Variance Reduced Domain Randomization for Reinforcement Learning With Policy Gradient

Yuankun Jiang, Chenglin Li, *Member, IEEE*, Wenrui Dai, *Member, IEEE*, Junni Zou, *Member, IEEE*, and Hongkai Xiong, *Senior Member, IEEE*

***Abstract*—By introducing randomness on the environments, domain randomization (DR) imposes diversity to the policy training of deep reinforcement learning, and thus improves its capability of generalization. The randomization of environments, however, introduces another source of variability for the estimate of policy gradients, in addition to the already high variance incurred by trajectory sampling. Therefore, with standard state-dependent baselines, the policy gradient methods may still suffer high variance, causing a low sample efficiency during the training of DR. In this paper, we theoretically derive a bias-free and state/environment-dependent optimal baseline for DR, and analytically show its ability to achieve further variance reduction over the standard constant and state-dependent baselines for DR. Based on our theory, we further propose a variance reduced domain randomization (VRDR) approach for policy gradient methods, to strike a tradeoff between the variance reduction and computational complexity for the practical implementation. By dividing the entire space of environments into some subspaces and then estimating the state/subspace-dependent baseline, VRDR enjoys a theoretical guarantee of variance reduction and faster convergence than the state-dependent baselines. Empirical evaluations on six robot control tasks with randomized dynamics demonstrate that VRDR not only accelerates the convergence of policy training, but can consistently achieve a better eventual policy with improved training stability.**

***Index Terms*—Deep reinforcement learning, policy gradient, domain randomization, variance reduction.**

I. Introduction

DEEP reinforcement learning (DRL) has achieved an impressive success on complex sequential decision-making tasks, such as playing video games at top human-level [1], continuous robot manipulation [2], [3], and traffic signal control [4]. For an extensive exploration, DRL requires a massive amount of samples to train the policy, which is usually done in a simulator constructed for the task. Due to lack of diversity, however, the policy trained by DRL tends to overfit to a specific training environment [5], [6], causing possibly a severe performance degradation (i.e., reality gap) when the policy learned in the simulator is transferred to a real deployment [7], [8]. To narrow this gap, domain randomization (DR) has been proposed to randomize the environment parameters that may vary the dynamics and observation in simulator, imposing diversity on the trained policy [9], [10]. Recent success in robot control has shown that DR enables the trained policy to generalize in real deployment, even with presence of fundamental modeling errors in the simulator [11], [12]. Besides, DR has also demonstrated its capability of robustness to contend with extremely rare environments [11].

Direct applications of DR on policy gradient methods, in essence, can be thought as the policy optimization for multiple randomly generated environments, which incurs additional variability of the observed data. Therefore, a critical challenge here is the low sample efficiency due to the extremely high variance of the gradient estimate, where the variability not only stems from the gradient approximation of expected return from a batch of sampled trajectories [13], but is also imposed by the randomization of environment parameters. In standard DRL, a commonly used method to reduce such a variance is to construct a bias-free and action-independent baseline that is subtracted from the expected return [14]. For example, a typical choice of baseline is the state value function of a policy. When directly applied to DR, as stated by [15], it is equivalent to learning a state value function that predicts the expected return over all the possible environments. Though remaining unbiased, such a state-dependent baseline may be a poor choice in DR, since the additional variability of randomized environments is not taken into account.

In this paper, we aim to address the high variance issue of policy gradient methods for domain randomization, with a particular focus on reducing the additional variance imposed by the randomization of environments. Our key insight is that the additional information on varying environments can be incorporated into the baseline to further reduce this variance. Theoretically, we derive the optimal state/environment-dependent baseline, and demonstrate that it improves consistently the variance reduction over baselines that either are constant or use the state information only. In order for the practical implementation to strike the tradeoff between the variance reduction performance and computational complexity for maintaining the state/environment-dependent baselines, we further propose a

Manuscript received 14 March 2023; revised 23 September 2023; accepted 20 October 2023. Date of publication 6 November 2023; date of current version 8 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grants 62250055, T2122024, 62125109, 61831018, 62371288, 62320106003, 61931023, 61932022, 61972256, 61971285, and 62120106007, in part by the Program of Shanghai Science and Technology Innovation Project under Grant 20511100100, and in part by the Key Research and Development Plan of Shandong Province under Grant 2020CXGC010701. Recommended for acceptance by X. Li. *(Corresponding author: Chenglin Li.)*

The authors are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yuankunjiang@sjtu.edu.cn; lcl1985@sjtu.edu.cn; daiwenrui@sjtu.edu.cn; zoujunni@sjtu.edu.cn; xionghongkai@sjtu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3330332, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3330332

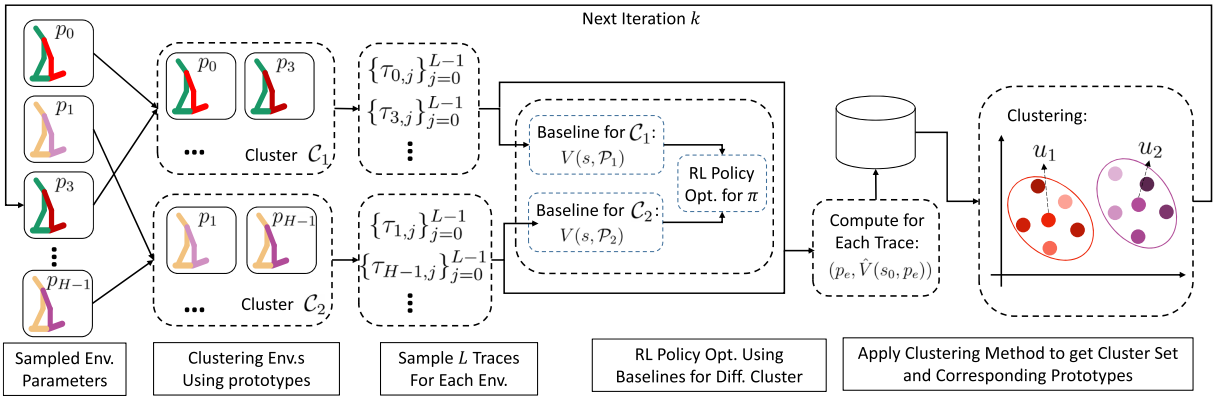


Fig. 1. System overview of the proposed variance reduced domain randomization (VRDR) approach for reinforcement learning with policy gradient. Different color of the Walker robot indicates different dynamics in each environment p. For example, the robots in environments p_0 and p_3 have the same dynamics of left legs indicated by green, whereas the dynamics of their right legs are also similar as indicated by different shades of red. Hence, p_0 and p_3 have similar overall system dynamics, and are grouped into the same cluster $\mathcal{C}_1$. In VRDR, we adopt the clustering method to group sampled environments and the trajectories sampled within them in DR settings, following the criterion proposed in Section V. Then, the policy is optimized by using policy gradient method on grouped trajectories with the baselines trained specifically for each cluster. Prototypes of these clusters are periodically computed and updated in the history buffer $\mathcal{B}$.

variance reduced domain randomization (VRDR) approach for policy gradient methods. The overall framework of VRDR is shown in Fig. 1. In VRDR, environments generated by sampled environment parameters are grouped into different clusters according to the prototype of each cluster. Then, the baseline for each cluster is trained on sampled traces from these environments, which is utilized to reduce the variance in policy gradient methods. Specifically, the clustering prototypes are computed based on the estimated policy performance (cumulative rewards of the trace) in each environment. Thus, VRDR can improve the sample efficiency while keeping a reasonable number of baselines associated with a set of specifically determined environment subspaces. Our main contributions of this paper can be summarized as follows.

- *Theoretically optimal state/environment-dependent baseline for DR:* For the policy gradient in a variety of environments that differ in dynamics, we derive theoretically an optimal baseline that depends on both the states and environments. We further quantify analytically the variance reduction improvement achieved by the proposed state/environment-dependent baseline over two common choices of baselines for DR, i.e., the constant and state-dependent baselines.
- *Criterion for constructing practical state/subspace-dependent baseline:* Since the accurate estimation of state/environment-dependent baseline for each possible environment is infeasible during practical implementation of DRL, we propose to alternatively divide the entire space of environment parameters into a limited number of subspaces, and estimate instead the optimal baseline for every pair of state and environment subspace. We further show that the clustering of environments into subspaces should follow the policy's expected returns on these environments, which can guarantee an improvement of variance reduction over the state-dependent baseline.
- *Variance reduced domain randomization (VRDR) Approach with empirical evaluation:* To strike the trade-off between the variance reduction performance and computational complexity for maintaining optimal baselines, we develop a variance reduced domain randomization (VRDR) approach for policy gradient methods. Specifically, VRDR learns an acceptable number of baselines for each pair of state and environment subspace, where the environment subspaces are determined based on the above criterion. We then conduct experiments on six robot control tasks with their fundamental physical parameters randomized, demonstrating that VRDR can generally accelerate the convergence of policy training in DR settings, as compared to other state-of-the-art methods. In addition to the faster convergence, VRDR can also achieve a better eventual policy with improved training stability.

The remainder of this paper is organized as follows. Section II reviews existing literature on domain randomization and variance reduction. Section III provides the preliminary of reinforcement learning, policy gradient method using domain randomization, and the impact of gradient's variance to the convergence. In Section IV, we derive the optimal baseline for domain randomization under several baseline choices, and theoretically show that the proposed optimal state/environment-dependent baseline achieves the minimal variance among them. In Section V, we further propose a practical state/subspace-dependent baseline and corresponding criterion to construct the subspaces, based on which variance reduction domain randomization (VRDR) is formally presented. Sections VI and VII presents experimental results and draws the conclusions, respectively.

II. Related Work

State-of-the-art methods that have endeavored to tackle the generalization and robustness of DRL [16], including DR, meta RL and robust RL, all implicitly or explicitly impose diversity on the policy training. DR and meta RL both tend to learn a policy that can generalize to a variety of environments [15], [17], which may differ in dynamics and reward function during training. By introducing an adversary that acts to disturb the dynamics or the

observation of the environment during training [18], the learned policy can gain robustness on deployment. The introduced diversity for training environments enables the generalization of DRL policy, while the consequent high variance aggravates the sample efficiency of DRL training. Current research on tackling this high variance problem can be broadly classified into two categories: optimization for the sampling distribution of environment parameters and variance reduction for the gradient descent.

A. Sampling Distribution Optimization

In standard DR, distribution for sampling the environment parameters is restricted to the uniform or Gaussian distribution. For policy optimization under DR, Mehta et al. [19] experimentally demonstrate that the uniform sampling of DR leads to a suboptimal and high-variance policy. They are thus motivated to develop the active domain randomization, which leverages stein variational policy gradient [20] to update the sampling distribution to select informative environments for generalization. In [10], we derive a lower bound for the worst-case policy performance in DR setting, and propose the monotonic robust policy optimization to update the sampling distribution to achieve a better worst-case performance. Fingerprint policy optimization (FPO) [21] utilizes Bayesian optimization to actively select the sampling distribution for improving the performance gained at each iteration. Similar to FPO, Mozifian et al. [22] propose to directly optimize the distribution to maximize the Monte-Carlo estimated policy's performance and stay close to a prior distribution. Neural posterior domain randomization [23] utilizes a likelihood-free inference to update the posterior sampling distribution to match with the observed data. However, the baseline optimization is seldom considered in the policy search by using policy gradient, while a direct application of the state-dependent baseline is commonly used as formulated in (8).

B. Variance Reduction for Gradient Descent

Like other stochastic gradient (SG) optimization methods, even when not under the DR setting, policy gradient methods may still suffer high variance due to the approximation of gradient from the randomly sampled subset of trajectories. In addition, poor training outliers are another important source of variance [24]. Hence, policy gradient-based RL methods may also benefit from methods for reducing variance in SG, upon which control variate is one of the prevalent techniques, such as stochastic variance-reduced gradient (SVRG) [25], SAGA [26], and their variants [27], [28], [29], [30]. Control variate leverages correlation between random estimates and typically subtracts the estimator by a baseline function, which is often zero in expectation for unbiasedness or can be nonzero for a better variance reduction. Variance reduced gradient algorithms like SVRG and STOchastic Recursive Momentum (Storm) [31] have been studied in the policy gradient for RL [32], [33]. Storm has also been utilized in meta-learning and tested in classification tasks [34].

The idea of control variate has also been widely used in RL, by subtracting a commonly used state-dependent baseline from the gradient estimate to reduce the variance without introducing bias [14]. Beyond considering state-dependent, several baseline functions that incorporate rich information have been developed. Cheng et al. [35] propose trajectory-wise baseline function that considers correlation across trajectories. Spooner et al. [36] exploit the independence structure of action space by incorporating prior knowledge, and propose a factored baseline to further reduce the variance in the problems with factored action space, which relates to the action-dependent baseline in multi-dimension single-agent [37], COMA in multi-agent systems [38], and multi-objective MDP. Guo et al. [39] propose a hindsight value function as baseline, which leverages the embeddings of the future states and rewards. For multi-agent system, Kuba et al. [40] derive the optimal baseline for policy gradient of each agent, which depends on the states and actions from the other agents. These methods are designed for standard RL and require further discussion under the context of policy optimization under DR.

