then gradually switching to the unbiased TrajCV gradient as the step size decays. But considering the imperfection of the Q-function estimate, we may also want to use other ordering instead of the natural one used in TrajCV. If the dynamics is relatively accurate and the computing resources for simulation are abundant, then although the residue is higher, the ordering (13) could actually be superior. The purpose of this paper is to provide new insights into these different choices, but we leave further discussion on the bias-variance trade-off as an interesting practical question to pursue in future work. In the experiment section later, we will focus on the natural ordering (10).

5 Experimental Results and Discussion

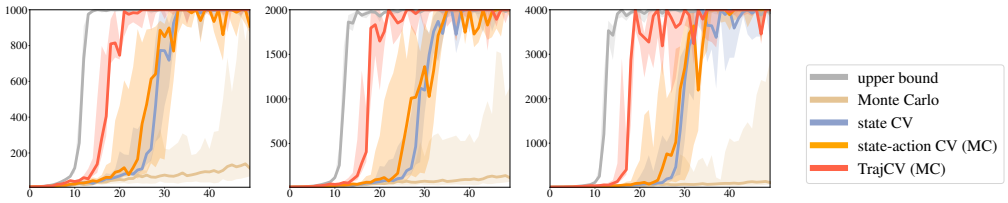
Although the focus of this paper is the theoretical insights, we illustrate our results with simulation of learning neural network policies to solve the CartPole balancing task in OpenAI Gym [41] powered by DART physics engine [42]. The policies are optimized using natural gradient descent [3]. Below we report in rewards, negative of costs, which is the natural performance measure provided in OpenAI Gym. The details of setup and implementation are provided in Appendix C.

[8] We sample all the action randomness $R_{t..h}$ first, execute the policy π in simulation with fixed randomness $R_{t..h}$, and then collect the statistics $N_tC_{t:h}$.



(a) $\sigma = 0.01$ (b) $\sigma = 0.09$ (c) $\sigma = 1.0$ (d) $\sigma = 9.0$

Figure 3: Components of $\mathbf{Tr}\left(\mathbf{Var}[G_t]\right)$, for $t = 100$, evaluated at policies generated under the "upper bound" setting. σ denotes the initial value of policy variance (defined in Theorem 3.1). The x-axis denotes the iteration number, and the y-axis is in log scale. The two vertical dashed lines mark the boundaries of iterations where the expected accumulated rewards is between 50 and 900.



(a) $h = 1000$ (MC) (b) $h = 2000$ (MC) (c) $h = 4000$ (MC)

Figure 4: Results of naive Monte Carlo estimate, state CV, state-action CV, and TrajCV on CartPole problems with horizon $h = 1000, 2000, 4000$, where 5 trajectories are collected in each iteration. (MC) in the legend indicates that $\mathbf{E}_{A_t|S_t}$ in state-action CV and TrajCV is approximated by 1000 samples. "Upper bound" emulates the results of noiseless estimates with $100,000$ samples per iteration.[9] The x- and y-axes are iteration number and accumulated rewards, respectively. The median of 8 random seeds is plotted, and the shaded area accounts for 25% percentile.

First, in Fig. 3, we corroborate the theoretical findings in Theorem 3.1 by empirically evaluating the components of $\mathbf{Tr}\left(\mathbf{Var}[G_t]\right)$ during learning at $t = 100$. We observe that the learning process on CartPole can be partitioned into *three stages* (as delineated by the dashed vertical lines in Fig. 3): 1) the initial exploration, where the policy performs very poorly and improves slowly, 2) the rapid improvement, where the policy performance increases steeply, and 3) the near convergence, where the policy reaches and stays at the peak performance. In the rapid improvement stage, due to the variance in the accumulated reward, $\mathbb{V}_{|S_t,A_t}$ is large, close to $\mathbb{V}_{A_t|S_t}$ and about 10 times of $\mathbb{V}_{S_t}$ as predicted by Theorem 3.1. Later on, when the policy is about to converge, $\mathbb{V}_{|S_t,A_t}$ drops because the performance becomes more consistent among trajectories especially for the case with small σ.

