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,

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

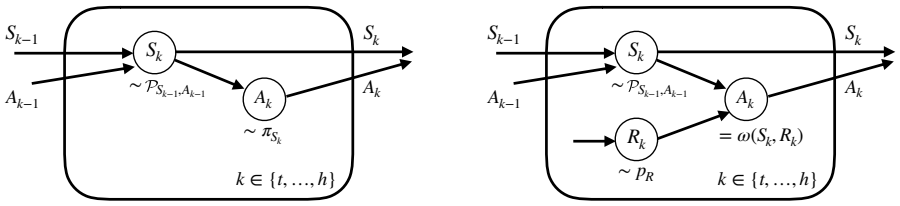
it gives the same control variate of the ordering below based on reparameterization

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

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

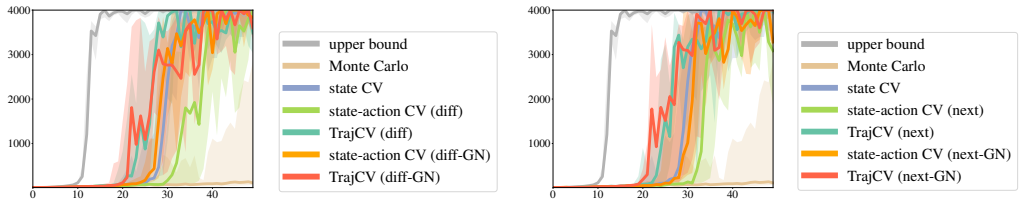
C.1 Setup

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.