For policy optimization under DR or meta-RL setting, Peng et al. [7] propose to combine the state and environment parameters as the input to the value function, namely DR with the combined state-dependent baseline (DRCS). The empirical evaluation later shows that DRCS may cause a degradation of the training process as compared to DR and VRDR. Mao et al. [41] propose an input-dependent baseline to reduce the variance of stochastic input process, which is similar to our state/environment-dependent baseline. Then, the multi-value-network (MVN) approach and meta-learning approach are proposed to learn the baseline. Meanwhile, Liu et al. [42] propose to learn the per-task control variate and the meta control variate for each meta-RL task, where the per-task variate is constructed by the action-dependent baseline. In essence, the MVN and per-task control variates approaches are the same, since they both aim to learn a baseline for each input or task. And both of these meta-learning approaches are built upon model-agnostic meta learning (MAML) [43] to scale when the number of input instantiations or tasks are large.

III. Background

A. Notations

Under the standard reinforcement learning (RL) setting, the environment is modeled as a Markov decision process (MDP) defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}_p, \Phi \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. For the convenience of derivation, we assume that they are both finite. $\mathcal{T}_p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environment transition model that is essentially determined by environment parameter $p \in \mathcal{P}$, with $\mathcal{P}$ denoting the space of environment parameters. In robot control, for example, environment parameter p can be a vector containing the rolling friction of each joint and the mass of torso and arms. Throughout the rest of this paper, by environment p we mean that the environment has dynamics determined by the parameter p. Finally, $\Phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function.

At each time step t, the RL agent observes its state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$ under the guidance of policy $\pi_\theta(\mathrm{a}_t|\mathrm{s}_t)$ parameterized by θ. It will then receive a reward $r_t = \Phi(s_t, a_t)$,

while the environment shifts to the next state s_{t+1} with probability $\mathcal{T}(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t, p)$. The goal of standard RL is to search for a policy π that maximizes the expected discounted return $\eta(\pi, p) = \mathbb{E}_\tau[R(\tau)]$ over all the possible trajectories of states and actions, where $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$ is the discounted return of trajectory $\tau = \{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{\infty}$ and $\gamma \in [0, 1]$ is the discount factor. We can then define the state value function as $V_\pi(s, p) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \mathrm{s}_t = s]$, the action value function as $Q_\pi(s, a, p) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \mathrm{s}_t = s, \mathrm{a}_t = a]$, and the advantage function as $A_\pi(s, a, p) = Q_\pi(s, a, p) - V_\pi(s, p)$.

B. Policy Gradient Methods for Domain Randomization

In domain randomization (DR), the environment parameter p is a random variable that follows the probability distribution P over $\mathcal{P}$. For the convenience of derivation, we assume a finite environment parameter space $\mathcal{P}$ with cardinality $|\mathcal{P}|$. By introducing DR, the goal of policy optimization is to maximize the expected return over all the possible environment parameters: $\mathbb{E}_{p\sim P}[\eta(\pi, p)]$. The policy gradient with an action-independent baseline [14] is then formulated as

$$\nabla_\theta \mathbb{E}_{p\sim P}\left[\eta(\pi, p)\right] = \mathbb{E}_P\left[\mathbb{E}_{\mu_\pi^p, \pi}\left[\nabla_\theta \log \pi_\theta(a|s)\left[Q_\pi(s, a, p) - b\right]\right]\right], \quad (1)$$

where $\mu_\pi^p(s) = \sum_{t=0}^{\infty} \gamma^t P_\pi(s_t = s|p)$ is defined as the discounted state visitation frequency, with $P_\pi(s_t = s|p)$ denoting the probability of shifting from the initial state s_0 to state s after t steps under policy π in environment p. For convenience, we further denote $g(\theta, s, a, p) \triangleq \nabla_\theta \log \pi_\theta(a|s)[Q_\pi(s, a, p) - b]$, which is the gradient estimate for the state-action pair under environment p. By using the Monte-Carlo estimate $\hat{Q}_\pi(s, a, \tau, p)$ of the action value function $Q_\pi(s, a, p)$ via computation based on the sampled trajectory τ, we can then have a Monte-Carlo estimate of g and denote it as $\hat{g}(\theta, s, a, \tau, p) \triangleq \nabla_\theta \log \pi_\theta(a|s)[\hat{Q}_\pi(s, a, \tau, p) - b]$. As long as the baseline b is action-independent, we have $\mathbb{E}_\mathrm{a}[\nabla_\theta \log \pi_\theta(a|s)b] = \nabla_\theta \mathbb{E}_\mathrm{a}[b] = 0$ and thus $\mathbb{E}_{P,\mu_\pi^p,\pi}[g] = \mathbb{E}_{P,\mu_\pi^p,\pi}[\nabla_\theta \log \pi_\theta(a|s) Q_\pi(s, a, p)] \triangleq \mathbb{E}[g]$, where the subscript is dropped for convenience. Therefore, by subtracting this action-independent baseline b, the variance of gradient estimate can be reduced without introducing any bias. In this paper, we consider reducing this variance of gradient estimate by minimizing the trace of covariance matrix w.r.t. the baseline function b as in [13], which is denoted as $Var(g) = \mathbb{E}[g^\mathrm{T} g] - \mathbb{E}[g]^\mathrm{T} \mathbb{E}[g]$.

C. Convergence Analysis w.r.t. Variance of g

The variability of sampling trajectories in policy gradient disturbs the convergence of training, which resembles the stochastic gradient, hence resulting in an inefficient sampling [44]. Let $\nabla J(\theta) = \nabla_\theta \mathbb{E}[\eta(\pi, p)]$ denote the policy gradient with the expectation taken over all randomness. By applying an arbitrary gradient estimate, the convergence of policy gradient is then upper bounded as follows.

Theorem III.1: [45] Under the Lipschitz continuity assumption of $\nabla J(\theta)$ w.r.t. θ, and for a proper choice of the step size α_k, we have

$$\mathbb{E}\|\nabla J(\theta)\|_2^2 \le \frac{L(J(\theta^*) - J(\theta_1))}{K/2} + \frac{L \max_k \sqrt{Var(g_k)}}{\sqrt{K}}, \quad (2)$$

where $J(\theta^*)$ is the maximum expected cumulative discounted reward that can be achieved by policy π_{θ^*}, L is the Lipschitz constant, K is the maximum number of iterations for policy gradient method, $g_k = g(\theta_k, s, a, p)$ is the gradient estimate at the kth iteration for (s, a) and p, and

$$\mathbb{E}\|\nabla J(\theta)\|_2^2 = \frac{\sum_{k=1}^{K}(\alpha_k - L/2\alpha_k^2)\mathbb{E}\|\nabla J(\theta_k)\|_2^2}{\sum_{i=1}^{K}(\alpha_i - L/2\alpha_i^2)}, \quad (3)$$

is the expectation taken over the distribution $P(\theta_k) = \frac{\alpha_k - L/2\alpha_k^2}{\sum_{i=1}^{K}(\alpha_i - L/2\alpha_i^2)}$ and all the other possible randomness.

According to the RHS of (2), the variance of gradient estimate dominates the convergence. Theorem III.1 thus suggests that, by reducing the variance of gradient estimate can the policy gradient methods enjoy an acceleration in their convergence. This then provides a theoretical guideline for derivation of the optimal baselines in the DR setting.

IV. Optimal Baselines for Domain Randomization

To derive the optimal baselines for DR, we formulate the following optimization problem that aims to minimize the variance of gradient estimate w.r.t. the baseline b [13]:

$$\min_b \mathbb{E}_{P,\mu_\pi^p,\pi}\Big[\underbrace{G(a, s)\left[Q_\pi(s, a, p) - b\right]^2}_{g^\mathrm{T} g}\Big] - \mathbb{E}[g]^\mathrm{T}\mathbb{E}[g]$$
$$\Leftrightarrow \quad \min_b \mathbb{E}_{P,\mu_\pi^p,\pi}\Big[G(a, s)\left[Q_\pi(s, a, p) - b\right]^2\Big], \quad (4)$$

where $G(a, s) \triangleq \nabla_\theta \log \pi_\theta(a|s)^\mathrm{T} \nabla_\theta \log \pi_\theta(a|s)$. Due to its independence of b, the second term $\mathbb{E}[g]^\mathrm{T}\mathbb{E}[g]$ does not affect the minimizer, and is thus omitted on the RHS of (4). In the following, we first derive for DR two common choices of baselines that are constant or depend only on the state. We then develop the optimal state/environment-dependent baseline, and show its ability to further reduce the variance incurred by the randomization of environment parameters.

A. Common Choices of Action-Independent Baselines

1) Optimal Constant Baseline: We first consider a constant baseline b_c that depends neither on the action nor on the state. The optimization problem in (4) then becomes minimizing the expectation of a quadratic function, which can be proved to be convex. Hence the optimal constant baseline b_c^* for DR-based policy gradient methods is

$$b_c^* = \frac{\mathbb{E}_{P,\mu_\pi^p,\pi}\left[G(a, s) Q_\pi(s, a, p)\right]}{\mathbb{E}_{P,\mu_\pi^p,\pi}\left[G(a, s)\right]} = \mathbb{E}_{P,\mu_\pi^p}\left[V'(s, p)\right]. \quad (5)$$

Proof: We denote the optimization problem w.r.t. the optimal constant baseline b_c as

$$\min_{b_c} F(b_c) \triangleq \mathbb{E}_{P,\mu_\pi^p,\pi}\left[G(a,s)\left[Q_\pi(s,a,p)-b_c\right]^2\right]. \quad (6)$$

Given convexity of the optimization problem, we can simply let the first-order derivative equal to zero and get the optimal constant baseline b_c^*

$$\frac{dF(b_c)}{db_c} = -2\mathbb{E}_{P,\mu_\pi^p,\pi}\left[G(a,s)[Q_\pi(s,a,p)-b_c]\right] = 0, \quad (7)$$

which solves $b_c^* = \frac{\mathbb{E}_{P,\mu_\pi^p,\pi}[G(a,s)Q_\pi(s,a,p)]}{\mathbb{E}_{P,\mu_\pi^p,\pi}[G(a,s)]}$ and completes the proof. □

This optimal baseline b_c^* can be understood as the expected state value function $V'(s,p)$ over all states and possible environments, where the state value function $V'(s,p)$ is computed as the weighted average of the action value function: $V'(s,p) = \mathbb{E}_\pi[\frac{G(a,s)}{\mathbb{E}_{P,\mu_\pi^p,\pi}[G(a,s)]}Q_\pi(s,a,p)]$.

2) Optimal State-Dependent Baseline: As in standard RL, the utilization of state-dependent baseline in the DR setting can be considered as an expected value function $b(s) = \mathbb{E}_P[\mathbb{E}_\pi[Q_\pi(s,a,p)]]$ for each state over all possible environments, which predicts the expected return over the distribution of dynamics caused by the variation of environment parameters. The optimal state-dependent baseline for DR is given by

$$b^*(s) = \frac{\mathbb{E}_{P(p|s)}\mathbb{E}_\pi[G(a,s)[Q_\pi(s,a,p)]]}{\mathbb{E}_\pi[G(a,s)]}. \quad (8)$$

Proof: We provide a proof sketch here. Please see Appendix A.1 for the detailed proof, available online. Similar to the problem formulation of constant baseline, we can formulate the variance minimization problem w.r.t. $b(s)$ as

$$\min_{b(s)} F(b(s)) \triangleq \mathbb{E}_{P,\mu_\pi^p,\pi}\left[G(a,s)[Q_\pi(s,a,p)-b(s)]^2\right]. \quad (9)$$

We consider all the variables to be discrete here to illustrate the derivation, and thus denote the state-dependent baseline as $b(s) = \{b(s_j)\}$ with a bit of ambiguity. By letting the first-order partial derivative equal to zero, we have

$$\frac{\partial F(b(s))}{\partial b(s)} = -2\sum_{j=1}^{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{P}|}P(p_i)\mu_\pi(s_j|p_i) \cdot \mathbb{E}_\pi\left[G(a,s_j)[Q_\pi(s_j,a,p_i)-b(s_j)]\right] = 0. \quad (10)$$

With the chain rule of probability, we have

$$P(p_i,s_j) = P(p_i)\mu_\pi(s_j|p_i) = P(s_j)\mu_\pi(p_i|s_j). \quad (11)$$

Substituting (11) into (10) and following the equivalence transformation of equation, we have $b^*(s_j) = \frac{\mathbb{E}_{P(p|s_j)}\mathbb{E}_\pi[G(a,s_j)[Q_\pi(s_j,a,p)]]}{\mathbb{E}_\pi[G(a,s_j)]}$ and $b^*(s)$ in (8) is its counterpart for the continuous state space. Note that we focus on the discrete environment parameter space in this paper, but the same result can be obtained for the continuous case by simply replacing the sum of p_i with the integration of p in the derivation. □

B. Optimal State/Environment-Dependent Baseline

In DR, the randomization of environment parameters imposes additional variability for the estimate of gradients, which aggravates the instability of RL training process. In the following subsections, we show that this variance can be effectively reduced by further incorporating into the baseline the information about varying environments. To this end, we propose a state/environment-dependent baseline $b(s,\mathcal{P}) = \{b(s,p_i)\}_{i=1}^{|\mathcal{P}|}$ for DR, and rewrite the variance minimization in (4) as

$$\min_{b(s,\mathcal{P})} F(b(s,\mathcal{P})) = \mathbb{E}_{p\sim P,\mu_\pi^p}\left[\mathbb{E}_\pi\left[G(a,s)\left[Q_\pi(s,a,p)-b(s,p)\right]^2\right]\right]. \quad (12)$$

Following the similar deduction as in [13], the *optimal state/environment baseline* can be derived as:

$$b^*(s,\mathcal{P}) = \{b^*(s,p_i)\}_{i=1}^{|\mathcal{P}|},$$
$$\text{where } b^*(s,p_i) = \frac{\mathbb{E}_\pi\left[G(a,s)Q_\pi(s,a,p_i)\right]}{\mathbb{E}_\pi\left[G(a,s)\right]}. \quad (13)$$

Proof: Still following the convexity of the optimization problem and letting the first-order partial derivative of each environment p_i equal to zero, we have

$$\frac{\partial F(b(s,\mathcal{P}))}{\partial b(s,p_i)} = -2\mathbb{E}_\pi\left[G(a,s)\left[Q_\pi(s,a,p)-b(s,p_i)\right]\right] = 0. \quad (14)$$

□

The above optimal state/environment baseline indicates requirements of maintaining a specific baseline for each environment to incorporate the environment-specific information. For the case of continuous environment parameter space, the optimal state/environment baseline can be similarly given as $b^*(s,p) = \mathbb{E}_\pi[G(a,s)Q(s,a,p)]/\mathbb{E}_\pi[G(a,s)]$. Note that if the environment parameter p_i is time-varying along episodes and considered as a stochastic input process that partially affects the dynamics, this optimal baseline $b^*(s,\mathcal{P})$ is equivalent to the optimal input-driven baseline as derived in [41].