Next, in Fig. 4, we compare naive Monte Carlo estimate (2), state-dependent CV (4), state-action CV (5), and TrajCV (11). We realized all algorithms with the same implementation of an on-policy value function approximator to facilitate a fair comparison; for the state-action CV and TrajCV, we used a Q-function estimate based on a biased physics simulator and the mentioned value function approximator. Overall, when more information is used to design the CVs (from state only, state-action, and then trajectory-wise) the convergence speed improves. In particular, as the problem horizon becomes longer, the gap becomes larger: the reward feedback becomes sparser, so the variance due to long-term trajectory starts to dominate, as shown in Fig. 4c. In Appendix C, we provide further results of CVs based on other choices of Q-function approximators.

These preliminary experimental results support the theoretical insights provided in Section 3 and Section 4, suggesting the importance of considering long-term effects in designing CVs, especially for problems with a long horizon. The fix turns out to be quite simple: just padding additional terms (cf. (11)) onto the existing CVs, which can be done using Q-function approximators available in existing CVs without new information. Interestingly we prove this simple idea is optimal. Important future work includes considering the different bias and variance trade-off discussed in Section 4.3.

[9] For the usual learners, the number of samples collected per iteration is often less than $5h$, and much less at the start of learning, because of early termination when the agent fails.

Acknowledgments

This work was partially supported by NSF CAREER award 1750483.

References

[1] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[2] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

[3] S. M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, pages 1531–1538, 2002.

[4] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

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

[6] C.-A. Cheng, X. Yan, N. Ratliff, and B. Boots. Predictor-corrector policy optimization. In *International Conference on Machine Learning*, 2019.

[7] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, pages 1008–1014, 2000.

[8] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots. Fast policy learning through imitation and reinforcement. In *Conference on Uncertainty in Artificial Intelligence*, 2018.

[9] L. Yang and Y. Zhang. Policy optimization with stochastic mirror descent. *arXiv preprint arXiv:1906.10462*, 2019.

[10] S. Ghadimi, G. Lan, and H. Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

[11] H. Kimura, S. Kobayashi, et al. An analysis of actor-critic algorithms using eligibility traces: reinforcement learning with imperfect value functions. *Journal of Japanese Society for Artificial Intelligence*, 15 (2):267–275, 2000.

[12] P. Thomas. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, pages 441–448, 2014.

[13] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[14] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

[15] W. Sun, J. A. Bagnell, and B. Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. In *International Conference on Learning Representations*, 2018.

[16] Y. Efroni, G. Dalal, B. Scherrer, and S. Mannor. Beyond the one step greedy approach in reinforcement learning. In *International Conference on Machine Learning*, 2019.

[17] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pages 278–287, 1999.

[18] E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[19] T. Jie and P. Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, pages 1000–1008, 2010.

[20] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*, 2017.

[21] H. Liu, Y. Feng, Y. Mao, D. Zhou, J. Peng, and Q. Liu. Action-depedent control variates for policy optimization via stein's identity. In *International Conference on Learning Representations*, 2018.

[22] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[23] G. Tucker, S. Bhupatiraju, S. Gu, R. E. Turner, Z. Ghahramani, and S. Levine. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*, 2018.

[24] S. Pankov. Reward-estimation variance elimination in sequential decision processes. *arXiv preprint arXiv:1811.06225*, 2018.

[25] C. Wu, A. Rajeswaran, Y. Duan, V. Kumar, A. M. Bayen, S. Kakade, I. Mordatch, and P. Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representation*, 2018.

[26] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[27] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[28] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[29] S. M. Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

[30] A. Vemula, W. Sun, and J. A. Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[31] S. M. Ross. *A course in simulation*. Prentice Hall PTR, 1990.