C. Variance Reduction Improvement of $b^*(s,\mathcal{P})$

Theorem IV.1: Let $Var^{b(s)}(g)$ and $Var^{b^*(s,\mathcal{P})}(g)$ denote the variance of gradient estimate by incorporating an arbitrary state-dependent baseline $b(s)$ and the optimal state/environment-dependent baseline $b^*(s,\mathcal{P})$, respectively. Compared to the state-dependent baseline $b(s)$, the variance can be further reduced by $b^*(s,\mathcal{P})$, with the following improvement:

$$Var^{b(s)}(g) - Var^{b^*(s,\mathcal{P})}(g) = \mathbb{E}_{P,\mu_\pi^p}\left[\sqrt{\mathbb{E}_\pi\left[G(a,s)\right]}b(s) - \frac{\mathbb{E}_\pi\left[G(a,s)Q_\pi(s,a,p)\right]}{\sqrt{\mathbb{E}_\pi\left[G(a,s)\right]}}\right]^2. \quad (15)$$

Proof: We provide a proof sketch here. Please see Appendix A.2 for the detailed proof, available online. The variance of

policy gradient using the optimal state/environment-dependent baseline $b^*(s,\mathcal{P})$ can be reformulated from the definition of variance as

$$Var^{b^*(s,\mathcal{P})}(g) = \mathbb{E}_{P,\mu_\pi^p}[X_p(s)] - \mathbb{E}_{P,\mu_\pi}\left[\frac{Y_p^2(s)}{Z}\right] - \mathbb{E}[g^{\mathrm{T}}]\mathbb{E}[g], \tag{16}$$

where we denote $X_p(s) \triangleq \mathbb{E}_\pi[G(a,s)Q_\pi^2(s,a,p)], Y_p(s) \triangleq \mathbb{E}_\pi[G(a,s)Q_\pi(s,a,p)]$ and $Z \triangleq \mathbb{E}_\pi[G(a,s)]$. Note that $Var^{b(s)}(g) = F(b(s)) - \mathbb{E}[g^{\mathrm{T}}]\mathbb{E}[g]$, by subtracting $Var^{b(s)}(g)$ with $Var^{b^*(s,\mathcal{P})}(g)$ in (16), we have

$$Var^{b(s)}(g) - Var^{b^*(s,\mathcal{P})}(g) = \mathbb{E}_{P,\mu_\pi^p}\mathbb{E}_\pi\Bigg[G(a,s) \cdot \left[(Q_\pi(s,a,p) - b(s))^2 - \left(Q_\pi(s,a,p) - \frac{Y_p(s)}{Z}\right)^2\right]\Bigg], \tag{17}$$

which can be reduced to the quadratic form in (15) and completes the proof. □

Guided by Theorem IV.1, we can then quantify the variance improvement achieved by the optimal state/environment-dependent baseline $b^*(s,\mathcal{P})$ over the optimal state-dependent baseline $b^*(s)$ by letting $b(s) = b^*(s)$, as shown in the following corollary.

Corollary IV.2: Let $Var^{b^*(s)}(g)$ denote the variance of gradient estimate with the optimal state-dependent baseline $b^*(s)$. The variance reduction improvement of $b^*(s,\mathcal{P})$ over $b^*(s)$ is

$$Var^{b^*(s)}(g) - Var^{b^*(s,\mathcal{P})}(g) = \mathbb{E}_{P,\mu_\pi^p}\left[\frac{\left(\mathbb{E}_{P(p|s)}[Y_p(s)] - Y_p(s)\right)^2}{\mathbb{E}_\pi[G(a,s)]}\right], \tag{18}$$

where $Y_p(s) \triangleq \mathbb{E}_\pi[G(a,s)Q_\pi(s,a,p)]$.

Similarly, we can demonstrate a variance improvement of the optimal state/environment-dependent baseline $b^*(s,\mathcal{P})$ over the optimal constant baseline b_c^* quantitatively in the following corollary.

Corollary IV.3: Let $Var^{b_c^*}(g)$ denote the variance of gradient estimate with the optimal constant baseline b_c^*. The variance reduction improvement of $b^*(s,\mathcal{P})$ over b_c^* is

$$Var^{b_c^*}(g) - Var^{b^*(s,\mathcal{P})}(g) = \mathbb{E}_{P,\mu_\pi^p}\Bigg[\mathbb{E}_\pi[G(a,s)] \left(\frac{\mathbb{E}_{P,\mu_\pi^p}[Y_p(s)]}{\mathbb{E}_{P,\mu_\pi^p,\pi}[G(a,s)]} - \frac{Y_p(s)}{\mathbb{E}_\pi[G(a,s)]}\right)^2\Bigg]. \tag{19}$$

Proof: We provide an intuition of the proof here, please refer to Appendix A.3 for the detailed proof, available online. The idea is that the application of the optimal constant baseline b_c^* can be thought as letting $b(s) = b_c^*$ for each state $s \in \mathcal{S}$. Hence, by directly applying Theorem IV.1, where we substitute $b(s)$ with $b(s) = b_c^*$, we can complete the proof of this corollary with several steps of equivalence transformation. □

In conclusion, we analytically justify that by incorporating the additional information about environments, the optimal state/environment-dependent baseline $b^*(s,\mathcal{P})$ can obtain the minimum variance for DR within the baselines that consider both the state and environment parameters.

V. Variance Reduced Domain Randomization

Though we have theoretically demonstrated the superiority of variance improvement for the optimal state/environment-dependent baseline compared to the constant and state-dependent baselines under DR setting, it is unfortunately intractable to maintain a specific baseline for each environment as suggested in (13), especially when the number of environments is large and the value of the parameters are continuous. This is comparable to the single environment case in standard RL, where the theoretically optimal state-dependent baseline is rarely used in practice due to the computational concern [37]. Rather, for both implementational and conceptual benefit, a common alternative choice of the state-dependent baseline $b(s)$ is the state value function $V(s)$. Based on similar implementations, a direct idea to tackle the intractability issue of the the optimal state/environment-dependent baseline is to learn a baseline $V(s,p)$ with the combination of state and environment parameters as input. However, as will be shortly shown in the experiments, learning a baseline with this combined state and environment parameters, in either the simple way or through the meta learning-based method, suffers from a performance degradation during training.

Fortunately, inspired by the fact that similar system dynamics often emerge on environments having the similar parameters and thus result in similar expected returns, we can practically apply the clustering approach for variance reduction. Thus, in this section, we present a practical clustering-based variance reduction method for domain randomization (VRDR), aiming to strike a tradeoff between the variance reduction performance and the computational complexity required for maintaining multiple state/environment-dependent baselines. In VRDR, a criterion is further provided to guide the clustering and avoid suffering from possible increase of the total variance due to an arbitrarily split of the entire environment parameter space.

A. Practical State/Subspace-Dependent Baseline

Specifically, we propose to maintain an acceptable number M of state/environment-dependent baselines in practice, where $M < |\mathcal{P}|$. The entire environment parameter space $\mathcal{P}$ is then divided as M subspaces $\{\mathcal{P}_j\}_{j=1}^M$, each of which has a size $|\mathcal{P}_j|$. For each subspace $\mathcal{P}_j$, we can compute the optimal state-only baseline as $b^*(s,\mathcal{P}_j) = \mathbb{E}_{P_j(p|s)}\mathbb{E}_\pi[G(a|s)Q_\pi(s,a,p)]/\mathbb{E}_\pi[G(a,s)]$. We then denote the state/subspace-dependent baseline as $b^M(s,\mathcal{P}) = \{b^*(s,\mathcal{P}_j)\}_{j=1}^M$ and the corresponding variance as $Var^M(g)$.

Theorem V.1: The variance difference of gradient estimate between the optimal state-dependent baseline and the

state/subspace-dependent baseline is

$$Var^{b^*(s)}(g) - Var^{M}(g) = \sum_{j=1}^{M} \sum_{p_i \in \mathcal{P}_j} P(p_i) \sum_s \mu_\pi(s|p_i) \frac{\left[\left(\mathbb{E}_{P(p|s)}[Y_p(s)] - Y_{p_i}(s)\right)^2 - \left(\mathbb{E}_{P_j(p|s)}[Y_p(s)] - Y_{p_i}(s)\right)^2\right]}{\mathbb{E}_\pi\left[G(a,s)\right]}, \tag{20}$$

where $Y_{p_i}(s)$ is $Y_p(s)$ with $p = p_i$ and p_i denotes the ith environment parameter. For a policy π to be updated at a certain step, and with a distribution $P(p)$ for sampling environment parameters, we have $Var^{b^*(s)}(g) \geq Var^{M}(g)$ if for each subspace $\mathcal{P}_j$

$$\mathbb{E}_{P(s)} Var^{\mathcal{P}}(Y_p(s)) \geq \mathbb{E}_{P_j(s)} Var^{\mathcal{P}_j}(Y_p(s)), \tag{21}$$

$$\text{or} \sum_{j=1}^{M} \mathbb{E}_{P(s)} Var^{\mathcal{P}}(Y_p(s)) \geq \sum_{j=1}^{M} \mathbb{E}_{P_j(s)} Var^{\mathcal{P}_j}(Y_p(s)), \tag{22}$$

where the variance $Var^{\mathcal{P}}(Y_p(s))$ and $Var^{\mathcal{P}_j}(Y_p(s))$ are computed over $\mathcal{P}$ and $\mathcal{P}_j$ with distribution $P(p|s)$ and $P_j(p|s)$, respectively, as follows:

$$Var^{\mathcal{P}}(Y_p(s)) = \mathop{\mathbb{E}}_{p \sim P(p|s)} (\mathbb{E}_{P(p|s)}[Y_p(s)] - Y_p(s))^2, \tag{23}$$

$$Var^{\mathcal{P}_j}(Y_p(s)) = \mathop{\mathbb{E}}_{p \sim P_j(p|s)} (\mathbb{E}_{P_j(p|s)}[Y_p(s)] - Y_p(s))^2. \tag{24}$$

Proof: We provide a proof sketch here, please see Appendix A.4 for the detailed proof, available online. With the same definition of Xp, Y_p, and Z as in Section IV-C, variance of the policy gradient estimate g using the state/subspace-dependent baseline can be rewritten as

$$Var^{M}(g) = \sum_{j=1}^{M} \sum_{p_i \in \mathcal{P}_j} P(p_i) \sum_s \mu_\pi(s|p_i) \cdot \left[X_{p_i} - \frac{Y_{p_i}^2(s)}{Z} + \left(\frac{\mathbb{E}_{P_j(p|s)} Y_p(s)}{\sqrt{Z}} - \frac{Y_{p_i}(s)}{\sqrt{Z}}\right)^2\right], \tag{25}$$

where $P(p_i)$ is the sampling probability of a certain environment p_i from the entire environment space $\mathcal{P}$. The variance of gradient estimate by incorporating the optimal state-dependent baseline is

$$Var^{b^*(s)}(g) = \sum_{j=1}^{M} \sum_{p_i \in \mathcal{P}_j} P(p_i) \sum_s \mu_\pi(s|p_i) \cdot \left[X_{p_i} - \frac{Y_{p_i}^2(s)}{Z} + \left(\frac{\mathbb{E}_{P(p|s)} Y_p(s)}{\sqrt{Z}} - \frac{Y_{p_i}(s)}{\sqrt{Z}}\right)^2\right]. \tag{26}$$

Then (20) can be obtained by subtraction between (25) and (26). If the condition in (21) holds for each subspace $\mathcal{P}_j$, we have

$$\sum_{j=1}^{M} \left[\sum_{p_i \in \mathcal{P}_j} P(p_i)\right] \sum_s P(s) Var^{\mathcal{P}}(Y_p(s)) \geq \sum_{j=1}^{M} \left[\sum_{p_i \in \mathcal{P}_j} P(p_i)\right] \sum_s P_j(s) Var^{\mathcal{P}_j}(Y_p(s)), \tag{27}$$

which is obtained by multiplying both side of (21) with the same positive terms. Since $Var^{\mathcal{P}}(Y_p(s))$ is independent of the randomness of p, incorporating (11) and then taking the expectation w.r.t. $P(s)$, we have

$$\sum_s P(s) \left[\sum_{j=1}^{M} \left[\sum_{p_i \in \mathcal{P}_j} P(p_i)\right] Var^{\mathcal{P}}(Y_p(s))\right] = \sum_{j=1}^{M} \sum_{p_i \in \mathcal{P}_j} P(p_i) \sum_s P(s|p_i) (\mathop{\mathbb{E}}_{P(p|s)}[Y_p(s)] - Y_{p_i}(s))^2, \tag{28}$$

which is the LHS of the (27) and the first term in (20) scaled by $\mathbb{E}_\pi[G(a,s)]$. The sampling probability of environment p_i within the subspace $\mathcal{P}_j$ can be formulated as $P_j(p_i) = \frac{P(p_i)}{\sum_{p_i \in \mathcal{P}_j} P(p_i)}$, which can be substituted to the definition of $Var^{\mathcal{P}_j}(Y_p(s))$ in (24). Then, by computing the weighted sum of subspace $\mathcal{P}_j$ over s and taking the expectation w.r.t. P_j sequentially, we have

$$\sum_{j=1}^{M} \left[\sum_{p_i \in \mathcal{P}_j} P(p_i)\right] \sum_s P_j(s) Var^{\mathcal{P}^j}(Y_p(s)) = \sum_{j=1}^{M} \sum_{p_i \in \mathcal{P}_j} P(p_i) \sum_s P(s|p_i) (\mathop{\mathbb{E}}_{P_j(p|s)}[Y_p(s)] - Y_{p_i}(s))^2, \tag{29}$$

which is the RHS of (27) and the second term in (20) scaled by $\mathbb{E}_\pi[G(a,s)]$. Combining (20), (27), (28), and (29), we have $Var^{b^*(s)}(g) - Var^{M}(g) \geq 0$ and complete the proof. □

Theorem V.1 can also be applied to the case of continuous environment parameter space, by computing the expectation of parameter p with the integration over a continuous range instead of the sum over a discrete set. Note that by increasing the number of subspaces, the state/subspace-dependent baseline $b^M(s, \mathcal{P})$ will approach the optimal state/environment-dependent baseline $b^*(s, \mathcal{P})$. However, we argue that the more baselines we employ, the more samples are required for estimating these baselines. As will be shortly shown in the experiments in Section VI, increasing M may not monotonically improve the policy's performance during the training process. Theorem V.1 indicates that we can reduce the variance of gradient estimate by holding the expected variance of $Y_p(s)$ over state s in each subspace not exceeding that in the entire environment space. Whereas it is intractable in RL to estimate the distribution of visiting each state, we therefore propose to consider only the initial state s_0, under the assumption of a common distribution for the initial states in all environments and the distribution has a full support of the whole state space. Then, given the condition that we consider the distribution of initial state s_0, (21) is equivalent to the following inequality:

$$\mathbb{E}_{P(s_0)}\left[Var^{\prime\mathcal{P}}(Y_p(s_0))\right] \geq \mathbb{E}_{P(s_0)}\left[Var^{\prime\mathcal{P}_j}(Y_p(s_0))\right], \tag{30}$$

where $Var^{\mathcal{P}}$ and $Var^{\mathcal{P}_j}$ are variance computed w.r.t. $P(p)$ and $P_j(p)$, respectively. Please refer to Appendix A.5 for the proof, available online. Next, We show that by holding the variance of $Y_p(s_0)$ w.r.t. p and s_0 less than the variance of $\mathbb{E}_{P(s_0)}[Y_p(s)]$, a mild condition of (30) that is equivalent to (22) under s_0 can be obtained.

Proposition V.2: We have the following inequality holds,

$$\sum_{j=1}^{M} \mathbb{E}_{P(s_0)} Var^{\mathcal{P}}(Y_p(s_0)) \geq \sum_{j=1}^{M} \mathbb{E}_{P(s_0)} Var^{\mathcal{P}_j}(Y_p(s_0)),$$

given the following condition of variance in the entire environment space $\mathcal{P}$ and each subspace $\mathcal{P}_j$,

$$Var^{\mathcal{P}}(\mathbb{E}_{P(s_0)} Y_p(s_0)) \geq \tilde{Var}^{\mathcal{P}_j}(Y_p(s_0)), \tag{31}$$

where $\tilde{Var}^{\mathcal{P}_j}(Y_p(s_0))$ takes variance w.r.t. $P(s_0)$ and $P_j(p)$

$$\tilde{Var}^{\mathcal{P}_j}(Y_p(s_0))) = \mathbb{E}_{P(s_0),P_j(p)}[(Y_p(s_0) - \mathbb{E}_{P(s_0),P_j(p)} Y_p(s_0))^2].$$

Proof: We provide proof sketch here, please refer to Appendix A.6 for the details, available online. Since a variance function is convex w.r.t. random variables, referring to Jensen's inequality, we have

$$\sum_{j=1}^{M} \mathbb{E}_{P(s_0)} Var^{\mathcal{P}}(Y_p(s_0)) \geq \sum_{j=1}^{M} Var^{\mathcal{P}}(\mathbb{E}_{P(s_0)} Y_p(s_0)). \tag{32}$$

For a baseline $b^{EM} = \{\mathbb{E}_{P_j(s)} b^*(s, \mathcal{P}_j)\}_{j=1}^{M}$, which can be considered as the constant baseline in each subspace, we have $Var^{EM}(g) > Var^{M}(g)$, where $Var^{EM}(g)$ denotes the variance of policy gradient using baseline b^{EM}. Then, we can obtain

$$\sum_{j=1}^{M} \tilde{Var}^{\mathcal{P}_j}(Y_p(s_0)) \geq \sum_{j=1}^{M} \mathbb{E}_{P(s_0)} Var^{\mathcal{P}_j}(Y_p(s_0)). \tag{33}$$

Given the condition as specified in (31) in Proposition V.2, and incorporating with (32) and (33), we can compete the proof. □

Proposition V.2 provides a practical guideline for constructing subspaces to reduce the variance of gradient estimate. However, it still introduces extra computation to estimating $G(a, s)$ in $Y_p(s)$. Under the same assumption about the loose correlation between log policy $G(s, a)$ and Q-value function as in [37], [41], we have the following proposition.

Proposition V.3: If the inner product of log policy $G(s, a)$ is loosely correlated to the Q-value function,[1] i.e., $\mathbb{E}_a[G(a, s) Q_\pi(s, a, p)] = \mathbb{E}_a[G(a, s)]\mathbb{E}_a[Q_\pi(s, a, p)]$,

$$Var^{\mathcal{P}}(\mathbb{E}_{P(s_0)} V_\pi(s_0, p)) \geq \tilde{Var}^{\mathcal{P}_j}(V_\pi(s_0, p)), \tag{34}$$

is a sufficient condition of the inequality in (30).

Proof: We provide a proof sketch here, please see Appendix A.7 for the detailed proof, available online. From (34), we have

$$\mathbb{E}_{P(s_0)}\left[Var^{\mathcal{P}}(V_\pi(s_0, p))\right] \geq \mathbb{E}_{P(s_0)}\left[Var^{\mathcal{P}_j}(V_\pi(s_0, p))\right]. \tag{35}$$

By multiplying both sides in (35) with a non-negative term $(\mathbb{E}_a[G(a, s_0)])^2$, we have

$$\mathbb{E}_{P(s_0)}\left[\mathbb{E}_a[G(a, s_0)]^2 Var^{\mathcal{P}}(V_\pi(s_0, p))\right] \geq \mathbb{E}_{P_j(s_0)}\left[\mathbb{E}_a[G(a, s_0)]^2 Var^{\mathcal{P}_j}(V_\pi(s_0, p))\right]. \tag{36}$$

By expanding the both sides with the definition of variance, we have

$$\begin{aligned} &\mathop{\mathbb{E}}_{P(s_0)}\left[\mathbb{E}_a[G(a, s_0)]^2 Var^{\mathcal{P}}(V_\pi(s_0, p))\right] \\ &= \mathop{\mathbb{E}}_{P(s_0)}\left[Var^{\mathcal{P}}(Y_p(s_0))\right], \\ &\quad \mathop{\mathbb{E}}_{P_j(s_0)}\left[\mathbb{E}_a[G(a, s_0)]^2 Var^{\mathcal{P}_j}(V_\pi(s_0, p))\right] \\ &= \mathop{\mathbb{E}}_{P_j(s_0)}\left[Var^{\mathcal{P}_j}(Y_p(s_0))\right], \end{aligned}$$

which can be plugged into (36) to complete the proof. □

The LHS of (34) can be considered as variance of policy's performance in the entire environment space. While the RHS of (34) can be expanded as

$$\tilde{Var}^{\mathcal{P}_j}(V_\pi(s_0, p)) = \mathbb{E}_{P_j(p)}\left[Var^{s_0}(V_\pi(s_0, p))\right] + Var^{\mathcal{P}_j}(\mathbb{E}_{P(s_0)} V_\pi(s_0, p)), \tag{37}$$

where the first term of the RHS is the averaged variance of performance in subspace $\mathcal{P}_j$, and the second term is the variance of policy's expected return in subspace $\mathcal{P}_j$. Please refer to (A30) for the expansion in Appendix A.7, available online. Since environments with similar dynamics will result in a similar return, clustering environments with similar dynamics will reduce the second term of the RHS of (37). In particular, the second term will degrade to zero when subspaces have size of one, where the sum of the first term in the overall space remains fixed for different division of subspaces. Hence, Proposition V.3 suggests to cluster environments into subspaces with similar expected return $\mathbb{E}_{P(s_0)}[V_\pi(s_0, p)]$ from initial state, since constructing state-dependent baselines for them can reduce variance of the policy gradient estimate.

B. VRDR Algorithm

Based on our theory, we propose a practical variance reduced domain randomization approach (VRDR) for policy gradient and summarize it in Algorithm 1, where the state/subspace-dependent baseline $b(s, \mathcal{P}_i) = \mathbb{E}_{p \sim P_i} V(s, p) \triangleq V(s, \mathcal{P}_i)$ is employed for each state and subspace pair. At each iteration k, as in Line 3, we sample H environments and group them

[1] Here, we intuitively explain the reasonability of the assumption of the loosely correlation. We take the Gaussian policy $\mathcal{N}(m_a, \sigma^2)$ that is commonly used in policy gradient methods for example. The policy takes s as input and outputs for action selections a Gaussian distribution with the expectation m_a and variance σ^2. The action $a = m_a$ has the minimum gradient of zero, i.e., $G(m_a, s) = 0$, since $\nabla_a \pi = -\frac{(a-m_a)}{\sqrt{2\pi}\sigma^3} \exp(-\frac{(a-m_a)^2}{2\delta^2})$ and $\nabla_a \pi(m_a) = 0$. However, before this policy is trained well to obtain a high Q-value, the action m_a is not guaranteed to achieve a high or low Q-value, hence the values of G and Q are not correlated.

Algorithm 1: Variance Reduced Domain Randomization (VRDR) for Policy Gradient.

1: **Initialize** policy π_0, number of environments sampled per iteration H, number of subspaces M, clustering interval N_c, maximum number of iterations K, subspace prototypes $\boldsymbol{u} = \{u_i\}_{i=1}^{M}$, baseline $V(s, \mathcal{P}_i)$ for subspace $\mathcal{P}_i$, sample buffer $\mathcal{B}$ of size $|\mathcal{B}|$ for clustering.
2: **for** $k = 0$ to $K - 1$ **do**
3: Sample a set of environment parameters $\{p_e\}_{e=0}^{H-1}$ uniformly and determine their cluster labels by $\min_i |p_e - u_i|$ to obtain the cluster set $\{\mathcal{C}_i\}_{i=1}^{M}$.
4: Sample L trajectories $\{\tau_{e,j}\}_{j=0}^{L-1}$ for each environment p_e using policy π_k.
5: Compute average return of environment p_e as $\hat{V}(s_0, p_e) = \sum_{j=0}^{L-1} R(\tau_{e,j})/L$ and store the tuple $(p_e, \hat{V}(s_0, p_e))$ in $\mathcal{B}$.
6: Use PPO for policy optimization to get the updated policy π_{k+1} on trajectories, with the baseline $V(s, \mathcal{P}_i)$ of the cluster that has been grouped to.
7: Update $V(s, \mathcal{P}_i)$ on the trajectories in cluster $\mathcal{C}_i$.
8: **if** $k\%N_c == 0$ and $k > 0$ **then**
9: Apply the clustering method on $\mathcal{B}$ to get the cluster set $\{\mathcal{C}_i\}_{i=1}^{M}$, and determine corresponding prototypes $\boldsymbol{u} = \{u_i\}_{i=1}^{M}$ by computing $u_i = \sum_{j \in \mathcal{C}_i} p_j / |\mathcal{C}_i|$.
10: **end if**
11: **end for**

Algorithm 2: Hierarchical Clustering.