[32] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[33] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[34] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[35] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[36] K. Ciosek and S. Whiteson. Expected policy gradients for reinforcement learning. *arXiv preprint arXiv:1801.03326*, 2018.

[37] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

[38] K. L. Chung. *A course in probability theory*. Academic press, 2001.

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

[40] L. Landau and E. Lifshitz. Statistical physics (course of theoretical physics vol 5). 1958.

[41] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[42] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500, feb 2018.

[43] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

Appendix

A Missing Proofs

A.1 Proof for Theorem 4.1

To understand how the ordering matters, we consider a toy example of estimating $\mathbf{E}_{X,Y}[f(X,Y)]$ of some function f of two random variables X and Y. We prove a basic lemma.

Lemma A.1. If X and Y are independent, then

$$\mathbf{Var}_X \mathbf{E}_Y \left[f(X,Y)\right] \leq \mathbf{E}_Y \mathbf{Var}_X \left[f(X,Y)\right] \tag{14}$$

Proof. This can be proved by Jensen's inequality.

$$\mathbf{Var}_X \mathbf{E}_Y \left[f\right] = \mathbf{E}_X \left(\mathbf{E}_Y \left[f - \mathbf{E}_X \left[f\right]\right]\right)^2 \leq \mathbf{E}_X \mathbf{E}_Y \left[\left(f - \mathbf{E}_{X,Y} \left[f\right]\right)^2\right] = \mathbf{E}_Y \mathbf{Var}_X \left[f(X,Y)\right]$$

□

Suppose we want to reduce the variance of estimating $\mathbf{E}_{X,Y}[f(X,Y)]$ with some CV $\phi(X,Y)$ but only knowing the distribution $P(X)$, not $P(Y)$. Lemma A.1 tells us that in decomposing the total variance of $f(X,Y)$ to design this CV (cf. Section 2.3) we should take the decomposition

$$\mathbf{Var}_Y \mathbf{E}_X \left[f(X,Y)\right] + \mathbf{E}_Y \mathbf{Var}_X \left[f(X,Y)\right] \tag{15}$$

instead of the decomposition

$$\mathbf{Var}_X \mathbf{E}_Y \left[f(X,Y)\right] + \mathbf{E}_X \mathbf{Var}_Y \left[f(X,Y)\right] \tag{16}$$

In other words, we should take the ordering $Y \to X$, instead of $X \to Y$, when invoking the law of total variance. The reason is that after choosing the optimal CV for each case to reduce the variance due to X (the information that we have access to), we are left with $\mathbf{Var}_Y \mathbf{E}_X \left[f(X,Y)\right]$ and $\mathbf{E}_X \mathbf{Var}_Y \left[f(X,Y)\right]$, respectively, for $Y \to X$ and $X \to Y$. By Lemma A.1, we see the $Y \to X$ has a smaller residue in variance. In other words, when we only have partial information about the distribution, we should arrange the random variables whose distribution we know to the latter stage of the ordering, so that the CV we design can leverage the sampled observations to compensate for the lack of prior.

We use this idea to prove the natural ordering (10) in optimal. In analogy of X and Y, we have the action randomness whose distribution is known (i.e. the policy) and the dynamics randomness, whose distribution is unknown.

The potential orderings we consider come from first reparameterizing the policy and then ordering the independent random variables R_t (cf. Section 4.3). The Bayes networks of the MDP with and without policy reparameterization are depicted in Fig. 5, based on which we draw conditional independent relations later in the proof. We note that the CV is determined by the ordering, not due to reparameterization. For the natural ordering,

$$S_t \to A_t \to S_{t+1} \to A_{t+1} \to \cdots \to S_h \to A_h, \tag{10}$$

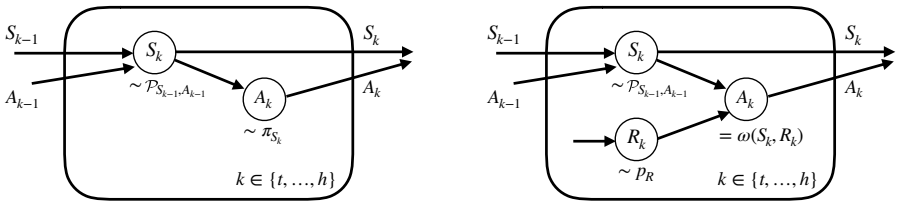
it gives the same control variate of the ordering below based on reparameterization