1: **Input** $\Omega = \{\hat{V}_\pi(p_c)\}_{c=1}^{H}$,
2: **Initialize** number of clusters M, current number of clusters $m = H$, prototypes $\{u_i = \hat{V}_\pi(p_i)\}_{i=1}^{H}$, distance matrix M_d
3: **while** $m \geq M$ **do**
4: **for** $i = 0$ to $m - 1$ **do**
5: **for** $j = 0$ to $m - 1$ **do**
6: $M_d(i, j) = |u_i - u_j|$
7: $M_d(i, j) = M_d(j, i)$
8: **end for**
9: **end for**
10: merge the two clusters u_{i^*} and u_{j^*} having the minimum distance: $u_{i^*} = u_{i^*} \cup u_{j^*}$
11: $m = m - 1$
12: **end while**
13: **Output** prototypes $\{u_i\}_{i=1}^{M}$

into M clusters by finding the nearest subspace prototype. Then, in Lines 4-7, we train the policy on the trajectories sampled on each environment p_e with the corresponding baseline $V(s, \mathcal{P}_i)$ and update this baseline on the corresponding trajectories. Specifically, in Line 7, we use the loss function $\mathcal{L}_V = \sum_{s_t \in \boldsymbol{\tau}_i} |V(s_t, \mathcal{P}_i) - \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}|^2$ to update the baseline $V(s, \mathcal{P}_i)$ for each cluster $\mathcal{C}_i$, where $\boldsymbol{\tau}_i$ is a set of trajectories from the environments in cluster $\mathcal{C}_i$.

During the clustering phase in Line 9, as guided by Proposition V.3, we apply the clustering method to samples in buffer $\mathcal{B}$ w.r.t. the average return $\hat{V}(s_0, p_e)$ from the initial state s_0, to get the M clusters with corresponding subspace prototypes $\{u_i\}_{i=1}^{M}$. Note that the clustering methods are usually applied to group a fixed dataset, with the labeling done once and for all. In VRDR, however, we perform clustering multiple times during training, hence the prototypes between two clustering phases would be similar but have different labels. In practice, to avoid the case where the new prototype close to a certain pre-clustering prototype is assigned with a different label, we relabel these new prototypes to get the minimum total distance between the pre- and post-clustering prototypes.

By holding (34), we have $\frac{Var^{b^*(s)}(g)}{Var^{M}(g)} > 1$. Then, based on Theorem III.1, it can be verified that VRDR gains a $\kappa = \sqrt{\frac{Var^{b^*(s)}(g)}{Var^{M}(g)}}$ $(\kappa > 1)$ acceleration on the convergence of policy gradient methods, as compared to the direct application of the optimal state-dependent baseline for DR.

VI. Experiments

In this section, we evaluate our proposed VRDR method on six simulated robot control tasks, where the fundamental environment parameters that affect dynamics are randomized for the generalization of trained policy.

A. Experiment Setup

The experimental environments are implemented based on the robot control simulator as described in [46]. We compare VRDR with three baseline algorithms: the uniform domain randomization (DR) [19], the multiple value network (MVN) [41], domain randomization with meta baseline (DRMB) [41], [42], and domain randomization with combined state-dependent baseline (DRCS) [7]. We utilize the proximal policy optimization (PPO) [47] to train a generalized policy for all the comparison algorithms.

- In DR, we uniformly sample the environment parameter p and collect trajectories by using the current policy on the sampled environments for policy update, where a value function that is learned from trajectories from all sampled environments is employed as the state-dependent baseline.
- In MVN, multiple value functions are learned for the pre-sampled environments at the beginning of training, while each value function V_i is utilized as a baseline for the gradient estimate of trajectories collected on environment parameters that are closest to the corresponding pre-sampled p_i.
- In DRMB, the meta learning method MAML proposed by [43] is applied to learn a state and environment dependent baseline.
- In DRCS, state s sampled from task and environment parameter p are combined to an augmented state as input to value function to construct the state-dependent baseline.

TABLE I
RANGE OF ENVIRONMENT PARAMETERS FOR EACH TASK

Environment	Environment Parameters	Training Range
Walker2D	Material density	$[750, 1250]$
	Sliding friction	$[0.3, 1.1]$
Hopper	Material density	$[750, 1250]$
	Sliding friction	$[0.5, 1.1]$
Halfcheetah	Total mass	$[5, 23]$
	Sliding friction	$[0.1, 1.5]$
Swimmer	Material density	$[3750, 4250]$
	Medium viscosity	$[0.0, 1.0]$
InvertedPendulum	Cart size	$[0.05, 0.25]$
	Pole length	$[0.50, 2.00]$
CartPole	Force magnitude	$[1.00, 20.00]$
	Pole length	$[0.05, 1.00]$
	Pole mass	$[0.01, 1.00]$

- For VRDR, we apply the hierarchical clustering method [48] as shown in Algorithm 2 to divide the entire environment space during training, and group the trajectories collected at each interaction as described in Algorithm 1.

We keep all the hyper-parameters about policy optimization for PPO the same as the implementation in [49], and purely discuss the gain brought by the variance reduction for gradient estimate in policy gradient methods. Our experiments are designed to answer the following questions.

- Can VRDR effectively improve the convergence rate of the policy training. Will the final reward achieved by the converged policy trained by VRDR outperform the other three baseline algorithms?
- How would all the comparison algorithms reduce the variance of policy gradient estimate? How would the variation of such variance relate to the algorithm's performance?
- How would the specific hyper-parameters in VRDR affect the performance of training process.

B. Training Curves With Randomized Dynamics

The six robot control tasks are as follows. 1) *Walker2D*: control a planar biped robot to run as fast as possible; 2) *Hopper*: control a planar monopod robot to hop as fast as possible; 3) *Halfcheetah*: control a planar cheetah robot to run fast; 4) *Swimmer*: control a three-link robot to swim forward as fast as possible in a viscous fluid; 5) *InvertedPendulum*: control a cart to balance a pendulum on the cart; 6) *CartPole*: a similar task as InvertedPendulum using the classic problem formulation. We modify the environment parameters in the configuration file to generate six environments with the same control task and different dynamics. The possible sampling range of the environment parameters is shown in Table I. Specifically, we randomize the material density that determines the mass and inertia, and the sliding friction acting along the tangent plane in Walker2D, Hopper, and Halfcheetah. We randomize the material density and viscosity of the medium in Swimmer. In InvertedPendulum, we randomize the cart size and pole length. In CartPole, we randomize the pole length, pole mass, and the magnitude of force that pushs the cart.

We run the training process on environments sampled on the preset parameter range for the same number of iterations K until all the algorithms are converged. At each iteration k, we generate $H = 100$ environments by uniformly sampling the parameter range in all the comparison algorithms. We run 15 seeds for each algorithm on all the environments to obtain the training results, with each seed generated at random by Python. The policy and value networks in each algorithm are trained for the same number of epochs and sampled trajectories at each iteration. We show the training curves of the six tasks, respectively, in Fig. 2(a)–(f). The solid curve is the average return on all the seeds at a certain iteration, while the shaded area denotes the standard deviation. It can be seen that by applying the proposed state/subspace-dependent baseline, our VRDR can improve the average return over DR in all the six tasks. Though DRMB achieves the fastest convergence occasionally in Swimmer and best score in CartPole, it fails to improve final return over DR in InvertedPendulum, Hopper, Halfcheetah, and Swimmer. DRMB also shows a significant instability during training process in InvertedPendulum and CartPole. For the other two comparison algorithms, MVN and DRCS almost show the worst performance on all the tasks and even fail in some tasks (e.g., MVN in Walker2D/InvertedPendulum/Hopper and DRCS in Swimmer). In comparison, VRDR can consistently accelerate the training process on all the six tasks, especially at the middle and later stages of iterations.

To validate the performance gain at the later stage of iterations, we further show in Table II the tabular comparison of final rewards achieved by different algorithms, where the performance is averaged over the last 200 continuous iterations of the corresponding training curves in Fig. 2. It can be seen that after convergence, VRDR can generally reach a better final reward than the other comparison algorithms, e.g., the highest average return in Walker2D, InvertedPendulum, Hopper, Halfcheetah, and Swimmer, and the second-highest average return in CartPole, where the score is close to the highest. In addition, VRDR achieves the lowest standard deviation of average return in most of the tasks, indicating that the converged policy trained by VRDR is also the most stable one. Specifically, for the tasks in which VRDR presents a close final score to other methods in training curves as shown in Fig. 2(b), (c), and (f), VRDR all demonstrates its stability.

Difference Between MVN and VRDR: i) MVN utilizes only M initially sampled environments for the policy training, and trains a value network for each sampled environment (i.e., M value networks in total) only based on samples from these M environments [41]. In comparison, VRDR samples different environments during the whole training process, and uses the samples from these consistently sampled environments for the value network update. Thus, the performance of MVN approach depends purely on the initially sampled M environments, while VRDR can leverage environments sampled from the whole parameter space and thus benefit from the diversity of samples as DR. Besides, MVN cannot scale very well if the range of environment parameters becomes larger. ii) The set of M environments in MVN are fixed during the whole training process, while the set of M environment subspaces (as well as the M state/subspace-dependent baseline $b^M(s, \mathcal{P}) = \{V_\pi(s, \mathcal{P}_j)\}_{j=1}^M$) in VRDR is consistently updated according to the clustering results. Hence,

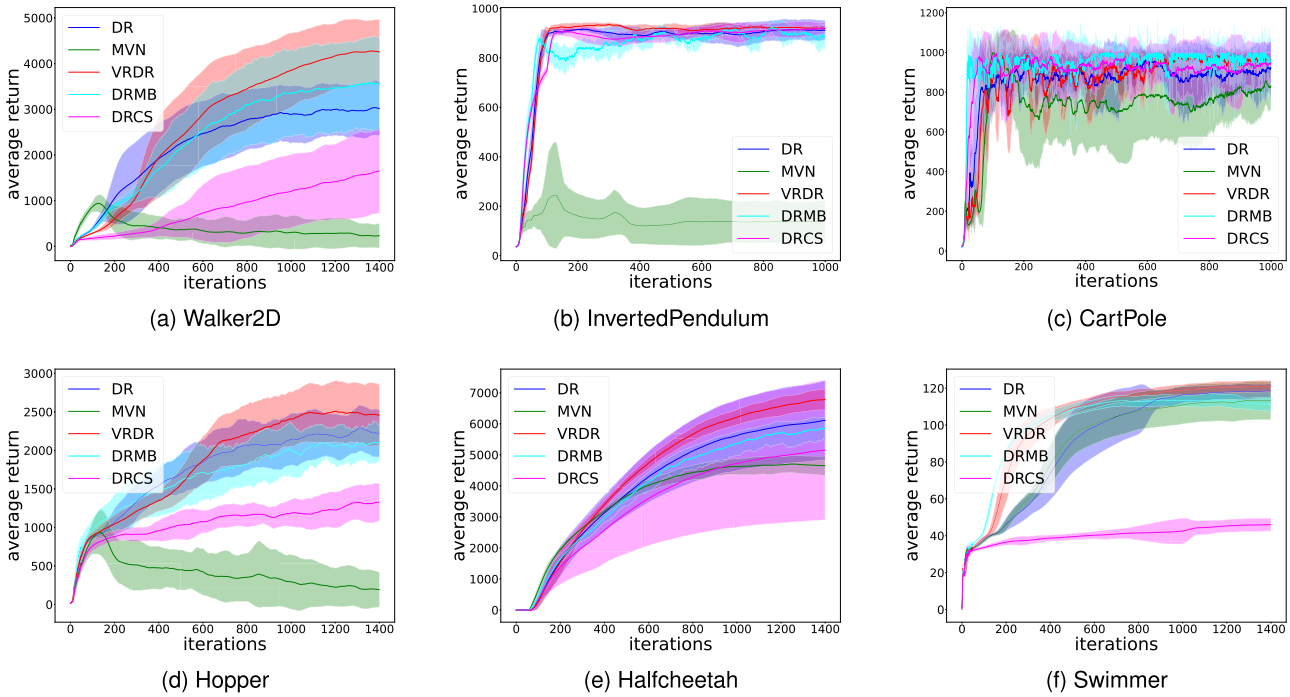


Fig. 2. Training curves of average return achieved by different algorithms for different tasks.

TABLE II
AVERAGED FINAL REWARDS IN TRAINING WITH STANDARD DEVIATIONS

ALGORITHM	WALKER2D	INVERTEDPENDULUM	CARTPOLE	HOPPER	HALFCHEETAH	SWIMMER
DR	2986.7 ± 584.5	911.7 ± 41.3	895.4 ± 128.7	2226.6 ± 292.1	6011.7 ± 1239.8	118.2 ± 4.1
MVN	257.8 ± 283.8	137.2 ± 81.5	799.7 ± 141.4	217.5 ± 251.1	4678.9 ± 284.6	113.0 ± 10.3
VRDR	$\mathbf{4239.4 \pm 709.1}$	$\mathbf{921.4 \pm 6.8}$	964.5 ± 34.8	$\mathbf{2481.1 \pm 394.7}$	$\mathbf{6688.4 \pm 308.3}$	$\mathbf{120.8 \pm 3.0}$
DRMB	3524.8 ± 986.4	898.7 ± 33.1	$\mathbf{969.0 \pm 80.5}$	2077.0 ± 240.1	5746.0 ± 381.8	111.9 ± 3.4
DRCS	1535.4 ± 899.6	914.5 ± 28.5	925.1 ± 129.2	1297.7 ± 232.6	5061.2 ± 2197.4	45.8 ± 3.4

The boldface indicates the best performing algorithm.

the distribution of samples from the environments for training each value network (state/subspace-dependent baseline) is adaptively adjusted by clustering to reduce the variance in the entire environment space, leading to a better performance of VRDR.

Deployment at the Testing Phase: Note that like in other policy gradient methods, the baseline function is not required to be deployed on the trained policy for policy gradient under the domain randomization settings. Specifically, the baseline function is only constructed during the training of policy gradient methods to reduce the variance of policy gradient, which is used by gradient descent to train and update the policy network. Then, at the testing (or deployment) phase, the trained policy will be directly deployed to perform a task without further computation of the policy gradient (and thus the baseline function). Hence, in the real-world test, the exact environment parameter is no longer needed by our VRDR to achieve a performance gain, as long as the real environment parameters locate within the range of generated environments by the simulator during training.