$$S_t \to R_t \to S_{t+1} \to R_{t+1} \to \cdots \to S_h \to R_h. \tag{17}$$

Suppose that given an ordering, we can compute its optimal CV. We define the variance left after applying that optimal CV associated with the ordering, the *residue* of that ordering. We will show that the residue is minimized at the natural ordering.

The proof consists of two steps.

1. We show that when dynamics is the MDP is unknown, an ordering is *feasible* to implement, if and only if, R_k appears before $S_{k+1..h}$ for all $t \leq k < h$. That is, a feasible ordering must be causal at least in actions: the action randomness that causes a state must be arranged before that state in the ordering. We prove this by contradiction. Assume otherwise S_u is



(a) before policy reparameterization (b) after policy reparameterization

Figure 5: Bayes networks for the random variables in G_t (2), before and after reparameterization. After policy is reparameterized, action A_k is decided by state S_k and action randomness R_k.

the *first* state before R_k satisfying $u > k$. We see that R_k and S_u are dependent, if none of the variables in $S_{k+1..u}$ is given. This observation can been inferred from the Bayes network that connect these random variables (Fig. 5b), i.e. the path from R_t to S_u is not blocked unless any of $S_{k+1..u}$ is observed [43]. Therefore, if we have an ordering that is violates the causality property defined above, the expectation over R_k required to define the difference estimator becomes intractable to compute, because the dynamics is *unknown*. This creates a contradiction.

2. We show that any feasible ordering can be transformed into the natural ordering in (10) using operations that do not increase the variance residue. We consider the following two operations
   (a) Suppose, in an ordering, there is $S_v \to S_u, v > u$, then we can exchange them without affecting variance residue.
   (b) Suppose, in a feasible ordering, there is $S_v \to R_k \to S_u$ with $v > u$ and $k \neq u, v$. Because this is a feasible ordering, we have $k + 1 \leq u < v$. This means that we can also move R_k after S_u. This change would not increase variance residue, because of the discussion after Lemma A.1. Then we change exchange the order of S_v and S_u too using the first operation.

   By using these two operations repeatedly, we can make all the states ordered by their subscripts, without increasing the residue. Finally, we can move R_k to just right after S_k without increasing residue using Lemma A.1 again. Thus, we arrive at the natural ordering in (17), which is the same as (10). In other words, the natural ordering is the optimal one among all feasible CVs that we can implement, which concludes the proof.

A.2 Proof of Theorem 3.1

Let $d_{\mathcal{A}}$ denote the dimension of $\mathcal{A}$. Suppose $d_{\mathcal{A}}$ is finite. To bound these variance terms, we derive some intermediate bounds. First, by the Gaussian assumption,

$$\pi_{S_t}(A_t) = (2\pi\sigma)^{-\frac{d_{\mathcal{A}}}{2}} \exp\left(\frac{-1}{2\sigma} \|A_t - \mu_\theta(S_t)\|^2\right)$$

we see that

$$N_t := \nabla \ln \pi(A_t|S_t) = \begin{bmatrix} \nabla_\theta \ln \pi(A_t|S_t) \\ \nabla_\sigma \ln \pi(A_t|S_t) \end{bmatrix} = \begin{bmatrix} \frac{-1}{\sigma} \nabla \mu_\theta(S_t)(A_t - \mu_\theta(S_t)) \\ \frac{1}{2\sigma^2} \|A_t - \mu_\theta(S_t)\|^2 - \frac{d_{\mathcal{A}}}{2\sigma} \end{bmatrix}$$

Therefore, for σ small enough, $\|N_t\| = O(\frac{\text{poly}(A_t)}{\sigma^2})$.

Second, by the assumption on boundedness of C, we have $C_{t:h} = O(h)$ and $Q_t := q^\pi(S_t, A_t) = O(h)$. We use these equalities to bound $\mathbf{E}_{|S_t}[N_t C_{t:h}]$. We observe that the identity that

$$\mathbf{E}_{|S_t}[N_t C_{t:h}] = \nabla \mathbf{E}_{A_t|S_t}[q^\pi(S_t, A_t)]$$

Under the assumption that q^π is analytic, q^π can be written in terms of an infinite sum of polynomials, i.e. $q^\pi(S_t, A_t) = \text{poly}_{S_t}(A_t)$, where the subscript remarks that these coefficients in the polynomial depends on S_t.

Now we are ready to bound $\mathbb{V}_{S_t}$, $\mathbb{V}_{A_t|S_t}$, and $\mathbb{V}_{|S_t,A_t}$. We recall that the expectation of polynomials over a Gaussian distribution depends only polynomially on the Gaussian's variance σ, with an order no less than 1. Therefore, for σ small enough, we have $\left\|\nabla \mathbf{E}_{A_t|S_t}[q^\pi(S_t, A_t)]\right\| = O(h)$ independent of σ, which implies that

$$\mathbb{V}_{S_t} = \mathbf{Tr}\left(\mathbf{Var}_{S_t}\left[\mathbf{E}_{|S_t}\left[N_t C_{t:h}\right]\right]\right) = o(h^2)$$

We can apply the same observation on the Gaussian expectation of polynomials and derive, for σ small enough,

$$\begin{aligned}\mathbb{V}_{A_t|S_t} &= \mathbf{Tr}\left(\mathbf{E}_{S_t}\left[\mathbf{Var}_{A_t|S_t}\left[N_t\left(\mathbf{E}_{|S_t,A_t}\left[C_{t:h}\right]\right)\right]\right]\right) \\ &= \mathbf{Tr}\left(\mathbf{E}_{S_t}\left[\mathbf{Var}_{A_t|S_t}\left[N_t q^\pi(S_t, A_t)\right]\right]\right) = O\left(\frac{h^2}{\sigma^4}\right)\end{aligned}$$

Similarly we can show

$$\mathbb{V}_{|S_t,A_t} = \mathbf{Tr}\left(\mathbf{E}_{S_t,A_t}[\mathbf{Var}_{|S_t,A_t}\left[N_t C_{t:h}\right]]\right) = O\left(\frac{h^2}{\sigma^4}\right)$$

This concludes the proof.

A.3 Bound for Variance of Policy Gradient

The variance of the policy gradient $\mathbf{Var}[G]$ can be bounded by the variance of policy gradient components $\{\mathbf{Var}[G_t]\}_{t=1}^n$. Appealing to the formula for the variance of the sum of two random variables

$$\mathbf{Var}[X+Y] = \mathbf{Var}[X] + \mathbf{Var}[Y] + 2\mathbf{Cov}[X, Y],$$

linearity of covariance

$$\mathbf{Cov}[X, Y+Z] = \mathbf{Cov}[X, Y] + \mathbf{Cov}[X, Z]$$

and Cauchy -Schwartz inequality

$$\mathbf{Cov}[X, Y] \leq \mathbf{Var}[X] + \mathbf{Var}[Y],$$

we can derive the following:

$$\begin{aligned}\mathbf{Var}[G] &= \mathbf{Var}\left[G_{1:h}\right] \\ &= \mathbf{Var}\left[G_1\right] + \mathbf{Var}\left[G_{2:h}\right] + \mathbf{Cov}[G_1, G_{2:h}] \\ &= \mathbf{Var}[G_1] + \sum_{t=2}^{h} \mathbf{Cov}[G_1, G_t] + \mathbf{Var}\left[G_{2:h}\right] \\ &= \sum_{t=1}^{h} \mathbf{Var}[G_t] + \sum_{u=1}^{h}\sum_{v=u+1}^{h} \mathbf{Cov}[G_u, G_v] \\ &\leq \sum_{t=1}^{h} \mathbf{Var}[G_t] + \sum_{u=1}^{h}\sum_{v=u+1}^{h} \left(\mathbf{Var}[G_u] + \mathbf{Var}[G_v]\right) \\ &= h\sum_{t=1}^{h} \mathbf{Var}[G_t]\end{aligned}$$

B Algorithm Example

Algorithm 1 specifies an instance of TrajCV, where Monte Carlo samples from a cheaper model simulator is used to approximate $\mathbf{E}_{A_t|S_t}[\widehat{Q}_t]$ (Line 8) and $\mathbf{E}_{A_t|S_t}[N_t\widehat{Q}_t]$ (Line 9). We discuss some other ways for approximation Appendix C.

In practice, the policy that's used for data collection may be different from the policy with respect to which the policy gradient is computed, e.g., when a whitening normalizer of the inputs to policy is

Algorithm 1: Policy gradient estimate with TrajCV

Input: policy π, single trajectory by running π: $\{s_t, a_t, c_t\}_{t=1}^h$, value function estimate $\widehat{v}$, deterministic dynamics estimate $\widehat{d}$, number of action samples I

Output: policy gradient estimate

for $t \leftarrow 1$ ***to*** h **do** `// collect statistics`
  $\widehat{q}_t \leftarrow \widehat{v}(\widehat{d}(s_t, a_t)) + c(s_t, a_t)$
  $n_t \leftarrow \nabla \log \pi_{s_t}(a_t)$
  for $i \leftarrow 1$ ***to*** I **do** `// Monte Carlo samples`
    Sample $a_i' \sim \pi_{s_t}$
    $\widehat{q}_i' \leftarrow \widehat{v}(\widehat{d}(s_t, a_i')) + c(s_t, a_i')$
    $n_i' \leftarrow \nabla \log \pi_{s_t}(a_i')$
  $\widetilde{\mathbf{E}}[\widehat{Q}_t] \leftarrow \frac{1}{I} \sum_{i=1}^I \widehat{q}_i'$
  $\widetilde{\mathbf{E}}[N_t \widehat{Q}_t] \leftarrow \frac{1}{I} \sum_{i=1}^I n_i' \widehat{q}_i'$
for $t \leftarrow 1$ ***to*** h **do** `// compute difference estimators`
  $\widetilde{G}_t^{\mathrm{Traj}} \leftarrow n_t c_{t:h} - \left(n_t \widehat{q}_t - \widetilde{\mathbf{E}}[N_t \widehat{Q}_t] \right) - n_t \sum_{k=t}^h \left(\widehat{q}_k - \widetilde{\mathbf{E}}[\widehat{Q}_t] \right)$ (11)
$\widetilde{G}^{\mathrm{Traj}} \leftarrow \widetilde{G}_{1:h}^{\mathrm{Traj}}$
return $\widetilde{G}^{\mathrm{Traj}}$

updated after data collection or when off-policy samples are utilized. Here we derive a TrajCV that takes this into account. Let π^d be the data collection policy and $W_t := \frac{\pi_{S_t}(A_t)}{\pi^d_{S_t}(A_t)}$ be the importance weights, and define $W_{a \to b} := \prod_{k=a}^b W_k$ for $b \geq a$ and $W_{a \to b} = 1$ for $b < a$, akin to : representing summation. Then we can write

$$\mathbf{E}_{\rho_\pi}[G_t] = \mathbf{E}_{\rho_{\pi^d}}[G_t W_{t \to h}] = \mathbf{E}_{\rho_{\pi^d}} \left[G_t W_{t \to h} - \sum_{k=t}^h W_{t \to (k-1)} \left(W_k N_t \widehat{Q}_k - \mathbf{E}_{A_k \sim \pi_{S_k}}[N_t \widehat{Q}_k] \right) \right]$$

Note that this TrajCV is unbiased, and when $\widehat{q} = q^\pi$, variance due to actions vanishes.