Limitations of Experiments: Since DRL experiments are commonly affected by i) specific hyperparameters of DRL algorithms; ii) choice of random seeds; iii) specific environment characteristics; and iv) codebases of algorithm implementation as stated in [50], the empirical study in this section also suffers from these effects. Our effort in alleviating these limitations are as follows. We adopt the same code implementation of PPO with the identical hyperparameter settings as in [49], for all the DRL-based comparison algorithms, to bypass the pitfalls of i) hyperparameter tuning and iv) code implementation of DRL. And we also evaluate these comparison algorithms on six different robot control tasks for 15 runs with different random seeds, to further reduce the impact of stochasticity stemmed from ii) the choice of random seeds and iii) specific environment characteristics.

C. Variance of Policy Gradient Estimate

In this section, we first evaluate the variance of policy gradient estimate by using the various baseline functions as discussed in this paper in a synthetic linear-quadratic-Gaussian (LQG) experiment, which generally yields a more stable results and thus allows a more robust measurement of the variance of policy gradient estimate. Then, we evaluate the variation of the variance

of policy gradient estimate during the training process of the six robot control tasks in Section VI-B. We demonstrate the superiority of variance reduction by using the state/subspace-dependent baseline function $b^M(s,\mathcal{P})$ in the LQG experiment with less stochasticity, and the better variance reduction performance of VRDR that is developed upon $b^M(s,\mathcal{P})$ in the robot control tasks.

1) Synthetic 2D Mass Point Experiments. LQG Under DR: We modify a synthetic LQG system evaluated in [51] to the DR settings, which can provide a near analytic and more robust evaluation of the variance of policy gradient estimate. We briefly describe the LQG system here, please refer to Appendix A.9 for the details, available online. The LQG system formulates a 2D point mass task, as follows:

$$r_t = -s_t^T Q_t s_t - a_t^T R_t a_t,$$

$$\mathcal{T}(s_{t+1}|s_t,a_t) = \mathcal{N}(\mu_{t+1}^{S|SA}, \Sigma_{t+1}^{S|SA}), \mu_{t+1}^{S|SA} = A_t s_t + B_t|_m a_t,$$

$$\pi(a_t) = \mathcal{N}(\mu_t^A, \Sigma_t^A), P(s_0) = \mathcal{N}(\mu_0^S, \Sigma_0^S),$$

where the reward function r_t defines the goal to move a point mass to reach and stay at coordinate (0,0). The dynamics are defined by a transition distribution $\mathcal{T}$ of Gaussian distribution, where the mean is modeled as a linear equation of current state-action pair (s_t, a_t) and the coefficient matrix B_t is determined by the time interval d_t and the mass of point m. The initial state distribution and control policy $\pi(a_t)$ are also Gaussian, hence the true value functions Q_π, V_π, policy gradient and its variance can all be computed analytically. We modify the original LQG to a DR setting and consider m as a random variable that follows a uniform distribution P over a discrete environment parameter space $\mathcal{P} = \{m_i = 0.2 + 0.08 \times i | i = 0, \ldots, |\mathcal{P}| - 1, |\mathcal{P}| = 1000\}$. The policy optimization is to maximize $\eta(\pi) = \mathbb{E}_{m,s_{0:T},a_{0:T}}[\sum_{t=0}^{T} \gamma^t r_t]$.

Variance Evaluation: In this LQG experiment, we follow the same approach of evaluating variance in [51] by estimating the log trace value of matrix terms, which are from decomposition of covariance matrix of policy gradient w.r.t. mean μ_t^A. The trace value of covariance matrix of Monte-Carlo policy gradient estimator $\hat{g}_t^\mu$ at time step t under domain randomization by sampling trajectory τ can be decomposed as

$$Var_{p,s,a,\tau}(\hat{g}_t^\mu) = \mathrm{Tr}(\Sigma_{\tau,t}^\mu) + \mathrm{Tr}(\Sigma_{b,t}^\mu) + \mathrm{Tr}(\Sigma_{s,t}^\mu) + \mathrm{Tr}(\Sigma_{g,t}^\mu), \tag{38}$$

where only the second term $\mathrm{Tr}(\Sigma_{b,t}^\mu)$ contains baseline b and the other terms remain the same when using different baseline functions. Please refer to Appendix A.8 for the detailed procedure of decomposition, available online. We evaluate the following six baseline functions:

- zero baseline $b_0 = 0$ (i.e., DR without baseline),
- MVN baseline $b_{MVN} = \{V_\pi(s, p_j)\}_{j=1}^M$,
- state-dependent baseline $b(s) = V_\pi(s)$,
- state/subspace-dependent baseline $b^M(s,\mathcal{P}) = \{V_\pi(s, \mathcal{P}_j)\}_{j=1}^M$,
- state/environment-dependent baseline $b(s,\mathcal{P}) = \{V_\pi(s, p_j)\}_{j=1}^{|\mathcal{P}|}$,
- constant baseline $b_c = \mathbb{E}_{p,s}[V_\pi(s,p)]$.

In LQG, the analytic state value function $V_\pi(s,p)$ for each environment is accessible, hence all the six baseline functions can be analytically computed, please refer to Appendix A.10 for the details, available online. For MVN and state/subspace-dependent baselines, the numbers of value function M are both set to 10.

Results Analysis: By using a specific baseline function b, we compute the trace of variance term $\Sigma_{b,t}^\mu$ of state s_t and a_t at time step t for a trajectory of length 100 as

$$\mathrm{Tr}(\Sigma_{b,t}^\mu) = \frac{1}{|\mathcal{P}|} \sum_i \mathbb{E}_{s_t,a_t} \Big[(Q_\pi(s_t,a_t,m_i) - b)^2 \, G_\mu(a_t,s_t) - g^\mu(s_t,m_i)^{\mathrm{T}} \left(g^\mu(s_t,m_i)\right) \Big], \tag{39}$$

and plot its curves of log value after 1, 50, 150, and 400 policy's updates using policy gradient methods in Fig. 4, where the horizontal axis is time horizon along the trajectory. For each time step, we sample 2000×100 state and action pairs to estimate $\log \mathrm{Tr}(\Sigma_{b,t}^\mu)$, which are from all the $|\mathcal{P}| = 1000$ environments with 2,000 samples for each. The solid curve is the average estimate of variance term over 50 runs with different random seeds, while the shaded area denotes standard deviation. It can be seen that the state/environment-dependent baseline $b(s,p)$ has the minimal variance of gradient estimate during the update process at each time step, since the true value function of each environment is correctly computed. In comparison, our state/subspace-dependent baseline $b^M(s,\mathcal{P})$ achieves the second least variance only higher than $b(s,p)$. As shown in Fig. 4(a), $b^M(s,\mathcal{P})$ and b_{MVN} present similar variance at the beginning of training updates, since the reward obtained from each environment is similar. But after several updates, when the clustering becomes more effective (since the reward obtained from each environment becomes distinct with the policy update), the MVN baseline b_{MVN} suffers a higher variance than ours. Besides, b_{MVN} presents the largest shaded area on these 50 runs, which indicates that a random selection of multiple environments for training multiple value functions will enlarge the variance. In addition, our $b^M(s,\mathcal{P})$ achieves a significant variance reduction as compared to the state-dependent baseline $b(s)$. Both the zero baseline b_0 and constant baseline b_c will result in a huge variance, while b_c is better than b_0.

2) Robot Control Tasks: For all the five comparison algorithms, we then evaluate the variation of the variance of policy gradient estimate during the training process. Here, the variance of gradient estimate is empirically approximated by taking 200 samples of time steps from all the time steps within every ten training iterations, and then using these samples, together with the policy network, to compute the gradient estimate $g(\theta,s,a,p)$ and the corresponding variance. Specifically in the tasks Walker2D, CartPole, InvertedPedulum, Hopper, Halfcheetah, and Swimmer, we run 5 random seeds for each algorithm, and show the training curves w.r.t. the variance of gradient estimate achieved by different algorithms in Figs. 3(a)–(f), respectively. The solid curve is the average variance over all the random seeds, while the shaded area denotes the standard deviations. Note that variance of the policy gradient for

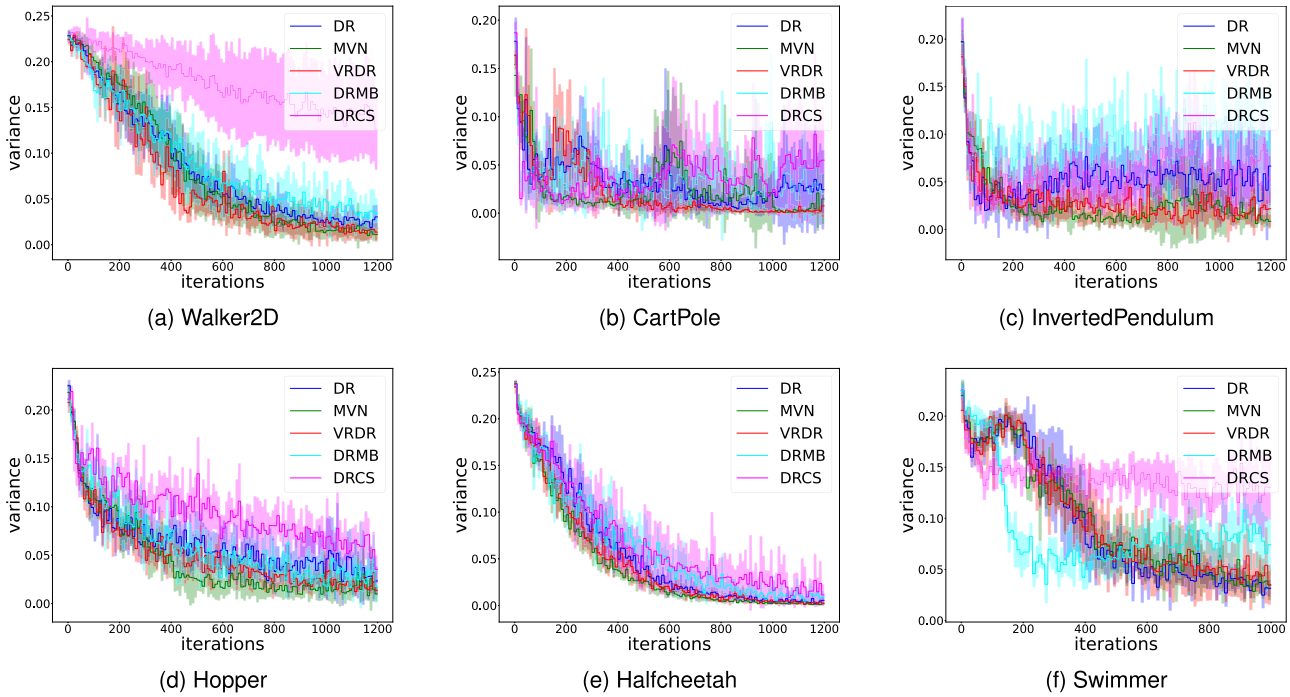


Fig. 3. Training curves of variance of gradient estimate achieved by different algorithms for different tasks.

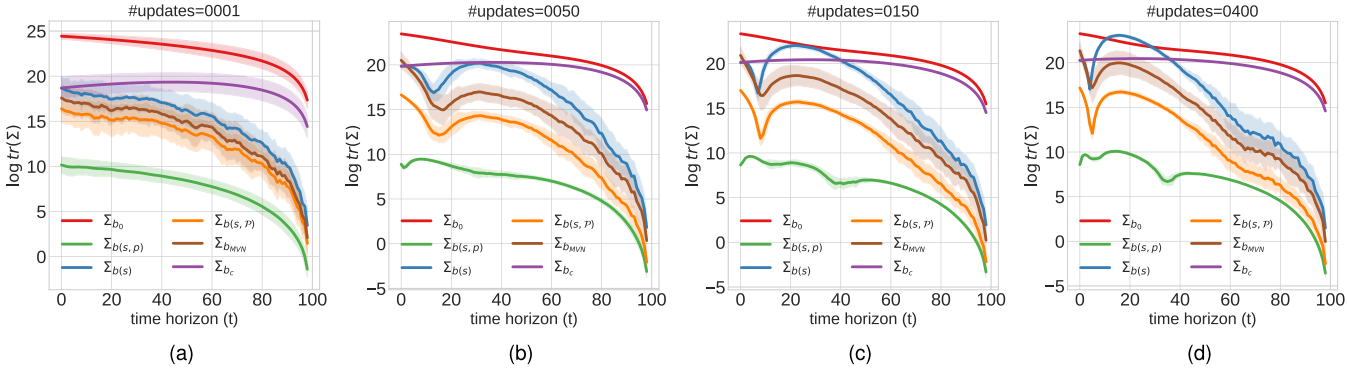


Fig. 4. Evaluation curves of variance term Σ_b (i.e., Eq. (38)) at each time step using different baseline functions on point mass task modelled by linear quadratic Gaussian. Horizontal axis is time horizon of the trajectory of total time steps $T = 100$. (a) after 1 update, (b) after 50 updates, (c) after 150 updates, (d) after 400 updates.

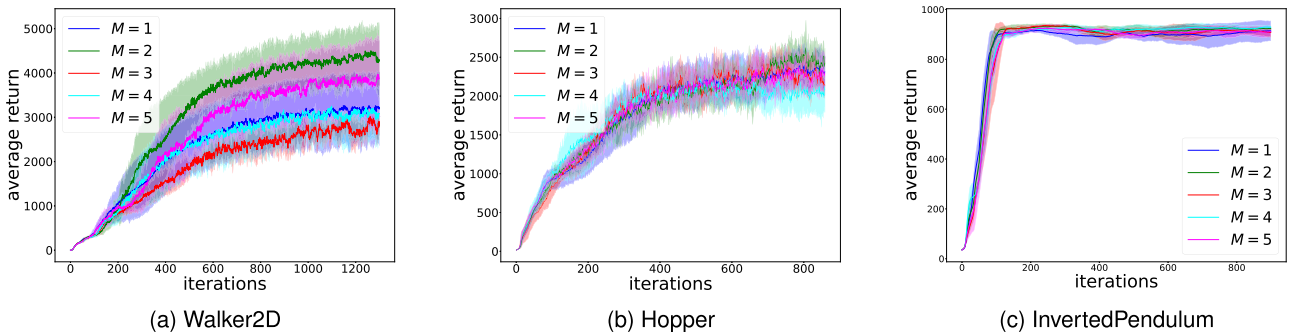
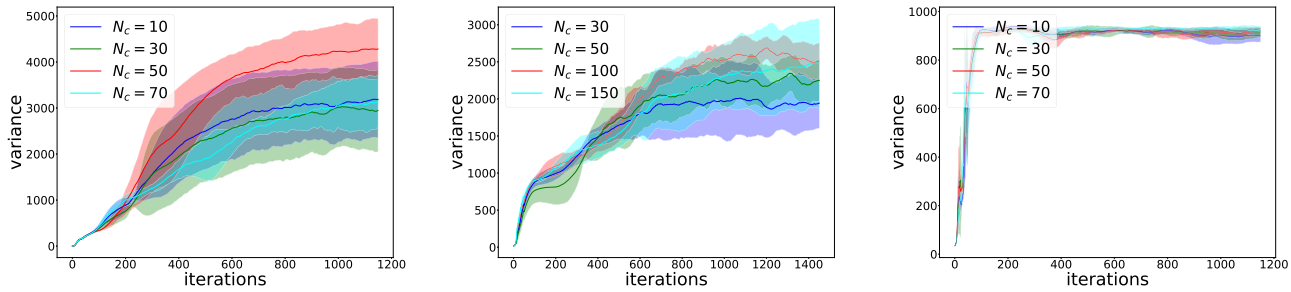


Fig. 5. VRDR's training curves of average return using different number of baselines M.



(a) Walker2D (b) Hopper (c) InvertedPendulum

Fig. 6. VRDR's training curves of average return using different clustering interval N_c.

TABLE III
UNSEEN RANGE OF ENVIRONMENT PARAMETERS FOR EACH TASK

Environment	Environment Parameters	Testing Range
Walker2D	Material density	$[400, 500]$
	Sliding friction	$[0.1, 2.0]$
Hopper	Material density	$[1250, 1600]$
	Sliding friction	$[0.1, 2.5]$
Halfcheetah	Total mass	$[1.0, 5.0]$
	Sliding friction	$[0.5, 1.5]$
Swimmer	Material density	$[3250, 3750]$
	Medium viscosity	$[0.0, 1.0]$
InvertedPendulum	Cart Size	$[0.05, 0.30]$
	Pole length	$[2.0, 3.0]$
CartPole	Force magnitude	$[20.0, 30.0]$
	Pole length	$[1.0, 2.0]$
	Pole mass	0.5

MVN is computed on samples from environments of the same number as its baselines, which is much less than the number of environments in other algorithms. Hence, the comparison with MVN is unfair and it is expected to have a lower variance. It can be seen that VRDR can achieve a lower variance of gradient estimate than the other comparison algorithms in Walker2D, CartPole, InvertedPendulum, Hopper, and Halfcheetah, especially at the middle and later stages of training iterations. In Swimmer, DRMB shows the lowest variance at the early stage, while VRDR and DR show lower and similar variance later. Such a variance reduction performance also coincides with the faster convergence behavior of VRDR and better converged performance as shown in Fig. 2(a)–(e). Besides, the lowest variance achieved by DRMB in Swimmer also coincides with the fastest convergence as shown in Fig. 2(f), while VRDR can achieve the best final performance in Swimmer. In addition, the standard deviation of the curves of VRDR is also smaller than the others, which might also interpret the improved training stability of VRDR.

D. Generalization to Unseen Environments

In this section, we evaluate the generalization capability of trained policies by DR, MVN, VRDR, DRMB, and DRCS on environments generated from unseen parameter range of Walker2D, Hopper, Halfcheetah, Swimmer, InvertedPendulum, and CartPole. The testing unseen range of environment parameters are listed in Table III. For all the six tasks, we generate the testing environments by uniformly selecting 20 points in each parameter range, which results in 400 environments in total for each task. The generalization performance of each algorithm is measured by the average rewards over 10 runs. The heatmaps of average return in these unseen environments achieved by DR, MVN, VRDR, DRMB, and DRCS are shown in Fig. 7(a)–(f), where a darker color indicates a higher performance. It can be seen that VRDR shows the best generalization performance in Walker2D, Hopper, Halfcheetah, Swimmer and InvertedPendulum, and the second best in CartPole with only a slight degradation than DRMB. In addition, DRMB achieves a broader range of generalization than MVN, DR, and DRCS on Walker2D, Hopper, Halfcheetah, and CartPole. And DR presents a better generalization performance than MVN. While DRCS almost fails to generalize in Walker2D and Simmer, though performing occasionally better than DR and MVN in Halfcheetah, InvertedPendulum, and CartPole.

E. Ablation Study of VRDR

1) Number of Baselines M: In Theorem IV.1, we show that by applying $b^*(s, \mathcal{P})$, we maintain a state/environment baseline $b^*(s, p_i)$ for each environment p_i and can obtain the minimum variance of gradient estimate. Theoretically, increasing M to $|\mathcal{P}|$ will achieve a better variance reduction improvement. Fig. 5(a)–(c) show the training curves of VRDR on Walker2D, Hopper, and InvertedPendulum with different numbers of baselines. Note that VRDR is reduced to DR when setting $M = 1$. Through ablation study on the choice of M, we find that the increase of M will decrease the number of training samples for approximation of each baseline if we sample the same number of environment at each iteration, which hence retards the convergence rate of VRDR. Therefore, the increase of M will not result in a monotonic performance improvement for VRDR in practice, which, however, may increase significantly the computational cost, since more baselines (subspaces) are employed and more samples are required for estimating these baselines. Thus, we choose a relative small candidate set $\{2, 3, 4, 5, 6\}$ for the practical tuning of M.

2) Clustering Interval N_c: Each new clustering result may change the group that each environment belong to, hence altering the value function and affecting the policy training in the next clustering interval. The underlying dynamic mechanisms vary

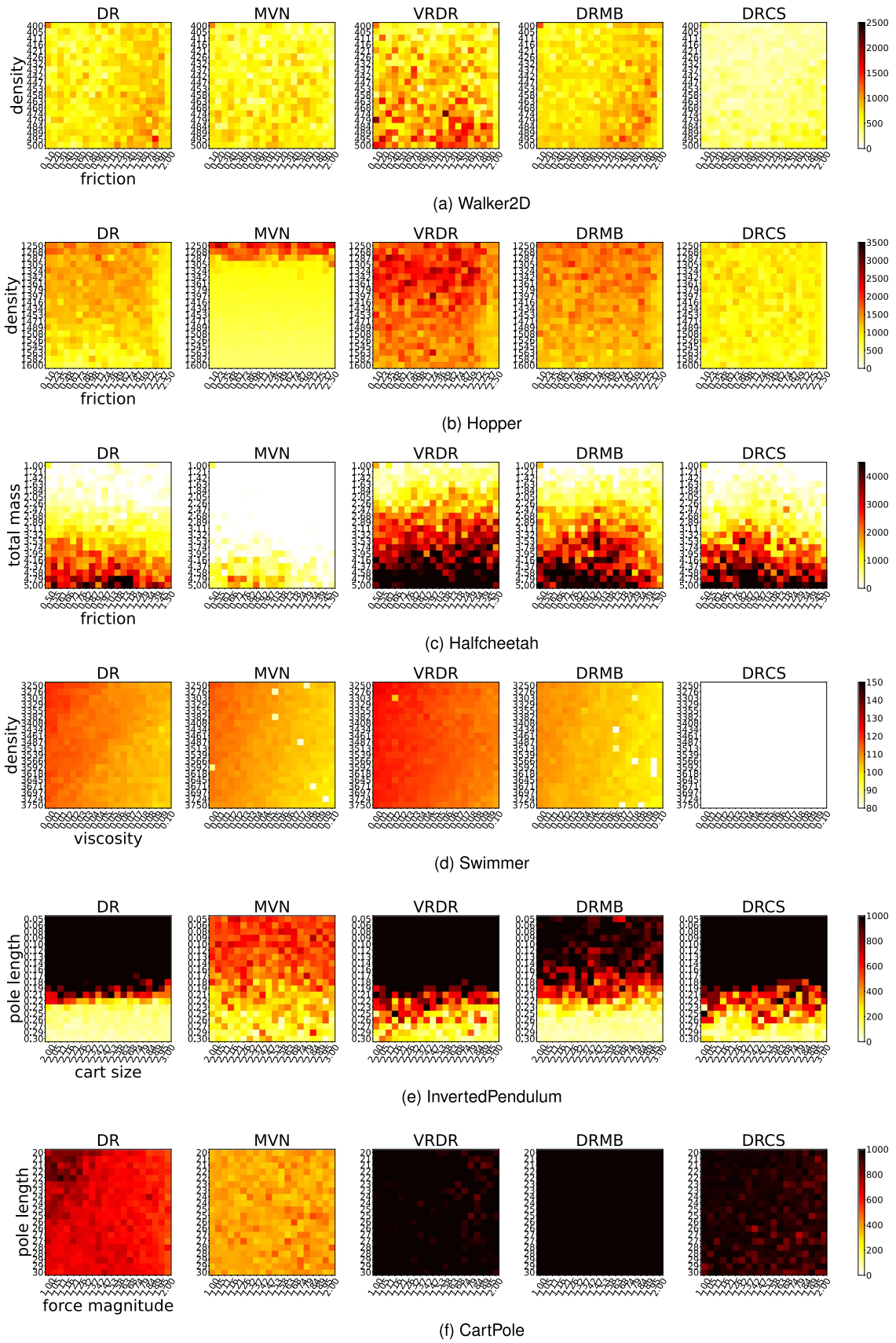


Fig. 7. Heatmap of return in unseen environments with policies trained by DR, MVN, VRDR, DRMB, and DRCS in the training environments.

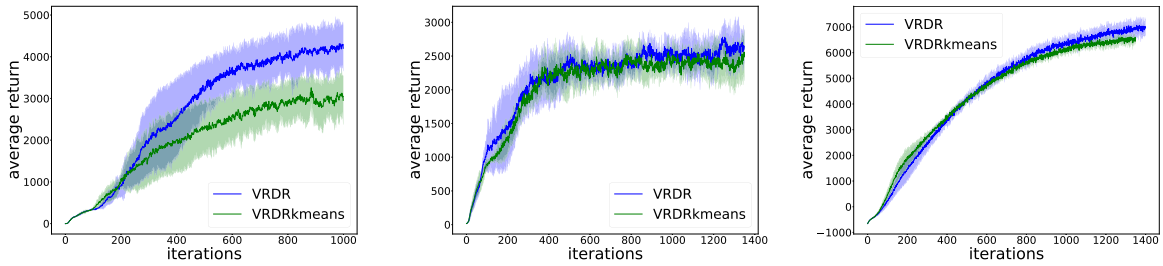


Fig. 8. Training curves of average return achieved by using k-means and hierarchical clustering in VRDR.

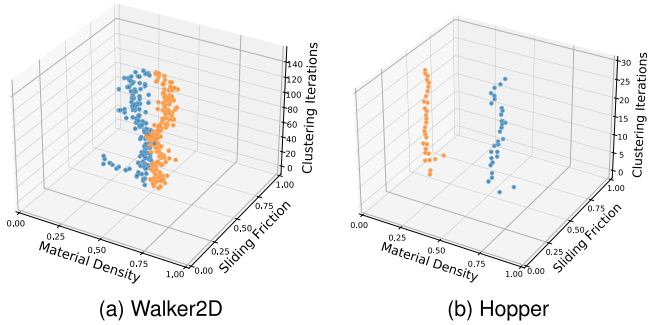


Fig. 9. Visualization of the clustering results during training of VRDR. The points of different colors are clustering prototypes of different clusters w.r.t. the clustering iterations, and the environment parameters are normalized.

in different environments, and thus the shifting of value function may affect the training with varying degrees. We show the training curves of VRDR with different clustering interval N_c on Walker2D, Hopper, and InvertedPendulum in Fig. 6(a)–(c). Through ablation study on the choice of N_c, we find that for the more complex mujoco tasks, a more significant performance difference of VRDR is observed within the range [50,100]. For example, the best performance can be achieved by VRDR by setting $N_c = 50$ for Walker2D, $N_c = 100$ for Hopper, $N_c = 100$ for Halfcheetah, and $N_c = 100$ for Swimmer, respectively. While for the relatively small and easy tasks like InvertedPendulum, they are not that sensitive to the different value N_c, but will still get better performance when N_c gets larger. Hence, a typical candidate set for tuning N_c can be determined as $\{30, 50, 70, 100, 150\}$.