C Experiment Details

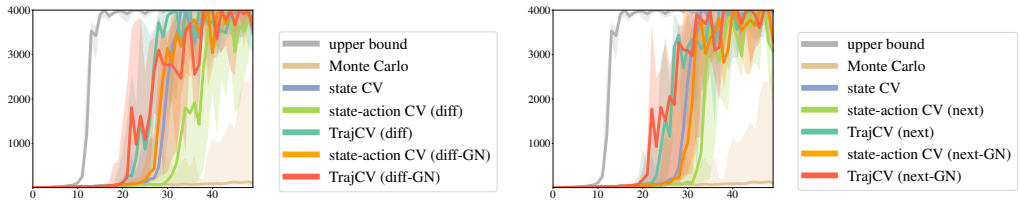
C.1 Setup

In CartPole, the reward function is the indicator function that equals to one when the pole is close to being upright and zero otherwise. This is a delayed reward problem in that the effective reward signal is revealed only when the task terminates prematurely before reaching the horizon, i.e. when the pole deviates from being upright. The start state is perturbed from being vertical and still by an offset uniformly sampled from $[-0.01, 0.01]^{d_\mathcal{S}}$, and the dynamics is deterministic.[10] The action space is continuous and Gaussian policies are considered in the experiments. The policy's mean function is a neural network with one hidden layers of 32 units and $\tanh$ activation, and a linear output layer. To be robust to outliers in data collection, the policy is optimized by natural gradient descent [3] with a KL-divergence safe guard on the policy change, such that a policy would change no more than 0.1 in the KL divergence averaged over the empirical state distribution on the data collected in each iteration.

C.2 Construction of Q-function Approximators

To facilitate a fair comparison across different CV techniques, we build all the CVs based on a an on-policy value function approximator $\widehat{v}$, which is a neural network with two hidden layers of 64 units each and $\tanh$ activation, and a linear output layer. In each iteration, we sample abundant data ($50,000$ state-action pairs) from a biased dynamics simulator (which is obtained by perturbing

[10] Symbol $d_\mathcal{S}$ denotes the dimension of $\mathcal{S}$.



(a) $h = 4000$ (diff) (b) $h = 4000$ (next)

Figure 6: The exact same settings as Fig. 4 except that *the state-action CV and TrajCV are given by* $\widehat{q}^{(\text{diff})}$ *and* $\widehat{q}^{(\text{diff-GN})}$ *(Fig. 6a), and* $\widehat{q}^{(\text{next})}$ *and* $\widehat{q}^{(\text{next-GN})}$ *(Fig. 6b).*

each underlying physical parameter relatively by 10%), and then fit $\widehat{v}$ to these biased Monte-Carlo estimates with a quadratic loss using ADAM (stepsize 0.001; $\beta_1 = 0.9$ and $\beta_2 = 0.999$) for $1,024$ batches with batchsize 128. The reason for using a biased dynamics simulator in lieu of the on-policy data from the real environment is that we only sample 5 trajectories per iteration, which amount to around 100 data points in the early iterations and can be too scarce to build a reasonable function approximator.