3) Clustering Methods: Considering that we require to specify the number of clusters in VRDR and the clusters should be generated with similar state values to satisfy (34), we additionally chose the k-means cluestering method in comparison to the originally adopted hierarchical clustering summarized in Algorithm 2. The comparison results of different clustering methods are shown in Fig. 8(a), (b) and (c). It can be seen that applying hierarchical clustering methods presents better average scores than applying the k-means clustering on Walker2D, Hopper, and Halfcheetah.

F. Analysis of Additional Computational Cost of VRDR

In DR, the computational cost contains mainly the feed-forward computation and back-propagation computation of the policy network and the value network. For N iterations, H trajectories with length of L_h, size of state space $|\mathcal{S}|$ and constant computational cost of feed-forward and back-propagation computation d_1, the computational cost of DR is $O(N \cdot H \cdot L_h \cdot |\mathcal{S}| \cdot d_1)$.

TABLE IV
TOTAL RUNNING TIME OF DR AND VRDR FOR EACH TASK (UNIT: SECOND)

Environment	DR	VRDR
Walker2D	$(11.23 \pm 1.28) \times 10^4$	$(11.22 \pm 0.99) \times 10^4$
Hopper	$(15.14 \pm 1.40) \times 10^4$	$(12.66 \pm 0.63) \times 10^4$
Halfcheetah	$(15.56 \pm 0.05) \times 10^4$	$(16.72 \pm 0.04) \times 10^4$
Swimmer	$(10.38 \pm 0.04) \times 10^4$	$(11.10 \pm 0.00) \times 10^4$
InvertedPendulum	$(12.42 \pm 0.28) \times 10^4$	$(13.30 \pm 0.06) \times 10^4$
CartPole	$(10.84 \pm 0.51) \times 10^4$	$(11.84 \pm 0.26) \times 10^4$

The additional computational cost introduced by VRDR is mainly from the clustering phase. Considering the hierachical clustering method in Algorithm 2. The maximum number of outer loop is $H - M < H$. At each outer loop, the computational cost of distance matrix is $m^2 \leq H^2$. Then, let the cost of distance computation in Line 6 of Algorithm 2 be d_2, the computational cost is $O(N \cdot H^3 \cdot d_2)$.

Note that compared with the simple computation of distance between two scalars u_i and u_j, the computational cost of feed-forward and back-propagation computation are implemented for the entire policy and value networks, both of which contain multiple layers and neurons. Therefore, in practice, d_1 is usually multiple order-of-magnitude larger than d_2. Thus, under the condition that L_h and $|\mathcal{S}|$ are comparable with H, term $O(N \cdot H \cdot L_h \cdot |\mathcal{S}| \cdot d_1)$ will become the dominate term in the overall computational complexity of VRDR, while the additional computational complexity introduced by the clustering phase of VRDR can be neglected.

Besides the theoretical analysis on computational cost, we further measure the total running time of DR and VRDR in training, as shown in Table IV, which are all computed over five random seeds. For each training task, we run DR and VRDR with the same number of iterations on the same computing device, where we randomly sample 100 environments at each iteration during training as in the common domain randomization settings. Though the training process may encounter different changes in the environment parameters due to the random sampling of environments, a large number of sampled environments and multiple runs with different random seeds can reduce the stochasticity of evaluation on the running time of

DR and VRDR. Surprisingly on Walker2D and Hopper, VRDR requires a slightly less running time than DR. The possible reason is that the length of trajectories is not fixed and thus the episode might be ended early due to fall of the robot, which randomly shortens the duration of an episode and affects the running time.

It can also be seen that VRDR requires additional seven to nine percent of running time on Halfcheetah, Swimmer, InvertedPendulum, and CartPole to complete the training compared to DR due to the additional computational cost, which is acceptable in real deployment. These tasks all have a fixed length of episode, hence the running time difference is mainly determined by the additional computational cost in VRDR.

G. Visualization of Clustering in VRDR

We show the clustering results (prototypes) during training processes of VRDR on Walker2D and Hopper in Fig. 9(a) and (b), respectively, when the number of subspace is set to $M = 2$. The clustering results are rolled out along the vertical axis. At the beginning of training, the prototypes are updated with larger steps, which corresponds to the poor performance in the early stage. Then, at the middle and later stages, the prototypes both tend to become steady when the training performance is getting stable.

VII. Conclusion

In this paper, we have focused on reducing the high variance of gradient estimate in policy gradient methods for DR with additional randomness on environment parameters. Specifically, we derived the optimal state/environment-dependent baseline, and verified that by incorporating the environment information further variance reduction could be achieved over the optimal constant or state-only baselines. We then proposed a variance reduced domain randomization (VRDR) approach for policy gradient methods, to strike a tradeoff between the variance reduction and computational complexity in practical implementation. We have validated VRDR on six robot control tasks, demonstrating an overall faster convergence speed of policy training, and eventually a better and stabilized policy performance.

References

[1] D. Ye et al., "Mastering complex control in MOBA games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 6672–6679.

[2] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare, "Contrastive behavioral similarity embeddings for generalization in reinforcement learning," 2021, *arXiv:2101.05265*.

[3] C. Huang, G. Wang, Z. Zhou, R. Zhang, and L. Lin, "Reward-adaptive reinforcement learning: Dynamic policy gradient optimization for bipedal locomotion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7686–7695, Jun. 2023.

[4] A. C. Egeaa, R. Zhanga, and N. Waltona, "Reinforcement learning for traffic signal control: Comparison with commercial systems," *Transp. Res. Procedia*, vol. 58, pp. 638–645, 2021.

[5] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1282–1289.

[6] C. Lyle, M. Rowland, W. Dabney, M. Kwiatkowska, and Y. Gal, "Learning dynamics and generalization in deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 14560–14581.

[7] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810.

[8] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6008–6014.

[9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[10] Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong, "Monotonic robust policy optimization with model discrepancy," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 4951–4960.

[11] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-efficient domain randomization with Bayesian optimization," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 911–918, Apr. 2021.

[12] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4955–4961.

[13] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *J. Mach. Learn. Res.*, vol. 5, pp. 1471–1530, 2004.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[15] O. M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.

[16] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "MetaDrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3461–3475, Mar. 2023.

[17] Z. Lin, G. Thomas, G. Yang, and T. Ma, "Model-based adversarial meta-reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, Art. no. 852.

[18] H. Zhang, H. Chen, D. S. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," in *Proc. Int. Conf. Learn. Representations*, 2020.

[19] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Proc. Conf. Robot Learn.*, 2020, pp. 1162–1176.

[20] Y. Liu, P. Ramachandran, Q. Liu, and J. Peng, "Stein variational policy gradient," in *Proc. Uncertainty Artif. Intell.*, 2017.

[21] S. Paul, M. A. Osborne, and S. Whiteson, "Fingerprint policy optimisation for robust reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5082–5091.

[22] M. Mozian, J. C. G. Higuera, D. Meger, and G. Dudek, "Learning domain randomization distributions for training robust locomotion policies," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020, pp. 6112–6117.

[23] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters, "Neural posterior domain randomization," in *Proc. Conf. Robot Learn.*, 2022, pp. 1532–1542.

[24] J. Bjorck, C. P. Gomes, and K. Q. Weinberger, "Is high variance unavoidable in RL? A case study in continuous control," in *Proc. Int. Conf. Learn. Representations*, 2022.

[25] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 315–323.

[26] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1646–1654.

[27] Y. Liu, F. Shang, H. Liu, L. Kong, L. Jiao, and Z. Lin, "Accelerated variance reduction stochastic ADMM for large-scale machine learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4242–4255, Dec. 2021.

[28] L. Liu, J. Liu, and D. Tao, "Variance reduced methods for non-convex composition optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5813–5825, Sep. 2022.

[29] Z. Liu, T. D. Nguyen, A. Ene, and H. Nguyen, "Adaptive accelerated (extra-) gradient methods with variance reduction," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 13947–13994.

[30] A. Han and J. Gao, "Improved variance reduction methods for riemannian non-convex optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7610–7623, Nov. 2022.

[31] A. Cutkosky and F. Orabona, "Momentum-based variance reduction in non-convex SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, Art. no. 1365.

[32] F. Huang, S. Gao, J. Pei, and H. Huang, "Momentum-based policy gradient methods," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4422–4433.
[33] P. Xu, F. Gao, and Q. Gu, "Sample efficient policy gradient methods with recursive variance reduction," in *Proc. Int. Conf. Learn. Representations*, 2020.
[34] H. Yang and J. Kwok, "Efficient variance reduction for meta-learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 25070–25095.
[35] C.-A. Cheng, X. Yan, and B. Boots, "Trajectory-wise control variates for variance reduction in policy gradient methods," in *Proc. Conf. Robot Learn.*, 2020, pp. 1379–1394.
[36] T. Spooner, N. Vadori, and S. Ganesh, "Factored policy gradients: Leveraging structure for efficient learning in MOMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 5481–5493.
[37] C. Wu et al., "Variance reduction for policy gradient with action-dependent factorized baselines," 2018, *arXiv:1803.07246*.
[38] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 363.
[39] J. Guo et al., "Hindsight value function for variance reduction in stochastic dynamic environment," 2021, *arXiv:2107.12216*.
[40] J. Kuba et al., "Settling the variance of multi-agent policy gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 13458–13470.
[41] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance reduction for reinforcement learning in input-driven environments," in *Proc. Int. Conf. Learn. Representations*, 2018.
[42] H. Liu, R. Socher, and C. Xiong, "Taming MAML: Efficient unbiased meta-reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4061–4071.
[43] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[44] P. Xu, F. Gao, and Q. Gu, "An improved convergence analysis of stochastic variance-reduced policy gradient," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 541–551.
[45] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2341–2368, 2013.
[46] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.
[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[48] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.
[49] P. Dhariwal et al., "OpenAI baselines," 2017. [Online]. Available: https://github.com/openai/baselines
[50] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 392.
[51] G. Tucker, S. Bhupatiraju, S. Gu, R. Turner, Z. Ghahramani, and S. Levine, "The mirage of action-dependent baselines in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5015–5024.

Yuankun Jiang received the BS degree from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, in 2018. He is currently working toward the PhD degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU). His research interests include robust reinforcement learning and generalization in reinforcement learning.

Chenglin Li (Member, IEEE) received the BS, MS, and PhD degrees in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2007, 2009 and 2015, respectively. From 2015 to 2017, he was a postdoctoral researcher with the Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. From 2017 to 2018, he was a senior researcher with the Chair of Media Technology (LMT), Technical University of Munich (TUM), Germany, supported by the Hildegard Maier Research Fellowship of the Alexander von Humboldt Foundation for postdoctoral researchers. He is currently a full professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. His current research interests include multimedia signal processing and communications, adaptive video streaming, and theories and applications of reinforcement learning and federated learning in multimedia communication systems. He was awarded the Microsoft Research Asia (MSRA) fellow, in 2011, the Alexander von Humboldt Research fellow, in 2017, and the ACM Multimedia Top Paper Award, in 2022.

Wenrui Dai (Member, IEEE) received the BS, MS, and PhD degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2005, 2008, and 2014. He is currently an associate professor with the Department of Computer Science and Engineering, SJTU. Before joining SJTU, he was with the Faculty of the University of Texas Health Science Center at Houston from 2018 to 2019. He was a postdoctoral scholar with the Department of Biomedical Informatics, University of California, San Diego from 2015 to 2018 and a postdoctoral scholar with the Department of Computer Science and Engineering, SJTU from 2014 to 2015. His research interests include learning-based image/video coding, image/signal processing and predictive modeling.

Junni Zou (Member, IEEE) received the MS and PhD degrees in communication and information system from Shanghai University, Shanghai, China, in 2004 and 2006, respectively. From 2006 to 2017, she was with the School of Communication and Information Engineering, Shanghai University, Shanghai, where she is a full professor. Since February 2017, she has been a full professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. From June 2011 to June 2012, she was with the Department of Electrical and Computer Engineering, University of California, San Diego (UCSD), as a visiting professor. Her research interests include multimedia networking and communications, machine learning and network resource optimization. She has published more than 100 international journal/conference papers on these topics. She was the co-author of a Top 10% Paper Award in VCIP 2016. She was granted National Science Fund for Outstanding Young Scholar, in 2016. She was the recipient of Shanghai Young Rising-Star Scientist, in 2011. She is the associate editor of *Digital Signal Processing*.

Hongkai Xiong (Senior Member, IEEE) received the PhD degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2003. He is a distinguished professor with both the Department of Electronic Engineering and Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU). Since then, he has been with the Department of Electronic Engineering, SJTU. He was an associate professor from 2005 to 2011 and an assistant professor from 2003 to 2005. From 2007 to 2008, he was a research scholar with the Department of Electrical and Computer Engineering, Carnegie Mellon University (CMU), Pittsburgh, Pennsylvania . During 2011–2012, he was a scientist with the Division of Biomedical Informatics, University of California (UCSD), San Diego, California. His research interests include multimedia signal processing, image and video coding, multimedia communication and networking, computer vision, biomedical informatics, machine learning and published more than 200 refereed journal and conference papers. He was the co-author of a Best Paper in IEEE BMSB 2013, a Best Student Paper in VCIP 2014, a Top 10% Paper Award in VCIP 2016, and a Top 10% Paper Award in MMSP 2011. He is the associate editor of *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*.