As mentioned, all the CVs are built using the above policy evaluation technique. (Different methods learn its own value function approximator on-the-fly along the progress of policy optimization.) For the state-dependent CV, the usage of $\widehat{v}$ is straightforward. For state-action CV and TrajCV, we use $\widehat{v}$ to further construct the needed Q-function approximator $\widehat{q}$. This is done as follows: First, we further train a deterministic function $\widehat{d}$ that maps the current state and action to next state using the same data collected from the true environment that are used for computing the policy gradient estimates. As policy optimization progresses, we aggregate the data from the past rounds to iteratively build this dynamics model (which is another neural network with two hidden layers of 64 units each and $\tanh$ activation and a linear output layer). This is done by updating it after the policy gradient step in each iteration to remove undesirable correlations. Next, we use the above value function approximator $\widehat{v}$ and the dynamics approximator $\widehat{d}$ to define a natural Q-function approximator $\widehat{q}^{(\text{dyn})}(s,a) = c(s,a) + \widehat{v}(\widehat{d}(s,a))$. Based on this basic $\widehat{q}^{(\text{dyn})}$, we explore several options of Q-function approximator for defining the state-action CV and TrajCV:

1. Monte Carlo (MC) : $\widehat{q}^{(\text{dyn})}(s,a)$. We use many samples of actions ($1,000$ in the experiments) to approximate $\mathbf{E}_{A_t|S_t}\left[\widehat{q}^{(\text{dyn})}(S_t,A_t)\right]$. To reduce variance, we use the same action randomness for different steps, i.e. using the same $1,000$ i.i.d. samples from p_R (defined in Section 4.3) in the evaluation for $\mathbf{E}_{A_t|S_t}$ with different t.
2. We also consider various Q-function approximators that are quadratic in action, so that $\mathbf{E}_{A_t|S_t}$ can be evaluated in closed-form. They are derived by different linearizations of the Q-function approximator $\widehat{q}$ as shown below.
   (a) $\widehat{q}^{(\text{next})}(s,a) = c(s,a) + \widehat{v}(\widehat{s}') + (a-m)^\top \nabla_m \widehat{d}(s,m) \nabla \widehat{v}(\widehat{s}')$,
   (b) $\widehat{q}^{(\text{next-GN})}(s,a) = \widehat{q}^{(\text{next})}(s,a) + \frac{1}{2}(a-m)^\top \nabla_m \widehat{d}(s,m) \nabla^2 \widehat{v}(\widehat{s}') \nabla_m \widehat{d}(s,m)^\top (a-m)$,
   (c) $\widehat{q}^{(\text{diff})}(s,a) = \widehat{v}(s) + (a-m)^\top \nabla_m (c(s,m) + \widehat{v}(\widehat{s}')) + \frac{1}{2}(a-m)^\top \nabla_m^2 c(s,m)(a-m)$,
   (d) $\widehat{q}^{(\text{diff-GN})}(s,a) = \widehat{q}^{(\text{diff})}(s,a) + \frac{1}{2}(a-m)^\top \nabla_m \widehat{d}(s,m) \nabla^2 \widehat{v}(\widehat{s}') \nabla_m \widehat{d}(s,m)^\top (a-m)$,

   where $m = \mu_\theta(s)$ is the mean of the Gaussian policy, $\widehat{s}' = \widehat{d}(s,m)$, and "GN" stands for Gauss-Newton. We assume $c(s,a)$ is quadratic in a for $\widehat{q}^{(\text{next})}$ and $\widehat{q}^{(\text{next-GN})}$.

Note to Practitioners We emphasize that constructing a Q-function approximator *indirectly* through a dynamics model and a value function approximator is not ideal for practical purposes. This approach would combine errors from two sources and can have worse performance than directly estimating a Q-function, e.g., through (simulated) Monte Carlo samples. However, we adopted this formulation to make the results of different CVs more comparable, removing the bias due to different value function approximators and evaluation techniques. While this construct is sufficient for the purpose of comparing theoretical properties here, we do remind that this scheme does not scale well to general high-dimensional problems.

C.3 Extra Experimental Results

The performance of different CVs using MC for approximating $\mathbf{E}_{A_t|S_t}$ is reported in Fig. 4. We provide the experimental results of these quadratic Q-function approximators in Fig. 6, where the setup is the same those in Fig. 4.

Finally, we note that because the recent technical report [24] essentially proposed the same equation (11) that TrajCV uses. We invite the readers to refer to their encouraging empirical results on simulated LQG tasks too.